LISACode Reference Manual

1.2

Generated by Doxygen 1.3.4

Tue May 22 19:43:01 2007

Contents

Chapter 1

Developers' Manual of LISACode

This document provides a description of the LISACode software for future developpers. Some information about the code execution (LISACode parameters) are also provided to users. This manual is divided in three sections:

- Introduction
- A description of the Code
- LISACode parameters

Chapter 2

LISACode Module Index

2.1 LISACode Modules

Here is a list of all modules:

Chapter 3

LISACode Namespace Index

3.1 LISACode Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 4

LISACode Hierarchical Index

4.1 LISACode Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 5

LISACode Class Index

5.1 LISACode Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 6

LISACode File Index

6.1 LISACode File List

Here is a list of all files with brief descriptions:

Chapter 7

LISACode Page Index

7.1 LISACode Related Pages

Here is a list of all related documentation pages:

Chapter 8

LISACode Module Documentation

8.1 Noise (directory Bruits)

Classes

- class Noise

  Noise base class.

- class NoiseFile

  Noise derived class to treat files with noise data.

- class NoiseFilter

  Noise derived class to treat noise filters.

- class NoiseWhite

  Noise derived class to treat white noise.

8.2 Detector (directory Dectecteur)

Classes

- class LISA

  This class contains and manages all the elements necessary to LISA satellites simulation.

- class PhoDetPhaMet

  Phasemeter photodiode class.

- class TrFctGW

  Gravitational Waves Transfer Function class.

Enumerations

- enum NOISEORIG { LA, OB, IM, OP }
- enum PDPMINTERF { S, TAU }

  Photodetector-phasemeter interferences type.

8.2.1 Enumeration Type Documentation

8.2.1.1 enum NOISEORIG

Enumeration values:

LA Laser Noise

OB Optical Bench Noise

IM Inertial Mass Noise

OP Optical Path Noise = Shot Noise + Others Optical Path Noises

Definition at line 49 of file LISACODE-PhoDetPhaMet.h.

8.2.1.2 enum PDPMINTERF

Photodetector-phasemeter interferences type.

Enumeration values:

S interferences between external and internal beams

TAU interferences between the two internal beams

Definition at line 56 of file LISACODE-PhoDetPhaMet.h.

8.3 Input data (directory Input_data)

Classes

- class ConfigSim

  Class to configure LISA simulation, that is, LISACode execution.

- struct NoiseSpec

  Noise specification structure.

8.4 Couple

Classes

- class Couple

  Couple management class.

8.5 Elliptic Filter

Classes

- struct QuadCell

  Elliptic cell structure.

Defines

- #define alog(A) log(A)
- #define alog10(A) log10(A)

Functions

- complex< double > I (0, 1)

  Pure imaginary=(0,1).

- void elli (double eps, double A, double fa, double fb, double fe, int NCellMax, int ∗NCells, complex< double > poles[ ], complex< double > zeros[ ], double CoefA[ ], double CoefB[ ], double CoefC[ ], double CoefD[ ])

  Poles, zeros and elliptic cells coefficients computation.

- double ak (double y)

  Integral filter parameter computation.

- double cak (double x)

  Developped filter parameter computation.

- double sn (double y, double A, double ak1, double ak3)

  Recursive or direct coefficients computation.

- double FilterQuadCell (double xn, QuadCell ∗Cell)

  Elliptic cell filtering step, depending on xn and Cell (type QuadCell) inputs.

- double FilterQuadCellChain (double xn, int NCells, QuadCell Cell[ ])

  Elliptic cells chain filtering step, depending on xn, number of cells NCells and Cell (type QuadCell) inputs.

- complex< double > TransfZQuadCell (complex< double > Z, QuadCell Cell)

  Elliptic cell Z transform.

- complex< double > TransfZQuadCellChain (complex< double > Z, int NCells, QuadCell Cell[ ])

  Elliptic cells chain Z transform, depending on Z, number of cells NCells and Cell (type QuadCell) inputs.

- double AbsRespFunctQuadCell (double f, QuadCell Cell)

  Frequency response modulus, depending on f frequency and Cell (type QuadCell) inputs.

- double AbsRespFunctQuadCellChain (double f, int NCells, QuadCell Cell[ ])

Elliptic cells chain frequency response modulus, depending on f frequency, number of cell NCells and Cell (type QuadCell) inputs.

- double HmQuadCell (QuadCell Cell)

  Returns max |1/D(w)|, where D(w) is the denominator of an ellitpic cell (type QuadCell).

- double KmQuadCell (QuadCell Cell)

  Returns max|Ell(w)|, where Ell(w) is the frequency response of an ellitpic cell (type QuadCell).

- void PoleMatching (int NCells, QuadCell Cell[ ])

  Matches nearest poles for a chain of elliptic cells.

- void OrderCellMaxNorm (int NCells, QuadCell Cell[ ])

  Orders cells according to the the max of inf norm.

- double CalcScalingFact (int NCells, QuadCell Cell[ ])

  Scale factors computation for an elliptic cells chain : a0 attributes are updated and global factor is returned.

- double CalcEllipticFilter (double fe, double at, double bp, double fb, double fa, int NCellMax, QuadCell **FilterCellsOut, int *NCellsOut)

  Computes filter coefficients from user specifications and returns the global scale factor.

- void CalcEllipticFilter4LISACode (double fe, double at, double bp, double fb, double fa, int NCellMax, double CellsCoef[ ][5], int *NCellsOut)

  Computes filter coefficients from LISA Code user specifications.
  The global scale factor is included in the first cell.

8.5.1 Define Documentation

8.5.1.1 #define alog(A) log(A)

Definition at line 15 of file LISACODE-EllipticFilter.h.

Referenced by cak(), and elli().

8.5.1.2 #define alog10(A) log10(A)

Definition at line 16 of file LISACODE-EllipticFilter.h.

8.5.2 Function Documentation

8.5.2.1 double AbsRespFunctQuadCell (double *f*, QuadCell *Cell*)

Frequency response modulus, depending on f frequency and Cell (type QuadCell) inputs.

$$\text{returned value} = \sqrt{\frac{(a0_{Cell} \cdot (a1_{Cell} + 2 \cdot cos(2 \cdot \pi \cdot f)))^2}{1 + b1_{Cell}^2 + b2_{Cell}^2 + 2 \cdot b1_{Cell} \cdot b2_{Cell} \cdot cos(2 \cdot \pi \cdot f) + 2 \cdot b2_{Cell} \cdot cos(4 \cdot \pi \cdot f)}}$$

Definition at line 410 of file LISACODE-EllipticFilter.cpp.

References QuadCell::a0, QuadCell::a1, QuadCell::b1, and QuadCell::b2.

Referenced by AbsRespFunctQuadCellChain().

8.5.2.2 double AbsRespFunctQuadCellChain (double *f*, int *NCells*, QuadCell *Cell*[ ])

Elliptic cells chain frequency response modulus, depending on f frequency, number of cell NCells and Cell (type QuadCell) inputs.

For each cell, AbsRespFunctQuadCell function is called.

Returned value is product of all cells frequency response modulus.

Definition at line 434 of file LISACODE-EllipticFilter.cpp.

References AbsRespFunctQuadCell().

8.5.2.3 double ak (double *y*)

Integral filter parameter computation.

cak function is called.

$$\text{returned value} = cak(1-y^2)$$

Definition at line 232 of file LISACODE-EllipticFilter.cpp.

References cak().

Referenced by elli().

8.5.2.4 double cak (double *x*)

Developped filter parameter computation.

Numerical problems that could occur when x is near to zero are avoided.

$$\text{returned value} = 1.38629436112+0.09666344259\cdot x+0.03590092383\cdot x^2+0.0374256371\cdot x^3+0.01451196212\cdot x^4$$

$$-alog(x)\cdot(0.5+0.12498593597\cdot x+0.068802485763\cdot x^2+0.03328355346\cdot x^3+0.004417870122\cdot x^4)$$

Definition at line 250 of file LISACODE-EllipticFilter.cpp.

References alog.

Referenced by ak(), and elli().

8.5.2.5 double CalcEllipticFilter (double *fe*, double *at*, double *bp*, double *fb*, double *fa*, int *NCellMax*, QuadCell ∗∗ *FilterCellsOut*, int ∗ *NCellsOut*)

Computes filter coefficients from user specifications and returns the global scale factor.

Inputs are:

- fe : sampling frequency [Hz]
- at : attenuation [dB]
- bp : oscillations in bandwidth [dB]

- fb : low transition frequency [Hz]
- fa : high transition frequency [Hz]
- NCellMax : maximum number of cells

Outputs are :

- FilterCellsOut : cells
- NCellsOut : number of cells

elli function is called and its outputs are NCellsOut, Poles, Zeros, CoefA, CoefB, CoefC, CoefD.

For all cells (index i=0,...,NCells-1), FilterCellsOut[i] are filled :

$$u = (0, 0)$$

$$a0 = 1$$
$$a1 = CoefB[i]$$
$$b1 = CoefD[i]$$
$$b2 = CoefC[i]$$
$$zero = Zeros[i]$$
$$pole = Poles[i]$$

PoleMatching, then OrderCellMaxNorm and CalcScalingFact functions are called using NCells and FilterCells arguments.

CalcScalingFact result is returned.

Definition at line 709 of file LISACODE-EllipticFilter.cpp.

References CalcScalingFact(), elli(), OrderCellMaxNorm(), PoleMatching(), and QuadCell::u.

Referenced by CalcEllipticFilter4LISACode().

8.5.2.6 void CalcEllipticFilter4LISACode (double *fe*, double *at*, double *bp*, double *fb*, double *fa*, int *NCellMax*, double *CellsCoef*[ ][5], int $*$ *NCellsOut*)

Computes filter coefficients from LISA Code user specifications.

The global scale factor is included in the first cell.

Inputs are:

- fe : sampling frequency [Hz]
- at : attenuation [dB]
- bp : oscillations in bandwidth [dB]
- fb : low transition frequency [Hz]
- fa : high transition frequency [Hz]
- NCellMax : maximum number of cells

Outputs are :

- CellsCoef : cells coefficients
- NCellsOut : number of cells

CalcEllipticFilter function is called : its outputs are NCellsOut, FilterCells, and its returned value is global scaling factor ag.

For all cells (index i=0,...,NCellsOut-1) :

$$u_{FilterCellsOut[i]} = (0,0)$$

$$CellsCoef[i][0] = b1_{FilterCells[i]}$$

$$CellsCoef[i][1] = b2_{FilterCells[i]}$$

$$CellsCoef[i][2] = a0_{FilterCells[i]}$$

$$CellsCoef[i][3] = a0_{FilterCells[i]} \cdot a1_{FilterCells[i]}$$

$$CellsCoef[i][4] = a0_{FilterCells[i]}$$

First cell is rescaled :

$$CellsCoef[0][2] = ag \cdot a0_{FilterCells[i]}$$

$$CellsCoef[0][3] = ag \cdot a0_{FilterCells[i]} \cdot a1_{FilterCells[i]}$$

$$CellsCoef[0][4] = ag \cdot a0_{FilterCells[i]}$$

Definition at line 828 of file LISACODE-EllipticFilter.cpp.

References QuadCell::b1, and CalcEllipticFilter().

Referenced by Filter::Filter().

8.5.2.7 double CalcScalingFact (int *NCells*, QuadCell *Cell*[ ])

Scale factors computation for an elliptic cells chain : a0 attributes are updated and global factor is returned.

Elliptic cells chain is defined by number of cells NCells and Cell (type QuadCell) inputs.

HmQuadCell and KmQuadCell functions are called for each cell. Results are : Hm_i, Km_i.

$$Hm_i = HmQuadCell(Cell[i])$$

$$Km_i = KmQuadCell(Cell[i])$$

For all cells execpt the last one (index i=0,...,NCells-2) :

$$ac_i = \frac{1}{a1_{Cell[i]}} - (b1_{Cell[i]} \cdot \frac{1 + b2_{Cell[i]}}{2 \cdot b2_{Cell[i]}}) \cdot Hm_{i+1}$$

$$n_i = ceil(\frac{log(ac_i)}{log(2) + \frac{1}{2}}$$

$$a0_{Cell[i]} = 2^{n_i}$$

For last cell:

$$norm = \frac{1}{Hm_0} \cdot \prod_{i=0}^{NCells-1} (a0_{Cell[i]} \cdot \frac{2 + a1_{Cell[i]}}{1 + b1_{Cell[i]} + b2_{Cell[i]}})$$

$$a0_{Cell[NCells-1]} = \frac{1}{norm}$$

$$\text{returned value : } ag = \frac{1}{Hm_0}$$

Definition at line 644 of file LISACODE-EllipticFilter.cpp.

References HmQuadCell(), and KmQuadCell().

Referenced by CalcEllipticFilter().

8.5.2.8 void elli (double *eps*, double *A*, double *fa*, double *fb*, double *fe*, int *NCellMax*, int $*$ *NCells*, complex< double > *poles*[ ], complex< double > *zeros*[ ], double *CoefA*[ ], double *CoefB*[ ], double *CoefC*[ ], double *CoefD*[ ])

Poles, zeros and elliptic cells coefficients computation.

Inputs are:

- eps : Oscillations in working bandwidth
- A : Weakening of attenuated band
- f : Low frequency transition edge [Hz]
- fb : High frequency transition edge [Hz]
- fe : Sampling frequency [Hz]
- NCellMax : Maximum number of cells

Outputs are :

- NCells : number of cells, must be positive and lower or equal to NCellMax
- poles : poles of the cells (imaginary part positive or null)
- zero : zeros of the cells (imaginary part positive or null)
- CoefA : coefficient A of the cells
- CoefB : coefficient B of the cells
- CoefC : coefficient C of the cells
- CoefD : coefficient D of the cells

Computations :

$\omega_c = fb \cdot 2 \cdot \pi$

$\omega_r = fa \cdot 2 \cdot \pi$

$T = 1/fe$

$dk1 = \frac{eps}{\sqrt{A^2-1}}$

$dk = \frac{tan(\omega_c \cdot \frac{T}{2})}{tan(\omega_r \cdot \frac{T}{2})}$

$dkp = \sqrt{1 - dk^2}$

$ak1 = ak(dk)$ using ak function

$ak2 = ak(dk1)$ using ak function

$ak3 = ak(dkp)$ using ak function

$ak4 = cak(dk1^2)$ using cak function

$N = \frac{1}{2} \cdot ceil\big(ceil(\frac{ak4 \cdot ak1}{ak2 \cdot ak3} + 1)\big)$

N is checked : $0 \leq N \leq NCellMax$

$U_0 = -\frac{ak3}{ak4} \cdot \frac{alog(1+\sqrt{(1+eps^2)})}{eps}$

- for $i = 0, \ldots, N-1$
  $xmag = 2 \cdot i \cdot \frac{ak1}{2 \cdot N}$
  $zeros[i] = -ak3 + I \cdot xmag$
  $poles[i] = U_0 + I \cdot xmag$
- for $i = 0, \ldots, 2 \cdot N - 1$
  $Q = real(zeros[mod(i, N)])$
  $R = imag(zeros[mod(i, N)])$
  $a1 = sn(Q, dkp, ak3, ak1)$ using sn function
  $b1 = sn(R, dk, ak1, ak3)$ using sn function
  $\sigma = \begin{cases} 0 & \text{if } i \leq N \\ a1 \cdot \sqrt{(1-a1^2)*(1-b1^2)} \cdot \frac{dn}{de} & else \end{cases}$
  $dn = \sqrt{1 - (dk \cdot b1)^2}$
  $de = 1 - (a1 \cdot dn)^2$
  $\omega = b1 \cdot \frac{\sqrt{(1-(dkp \cdot a1)^2)}}{de}$
  $C[i] = -2 \cdot \sigma \cdot \omega_c$
  $D[i] = (\sigma^2 + \omega^2) \cdot {\omega_c}^2$
  $\sigma = \sigma \cdot tan(\omega_c \cdot \frac{T}{2})$
  $\omega = \omega \cdot tan(\omega_c \cdot \frac{T}{2})$
  $\begin{cases} \text{if } i \leq N & zeros[i] = \sigma + I \cdot \omega \\ \text{else} & poles[i] = \sigma + I \cdot \omega \end{cases}$
- for $i = 2 \cdot N - 1, \ldots, 0$
  $\begin{cases} \text{if } i \leq N-1 & (X, Y) = (real(zeros[i]), imag(zeros[i])) \\ else & (X, Y) = (real(poles[i]), imag(poles[i])) \end{cases}$
  $Re = \frac{1-X^2-Y^2}{(1-X)^2+Y^2}$
  $V = \frac{2 \cdot Y}{(1-X)^2+Y^2}$
  $c1 = -2 \cdot Re$
  $d1 = Re^2 + V^2$
  $\begin{cases} \text{if } i \leq N-1 & \begin{cases} zeros[i] = Re + I \cdot V \\ CoefB[i] = c1 \\ CoefA[i] = d1 \end{cases} \\ else & \begin{cases} poles[i-N] = Re + I \cdot V \\ CoefD[i-N] = c1 \\ CoefC[i-N] = d1 \end{cases} \end{cases}$

Definition at line 89 of file LISACODE-EllipticFilter.cpp.

References ak(), alog, cak(), omega, and sn().

Referenced by CalcEllipticFilter().

8.5.2.9 double FilterQuadCell (double *xn*, QuadCell $*$ *Cell*)

Elliptic cell filtering step, depending on xn and Cell (type QuadCell) inputs.

Computations :

$$u = x_n - b2_{Cell} \cdot u_{Cell}[1] - -b1_{Cell} \cdot u_{Cell}[0]$$

$$\text{returned value} = a0_{Cell} \cdot (u + a1_{Cell} \cdot u_{Cell}[0] + u_{Cell}[1])$$

Cell u memory attribute is updated :

$$\text{new } u_{Cell} = \begin{pmatrix} u \\ \text{old } u_{Cell}[0] \end{pmatrix}$$

Definition at line 330 of file LISACODE-EllipticFilter.cpp.

References QuadCell::a0, QuadCell::a1, QuadCell::b1, QuadCell::b2, and QuadCell::u.

Referenced by FilterQuadCellChain().

8.5.2.10 double FilterQuadCellChain (double *xn*, int *NCells*, QuadCell *Cell*[ ])

Elliptic cells chain filtering step, depending on xn, number of cells NCells and Cell (type QuadCell) inputs.

For each cell, FilterQuadCell function is called.

Cells are updated and returned value depends on last cell.

Definition at line 352 of file LISACODE-EllipticFilter.cpp.

References FilterQuadCell().

8.5.2.11 double HmQuadCell (QuadCell *Cell*)

Returns max |1/D(w)|, where D(w) is the denominator of an ellitpic cell (type QuadCell).

$$\text{returned value} = \frac{1}{(1 - b2_{Cell}) \cdot \sqrt{1 - \frac{b1_{Cell}{}^2}{4 \cdot b2_{Cell}}}}$$

Definition at line 455 of file LISACODE-EllipticFilter.cpp.

References QuadCell::b1, and QuadCell::b2.

Referenced by CalcScalingFact(), and KmQuadCell().

8.5.2.12 complex<double> I (0, 1) `[static]`

Pure imaginary=(0,1).

8.5.2.13 double KmQuadCell (QuadCell *Cell*)

Returns max|Ell(w)|, where Ell(w) is the frequency response of an ellitpic cell (type QuadCell).

HmQuadCell is called and returned value $= a0_{Cell} \cdot (a1_{Cell} - b1_{Cell} \cdot \frac{1+b2_{Cell}}{2\cdot+b2_{Cell}}) \cdot HmQuadCell(Cell)$

Definition at line 474 of file LISACODE-EllipticFilter.cpp.

References QuadCell::a0, QuadCell::a1, QuadCell::b1, QuadCell::b2, and HmQuadCell().

Referenced by CalcScalingFact(), and OrderCellMaxNorm().

8.5.2.14 void OrderCellMaxNorm (int *NCells*, QuadCell *Cell*[ ])

Orders cells according to the the max of inf norm.

Elliptic cells chain is defined by number of cells NCells and Cell (type QuadCell) inputs.

First, KmQuadCell function is called for each cell.

Then, cells are ordered according to the the max of inf norm.

Definition at line 577 of file LISACODE-EllipticFilter.cpp.

References KmQuadCell().

Referenced by CalcEllipticFilter().

8.5.2.15 void PoleMatching (int *NCells*, QuadCell *Cell*[ ])

Matches nearest poles for a chain of elliptic cells.

Elliptic cells chain is defined by number of cells NCells and Cell (type QuadCell) inputs.

First, cells are ordered according to the distance between the pole and the unity circle.

For all cells (i index) the nearest zero (jmin index) corresponding to a pole is found by minimizing $dmin = min_{j=i}^{NCells}(zero_{Cell[j]} - pole_{Cell[i]})$. If $i \neq jmin$, a0, a1 and zero attributes of Cell[i] and Cell[jmin] are inverted.

Definition at line 495 of file LISACODE-EllipticFilter.cpp.

Referenced by CalcEllipticFilter().

8.5.2.16 double sn (double *y*, double *A*, double *ak1*, double *ak3*)

Recursive or direct coefficients computation.

Inputs are:

- y
- A
- ak1
- ak3

Computations :

$$ns = \sqrt{\frac{50 \cdot ak1}{\pi \cdot ak3}} + 2$$

$$x = \frac{y}{2 \cdot ak1}$$

$$q = e^{-\frac{\pi \cdot ak3}{ak1}}$$

$$\text{returned value} = 2 \cdot \frac{q^{\frac{1}{4}} \cdot sin(\pi \cdot x) + \sum_{N=1}^{ns} ((-1)^N \cdot q^{(N+\frac{1}{2})^2} \cdot sin((2 \cdot N + 1) \cdot \pi \cdot x))}{\left(1 + 2 \cdot \sum_{N=1}^{ns} ((-1)^N \cdot q^{N^2} \cdot cos(2 \cdot N \cdot \pi \cdot x))\right) \cdot \sqrt{A}}$$

Definition at line 293 of file LISACODE-EllipticFilter.cpp.

Referenced by elli().

8.5.2.17 complex<double> TransfZQuadCell (complex< double > *Z*, QuadCell *Cell*)

Elliptic cell Z transform.

$$\text{returned value} = \frac{a0_{Cell} \cdot \frac{a1_{Cell} + \frac{1}{Z}}{Z}}{1 + \frac{b1_{Cell} + \frac{b2_{Cell} Z}{Z}}{Z}}$$

Definition at line 371 of file LISACODE-EllipticFilter.cpp.

References QuadCell::a0, QuadCell::a1, QuadCell::b1, and QuadCell::b2.

Referenced by TransfZQuadCellChain().

8.5.2.18 complex<double> TransfZQuadCellChain (complex< double > *Z*, int *NCells*, QuadCell *Cell*[ ])

Elliptic cells chain Z transform, depending on Z, number of cells NCells and Cell (type QuadCell) inputs.

For each cell, TransfZQuadCell function is called.

Returned value is product of all cells Z transform.

Definition at line 390 of file LISACODE-EllipticFilter.cpp.

References TransfZQuadCell().

8.6 Filter

Classes

- class Filter

  filter management class.

8.7 Matrix

Classes

- class Mat

  (3x3) matrix management class.

8.8 Mathematical Tools (directory Outils_Math)

Modules

- Couple
- Elliptic Filter
- Filter
- Matrix
- Angles handling
- RandomMT
- Serie
- Vector

8.9 Angles handling

Classes

- class MathUtils

  Angle conversion class.

Defines

- #define SWAP(a, b) tempr=(a);(a)=(b);(b)=tempr
- #define MIN(a, b) (((a)<(b))?(a):(b))
- #define MAX(a, b) (((a)>(b))?(a):(b))

8.9.1 Define Documentation

8.9.1.1 #define MAX(a, b) (((a)>(b))?(a):(b))

Definition at line 29 of file LISACODE-MathUtils.h.

Referenced by Filter::getDepth(), PhoDetPhaMet::init(), and main().

8.9.1.2 #define MIN(a, b) (((a)<(b))?(a):(b))

Definition at line 28 of file LISACODE-MathUtils.h.

8.9.1.3 #define SWAP(a, b) tempr=(a);(a)=(b);(b)=tempr

Definition at line 27 of file LISACODE-MathUtils.h.

8.10 RandomMT

Classes

- class RandomMT

  Mersenne twister random generator class.

Defines

- #define RANDOM_GENERATOR TRandomMersenne

  Mersenne twister random generator class.

8.10.1 Define Documentation

8.10.1.1 RANDOM_GENERATOR TRandomMersenne

Mersenne twister random generator class.

Definition at line 35 of file LISACODE-Random.h.

8.11 Serie

Classes

- class Serie

  Serie interpolation class.

- class SerieC

  complex serie interpolation class.

Enumerations

- enum INTERP {
  TRU, LIN, CUB, LAG,
  SIN }

  Interpolation type.

8.11.1 Enumeration Type Documentation

8.11.1.1 enum INTERP

Interpolation type.

Enumeration values:

TRU Truncated interpolation

LIN Linear interpolation

CUB Cubic interpolation

LAG Lagrange interpolation

SIN Truncated sinc interpolation

Definition at line 36 of file LISACODE-Serie.h.

Referenced by ConfigSim::getTDIInterp().

8.12 Vector

Classes

- class Vect

  3 components vector management class.

8.13 Gravitational waves (directory Ondes_Gravit)

Classes

- class GW

  Gravitational Waves parameters are described in this class.

- class GWBinary

  Gravitational Waves parameters for a monochromatic binary system are defined in this class.

- class GWFile

  Gravitational Waves file management.

- class GWMono

  Gravitational Waves instantaneous parameters h_plus and h_cross are described in this class.

- class GWNewton2

  Gravitational Waves binary system parameters computation.

- class GWPeriGate

  Gravitational Waves periodic gate signal.

8.14 Background (directory Background)

Classes

- class Background

  Background signal received by phasemeters is described in this class.

- class BackgroundGalactic

  Background Galactic signal received by phasemeters is described in this class.

8.15 USO clock (directory USO_Temps)

Classes

- class USOClock

  Ultra Stable Oscillator based satellite time is defined in this class.

8.16 TDI handling (directory TDI)

Modules

- TDI
- TDI_InterData
- TDITools

8.17 TDI

Classes

- class TDI

  Time Delay Interferometry combinaison class.

8.18 TDI_InterData

Classes

- class TDI_InterData

    Time Delay Interferometry interpolated signal class.

8.19 TDITools

Classes

- class TDITools

  Time Delay Interferometry tools class.

8.20 Geometry (directory Orbitographie)

Classes

- class Geometry

    Orbit geometry class.

8.21 Main (directory Main)

Functions

- int main (int argc, char *const argv[ ])

  LISA simulator.

  - *Initialization.*
    Random generator is initialized.
    Config is a ConfigSim instance created with data read from "ConfigRefBase" file.
    RecordPDPM is a Memory vector where spacecraft signals wil be recorded.
    LISACode is a LISA instance created with Config and RecordPDPM.
    Eta signals are created.
    TDI generators are created using approximative delay computation specified in Config.
  - *Data processing first step : time $t = 0, \ldots, tMemTDI + tTDIShift$ with tStepMes timsetep.*
    Signals are stored.
    LISA::MakeOneStepOfTime method is called.
    Delays are recorded.
    Positions are recorded.
  - *Data processing second step : when there are enough data, TDI is computed and results are stored in file, while time $t \leq tmax$ with tStepMes timsetep.*
    TDI is computed using TDI_InterData::ComputeEta method.
    Delays are recorded.
    Positions are recorded.
    .

8.21.1 Function Documentation

8.21.1.1 int main (int *argc*, char *const *argv*[ ])

LISA simulator.

- Initialization.

  Random generator is initialized.

  Config is a ConfigSim instance created with data read from "ConfigRefBase" file.

  RecordPDPM is a Memory vector where spacecraft signals wil be recorded.

  LISACode is a LISA instance created with Config and RecordPDPM.

  Eta signals are created.

  TDI generators are created using approximative delay computation specified in Config.

- Data processing first step : time $t = 0, \ldots, tMemTDI + tTDIShift$ with tStepMes timsetep.

  Signals are stored.

  LISA::MakeOneStepOfTime method is called.

  Delays are recorded.

  Positions are recorded.

- Data processing second step : when there are enough data, TDI is computed and results are stored in file, while time $t \leq tmax$ with tStepMes timsetep.

  TDI is computed using TDI_InterData::ComputeEta method.

  Delays are recorded.

Positions are recorded.

.

Definition at line 35 of file LISACODE-DnonGW.cpp.

References TrFctGW::deltanu(), ConfigSim::getArmlength(), ConfigSim::getGWs(), ConfigSim::getOrbInitRot(), ConfigSim::getOrbMove(), ConfigSim::getOrbOrder(), ConfigSim::getOrbStartTime(), ConfigSim::gettDisplay(), ConfigSim::gettMax(), ConfigSim::gettStepMes(), Geometry::gposition(), Vect::p, Geometry::tdelay(), and Geometry::VectNormal().

8.22 Memory (directory Memoire)

Classes

- class Memory

  Memory management class.

- class MemoryReadDisk

  Class to manage disk reading.

- class MemoryWriteDisk

  Class to manage disk writting.

Chapter 9

LISACode Namespace Documentation

9.1 std Namespace Reference

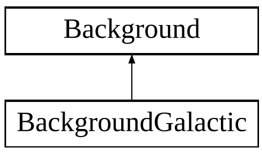
Chapter 10

LISACode Class Documentation

10.1 Background Class Reference

```
#include <LISACODE-Background.h>
```

Inheritance diagram for Background::



10.1.1 Detailed Description

Background signal received by phasemeters is described in this class.

Background signal depends on satellites position given by LISAGeo attribute.

Definition at line 39 of file LISACODE-Background.h.

Public Member Functions

- Background ()

  Constructs an instance and initializes its attribute (LISAGeo) with default values.

- Background (Geometry ∗LISAGeo_n)

  Constructs an instance and initializes its attribute with input LISAGeo_n.

- virtual ∼Background ()

  Destructor.

- void setGeometry (Geometry ∗LISAGeo_n)

  Sets LISAGeo attribute with input.

- virtual double deltanu (int iSC, int IndirSens, double t)

Gives deltanu.

Protected Attributes

- Geometry ∗ LISAGeo
  LISA orbit description.

10.1.2 Constructor & Destructor Documentation

10.1.2.1 Background::Background ()

Constructs an instance and initializes its attribute (LISAGeo) with default values.

- LISAGeo = Geometry instance with default attributes.

Definition at line 22 of file LISACODE-Background.cpp.

References LISAGeo.

10.1.2.2 Background::Background (Geometry ∗ *LISAGeo_n*)

Constructs an instance and initializes its attribute with input *LISAGeo_n*.

- LISAGeo = LISAGeo_n input

Definition at line 33 of file LISACODE-Background.cpp.

References LISAGeo.

10.1.2.3 Background::∼Background () `[virtual]`

Destructor.

Definition at line 41 of file LISACODE-Background.cpp.

10.1.3 Member Function Documentation

10.1.3.1 double Background::deltanu (int *iSC*, int *IndirSens*, double *t*) `[virtual]`

Gives deltanu.

Virtual unused method.

Returns:
0.0

Reimplemented in BackgroundGalactic.

Definition at line 64 of file LISACODE-Background.cpp.

Referenced by PhoDetPhaMet::gGWB().

10.1.3.2 void Background::setGeometry (Geometry * *LISAGeo_n*)

Sets LISAGeo attribute with input.

- LISAGeo = *LISAGeo_n* input

Definition at line 52 of file LISACODE-Background.cpp.

References LISAGeo.

Referenced by LISA::LISA().

10.1.4 Member Data Documentation

10.1.4.1 Geometry* Background::LISAGeo `[protected]`

LISA orbit description.

Definition at line 45 of file LISACODE-Background.h.

Referenced by Background(), BackgroundGalactic::deltanu(), and setGeometry().

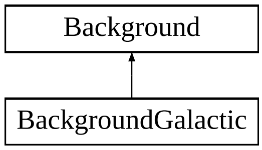
The documentation for this class was generated from the following files:

- LISACODE-Background.h
- LISACODE-Background.cpp

10.2 BackgroundGalactic Class Reference

```
#include <LISACODE-BackgroundGalactic.h>
```

Inheritance diagram for BackgroundGalactic::



10.2.1 Detailed Description

Background Galactic signal received by phasemeters is described in this class.

Definition at line 35 of file LISACODE-BackgroundGalactic.h.

Public Member Functions

- BackgroundGalactic ()

    Constructs an instance and initializes its attributes with default values.

- BackgroundGalactic (char ∗FileName, double Factor)

    Constructs an instance and initializes its attributes with default values and inputs.

- BackgroundGalactic (Geometry ∗LISAGeo_n, char ∗FileName, double Factor)

    Constructs an instance and initializes its attributes with default values and inputs.

- ∼BackgroundGalactic ()

    Destructor.

- void ReadFile (char ∗FileName, double Factor_n)

    Reads signal samples and associated times from file specified in argument.

- double deltanu (int iSC, int IndirSens, double t)

    Gives frequency fluctuation.

- void setGeometry (Geometry ∗LISAGeo_n)

    Sets LISAGeo attribute with input.

Protected Attributes

- vector< double > TimeList

    Vector of time values associated to signal samples.

- vector< vector< double > > SignalList

    Set of signals received by each photometer.

- int NbData

    Number of data or samples.

- int iRead

    Last bin or index read.

- double tmp_t

    Last time value read.

- double tmp_ci

    Last bin time lower or equal to current time.

- double tmp_cip1

    First bin time greater than current time.

- vector< double > tmp_Sig_i

    Signal value corresponding to tmp_ci.

- vector< double > tmp_Sig_ip1

    Signal value corresponding to tmp_cip1.

- Geometry * LISAGeo

    LISA orbit description.

10.2.2 Constructor & Destructor Documentation

10.2.2.1 BackgroundGalactic::BackgroundGalactic ()

Constructs an instance and initializes its attributes with default values.

Default values are:

- LISAGeo = Geometry instance with default attributes.
- TimeList = NULL
- SignalList = NULL
- NbData = 0
- iRead = -1
- tmp_t = -1.0

Definition at line 29 of file LISACODE-BackgroundGalactic.cpp.

References iRead, NbData, SignalList, TimeList, and tmp_t.

10.2.2.2 BackgroundGalactic::BackgroundGalactic (char * *FileName*, double *Factor*)

Constructs an instance and initializes its attributes with default values and inputs.

It calls ReadFile.

Parameters:

FileName: name of file containing signal values and associated times

Factor: scale factor applied to read signal values

Attributes are set as follows

- LISAGeo = Geometry instance with default attributes.
- TimeList is filled by ReadFile method.
- SignalList is filled by ReadFile method.
- NbData is filled by ReadFile method.
- iRead = -1
- tmp_t = -1.0

Definition at line 53 of file LISACODE-BackgroundGalactic.cpp.

References iRead, NbData, ReadFile(), SignalList, TimeList, and tmp_t.

10.2.2.3 BackgroundGalactic::BackgroundGalactic (Geometry * *LISAGeo_n*, char * *FileName*, double *Factor*)

Constructs an instance and initializes its attributes with default values and inputs.

It calls ReadFile.

Parameters:

LISAGeo_n: LISA orbit.

FileName: name of file containing signal values and associated times

Factor: scale factor applied to read signal values

Attributes will be set to next values:

- LISAGeo = LISAGeo_n input
- TimeList is filled by ReadFile method.
- SignalList is filled by ReadFile method.
- NbData is filled by ReadFile method.
- iRead = -1
- tmp_t = -1.0

Definition at line 80 of file LISACODE-BackgroundGalactic.cpp.

References iRead, NbData, ReadFile(), SignalList, TimeList, and tmp_t.

10.2.2.4 BackgroundGalactic::∼BackgroundGalactic ()

Destructor.

Definition at line 94 of file LISACODE-BackgroundGalactic.cpp.

10.2.3 Member Function Documentation

10.2.3.1 double BackgroundGalactic::deltanu (int *iSC*, int *IndirSens*, double *t*) [virtual]

Gives frequency fluctuation.

Parameters:

iSC: spacecraft index (1,2 or 3 for each LISA satellite)

IndirSens: direction flag (0 if the optical bench is in the direct direction, else 1)

t: time

It computes *tGeo* to obtain deltanu:

$tGeo = mod(t + t0_{LISAGeo}, 31557600)$ (annual periodicity)

If $tGeo \notin [TimeList[0], TimeList[size(TimeList) - 2])$, returned value =0.

Else:

iRead index is found, such as $TimeList[iRead] \leq tGeo < TimeList[iRead + 1]$

$$\text{returned value} = \frac{TimeList[iRead + 1] - tGeo}{TimeList[iRead + 1] - TimeList[iRead]} \cdot tmp\_Sig\_i[iSC + 3 \cdot IndirSens - 1]$$

$$+ \frac{tGeo - TimeList[iRead]}{TimeList[iRead + 1] - TimeList[iRead]} \cdot tmp\_Sig\_ip1[iSC + 3 \cdot IndirSens - 1]$$

Reimplemented from Background.

Definition at line 165 of file LISACODE-BackgroundGalactic.cpp.

References Geometry::gett0(), iRead, Background::LISAGeo, PRECISION, SignalList, TimeList, tmp_ci, tmp_cip1, tmp_Sig_i, tmp_Sig_ip1, and tmp_t.

10.2.3.2 void BackgroundGalactic::ReadFile (char ∗ *FileName*, double *Factor*)

Reads signal samples and associated times from file specified in argument.

NbData is set to number of signal samples read from file.

It runs as follows: File's lines beginning with "#" character are ignored. While end of file is not reached :

- time is read, then pushed back into TimeList attribute
- 6 signals are read, then multiplied by *Factor* input and pushed back into SignalList attribute

Definition at line 113 of file LISACODE-BackgroundGalactic.cpp.

References NbData, ReadFile(), SignalList, and TimeList.

Referenced by BackgroundGalactic(), and ReadFile().

10.2.3.3 void Background::setGeometry (Geometry ∗ *LISAGeo_n*) [inherited]

Sets LISAGeo attribute with input.

- LISAGeo = *LISAGeo_n* input

Definition at line 52 of file LISACODE-Background.cpp.

References Background::LISAGeo.

Referenced by LISA::LISA().

10.2.4 Member Data Documentation

10.2.4.1 BackgroundGalactic::iRead [protected]

Last bin or index read.

Referenced by BackgroundGalactic(), and deltanu().

10.2.4.2 Geometry∗ Background::LISAGeo [protected, inherited]

LISA orbit description.

Definition at line 45 of file LISACODE-Background.h.

Referenced by Background::Background(), deltanu(), and Background::setGeometry().

10.2.4.3 BackgroundGalactic::NbData [protected]

Number of data or samples.

Referenced by BackgroundGalactic(), and ReadFile().

10.2.4.4 vector< vector<double> > BackgroundGalactic::SignalList [protected]

Set of signals received by each photometer.

Parameters:

SignalList[i][j] is the j-th sample of signal received by the i-th photometer.

Definition at line 45 of file LISACODE-BackgroundGalactic.h.

Referenced by BackgroundGalactic(), deltanu(), and ReadFile().

10.2.4.5 vector<double> BackgroundGalactic::TimeList [protected]

Vector of time values associated to signal samples.

Definition at line 39 of file LISACODE-BackgroundGalactic.h.

Referenced by BackgroundGalactic(), deltanu(), and ReadFile().

10.2.4.6 BackgroundGalactic::tmp_ci [protected]

Last bin time lower or equal to current time.

Referenced by deltanu().

10.2.4.7 BackgroundGalactic::tmp_cip1 [protected]

First bin time greater than current time.

Referenced by deltanu().

10.2.4.8 BackgroundGalactic::tmp_Sig_i [protected]

Signal value corresponding to tmp_ci.

Referenced by deltanu().

10.2.4.9 BackgroundGalactic::tmp_Sig_ip1 [protected]

Signal value corresponding to tmp_cip1.

Referenced by deltanu().

10.2.4.10 BackgroundGalactic::tmp_t [protected]

Last time value read.

Referenced by BackgroundGalactic(), and deltanu().

The documentation for this class was generated from the following files:

- LISACODE-BackgroundGalactic.h
- LISACODE-BackgroundGalactic.cpp

10.3 ConfigSim Class Reference

`#include <LISACODE-ConfigSim.h>`

10.3.1 Detailed Description

Class to configure LISA simulation, that is, LISACode execution.

Definition at line 82 of file LISACODE-ConfigSim.h.

Public Member Functions

- ConfigSim ()

  Base constructor.

- ConfigSim (char ∗NameReadFile_n)

  Constructor setting configuration file.

- ∼ConfigSim ()

  Destructor.

- void DefaultConfig (char ∗NameConfigFile_n)

  It sets default simulation parameters.

- double gettStepPhy ()

  It returns physical time step, that is the value of tStepPhy attribute.

- double gettMax ()

  It returns maximal simulation duration, that is the value of tMax attribute.

- double gettStepMes ()

  It returns tStepMes attribute.

- double gettMemNoiseFirst ()

  It returns tMemNoiseFirst attribute.

- double gettMemNoiseLast ()

  It returns tMemNoiseLast attribute.

- double gettMemSig ()

  It returns tMemSig attribute.

- double gettDisplay ()

  It returns is the value of tDisplay attribute.

- double gettDeltaTDIDelay ()

  It returns tDeltaTDIDelay attribute.

- INTERP getTDIInterp ()

It returns TDIInterp attribute.

- double getTDIInterpUtilVal ()

  It returns TDIInterpUtilVal attribute.

- double getArmlength ()

  It returns the value of Armlength attribute.

- double getOrbStartTime ()

  It returns OrbStartTime attribute.

- double getOrbInitRot ()

  It returns OrbInitRot attribute.

- bool getTDIDelayApprox ()

  It returns TDIDelayApprox attribute.

- int getOrbMove ()

  It returns the value of OrbMove attribute.

- int getOrbOrder ()

  It returns the value of OrbOrder attribute.

- double getLaserPower ()

  It returns the laser power, that is the value of LaserPower attribute.

- bool getPhaMetFilter ()

  It returns 1 if a filter is applied in phasemeters, that is the value of PhaMetFilterON attribute.

- vector< double > getPhaMetFilterParam ()

  It returns PhaMetFilterParam attribute.

- GW * getGW ()

  It returns the first element of the vector GWs.

- vector< GW * > * getGWs ()

  It returns a pointer to the GWs attribute.

- char * getFileNameSig (int iSC)

  Returns FileNameSig corresponding to iSC input.

- char * getFileNameTDI ()

  It returns FileNameTDI attribute.

- char * getFileNameDelays ()

  It returns FileNameDelays attribute.

- char * getFileNamePositions ()

  It returns FileNamePositions attribute.

- vector< Noise * > getNoises ()

  It returns Noises attribute.

- Background * getGWB ()

  It returns GWB attribute.

- vector< USOClock > getUSOs ()

  It returns USOs attribute.

- int NbGenTDI ()

  It returns TDIsName attribute.

- char * getNameGenTDI (int iGen)

  Returns NameGenTDI corresponding to iGen input.

- vector< int > getGenTDIPacks (int iGen)

  Returns TDIsPack corresponding to iGen input.

- int getNbMaxDelays ()

  It returns NbMaxDelays attribute plus 1.

- void ReadFile ()

  Reads configuration file.

- void ReadASCIIFile ()

  Reads ASCII configuration file.

- void ReadXMLFile ()

  Reads XML configuration file.

- char * gXMLUnit (const char In[ ], double &Fact)

  Character to double conversion.

- double gXMLTime (ezxml_t param)

  Extracts time parameter from XML sctructure.

- double gXMLAngle (ezxml_t param)

  Extracts angle parameter from XML sctructure.

- double gXMLFrequency (ezxml_t param)

  Extracts frequency parameter from XML sctructure.

- double gXMLAstroMass (ezxml_t param)

  Extracts AstroMass parameter from XML sctructure.

- double gXMLAstroDistance (ezxml_t param)

  Extracts AstroDistance parameter from XML sctructure.

- char * gXMLTimeSeries (ezxml_t series, const char *type, const char *encoding, int &length, int &record)

Extracts TimeSeries parameters from XML sctructure.

- void NoisePlace (NoiseSpec tmp_noise, int iSC, int IndDir, int InstrumentIndex)

  Adds a noise specification structure.

- void NoisesCreation ()

  Creation of noises.

- double tMaxDelay ()

  Computes maximal time travel for one delay.

- double tMinDelay ()

  Computes minimal time travel for one delay.

- double tMemNecInterpTDI ()

  Computes memory time during which data must be saved for apply TDI interpolation.

- bool getNoNoise ()

  Checks if there are noises or not.

Private Attributes

- char ∗ ConfigFileName

  File name of simulation parameters. It is called configuration file.

- double tStepPhy

  Physical time step.

- double tMax

  It is the maximal simulation duration time.

- double tStepMes

  It is the time step between two phasemeter measures.

- double tMemNoiseFirst

  time shift between noise computation time and current time.

- double tMemNoiseLast

  time shift between time of last computed noise and current time.

- double tMemSig

  Duration of satellite memory for signal storage.

- double tDisplay

  Time step for screen display.

- double tDeltaTDIDelay

  Inaccuracy on wave time propagation with a length of a LISA arm.

- INTERP TDIInterp

  TDI interpolator type.

- double TDIInterpUtilVal

  Value used for TDI interpolation.

- double Armlength

  Nominal LISA arm length.

- double OrbStartTime

  Orbits start time.

- double OrbInitRot

  Angle (radians) giving rotation of the satellites triangle (from the basical position) at initial time (t=0).

- int OrbOrder

  Order for time propagation computing.

- int OrbMove

  Satellites motion.

- bool TDIDelayApprox

  Approximated TDI delays computation or not.

- double LaserPower

  Laser Power in Watts.

- bool PhaMetFilterON

  Phasemeter Filter (On, Off).

- vector< double > PhaMetFilterParam

  Phasemeter Filter Parameters.

- vector< GW ∗ > GWs

  Gravitational Waves Parameters.

- Background ∗ GWB

  Background signals

- vector< vector< NoiseSpec > > NoisesData

  noise specifications

- vector< Noise ∗ > Noises

  satellites noise

- vector< USOClock > USOs

  satellites USO time

- char FileNameSigSC1 [256]

  FileName for Space Craft 1 Signal.

- char FileNameSigSC2 [256]

  FileName for Space Craft 2 Signal.

- char FileNameSigSC3 [256]

  FileName for Space Craft 3 Signal.

- char FileNameTDI [256]

  FileName for TDI generators.

- char FileNameDelays [256]

  FileName for Delays.

- char FileNamePositions [256]

  FileName for Positions.

- vector< string > TDIsName

  Name of TDI.

- vector< vector< int > > TDIsPacks

  Vector of TDI data.

- int NbMaxDelays

  Maximum Delays Number.

10.3.2 Constructor & Destructor Documentation

10.3.2.1 ConfigSim::ConfigSim ()

Base constructor.

It sets the default configuration file from which all execution (simulation) parameters are set. Base paremeters are set by calling DefaultConfig, the others are read from the file by ReadFile.

Definition at line 24 of file LISACODE-ConfigSim.cpp.

References DefaultConfig(), and ReadFile().

10.3.2.2 ConfigSim::ConfigSim (char * *NameConfigFile_n*)

Constructor setting configuration file.

It takes the configuration file given in input to set all execution (simulation) parameters. The behaviour is Base paremeters are set by calling DefaultConfig, the others are read from the file by ReadFile. /param NameConfigFile_n: configuration file name.

Definition at line 40 of file LISACODE-ConfigSim.cpp.

References DefaultConfig(), and ReadFile().

10.3.2.3 ConfigSim::∼ConfigSim ()

Destructor.

It does not do any particular action.

Definition at line 57 of file LISACODE-ConfigSim.cpp.

10.3.3 Member Function Documentation

10.3.3.1 void ConfigSim::DefaultConfig (char ∗ *NameConfigFile_n*)

It sets default simulation parameters.

The default parameters are:

- DefVectNoise = 0
- tStepPhy = 0.5
- tMax = 65736.0
- tStepMes = 1.0
- tMemNoiseFirst = 5.0
- tMemNoiseLast = -30.0
- tMemSig = 100.0
- tDisplay = 1000.0
- tDeltaTDIDelay = 0.0
- TDIInterp = LAG
- TDIInterpUtilVal = 20
- Armlength = L0_m_default
- OrbStartTime = 0.0
- OrbInitRot = 0.0
- OrbMove = 1
- OrbOrder = 2
- TDIDelayApprox = false
- LaserPower = LaserPower_W_default
- PhaMetFilterON = true
- Phasemeter Filter Parameters :
- attenuation : PhaMetFilterParam[0]= 180.0 [dB]
- oscillations in bandwidth : PhaMetFilterParam[1]= 0.1 [dB]
- low transition frequency in factor of frequency of measurment : PhaMetFilterParam[2]= 0.1

- high transition frequency in factor of frequency of measurment : PhaMetFilterParam[3]= 0.3
- filename for Space Craft 1 Signal : FileNameSigSC1= "None"
- filename for Space Craft 2 Signal : FileNameSigSC2= "None"
- filename for Space Craft 3 Signal : =FileNameSigSC3 "None"
- filename for delays : FileNameDelays= "None"
- fileNameP for positions : FileNamePositions= "None"
- filename for TDI generators : FileNameTDI= "Def_SignalTDI.txt"
- GWB = NULL
- NoisesData = 24 NULL vectors
- USOs = 3 USOClock instances set to 0.0
- NbMaxDelays = 0
- ConfigFileName = NameConfigFile_n input

Definition at line 104 of file LISACODE-ConfigSim.cpp.

References Armlength, ConfigFileName, FileNameDelays, FileNamePositions, FileNameSigSC1, FileNameSigSC2, FileNameSigSC3, FileNameTDI, GWB, L0_m_default, LAG, LaserPower, LaserPower_W_default, NbMaxDelays, Noises, NoisesData, OrbInitRot, OrbMove, OrbOrder, OrbStartTime, PhaMetFilterON, PhaMetFilterParam, tDeltaTDIDelay, TDIDelayApprox, TDIInterp, TDIInterpUtilVal, tDisplay, tMax, tMemNoiseFirst, tMemNoiseLast, tMemSig, tStepMes, tStepPhy, and USOs.

Referenced by ConfigSim().

10.3.3.2 double ConfigSim::getArmlength () [inline]

It returns the value of Armlength attribute.

Definition at line 270 of file LISACODE-ConfigSim.h.

References Armlength.

Referenced by LISA::LISA(), and main().

10.3.3.3 char* ConfigSim::getFileNameDelays () [inline]

It returns FileNameDelays attribute.

Definition at line 297 of file LISACODE-ConfigSim.h.

References FileNameDelays.

Referenced by main().

10.3.3.4 char* ConfigSim::getFileNamePositions () [inline]

It returns FileNamePositions attribute.

Definition at line 299 of file LISACODE-ConfigSim.h.

References FileNamePositions.

Referenced by main().

10.3.3.5 char * ConfigSim::getFileNameSig (int *iSC*)

Returns FileNameSig corresponding to iSC input.

iSC : spacecraft number (expected values : 1, 2, 3)

Definition at line 152 of file LISACODE-ConfigSim.cpp.

References FileNameSigSC1, FileNameSigSC2, and FileNameSigSC3.

Referenced by main().

10.3.3.6 char* ConfigSim::getFileNameTDI () `[inline]`

It returns FileNameTDI attribute.

Definition at line 295 of file LISACODE-ConfigSim.h.

References FileNameTDI.

Referenced by main().

10.3.3.7 vector< int > ConfigSim::getGenTDIPacks (int *iGen*)

Returns TDIsPack corresponding to iGen input.

iGen : TDI generator index (expected values : [0, size of TDIsPacks[])

Definition at line 199 of file LISACODE-ConfigSim.cpp.

References TDIsPacks.

Referenced by main().

10.3.3.8 GW* ConfigSim::getGW () `[inline]`

It returns the first element of the vector GWs.

Definition at line 289 of file LISACODE-ConfigSim.h.

References GWs.

10.3.3.9 Background* ConfigSim::getGWB () `[inline]`

It returns GWB attribute.

Definition at line 303 of file LISACODE-ConfigSim.h.

References GWB.

Referenced by LISA::LISA().

10.3.3.10 vector<GW *>* ConfigSim::getGWs () `[inline]`

It returns a pointer to the GWs attribute.

Definition at line 291 of file LISACODE-ConfigSim.h.

References GWs.

Referenced by LISA::LISA(), and main().

10.3.3.11 double ConfigSim::getLaserPower () [inline]

It returns the laser power, that is the value of LaserPower attribute.

Definition at line 282 of file LISACODE-ConfigSim.h.

References LaserPower.

10.3.3.12 char ∗ ConfigSim::getNameGenTDI (int *iGen*)

Returns NameGenTDI corresponding to iGen input.

iGen : TDI generator index (expected values : [0, size of TDIsPacks[])

Definition at line 178 of file LISACODE-ConfigSim.cpp.

References TDIsName, and TDIsPacks.

Referenced by main().

10.3.3.13 int ConfigSim::getNbMaxDelays () [inline]

It returns NbMaxDelays attribute plus 1.

Definition at line 313 of file LISACODE-ConfigSim.h.

References NbMaxDelays.

Referenced by main().

10.3.3.14 vector<Noise ∗> ConfigSim::getNoises () [inline]

It returns Noises attribute.

Definition at line 301 of file LISACODE-ConfigSim.h.

References Noises.

Referenced by LISA::LISA().

10.3.3.15 bool ConfigSim::getNoNoise ()

Checks if there are noises or not.

Returns FALSE if all noises in Noises are NULL, else TRUE.

Definition at line 2765 of file LISACODE-ConfigSim.cpp.

References Noises.

Referenced by LISA::LISA(), and main().

10.3.3.16 double ConfigSim::getOrbInitRot () [inline]

It returns OrbInitRot attribute.

Definition at line 274 of file LISACODE-ConfigSim.h.

References OrbInitRot.

Referenced by LISA::LISA(), and main().

10.3.3.17 int ConfigSim::getOrbMove () [inline]

It returns the value of OrbMove attribute.

Definition at line 278 of file LISACODE-ConfigSim.h.

References OrbMove.

Referenced by LISA::LISA(), and main().

10.3.3.18 int ConfigSim::getOrbOrder () [inline]

It returns the value of OrbOrder attribute.

Definition at line 280 of file LISACODE-ConfigSim.h.

References OrbOrder.

Referenced by LISA::LISA(), and main().

10.3.3.19 double ConfigSim::getOrbStartTime () [inline]

It returns OrbStartTime attribute.

Definition at line 272 of file LISACODE-ConfigSim.h.

References OrbStartTime.

Referenced by LISA::LISA(), and main().

10.3.3.20 bool ConfigSim::getPhaMetFilter () [inline]

It returns 1 if a filter is applied in phasemeters, that is the value of PhaMetFilterON attribute.

Definition at line 285 of file LISACODE-ConfigSim.h.

References PhaMetFilterON.

Referenced by LISA::LISA().

10.3.3.21 vector<double> ConfigSim::getPhaMetFilterParam () [inline]

It returns PhaMetFilterParam attribute.

Definition at line 287 of file LISACODE-ConfigSim.h.

References PhaMetFilterParam.

Referenced by LISA::LISA().

10.3.3.22 double ConfigSim::gettDeltaTDIDelay () [inline]

It returns tDeltaTDIDelay attribute.

Definition at line 264 of file LISACODE-ConfigSim.h.

References tDeltaTDIDelay.

Referenced by main().

10.3.3.23 bool ConfigSim::getTDIDelayApprox () [inline]

It returns TDIDelayApprox attribute.

Definition at line 276 of file LISACODE-ConfigSim.h.

References TDIDelayApprox.

Referenced by main().

10.3.3.24 INTERP ConfigSim::getTDIInterp () [inline]

It returns TDIInterp attribute.

Definition at line 266 of file LISACODE-ConfigSim.h.

References INTERP, and TDIInterp.

Referenced by main().

10.3.3.25 double ConfigSim::getTDIInterpUtilVal () [inline]

It returns TDIInterpUtilVal attribute.

Definition at line 268 of file LISACODE-ConfigSim.h.

References TDIInterpUtilVal.

Referenced by main().

10.3.3.26 double ConfigSim::gettDisplay () [inline]

It returns is the value of tDisplay attribute.

Definition at line 262 of file LISACODE-ConfigSim.h.

References tDisplay.

Referenced by main().

10.3.3.27 double ConfigSim::gettMax () [inline]

It returns maximal simulation duration, that is the value of tMax attribute.

Definition at line 252 of file LISACODE-ConfigSim.h.

References tMax.

Referenced by main().

10.3.3.28 double ConfigSim::gettMemNoiseFirst () [inline]

It returns tMemNoiseFirst attribute.

Definition at line 256 of file LISACODE-ConfigSim.h.

References tMemNoiseFirst.

10.3.3.29 double ConfigSim::gettMemNoiseLast () [inline]

It returns tMemNoiseLast attribute.

Definition at line 258 of file LISACODE-ConfigSim.h.

References tMemNoiseLast.

10.3.3.30 double ConfigSim::gettMemSig () [inline]

It returns tMemSig attribute.

Definition at line 260 of file LISACODE-ConfigSim.h.

References tMemSig.

Referenced by LISA::LISA().

10.3.3.31 double ConfigSim::gettStepMes () [inline]

It returns tStepMes attribute.

Definition at line 254 of file LISACODE-ConfigSim.h.

References tStepMes.

Referenced by LISA::LISA(), and main().

10.3.3.32 double ConfigSim::gettStepPhy () [inline]

It returns physical time step, that is the value of tStepPhy attribute.

Definition at line 250 of file LISACODE-ConfigSim.h.

References tStepPhy.

Referenced by LISA::LISA(), and main().

10.3.3.33 vector<USOClock> ConfigSim::getUSOs () [inline]

It returns USOs attribute.

Definition at line 305 of file LISACODE-ConfigSim.h.

References USOs.

Referenced by LISA::LISA().

10.3.3.34 double ConfigSim::gXMLAngle (ezxml_t *param*)

Extracts angle parameter from XML sctructure.

Checks if unit is degree.

Definition at line 2418 of file LISACODE-ConfigSim.cpp.

References MathUtils::deg2rad(), ezxml_attr(), ezxml_t, ezxml_txt, and gXMLUnit().

Referenced by ReadXMLFile().

10.3.3.35 double ConfigSim::gXMLAstroDistance (ezxml_t *param*)

Extracts AstroDistance parameter from XML sctructure.

Checks if unit is Parsec.

Definition at line 2472 of file LISACODE-ConfigSim.cpp.

References ezxml_attr(), ezxml_t, ezxml_txt, and gXMLUnit().

Referenced by ReadXMLFile().

10.3.3.36 double ConfigSim::gXMLAstroMass (ezxml_t *param*)

Extracts AstroMass parameter from XML sctructure.

Checks if unit is SolarMass.

Definition at line 2456 of file LISACODE-ConfigSim.cpp.

References ezxml_attr(), ezxml_t, ezxml_txt, and gXMLUnit().

Referenced by ReadXMLFile().

10.3.3.37 double ConfigSim::gXMLFrequency (ezxml_t *param*)

Extracts frequency parameter from XML sctructure.

Checks if unit is MilliHertz.

Definition at line 2437 of file LISACODE-ConfigSim.cpp.

References ezxml_attr(), ezxml_t, ezxml_txt, and gXMLUnit().

Referenced by ReadXMLFile().

10.3.3.38 double ConfigSim::gXMLTime (ezxml_t *param*)

Extracts time parameter from XML sctructure.

Checks if unit is second.

Definition at line 2401 of file LISACODE-ConfigSim.cpp.

References ezxml_attr(), ezxml_t, ezxml_txt, and gXMLUnit().

Referenced by ReadXMLFile().

10.3.3.39 char ∗ ConfigSim::gXMLTimeSeries (ezxml_t *series*, const char ∗ *typedata*, const char ∗ *encoding*, int & *length*, int & *records*)

Extracts TimeSeries parameters from XML sctructure.

Sets parameters :

- typedata and encoding (extracted from "Stream" child of series input)
- length and records (extracted from "Array" child of series input) Returns filenename (extracted from "Array" child of series input).

Definition at line 2494 of file LISACODE-ConfigSim.cpp.

References ezxml_attr(), ezxml_child(), ezxml_t, ezxml_txt, and ezxml::next.

Referenced by ReadXMLFile().

10.3.3.40 char * ConfigSim::gXMLUnit (const char *In*[ ], double & *Fact*)

Character to double conversion.

Fact double output is converted from In charcater pointer input. This function returns end of string character pointer.

Definition at line 2387 of file LISACODE-ConfigSim.cpp.

Referenced by gXMLAngle(), gXMLAstroDistance(), gXMLAstroMass(), gXMLFrequency(), g-XMLTime(), and ReadXMLFile().

10.3.3.41 int ConfigSim::NbGenTDI () `[inline]`

It returns TDIsName attribute.

Definition at line 307 of file LISACODE-ConfigSim.h.

References TDIsName.

Referenced by main().

10.3.3.42 void ConfigSim::NoisePlace (NoiseSpec *tmp_noise*, int *iSC*, int *IndDir*, int *InstrumentIndex*)

Adds a noise specification structure.

A noise specification structure NoiseSpec containing tm_noise input information is pushed. The place where it is pushed is defined by inputs:

- iSC : spacecraft number
- IndDir : directional index
- InstrumentIndex : instrument index

Definition at line 2550 of file LISACODE-ConfigSim.cpp.

References NoisesData.

Referenced by ReadXMLFile().

10.3.3.43 void ConfigSim::NoisesCreation ()

Creation of noises.

Number of created noises is given by size of NoisesData.

Definition at line 2578 of file LISACODE-ConfigSim.cpp.

References Armlength, L0_m_default, LaserPower, LaserPower_W_default, Noises, NoisesData, NoiseSpec::NStr, NoiseSpec::NType, NoiseSpec::NVal0, NoiseSpec::NVal1, NoiseSpec::NVal2, PRECISION, tMemNoiseFirst, tMemNoiseLast, tStepMes, and tStepPhy.

Referenced by ReadASCIIFile(), and ReadXMLFile().

10.3.3.44 void ConfigSim::ReadASCIIFile ()

Reads ASCII configuration file.

Data are read :

- temporal
- interpolation
- accuracy
- orbits
- detector
- gravitational waves background
- record
- gravitational waves
- noises
- USO Clocks
- Phasemeter
- TDI generators

Definition at line 249 of file LISACODE-ConfigSim.cpp.

References Armlength, ConfigFileName, MathUtils::deg2rad(), FileNameDelays, FileNamePositions, FileNameSigSC1, FileNameSigSC2, FileNameSigSC3, FileNameTDI, GWB, GWs, LAG, LaserPower, NbMaxDelays, NoisesCreation(), NoisesData, NoiseSpec::NStr, NoiseSpec::NType, NoiseSpec::NVal0, NoiseSpec::NVal1, NoiseSpec::NVal2, OrbInitRot, OrbMove, OrbOrder, OrbStartTime, PhaMetFilterON, PhaMetFilterParam, tDeltaTDIDelay, TDIDelayApprox, TDIInterp, TDIInterpUtilVal, TDIsName, TDIsPacks, tDisplay, tMax, tMaxDelay(), tMemNoiseFirst, tMemNoiseLast, tMemSig, tStepMes, tStepPhy, and USOs.

Referenced by ReadFile().

10.3.3.45 void ConfigSim::ReadFile ()

Reads configuration file.

Calls ReadXMLFile or ReadASCIIFile depending on ConfigFileName type

Definition at line 214 of file LISACODE-ConfigSim.cpp.

References ConfigFileName, ReadASCIIFile(), and ReadXMLFile().

Referenced by ConfigSim().

10.3.3.46 void ConfigSim::ReadXMLFile ()

Reads XML configuration file.

Data are read :

- Time Data
- Interpolation Data
- Precision TDI Data
- Orbits Data
- Detector Data
- USO Data
- Record Data

Definition at line 1372 of file LISACODE-ConfigSim.cpp.

References Armlength, ConfigFileName, ezxml_attr(), ezxml_child(), ezxml_free(), ezxml_parse_-file(), ezxml_t, ezxml_txt, FileNameDelays, FileNamePositions, FileNameSigSC1, FileNameSigSC2, FileNameSigSC3, FileNameTDI, GWs, gXMLAngle(), gXMLAstroDistance(), gXMLAstroMass(), g-XMLFrequency(), gXMLTime(), gXMLTimeSeries(), gXMLUnit(), LAG, LaserPower, NbMaxDelays, ezxml::next, NoisePlace(), NoisesCreation(), NoiseSpec::NStr, NoiseSpec::NType, NoiseSpec::NVal0, NoiseSpec::NVal1, NoiseSpec::NVal2, OrbInitRot, OrbMove, OrbOrder, OrbStartTime, PhaMetFilterON, PhaMetFilterParam, tDeltaTDIDelay, TDIDelayApprox, TDIInterp, TDIInterpUtilVal, TDIsName, TDIs-Packs, tDisplay, tMax, tMaxDelay(), tMemNoiseFirst, tMemNoiseLast, tStepMes, tStepPhy, ezxml::txt, and USOs.

Referenced by ReadFile().

10.3.3.47 double ConfigSim::tMaxDelay ()

Computes maximal time travel for one delay.

Maximal time travel for one delay is $tMaxDelay = tStepMes \cdot ceil(\frac{6 \cdot Armlength}{5 \cdot tStepMes \cdot C})$.

Definition at line 2731 of file LISACODE-ConfigSim.cpp.

References Armlength, c_SI, and tStepMes.

Referenced by main(), ReadASCIIFile(), and ReadXMLFile().

10.3.3.48 double ConfigSim::tMemNecInterpTDI ()

Computes memory time during which data must be saved for apply TDI interpolation.

Memory time during which data must be saved for apply TDI interpolation is :

$$tMemNecInterpTDI = \begin{cases} \left(2 + ceil(\frac{TDIInterpUtilVal}{2})\right) \cdot tStepMes & \text{if (TDIInterp = LAG)} \\ 2 \cdot tStepMes & else \end{cases}$$

Definition at line 2753 of file LISACODE-ConfigSim.cpp.

References LAG, TDIInterp, TDIInterpUtilVal, and tStepMes.

Referenced by main().

10.3.3.49 double ConfigSim::tMinDelay ()

Computes minimal time travel for one delay.

Minimal time travel for one delay is $tMinDelay = tStepMes \cdot ceil \frac{4 \cdot Armlength}{5 \cdot tStepMes \cdot C}$.

Definition at line 2740 of file LISACODE-ConfigSim.cpp.

References Armlength, c_SI, and tStepMes.

Referenced by main().

10.3.4 Member Data Documentation

10.3.4.1 double ConfigSim::Armlength [private]

Nominal LISA arm length.

Definition at line 132 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getArmlength(), NoisesCreation(), ReadASCIIFile(), ReadXMLFile(), t-MaxDelay(), and tMinDelay().

10.3.4.2 char* ConfigSim::ConfigFileName [private]

File name of simulation parameters. It is called configuration file.

Definition at line 88 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), ReadASCIIFile(), ReadFile(), and ReadXMLFile().

10.3.4.3 ConfigSim::FileNameDelays [private]

FileName for Delays.

6 time delays between satellites are stored in this file.

Referenced by DefaultConfig(), getFileNameDelays(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.4 ConfigSim::FileNamePositions [private]

FileName for Positions.

Referenced by DefaultConfig(), getFileNamePositions(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.5 ConfigSim::FileNameSigSC1 [private]

FileName for Space Craft 1 Signal.

4 phasemeters data are stored in this file.

Referenced by DefaultConfig(), getFileNameSig(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.6 ConfigSim::FileNameSigSC2 [private]

FileName for Space Craft 2 Signal.

4 phasemeters data are stored in this file.

Referenced by DefaultConfig(), getFileNameSig(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.7 ConfigSim::FileNameSigSC3 [private]

FileName for Space Craft 3 Signal.

4 phasemeters data are stored in this file.

Referenced by DefaultConfig(), getFileNameSig(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.8 ConfigSim::FileNameTDI [private]

FileName for TDI generators.

Referenced by DefaultConfig(), getFileNameTDI(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.9 Background* ConfigSim::GWB [private]

Background signals

Definition at line 196 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getGWB(), and ReadASCIIFile().

10.3.4.10 vector< GW *> ConfigSim::GWs [private]

Gravitational Waves Parameters.

Definition at line 194 of file LISACODE-ConfigSim.h.

Referenced by getGW(), getGWs(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.11 double ConfigSim::LaserPower [private]

Laser Power in Watts.

It specifies the beam power at the laser output.

Definition at line 177 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getLaserPower(), NoisesCreation(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.12 int ConfigSim::NbMaxDelays [private]

Maximum Delays Number.

Definition at line 235 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getNbMaxDelays(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.13 vector<Noise *> ConfigSim::Noises [private]

satellites noise

Definition at line 200 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getNoises(), getNoNoise(), and NoisesCreation().

10.3.4.14 vector< vector <NoiseSpec> > ConfigSim::NoisesData [private]

noise specifications

Definition at line 198 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), NoisePlace(), NoisesCreation(), and ReadASCIIFile().

10.3.4.15 double ConfigSim::OrbInitRot [private]

Angle (radians) giving rotation of the satellites triangle (from the basical position) at initial time (t=0).

Basical position (OrbInitRot=0) is a triangle pointing to the bottom with satellite 1 on the bottom, satellite 2 on negative Y axis (on the right hand side seeing on X axis) and satelite 3 on Y positive axis (on the left hand side seeing on X axis).

OrbInitRot not null gives rotation angle to obtain the new starting position of the satellites.

Definition at line 153 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getOrbInitRot(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.16 int ConfigSim::OrbMove [private]

Satellites motion.

It indicates if satellites are moving (On) or fixed (Off).

Definition at line 170 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getOrbMove(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.17 int ConfigSim::OrbOrder [private]

Order for time propagation computing.

This parameter specifies the order to compute the round-trip time of photons between two satellites. The possible order values are :

- 0 : classical computing
- 1 : time propagation is computed using special relativity.
- 2 : time propagation is computed using general relativity. If the satellites are fixed, only $Order = 0$ is valid.

Definition at line 164 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getOrbOrder(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.18 double ConfigSim::OrbStartTime [private]

Orbits start time.

This parameter allows to start simulation with a satellites position different to base configuration. In the base configuration satellite 1 is on x axis under the ecliptic plan. Satellites 2 and 3 are over the ecliptic plan, 2 in $y < 0$ and 3 in $y > 0$.

Definition at line 141 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getOrbStartTime(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.19 bool ConfigSim::PhaMetFilterON [private]

Phasemeter Filter (On, Off).

It specifies if a low pass filter is applied to the phasemeters signals.

If variable is set to 1 the signals are filtered.

Definition at line 183 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getPhaMetFilter(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.20 vector<double> ConfigSim::PhaMetFilterParam [private]

Phasemeter Filter Parameters.

Phasemeter Filter Parameters :

- attenuation [dB]
- oscillations in bandwidth [dB]
- low transition frequency in factor of frequency of measurment
- high transition frequency in factor of frequency of measurment

Definition at line 192 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getPhaMetFilterParam(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.21 ConfigSim::tDeltaTDIDelay [private]

Inaccuracy on wave time propagation with a length of a LISA arm.

It is an error added to the exact time propagation before to its use by TDI.

Referenced by DefaultConfig(), gettDeltaTDIDelay(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.22 bool ConfigSim::TDIDelayApprox [private]

Approximated TDI delays computation or not.

Definition at line 172 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getTDIDelayApprox(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.23 INTERP ConfigSim::TDIInterp [private]

TDI interpolator type.

Definition at line 127 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getTDIInterp(), ReadASCIIFile(), ReadXMLFile(), and tMemNecInterpTDI().

10.3.4.24 double ConfigSim::TDIInterpUtilVal [private]

Value used for TDI interpolation.

Definition at line 129 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getTDIInterpUtilVal(), ReadASCIIFile(), ReadXMLFile(), and tMemNecInterpTDI().

10.3.4.25 vector<string> ConfigSim::TDIsName [private]

Name of TDI.

Definition at line 231 of file LISACODE-ConfigSim.h.

Referenced by getNameGenTDI(), NbGenTDI(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.26 vector< vector<int> > ConfigSim::TDIsPacks [private]

Vector of TDI data.

Definition at line 233 of file LISACODE-ConfigSim.h.

Referenced by getGenTDIPacks(), getNameGenTDI(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.27 ConfigSim::tDisplay [private]

Time step for screen display.

Referenced by DefaultConfig(), gettDisplay(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.28 ConfigSim::tMax [private]

It is the maximal simulation duration time.

Referenced by DefaultConfig(), gettMax(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.29 ConfigSim::tMemNoiseFirst [private]

time shift between noise computation time and current time.

Noises are computed tMemNoiseFirst second(s) before current time.

Referenced by DefaultConfig(), gettMemNoiseFirst(), NoisesCreation(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.30 ConfigSim::tMemNoiseLast [private]

time shift between time of last computed noise and current time.

later noises are eleminated

Referenced by DefaultConfig(), gettMemNoiseLast(), NoisesCreation(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.31 ConfigSim::tMemSig [private]

Duration of satellite memory for signal storage.

Referenced by DefaultConfig(), gettMemSig(), and ReadASCIIFile().

10.3.4.32 ConfigSim::tStepMes [private]

It is the time step between two phasemeter measures.

It corresponds to the time step between two output data samples.

Referenced by DefaultConfig(), gettStepMes(), NoisesCreation(), ReadASCIIFile(), ReadXMLFile(), tMaxDelay(), tMemNecInterpTDI(), and tMinDelay().

10.3.4.33 ConfigSim::tStepPhy [private]

Physical time step.

It is the smaller time step of the simulation. It is used to simulate continuous phenomena.

Referenced by DefaultConfig(), gettStepPhy(), NoisesCreation(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.34 vector<USOClock> ConfigSim::USOs [private]

satellites USO time

Definition at line 202 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getUSOs(), ReadASCIIFile(), and ReadXMLFile().

The documentation for this class was generated from the following files:

- LISACODE-ConfigSim.h
- LISACODE-ConfigSim.cpp
- LISACODE-ConfigSim_s.cpp

10.4 Couple Class Reference

`#include <LISACODE-Couple.h>`

10.4.1 Detailed Description

Couple management class.

Definition at line 31 of file LISACODE-Couple.h.

Public Member Functions

- Couple ()

  Constructs an instance and initializes it with default values: (0,0).

- Couple (double, double)

  Constructs an instance and initializes it with inputs.

- ∼Couple ()

  Destructor.

Public Attributes

- double x

  First component.

- double y

  Second component.

Friends

- Couple operator+ (Couple, Couple)

  2 couples addiction.

- Couple operator- (Couple, Couple)

  2 couples subtraction.

- Couple operator ∗ (Couple, Couple)

  ?? where operator∗ (Couple,Couple) is defined?

- Couple operator ∗ (double, Couple)

  Product of a couple by a scalar.

- Couple operator ∗ (Couple, double)

  Product of a couple by a scalar.

- Couple operator/ (Couple, double)

Division of a couple by a scalar .

10.4.2 Constructor & Destructor Documentation

10.4.2.1 Couple::Couple ()

Constructs an instance and initializes it with default values: (0,0).

Definition at line 18 of file LISACODE-Couple.cpp.

References x, and y.

10.4.2.2 Couple::Couple (double *xvalue*, double *yvalue*)

Constructs an instance and initializes it with inputs.

- x = xvalue input
- y = yvalue input

Definition at line 29 of file LISACODE-Couple.cpp.

References x, and y.

10.4.2.3 Couple::∼Couple ()

Destructor.

Definition at line 37 of file LISACODE-Couple.cpp.

10.4.3 Friends And Related Function Documentation

10.4.3.1 Couple operator ∗ (Couple *z1*, double *a*) `[friend]`

Product of a couple by a scalar.

Definition at line 87 of file LISACODE-Couple.cpp.

10.4.3.2 Couple operator ∗ (double *a*, Couple *z1*) `[friend]`

Product of a couple by a scalar.

Definition at line 78 of file LISACODE-Couple.cpp.

10.4.3.3 Couple operator ∗ (Couple *z1*, Couple *z2*) `[friend]`

?? where operator∗ (Couple,Couple) is defined?

Definition at line 65 of file LISACODE-Couple.cpp.

10.4.3.4 Couple operator+ (Couple *z1*, Couple *z2*) `[friend]`

2 couples addiction.

Definition at line 47 of file LISACODE-Couple.cpp.

10.4.3.5 Couple operator- (Couple *z1*, Couple *z2*) `[friend]`

2 couples subtraction.

Definition at line 56 of file LISACODE-Couple.cpp.

10.4.3.6 Couple operator/ (Couple *z1*, double *a*) `[friend]`

Division of a couple by a scalar .

Definition at line 95 of file LISACODE-Couple.cpp.

10.4.4 Member Data Documentation

10.4.4.1 Couple::x

First component.

Referenced by Couple(), GWNewton2::hbin(), GWBinary::hbin(), operator *(), operator+(), operator-(), operator/(), Geometry::position(), GWFile::ReadFile(), and Geometry::velocity().

10.4.4.2 Couple::y

Second component.

Referenced by Couple(), GWNewton2::hbin(), GWBinary::hbin(), operator *(), operator+(), operator-(), operator/(), Geometry::position(), GWFile::ReadFile(), and Geometry::velocity().

The documentation for this class was generated from the following files:

- LISACODE-Couple.h
- LISACODE-Couple.cpp

10.5 ezxml Struct Reference

```
#include <ezxml.h>
```

Public Attributes

- char ∗ name

  Tag name.

- char ∗∗ attr

  Tag attributes { name, value, name, value, ... NULL }.

- char ∗ txt

  Tag character content, empty string if none.

- size_t off

  Tag offset from start of parent tag character content.

- ezxml_t next

  Next tag with same name in this section at this depth.

- ezxml_t sibling

  Next tag with different name in same section and depth.

- ezxml_t ordered

  next tag, same section and depth, in original order.

- ezxml_t child

  Head of sub tag list, NULL if none.

- ezxml_t parent

  Parent tag, NULL if current tag is root tag.

- short flags

  Additional information.

10.5.1 Member Data Documentation

10.5.1.1 char∗∗ ezxml::attr

Tag attributes { name, value, name, value, ... NULL }.

Definition at line 57 of file ezxml.h.

Referenced by ezxml_add_child(), ezxml_attr(), ezxml_free(), ezxml_new(), ezxml_open_tag(), ezxml_set_attr(), and ezxml_toxml_r().

10.5.1.2 ezxml_t ezxml::child

Head of sub tag list, NULL if none.

Definition at line 69 of file ezxml.h.

Referenced by ezxml_add_child(), ezxml_child(), ezxml_free(), ezxml_remove(), and ezxml_toxml_r().

10.5.1.3 ezxml::flags

Additional information.

Referenced by ezxml_char_content(), ezxml_free(), ezxml_set_attr(), ezxml_set_flag(), and ezxml_set_-txt().

10.5.1.4 char* ezxml::name

Tag name.

Definition at line 55 of file ezxml.h.

Referenced by ezxml_add_child(), ezxml_attr(), ezxml_char_content(), ezxml_child(), ezxml_close_tag(), ezxml_free(), ezxml_new(), ezxml_open_tag(), ezxml_parse_str(), ezxml_proc_inst(), ezxml_remove(), ezxml_toxml(), and ezxml_toxml_r().

10.5.1.5 ezxml_t ezxml::next

Next tag with same name in this section at this depth.

Definition at line 63 of file ezxml.h.

Referenced by ezxml_add_child(), ezxml_idx(), ezxml_remove(), ConfigSim::gXMLTimeSeries(), and ConfigSim::ReadXMLFile().

10.5.1.6 size_t ezxml::off

Tag offset from start of parent tag character content.

Definition at line 61 of file ezxml.h.

Referenced by ezxml_add_child(), and ezxml_toxml_r().

10.5.1.7 ezxml_t ezxml::ordered

next tag, same section and depth, in original order.

Definition at line 67 of file ezxml.h.

Referenced by ezxml_add_child(), ezxml_free(), ezxml_remove(), ezxml_toxml(), and ezxml_toxml_r().

10.5.1.8 ezxml_t ezxml::parent

Parent tag, NULL if current tag is root tag.

Definition at line 71 of file ezxml.h.

Referenced by ezxml_add_child(), ezxml_attr(), ezxml_close_tag(), ezxml_error(), ezxml_free(), ezxml_-pi(), ezxml_remove(), ezxml_toxml(), and ezxml_toxml_r().

10.5.1.9 ezxml_t ezxml::sibling

Next tag with different name in same section and depth.

Definition at line 65 of file ezxml.h.

Referenced by ezxml_add_child(), ezxml_child(), and ezxml_remove().

10.5.1.10 char* ezxml::txt

Tag character content, empty string if none.

Definition at line 59 of file ezxml.h.

Referenced by ezxml_add_child(), ezxml_char_content(), ezxml_free(), ezxml_new(), ezxml_open_tag(), ezxml_set_txt(), ezxml_toxml_r(), and ConfigSim::ReadXMLFile().

The documentation for this struct was generated from the following file:

- ezxml.h

10.6 ezxml_root Struct Reference

Public Attributes

- ezxml xml
- ezxml_t cur
- char ∗ m
- size_t len
- char ∗ u
- char ∗ s
- char ∗ e
- char ∗∗ ent
- char ∗∗∗ attr
- char ∗∗∗ pi
- short standalone
- char err [EZXML_ERRL]

10.6.1 Member Data Documentation

10.6.1.1 char∗∗∗ ezxml_root::attr

Definition at line 51 of file ezxml.c.

Referenced by ezxml_attr(), ezxml_free(), ezxml_internal_dtd(), ezxml_new(), ezxml_parse_str(), and ezxml_toxml().

10.6.1.2 ezxml_t ezxml_root::cur

Definition at line 44 of file ezxml.c.

Referenced by ezxml_char_content(), ezxml_close_tag(), ezxml_new(), ezxml_open_tag(), and ezxml_-parse_str().

10.6.1.3 char∗ ezxml_root::e

Definition at line 49 of file ezxml.c.

Referenced by ezxml_free(), and ezxml_parse_str().

10.6.1.4 char∗∗ ezxml_root::ent

Definition at line 50 of file ezxml.c.

Referenced by ezxml_char_content(), ezxml_free(), ezxml_internal_dtd(), ezxml_new(), and ezxml_-parse_str().

10.6.1.5 char ezxml_root::err[EZXML_ERRL]

Definition at line 54 of file ezxml.c.

Referenced by ezxml_err(), ezxml_internal_dtd(), and ezxml_new().

10.6.1.6 size_t ezxml_root::len

Definition at line 46 of file ezxml.c.

Referenced by ezxml_free(), ezxml_parse_fd(), and ezxml_parse_fp().

10.6.1.7 char∗ ezxml_root::m

Definition at line 45 of file ezxml.c.

Referenced by ezxml_free(), and ezxml_parse_str().

10.6.1.8 char∗∗∗ ezxml_root::pi

Definition at line 52 of file ezxml.c.

Referenced by ezxml_free(), ezxml_new(), ezxml_pi(), ezxml_proc_inst(), and ezxml_toxml().

10.6.1.9 char∗ ezxml_root::s

Definition at line 48 of file ezxml.c.

Referenced by ezxml_err(), ezxml_free(), and ezxml_parse_str().

10.6.1.10 short ezxml_root::standalone

Definition at line 53 of file ezxml.c.

Referenced by ezxml_internal_dtd(), and ezxml_proc_inst().

10.6.1.11 char∗ ezxml_root::u

Definition at line 47 of file ezxml.c.

Referenced by ezxml_free(), and ezxml_parse_str().

10.6.1.12 struct ezxml ezxml_root::xml

Definition at line 43 of file ezxml.c.

Referenced by ezxml_attr(), ezxml_err(), ezxml_new(), ezxml_parse_fd(), ezxml_parse_fp(), ezxml_-parse_str(), ezxml_pi(), ezxml_proc_inst(), and ezxml_toxml().

The documentation for this struct was generated from the following file:

- ezxml.c

10.7 Filter Class Reference

`#include <LISACODE-Filter.h>`

10.7.1 Detailed Description

filter management class.

Definition at line 40 of file LISACODE-Filter.h.

Public Member Functions

- Filter ()

  Constructs an instance and initializes it with default values.

- Filter (vector< vector< double > > alpha_n, vector< vector< double > > beta_n)

  Constructs an instance and initializes it with default values and inputs.

- Filter (vector< vector< double > > alpha_n, vector< vector< double > > beta_n, int NbDataStabilization_n)

  Constructs an instance and initializes it with inputs.

- Filter (double fe, double at, double bp, double fb, double fa)

  Constructs an instance and initializes it with inputs.

- ~Filter ()

  Destructor.

- void init (vector< vector< double > > alpha_n, vector< vector< double > > beta_n, int NbDataStabilization_n)

  Initializes an instance with default values and inputs.

- int getDepth ()

  Gives maximum of alpha or beta attribute size.

- int getNbDataStab ()

  Returns NbDataStab attribute.

- vector< vector< double > > getAlpha ()

  Returns alpha attribute.

- vector< vector< double > > getBeta ()

  Returns beta attribute.

- void App (int StartBin, const vector< double > &RawData, vector< double > &FilterData)

  Appends data from RawData input (starting at StartBin index) to TmpData attribute and ti FilterData output.

Protected Attributes

- vector< vector< double > > alpha

  Alpha parameters list.

- vector< vector< double > > beta
- int NbDataStab

  Number of data for stabilization.

- vector< vector< double > > TmpData

  Temporary data.

10.7.2 Constructor & Destructor Documentation

10.7.2.1 Filter::Filter ()

Constructs an instance and initializes it with default values.

- alpha = empty
- beta = 1 element with value = 1
- TmpData = 2 elements

Definition at line 29 of file LISACODE-Filter.cpp.

References alpha, beta, NbDataStab, and TmpData.

10.7.2.2 Filter::Filter (vector< vector< double > > *alpha_n*, vector< vector< double > > *beta_n*)

Constructs an instance and initializes it with default values and inputs.

init method is called to set attributes :

- alpha = alpha_n input
- beta = beta_n input
- NbDataStab = 0
- TmpData = alpha_n size + 1 null vectors

Definition at line 48 of file LISACODE-Filter.cpp.

References init().

10.7.2.3 Filter::Filter (vector< vector< double > > *alpha_n*, vector< vector< double > > *beta_n*, int *NbDataStabilization_n*)

Constructs an instance and initializes it with inputs.

init method is called to set attributes :

- alpha = alpha_n input
- beta = beta_n input
- NbDataStab = NbDataStab_n input
- TmpData = (alpha_n size + 1) null vectors

Definition at line 62 of file LISACODE-Filter.cpp.

References init().

10.7.2.4 Filter::Filter (double *fe*, double *at*, double *bp*, double *fb*, double *fa*)

Constructs an instance and initializes it with inputs.

Parameters:

fe sampling frequency

at attenuation

bp oscillations in bandwidth

fb low transition frequency

fa high transition frequency

CalcEllipticFilter4LISACode method is called with inputs and 30 as maximum number of cells.

Then, CalcEllipticFilter4LISACode outputs are used :

- CellsCoef Cells coefficients
- NCellsOut number of cells

init method is called to set attributes :

- alpha = 2 first coefficients opposite, for each cell (using CalcEllipticFilter4LISACode outputs)
- beta = 3 last coefficients, for each cell
- NbDataStab = $\frac{100}{fe}$
- TmpData = (alpha_n size + 1 null) vectors

Definition at line 89 of file LISACODE-Filter.cpp.

References CalcEllipticFilter4LISACode(), and init().

10.7.2.5 Filter::∼Filter ()

Destructor.

Definition at line 126 of file LISACODE-Filter.cpp.

10.7.3 Member Function Documentation

10.7.3.1 void Filter::App (int *StartBin*, const vector< double > & *RawData*, vector< double > & *FilterData*)

Appends data from RawData input (starting at StartBin index) to TmpData attribute and ti FilterData output.

RawData is first copied in TmpData, then filtering is applied.

For all TmpData elements (index :iFil=0,...,size(α)), a recursive computation is done :

for $i = 0, \ldots, StartBin$

$$TmpData[iFil+1][i] = \sum_{k=0}^{size(\alpha[iFil])} \alpha[iFil][k] \cdot TmpData[iFil+1][k+i+1]$$

$$+ \sum_{k=0}^{size(\beta[iFil])} \beta[iFil][k] \cdot TmpData[iFil][k+i]$$

Then last TmpData array is copied into FilterData.

Definition at line 191 of file LISACODE-Filter.cpp.

References alpha, beta, and TmpData.

Referenced by NoiseFilter::generNoise(), PhoDetPhaMet::IntegrateSignal(), and NoiseFilter::loadNoise().

10.7.3.2 vector< vector <double> > Filter::getAlpha () [inline]

Returns alpha attribute.

Definition at line 79 of file LISACODE-Filter.h.

References alpha.

Referenced by NoiseFilter::getFilterAlpha().

10.7.3.3 vector< vector <double> > Filter::getBeta () [inline]

Returns beta attribute.

Definition at line 81 of file LISACODE-Filter.h.

References beta.

Referenced by NoiseFilter::getFilterBeta().

10.7.3.4 int Filter::getDepth ()

Gives maximum of alpha or beta attribute size.

Definition at line 165 of file LISACODE-Filter.cpp.

References alpha, beta, and MAX.

Referenced by PhoDetPhaMet::init(), and NoiseFilter::loadNoise().

10.7.3.5 int Filter::getNbDataStab ()

Returns NbDataStab attribute.

Definition at line 176 of file LISACODE-Filter.cpp.

References NbDataStab.

Referenced by PhoDetPhaMet::gettStab(), PhoDetPhaMet::init(), and NoiseFilter::loadNoise().

10.7.3.6 void Filter::init (vector< vector< double > > *alpha_n*, vector< vector< double > > *beta_n*, int *NbDataStabilization_n*)

Initializes an instance with default values and inputs.

- alpha = alpha_n input
- beta = beta_n input
- NbDataStab = NbDataStabilization_n input
- TmpData = (alpha_n size + 1) null vectors

Definition at line 138 of file LISACODE-Filter.cpp.

References alpha, beta, NbDataStab, and TmpData.

Referenced by Filter(), and NoiseFilter::NoiseFilter().

10.7.4 Member Data Documentation

10.7.4.1 vector< vector<double> > Filter::alpha `[protected]`

Alpha parameters list.

Definition at line 44 of file LISACODE-Filter.h.

Referenced by App(), Filter(), getAlpha(), getDepth(), and init().

10.7.4.2 vector< vector<double> > Filter::beta `[protected]`

Beta parameters list.

Definition at line 46 of file LISACODE-Filter.h.

Referenced by App(), Filter(), getBeta(), getDepth(), and init().

10.7.4.3 int Filter::NbDataStab `[protected]`

Number of data for stabilization.

Definition at line 48 of file LISACODE-Filter.h.

Referenced by Filter(), getNbDataStab(), and init().

10.7.4.4 vector< vector<double> > Filter::TmpData [protected]

Temporary data.

Definition at line 50 of file LISACODE-Filter.h.

Referenced by App(), Filter(), and init().

The documentation for this class was generated from the following files:

- LISACODE-Filter.h
- LISACODE-Filter.cpp

10.8 Geometry Class Reference

`#include <LISACODE-Geometry.h>`

10.8.1 Detailed Description

Orbit geometry class.

Definition at line 39 of file LISACODE-Geometry.h.

Public Member Functions

- Geometry ()

  Constructs an instance and initializes it with default values.

- Geometry (double t0_n, double rot0_n)

  Constructs an instance and initializes it with default values and t0_n and rot0_n inputs.

- Geometry (double t0_n, double rot0_n, double L0m_n)

  Constructs an instance and initializes it with default values and t0_n, rot0_n and L0m_n inputs.

- Geometry (double t0_n, double rot0_n, double L0m_n, int order_default_n, int move_n)

  Constructs an instance and initializes it with default values and t0_n, rot0_n, L0m_n, order_default_n and move_n inputs.

- Geometry (double t0_n, double rot0_n, double L0m_n, int order_default_n, int move_n, double tStep)

  Constructs an instance and initializes it with t0_n, rot0_n, L0m_n, order_default_n, move_n and tStep inputs.

- ∼Geometry ()

  Destructor.

- void init (double t0_n, double rot0_n, double L0m_n, int order_default_n, int move_n, double tStep)

  Initializes attrubutes using t0_n, rot0_n, L0m_n, order_default_n, move_n and tStep inputs.

- double getL0 ()

  Returns L0m attribute.

- double gett0 ()

  Returns t0 attribute.

- Couple exanom (int nb, double ts)

  Returns the cosinus and sinus of the eccentric anomaly, depending on time ts (s) and spacecraft number nb=[1,3].

- Vect position (int nb, double t)

  Returns the position of the spacecraft in the barycentric frame, depending on time ts (s) and spacecraft number nb=[1,3].

- Vect velocity (int nb, double t)

  Returns the velocity of the spacecraft in the barycentric frame, depending on time ts (s) and spacecraft number nb=[1,3].

- double tdelay (int em, int rec, int order, double trec)

  Gets delay for specified order, depending on emitter em, receiver rec, reception time trec inputs.

- double tdelayOrderContribution (int em, int rec, int order, double trec)

  Gets contribution of specified order, depending on emitter em, receiver rec, reception time trec inputs.

- double ArmVelocity (int em, int rec, double trec)

  Gets spacecrafts relative velocity along an arm.

- Vect VectNormal (double t)

  Gets unitary vector normal to plane defined by 3 satellites, depending on t time input.

- Vect gposition (int nb, double t)

  Gives direct value for position, depending on spacecraft number nb and time t inputs.

- double gtdelay (int em, int rec, int order, double trec)

  Gives delay, depending on transmitter em, receiver rec, order and trec time inputs.

Protected Attributes

- double L0m

  Nominal arm length in meters.

- double alpha

  Orbital parameter.

- double nu

  Orbital angular constant.

- double tmu

  Orbital constant.

- double cmu

  Orbital constant.

- double smu

  Orbital constant.

- double e

  Eccentricity.

- double sqrtee

  Eccentricity derived constant.

- double arot

  Phase between satellites : constant, $arot = \frac{2 \cdot \pi}{3}$.

- double t0

  Initial time (seconds).

- double rot0

  Initial rotation (radians).

- vector< double > rot

  Satellites phase.

- vector< double > crot

  Satellites phase cosinus.

- vector< double > srot

  Satellites phase sinus.

- int move

  0 for LISA fixed, 1 for LISA moved (default)

- int order_default

  if -1 read the specified order else by-pass the specified order

- Vect SCposStore [3]

  Stored positions.

- double tStorePos

  last position computation time.

- double tRangeStorePos

  last position storage time.

- double DelayStore [6]

  Stored delays for each one of six arms.

- double tStoreDelay

  Last delay computation time.

- double tRangeStoreDelay

  Last delay storage time.

10.8.2 Constructor & Destructor Documentation

10.8.2.1 Geometry::Geometry ()

Constructs an instance and initializes it with default values.

init is called with the following arguments :

- t0_n = 0.0
- rot0_n = 0.0
- L0m_n = L0_m_default
- order_default_n = -1
- move_n = 1
- tStep = 10.0

Definition at line 27 of file LISACODE-Geometry.cpp.

References init(), and L0_m_default.

10.8.2.2 Geometry::Geometry (double *t0_n*, double *rot0_n*)

Constructs an instance and initializes it with default values and t0_n and rot0_n inputs.

init is called with the following arguments :

- t0_n = t0_n input
- rot0_n = rot0_n input
- L0m_n = L0_m_default
- order_default_n = -1
- move_n = 1
- tStep = 10.0

Definition at line 43 of file LISACODE-Geometry.cpp.

References init(), and L0_m_default.

10.8.2.3 Geometry::Geometry (double *t0_n*, double *rot0_n*, double *L0m_n*)

Constructs an instance and initializes it with default values and t0_n, rot0_n and L0m_n inputs.

init is called with the following arguments :

- t0_n = t0_n input
- rot0_n = rot0_n input
- L0m_n = L0m_n input
- order_default_n = -1
- move_n = 1
- tStep = 10.0

Definition at line 59 of file LISACODE-Geometry.cpp.

References init().

10.8.2.4 Geometry::Geometry (double *t0_n*, double *rot0_n*, double *L0m_n*, int *order_default_n*, int *move_n*)

Constructs an instance and initializes it with default values and t0_n, rot0_n, L0m_n, order_default_n and move_n inputs.

init is called with the following arguments :

- t0_n = t0_n input
- rot0_n = rot0_n input
- L0m_n = L0m_n input
- order_default_n = order_default_n input
- move_n = move_n input
- tStep = 10.0

Definition at line 75 of file LISACODE-Geometry.cpp.

References init().

10.8.2.5 Geometry::Geometry (double *t0_n*, double *rot0_n*, double *L0m_n*, int *order_default_n*, int *move_n*, double *tStep*)

Constructs an instance and initializes it with t0_n, rot0_n, L0m_n, order_default_n, move_n and tStep inputs.

init is called with the following arguments :

- t0_n = t0_n input
- rot0_n = rot0_n input
- L0m_n = L0m_n input
- order_default_n = order_default_n input
- move_n = move_n input
- tStep = tStep input

Definition at line 90 of file LISACODE-Geometry.cpp.

References init().

10.8.2.6 Geometry::∼Geometry ()

Destructor.

Definition at line 99 of file LISACODE-Geometry.cpp.

10.8.3 Member Function Documentation

10.8.3.1 double Geometry::ArmVelocity (int *em*, int *rec*, double *trec*)

Gets spacecrafts relative velocity along an arm.

Parameters:

em emitter

rec receiver

trec reception time

Inputs are checked: em and rec expected values are 1,2 and 3.

Returns:

$(\overrightarrow{v_j} - \overrightarrow{v_i}) \cdot \overrightarrow{n}$,
where :
ri is computed using position method with rec and trec inputs,
vi is computed using velocity method with rec and trec inputs, then divided by c_SI,
rj is computed using position method with em and trec inputs,
vj is computed using velocity method with em and trec inputs, then divided by c_SI,
$\overrightarrow{r_{ij}} = \overrightarrow{r_j} - \overrightarrow{r_i}$ and $\overrightarrow{n} = -\frac{\overrightarrow{r_{ij}}}{\left\|\overrightarrow{r_{ij}}\right\|}$.
c_SI and gamma_u constatnts are used

Definition at line 465 of file LISACODE-Geometry.cpp.

References position(), Vect::unit(), and velocity().

10.8.3.2 Couple Geometry::exanom (int *nb*, double *ts*)

Returns the cosinus and sinus of the eccentric anomaly, depending on time ts (s) and spacecraft number nb=[1,3].

Returns:

$zpsi = (cos(ex_2), sin(ex_2))$,
where :
$t = t0 + move \cdot ts$,
$\psi = \omega \cdot t - rot_{nb-1}$,
$ex_0 = \psi + e \cdot sin(\psi) \cdot cos(\frac{1-3 \cdot cos(\psi)^2}{2})$,
and $ex_{k=1,2,3} = ex_{k-1} - \frac{ex_{k-1} - e \cdot sin(ex_{k-1}) - \psi}{1 - e \cdot cos(ex_{k-1})}$.

Definition at line 209 of file LISACODE-Geometry.cpp.

References e, move, omega, rot, and t0.

Referenced by position(), and velocity().

10.8.3.3 double Geometry::getL0 () `[inline]`

Returns L0m attribute.

Definition at line 117 of file LISACODE-Geometry.h.

References L0m.

10.8.3.4 double Geometry::gett0 () [inline]

Returns t0 attribute.

Definition at line 119 of file LISACODE-Geometry.h.

References t0.

Referenced by BackgroundGalactic::deltanu().

10.8.3.5 Vect Geometry::gposition (int *nb*, double *t*)

Gives direct value for position, depending on spacecraft number nb and time t inputs.

position method is used to set SCposStore attribute.

if $abs(t - tStorePos) > tRangeStorePos$,

$$tStorePos = t \text{ and } SCposStore[i-1] = position(i,t) \text{ for i=1,2,3,4}$$

Returns:

$SCposStore[nb - 1]$

Definition at line 520 of file LISACODE-Geometry.cpp.

References position(), SCposStore, tRangeStorePos, and tStorePos.

Referenced by TrFctGW::deltanu(), LISA::gPosSC(), and main().

10.8.3.6 double Geometry::gtdelay (int *em*, int *rec*, int *order*, double *trec*)

Gives delay, depending on transmitter em, receiver rec, order and trec time inputs.

tdelay method is used to set DelayStore attribute

$$bs = \begin{cases} 3 & \text{if (mod(em,3)+1=rec)} \\ 0 & \text{else} \end{cases}$$

If (abs(trec-tStoreDelay)>tRangeStorePos)

$$\begin{cases} StoreDelay = trec \\ \text{for } i = 1,2,3,4 & \begin{cases} DelayStore[i-1] = tdelay(i,i,mod((i+1),3)+1,2,0) \\ DelayStore[i+2] = tdelay(i,i,mod(i,3)+1,2,0) \end{cases} \end{cases}$$

Returns:

$DelayStore[bs + em - 1]$

Definition at line 545 of file LISACODE-Geometry.cpp.

References DelayStore, tdelay(), tRangeStoreDelay, and tStoreDelay.

Referenced by TrFctGW::deltanu(), LISA::gArmLength(), and LISA::gDelayT().

10.8.3.7 void Geometry::init (double *t0_n*, double *rot0_n*, double *L0m_n*, int *order_default_n*, int *move_n*, double *tStep*)

Initializes attrubutes using t0_n, rot0_n, L0m_n, order_default_n, move_n and tStep inputs.

Inputs are checked.

tdelay method is used to set DelayStore attribute

position method is used to set SCposStore attribute

Attributes are set :

- t0 = t0_n input (expected to be positive or null)
- rot0 = rot0_n input
- L0m = L0m_n input (expected to be positive or null)
- order_default = order_default_n input
- move = move_n input (expected to be 0 or 1)
- alpha = $\frac{L0m}{2 \cdot Rgc}$
- nu = $\frac{\pi}{3} + \frac{5 \cdot \alpha}{8}$
- tmu = $\frac{\alpha \cdot sin(\nu)}{sin(\frac{\pi}{3})} + \alpha \cdot cos(\nu)$
- cmu = $\frac{1}{\sqrt{1+t_\mu^2}}$
- smu = $t_\mu \cdot c_\mu$
- e = $\sqrt{1 + \frac{4}{\sqrt{3}} \cdot cos(\nu) \cdot \alpha + \frac{4}{3}\alpha^2} - 1$
- sqrtee = $\sqrt{1 - e^2}$
- arot = $\frac{2 \cdot \pi}{3}$
- rot : $rot[i] = i \cdot arot \cdot rot0$ for i=1,2,3
- crot : $crot[i] = cos(rot_i)$ for i=1,2,3
- srot : $srot[i = 0, 1, 2] = sin(rot_i)$ for i=1,2,3
- SCposStore : $SCposStore[i] = position(i, 0)$ for i=1,2,3,4
- DelayStore : for i=1,2,3,4 $\left\{ \begin{array}{l} DelayStore[i-1] = tdelay(i, i, mod((i+1), 3) + 1, 2, 0) \\ DelayStore[i+2] = tdelay(i, i, mod(i, 3) + 1, 2, 0) \end{array} \right.$
- tRangeStorePos = tRangeStorePos_default
- tRangeStoreDelay = $min(tStep, tRangeStoreDelay\_default)$

Definition at line 142 of file LISACODE-Geometry.cpp.

References alpha, arot, cmu, crot, DelayStore, e, L0m, move, nu, order_default, position(), Rgc, rot, rot0, SCposStore, smu, sqrtee, srot, t0, tdelay(), tmu, tRangeStoreDelay, tRangeStoreDelay_default, tRangeStorePos, and tRangeStorePos_default.

Referenced by Geometry().

10.8.3.8 Vect Geometry::position (int *nb*, double *t*)

Returns the position of the spacecraft in the barycentric frame, depending on time ts (s) and spacecraft number nb=[1,3].

crot, srot and sqrtee attributes are used, as Rgc constant.

Eccentric anomaly zpsi=(cpsi,spsi) is computed using exanom method.

Returns:

$$\overrightarrow{r} = \begin{pmatrix} Rgc \cdot (cpsi - e) \cdot cmu \cdot crot_{nb-1} - sqrtee \cdot spsi \cdot srot_{nb-1} \\ Rgc \cdot (cpsi - e) \cdot cmu \cdot srot_{nb-1} - sqrtee \cdot spsi \cdot crot_{nb-1} \\ -Rgc \cdot smu \cdot (cpsi - e) \end{pmatrix}$$

Definition at line 249 of file LISACODE-Geometry.cpp.

References cmu, crot, e, exanom(), Vect::p, Rgc, smu, sqrtee, srot, Couple::x, and Couple::y.

Referenced by ArmVelocity(), gposition(), init(), tdelay(), tdelayOrderContribution(), and VectNormal().

10.8.3.9 double Geometry::tdelay (int *em*, int *rec*, int *order*, double *trec*)

Gets delay for specified order, depending on emitter em, receiver rec, reception time trec inputs.

Inputs are checked:

- em and rec expected values are 1,2 and 3
- order expected values are 0, 1 (for 1/2) and 2 (for 1)

Computation:

Returns:

$delay = \sum_{i=0}^{order} c_i$,
where :
ri is computed using position method with rec and trec inputs,
vi is computed using velocity method with rec and trec inputs, then divided by c_SI,
rj is computed using position method with em and trec inputs,
vj is computed using velocity method with em and trec inputs, then divided by c_SI,
$\overrightarrow{r_{ij}} = \overrightarrow{r_j} - \overrightarrow{r_i}$,
$c_0 = \frac{\left\|\overrightarrow{r_{ij}}\right\|}{C}$,
$\overrightarrow{n} = -\frac{\overrightarrow{r_{ij}}}{\left\|\overrightarrow{r_{ij}}\right\|}$,
$c_1 = t_{ij} \cdot \overrightarrow{n} \cdot \overrightarrow{v_j}$,
$c_2 = c_{21} + c_{22} + c_{23}$,
$c_{21} = \frac{t_{ij}}{2} \cdot (\overrightarrow{v_j}^2 + (\overrightarrow{n} \cdot \overrightarrow{v_j})^2)$,
$c_{22} = -\frac{{t_{ij}}^2}{2} \cdot C \cdot RSchw$,
$c_{23} = -\frac{RSchw}{C} \cdot (1 + \gamma_u) \cdot log\left(\frac{\sqrt{(C \cdot t_{ij} \cdot \overrightarrow{n} \cdot \overrightarrow{r_{ij}})^2 + {r_i}^2 + C \cdot t_{ij} + \overrightarrow{n} \cdot \overrightarrow{r_i}}}{\overrightarrow{n} \cdot \overrightarrow{r_i} + \left\|\overrightarrow{r_i}\right\|}\right)$.
c_SI and gamma_u constants are used.

Definition at line 326 of file LISACODE-Geometry.cpp.

References c_SI, gamma_u, Vect::norme(), order_default, position(), RSchw, Vect::unit(), and velocity().

Referenced by gtdelay(), init(), and main().

10.8.3.10 double Geometry::tdelayOrderContribution (int *em*, int *rec*, int *order*, double *trec*)

Gets contribution of specified order, depending on emitter em, receiver rec, reception time trec inputs.

Inputs are checked:

- em and rec expected values are 1,2 and 3
- order expected values are 1 (for 1/2) and 2 (for 1)

Returns:

$$\begin{cases} c_1 & \text{if } order = 1 \\ c_2 & \text{if } order = 2 \\ 0 & \text{else} \end{cases}$$

where :

ri is computed using position method with rec and trec inputs,

vi is computed using velocity method with rec and trec inputs, then divided by c_SI,

rj is computed using position method with em and trec inputs,

vj is computed using velocity method with em and trec inputs, then divided by c_SI,

$\overrightarrow{r_{ij}} = \overrightarrow{r_j} - \overrightarrow{r_i}$,

$\overrightarrow{n} = -\frac{\overrightarrow{r_{ij}}}{\left\|\overrightarrow{r_{ij}}\right\|}$,

$c_1 = t_{ij} \cdot \overrightarrow{n} \cdot \overrightarrow{v_j}$,

$c_2 = c_{21} + c_{22} + c_{23}$,

$c_{21} = \frac{t_{ij}}{2} \cdot (\overrightarrow{v_j}^2 + (\overrightarrow{n} \cdot \overrightarrow{v_j})^2)$,

$c_{22} = -\frac{{t_{ij}}^2}{2} \cdot C \cdot RSchw$,

$c_{23} = -\frac{RSchw}{C} \cdot (1 + \gamma_u) \cdot log(\frac{\sqrt{(C \cdot t_{ij} \cdot \overrightarrow{n} \cdot \overrightarrow{r_{ij}})^2 + {r_i}^2 + C \cdot t_{ij} + \overrightarrow{n} \cdot \overrightarrow{r_i}}}{\overrightarrow{n} \cdot \overrightarrow{r_i} + \left\|\overrightarrow{r_i}\right\|})$.

c_SI and gamma_u constants are used

Definition at line 403 of file LISACODE-Geometry.cpp.

References c_SI, gamma_u, Vect::norme(), position(), RSchw, Vect::unit(), and velocity().

10.8.3.11 Vect Geometry::VectNormal (double *t*)

Gets unitary vector normal to plane defined by 3 satellites, depending on t time input.

position method is used to set SCposStore attribute

Returns:

$$\frac{(\overrightarrow{position(2,t)} - \overrightarrow{position(3,t)}) \bigwedge (\overrightarrow{position(3,t)} - \overrightarrow{position(1,t)})}{\left\|(\overrightarrow{position(2,t)} - \overrightarrow{position(3,t)}) \bigwedge (\overrightarrow{position(3,t)} - \overrightarrow{position(1,t)})\right\|}$$

Definition at line 494 of file LISACODE-Geometry.cpp.

References Vect::p, position(), and Vect::unit().

Referenced by main().

10.8.3.12 Vect Geometry::velocity (int *nb*, double *t*)

Returns the velocity of the spacecraft in the barycentric frame, depending on time ts (s) and spacecraft number nb=[1,3].

crot, srot and sqrtee attributes are used, as Rgc constant.

Eccentric anomaly zpsi=(cpsi,spsi) is computed using exanom method.

Returns:

$$\overrightarrow{v} = \left(\begin{array}{l} \dot{\psi} \cdot (-spsi \cdot cmu \cdot crot_{nb-1} - sqrtee \cdot cpsi \cdot srot_{nb-1}) \\ \dot{\psi} \cdot (-spsi \cdot cmu \cdot srot_{nb-1} + sqrtee \cdot cpsi \cdot crot_{nb-1}) \\ \dot{\psi} \cdot smu \cdot spsi \end{array} \right),$$

where $\dot{\psi} = \frac{\omega \cdot Rgc}{1 - e \cdot cpsi}$

Definition at line 280 of file LISACODE-Geometry.cpp.

References cmu, crot, e, exanom(), omega, Vect::p, Rgc, smu, sqrtee, srot, Couple::x, and Couple::y.

Referenced by ArmVelocity(), tdelay(), and tdelayOrderContribution().

10.8.4 Member Data Documentation

10.8.4.1 Geometry::alpha `[protected]`

Orbital parameter.

Referenced by init().

10.8.4.2 Geometry::arot `[protected]`

Phase between satellites : constant, $arot = \frac{2 \cdot \pi}{3}$.

Referenced by init().

10.8.4.3 Geometry::cmu `[protected]`

Orbital constant.

Referenced by init(), position(), and velocity().

10.8.4.4 Geometry::crot `[protected]`

Satellites phase cosinus.

Referenced by init(), position(), and velocity().

10.8.4.5 double Geometry::DelayStore[6] `[protected]`

Stored delays for each one of six arms.

Each arm corresponds to a pair of emitter (em) and receiver (rec):

- arm 2: (em,rec)=(1,3)

- arm 3: (em,rec)=(2,1)
- arm 1: (em,rec)=(3,2)
- arm 6: (em,rec)=(1,2)
- arm 5: (em,rec)=(2,3)
- arm 4: (em,rec)=(3,1)

Definition at line 97 of file LISACODE-Geometry.h.

Referenced by gtdelay(), and init().

10.8.4.6 Geometry::e `[protected]`

Eccentricity.

Referenced by exanom(), init(), position(), and velocity().

10.8.4.7 double Geometry::L0m `[protected]`

Nominal arm length in meters.

Nominal distance between two satellites.

Definition at line 47 of file LISACODE-Geometry.h.

Referenced by getL0(), and init().

10.8.4.8 int Geometry::move `[protected]`

0 for LISA fixed, 1 for LISA moved (default)

Definition at line 78 of file LISACODE-Geometry.h.

Referenced by exanom(), and init().

10.8.4.9 Geometry::nu `[protected]`

Orbital angular constant.

Referenced by init().

10.8.4.10 int Geometry::order_default `[protected]`

if -1 read the specified order else by-pass the specified order

Definition at line 80 of file LISACODE-Geometry.h.

Referenced by init(), and tdelay().

10.8.4.11 Geometry::rot `[protected]`

Satellites phase.

Referenced by exanom(), and init().

10.8.4.12 Geometry::rot0 [protected]

Initial rotation (radians).

Referenced by init().

10.8.4.13 Vect Geometry::SCposStore[3] [protected]

Stored positions.

Definition at line 82 of file LISACODE-Geometry.h.

Referenced by gposition(), and init().

10.8.4.14 Geometry::smu [protected]

Orbital constant.

Referenced by init(), position(), and velocity().

10.8.4.15 Geometry::sqrtee [protected]

Eccentricity derived constant.

Referenced by init(), position(), and velocity().

10.8.4.16 Geometry::srot [protected]

Satellites phase sinus.

Referenced by init(), position(), and velocity().

10.8.4.17 Geometry::t0 [protected]

Initial time (seconds).

Referenced by exanom(), gett0(), and init().

10.8.4.18 Geometry::tmu [protected]

Orbital constant.

Referenced by init().

10.8.4.19 double Geometry::tRangeStoreDelay [protected]

Last delay storage time.

Definition at line 101 of file LISACODE-Geometry.h.

Referenced by gtdelay(), and init().

10.8.4.20 double Geometry::tRangeStorePos [protected]

last position storage time.

Definition at line 86 of file LISACODE-Geometry.h.

Referenced by gposition(), and init().

10.8.4.21 double Geometry::tStoreDelay [protected]

Last delay computation time.

Definition at line 99 of file LISACODE-Geometry.h.

Referenced by gtdelay().

10.8.4.22 double Geometry::tStorePos [protected]

last position computation time.

Definition at line 84 of file LISACODE-Geometry.h.

Referenced by gposition().

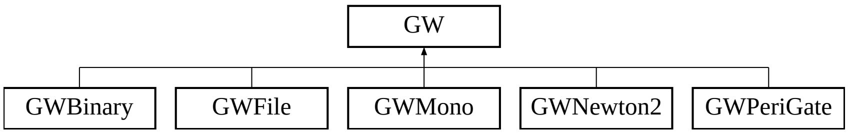
The documentation for this class was generated from the following files:

- LISACODE-Geometry.h
- LISACODE-Geometry.cpp
- LISACODE-Geometry_new.cpp

10.9 GW Class Reference

```
#include <LISACODE-GW.h>
```

Inheritance diagram for GW::



10.9.1 Detailed Description

Gravitational Waves parameters are described in this class.

Definition at line 37 of file LISACODE-GW.h.

Public Member Functions

- GW ()

  Constructs an instance and initializes it with default values.

- GW (double Beta_n, double Lambda_n)

  Constructs an instance and initializes it with default values and inputs.

- GW (double Beta_n, double Lambda_n, double AnglPol_n)

  Constructs an instance and initializes it with default values and inputs.

- virtual ~GW ()

  Destructor.

- double getBeta () const

  Returns Beta attribute.

- void setBeta (double Beta_n)

  Sets Beta and DirProp attributes.

- double getLambda () const

  Returns Lambda attribute.

- void setLambda (double Lambda_n)

  Sets Lambda attribute.

- vector< double > getDirProp () const

  Returns DirProp attribute.

- void setDirProp (vector< double > DirProp_n)

  Sets DirProp attribute.

- double getAnglPol ()

  Returns AnglPol attribute.

- void setAnglPol (double AnglPol_n)

  Sets AnglPol attribute.

- virtual double hp (double t)

  Gives h_plus polarisation component, depending on time t *input.*

- virtual double hc (double t)

  Gives h_cross polarisation component, depending on time t *input.*

- void CalculDirProp ()

  Computes DirProp attribute components, depending on Lambda and Beta attributes.

Protected Attributes

- double Beta

  Source direction angle (in the heliocentric reference frame) in radians.

- double Lambda

  Source direction angle (in the heliocentric reference frame) in radians.

- vector< double > DirProp

  Source direction unit vector (in the heliocentric reference frame).

- double AnglPol

  Polarisation angle.

10.9.2 Constructor & Destructor Documentation

10.9.2.1 GW::GW ()

Constructs an instance and initializes it with default values.

- Beta = 0
- Lambda = 0
- AnglPol = 0
- CalculDirProp method is called to fill DirProp attribute

Definition at line 24 of file LISACODE-GW.cpp.

References AnglPol, Beta, CalculDirProp(), DirProp, and Lambda.

10.9.2.2 GW::GW (double *Beta_n*, double *Lambda_n*)

Constructs an instance and initializes it with default values and inputs.

Inputs are checked:

$$\beta_n \in [-\frac{\pi}{2}, -\frac{\pi}{2}]$$

$$\lambda_n \in [0, 2 \cdot \pi]$$

- Beta = Beta_n
- Lambda = Lambda_n
- AnglPol = 0
- CalculDirProp method is called to fill DirProp attribute

Definition at line 44 of file LISACODE-GW.cpp.

References AnglPol, Beta, CalculDirProp(), DirProp, and Lambda.

10.9.2.3 GW::GW (double *Beta_n*, double *Lambda_n*, double *AnglPol_n*)

Constructs an instance and initializes it with default values and inputs.

Inputs are checked:

$$\beta_n \in [-\frac{\pi}{2}, -\frac{\pi}{2}]$$

$$\lambda_n \in [0, 2 \cdot \pi]$$

$$AnglPol_n \in [0, 2 \cdot \pi]$$

- Beta = Beta_n
- Lambda = Lambda_n
- AnglPol = AnglPol_n
- CalculDirProp method is called to fill DirProp attribute

Definition at line 69 of file LISACODE-GW.cpp.

References AnglPol, Beta, CalculDirProp(), DirProp, and Lambda.

10.9.2.4 GW::∼GW () `[virtual]`

Destructor.

Definition at line 86 of file LISACODE-GW.cpp.

10.9.3 Member Function Documentation

10.9.3.1 void GW::CalculDirProp ()

Computes DirProp attribute components, depending on Lambda and Beta attributes.

$$DirProp = \begin{pmatrix} -cos(\lambda) \cdot cos(\beta) \\ -sin(\lambda) \cdot cos(\beta) \\ -sin(\beta) \end{pmatrix}$$

Definition at line 203 of file LISACODE-GW.cpp.

References Beta, DirProp, and Lambda.

Referenced by GW(), GWBinary::GWBinary(), setBeta(), and setLambda().

10.9.3.2 double GW::getAnglPol () `[inline]`

Returns AnglPol attribute.

Definition at line 70 of file LISACODE-GW.h.

References AnglPol.

10.9.3.3 double GW::getBeta () const `[inline]`

Returns Beta attribute.

Definition at line 59 of file LISACODE-GW.h.

References Beta.

10.9.3.4 vector<double> GW::getDirProp () const `[inline]`

Returns DirProp attribute.

Definition at line 66 of file LISACODE-GW.h.

References DirProp.

10.9.3.5 double GW::getLambda () const `[inline]`

Returns Lambda attribute.

Definition at line 62 of file LISACODE-GW.h.

References Lambda.

10.9.3.6 double GW::hc (double *t*) `[virtual]`

Gives h_cross polarisation component, depending on time *t* input.

Virtual unused method. Returned value is constant : 0.

Reimplemented in GWBinary, GWFile, GWMono, GWNewton2, and GWPeriGate.

Definition at line 191 of file LISACODE-GW.cpp.

10.9.3.7 double GW::hp (double *t*) [virtual]

Gives h_plus polarisation component, depending on time *t* input.

Virtual unused method. Returned value is constant : 1.

Reimplemented in GWBinary, GWFile, GWMono, GWNewton2, and GWPeriGate.

Definition at line 181 of file LISACODE-GW.cpp.

10.9.3.8 void GW::setAnglPol (double *AnglPol_n*)

Sets AnglPol attribute.

Input is checked:

$$AnglPol_n \in [0, 2 \cdot \pi]$$

Reimplemented in GWMono.

Definition at line 168 of file LISACODE-GW.cpp.

References AnglPol.

10.9.3.9 void GW::setBeta (double *Beta_n*)

Sets Beta and DirProp attributes.

Input is checked:

$$\beta_n \in [-\frac{\pi}{2}, -\frac{\pi}{2}]$$

CalculDirProp method is called to fill DirProp attribute

Definition at line 100 of file LISACODE-GW.cpp.

References Beta, and CalculDirProp().

10.9.3.10 void GW::setDirProp (vector< double > *DirProp_n*)

Sets DirProp attribute.

Input is checked: *DirProp_n* must be a 3 components vector.

Computations: $DirProp_{norm}$ is normalized DirProp_n

- $$\beta = asin(DirProp_{norm}[2])$$
- $$\lambda = mod(atan(\frac{-DirProp_{norm}[1]}{-DirProp_{norm}[0]}), 2 \cdot \pi)$$

Definition at line 133 of file LISACODE-GW.cpp.

References Beta, DirProp, Lambda, and PRECISION.

10.9.3.11 void GW::setLambda (double *Lambda_n*)

Sets Lambda attribute.

Input is checked:

$$\lambda_n \in [0, 2 \cdot \pi]$$

CalculDirProp method is called to set DirProp attribute

Definition at line 115 of file LISACODE-GW.cpp.

References CalculDirProp(), and Lambda.

10.9.4 Member Data Documentation

10.9.4.1 double GW::AnglPol [protected]

Polarisation angle.

Reimplemented in GWMono, and GWPeriGate.

Definition at line 48 of file LISACODE-GW.h.

Referenced by getAnglPol(), GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), and setAnglPol().

10.9.4.2 GW::Beta [protected]

Source direction angle (in the heliocentric reference frame) in radians.

Referenced by CalculDirProp(), getBeta(), GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), setBeta(), and setDirProp().

10.9.4.3 vector<double> GW::DirProp [protected]

Source direction unit vector (in the heliocentric reference frame).

Definition at line 46 of file LISACODE-GW.h.

Referenced by CalculDirProp(), getDirProp(), GW(), and setDirProp().

10.9.4.4 GW::Lambda [protected]

Source direction angle (in the heliocentric reference frame) in radians.

Referenced by CalculDirProp(), getLambda(), GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), setDirProp(), and setLambda().

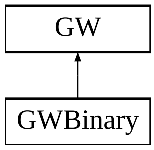
The documentation for this class was generated from the following files:

- LISACODE-GW.h
- LISACODE-GW.cpp

10.10 GWBinary Class Reference

```
#include <LISACODE-GWBinary.h>
```

Inheritance diagram for GWBinary::



10.10.1 Detailed Description

Gravitational Waves parameters for a monochromatic binary system are defined in this class.

Definition at line 34 of file LISACODE-GWBinary.h.

Public Member Functions

- GWBinary ()

  Constructs an instance and initializes it with default values.

- GWBinary (double Beta_n, double Lambda_n, double AnglPol_n, double bM1, double bM2, double bforb, double binc, double bphi0, double br)

  Constructs an instance and initializes it with default values and inputs.

- ~GWBinary ()

  Destructor.

- double getM1 ()

  Returns M1 attribute.

- double getM2 ()

  Returns M2 attribute.

- double getForb ()

  Returns forb attribute.

- double getInc ()

  Returns inc attribute.

- double getPhi0 ()

  Returns phi0 attribute.

- double getDistance ()

  Returns r attribute.

- void setM1 (double M1_n)

  Sets M1 attribute.

- void setM2 (double M2_n)

  Sets M2 attribute.

- void setForb (double forb_n)

  Sets forb attribute.

- void setInc (double inc_n)

  Sets inc attribute.

- void setPhi0 (double phi0_n)

  Sets phi0 attribute.

- void setDistance (double r_n)

  Sets r attribute.

- void init ()

  Sets polarized amplitudes attributes Ap and Ac.

- double hp (double t)

  Gives h_plus polarisation component depending on t *input time.*

- double hc (double t)

  Gives h_cross polarisation component depending on t *input time.*

- Couple hbin (double t)

  Gives (h_plus,h_cross)=(hp,hc) polarisation components as a Couple depending on t *input time.*

- double getBeta () const

  Returns Beta attribute.

- void setBeta (double Beta_n)

  Sets Beta and DirProp attributes.

- double getLambda () const

  Returns Lambda attribute.

- void setLambda (double Lambda_n)

  Sets Lambda attribute.

- vector< double > getDirProp () const

  Returns DirProp attribute.

- void setDirProp (vector< double > DirProp_n)

  Sets DirProp attribute.

- double getAnglPol ()

  Returns AnglPol attribute.

- void setAnglPol (double AnglPol_n)

Sets AnglPol attribute.

- void CalculDirProp ()

  Computes DirProp attribute components, depending on Lambda and Beta attributes.

Protected Attributes

- double M1

  First star mass (in solar masses).

- double M2

  Second star mass (in solar masses).

- double forb

  orbital frequency (Hertz)

- double inc

  inclination angle ($[0, 2 \cdot \pi[$)

- double phi0

  initial phase ($[0, 2 \cdot \pi[$)

- double r

  Distance to the source in kpc (kiloparsec).

- double Ap

  Polarized amplitude.

- double Ac

  Polarized amplitude.

- double Beta

  Source direction angle (in the heliocentric reference frame) in radians.

- double Lambda

  Source direction angle (in the heliocentric reference frame) in radians.

- vector$<$ double $>$ DirProp

  Source direction unit vector (in the heliocentric reference frame).

- double AnglPol

  Polarisation angle.

10.10.2 Constructor & Destructor Documentation

10.10.2.1 GWBinary::GWBinary ()

Constructs an instance and initializes it with default values.

$$\beta = 0.917311 - \frac{\pi}{2} \text{ rad}$$
$$\lambda = 2.97378 - \pi \text{ rad}$$
$$AnglPol = 2.46531 \text{ rad}$$
$$M_1 = \frac{M_{Sun}}{2} \text{ kg}$$
$$M_2 = 0.033 \cdot M_{Sun} \text{ kg, where } M_{Sun} = 1.9889 \cdot 10^{30} \text{ kg}$$
$$f_{orb} = \frac{1}{1028.76} \text{ Hz}$$
$$inc = 1.53921 \text{ rad}$$
$$\phi_0 = 0 \text{ rad}$$
$$r = \frac{kpc_m}{10} \text{ m, where } kpc_m = 3.0856675807 \cdot 10^{19} \text{ m}$$

- init method is called to fill polarized amplitudes attributes Ap and Ac.
- CalculDirProp method is called to fill DirProp attribute

Definition at line 31 of file LISACODE-GWBinary.cpp.

References GW::AnglPol, GW::Beta, GW::CalculDirProp(), forb, inc, init(), kpc_m, GW::Lambda, M1, M2, MS_SI, phi0, and r.

10.10.2.2 GWBinary::GWBinary (double *Beta_n*, double *Lambda_n*, double *AnglPol_n*, double *bM1*, double *bM2*, double *bforb*, double *binc*, double *bphi0*, double *br*)

Constructs an instance and initializes it with default values and inputs.

Inputs are checked : *bM1*, *bM2*, *bforb* and *br* must be positive or null

$$\beta = \beta_n \text{ rad}$$
$$\lambda = \lambda_n \text{ rad}$$
$$AnglPol = AnglPol_n \text{ rad}$$
$$M_1 = bM_1 \cdot M_{Sun} \text{ kg}$$
$$M_2 = bM_2 \cdot M_{Sun} \text{ kg, where } M_{Sun} = 1.9889 \cdot 10^{30} \text{ kg}$$
$$f_{orb} = bf_{orb} \text{ Hz}$$
$$inc = binc \text{ rad}$$
$$\phi_0 = b\phi_0 \text{ rad}$$
$$r = br \cdot kpc_m \text{ m, where } kpc_m = 3.0856675807 \cdot 10^{19} \text{ m}$$

- init method is called to fill polarized amplitudes attributes Ap and Ac.
- CalculDirProp method is called to fill DirProp attribute

Definition at line 71 of file LISACODE-GWBinary.cpp.

References forb, inc, init(), kpc_m, M1, M2, MS_SI, phi0, and r.

10.10.2.3 GWBinary::∼GWBinary ()

Destructor.

Definition at line 105 of file LISACODE-GWBinary.cpp.

10.10.3 Member Function Documentation

10.10.3.1 void GW::CalculDirProp () `[inherited]`

Computes DirProp attribute components, depending on Lambda and Beta attributes.

$$DirProp = \begin{pmatrix} -cos(\lambda) \cdot cos(\beta) \\ -sin(\lambda) \cdot cos(\beta) \\ -sin(\beta) \end{pmatrix}$$

Definition at line 203 of file LISACODE-GW.cpp.

References GW::Beta, GW::DirProp, and GW::Lambda.

Referenced by GW::GW(), GWBinary(), GW::setBeta(), and GW::setLambda().

10.10.3.2 double GW::getAnglPol () `[inline, inherited]`

Returns AnglPol attribute.

Definition at line 70 of file LISACODE-GW.h.

References GW::AnglPol.

10.10.3.3 double GW::getBeta () const `[inline, inherited]`

Returns Beta attribute.

Definition at line 59 of file LISACODE-GW.h.

References GW::Beta.

10.10.3.4 vector<double> GW::getDirProp () const `[inline, inherited]`

Returns DirProp attribute.

Definition at line 66 of file LISACODE-GW.h.

References GW::DirProp.

10.10.3.5 double GWBinary::getDistance () `[inline]`

Returns r attribute.

Definition at line 89 of file LISACODE-GWBinary.h.

References r.

10.10.3.6 double GWBinary::getForb () [inline]

Returns forb attribute.

Definition at line 83 of file LISACODE-GWBinary.h.

References forb.

10.10.3.7 double GWBinary::getInc () [inline]

Returns inc attribute.

Definition at line 85 of file LISACODE-GWBinary.h.

References inc.

10.10.3.8 double GW::getLambda () const [inline, inherited]

Returns Lambda attribute.

Definition at line 62 of file LISACODE-GW.h.

References GW::Lambda.

10.10.3.9 double GWBinary::getM1 () [inline]

Returns M1 attribute.

Definition at line 79 of file LISACODE-GWBinary.h.

References M1.

10.10.3.10 double GWBinary::getM2 () [inline]

Returns M2 attribute.

Definition at line 81 of file LISACODE-GWBinary.h.

References M2.

10.10.3.11 double GWBinary::getPhi0 () [inline]

Returns phi0 attribute.

Definition at line 87 of file LISACODE-GWBinary.h.

References phi0.

10.10.3.12 Couple GWBinary::hbin (double *t*)

Gives (h_plus,h_cross)=(hp,hc) polarisation components as a Couple depending on *t* input time.

Returns:

$$\text{Returned value} = \begin{pmatrix} A_p \cdot cos(4 \cdot \pi \cdot f_{orb} \cdot t + \phi_0) \\ A_c \cdot sin(4 \cdot \pi \cdot f_{orb} \cdot t + \phi_0) \end{pmatrix}$$

Definition at line 244 of file LISACODE-GWBinary.cpp.

References Ac, Ap, forb, hbin(), phi0, Couple::x, and Couple::y.

Referenced by hbin().

10.10.3.13 double GWBinary::hc (double *t*) `[virtual]`

Gives h_cross polarisation component depending on *t* input time.

$$\text{Returned value} = A_c \cdot sin(4 \cdot \pi \cdot f_{orb} \cdot t + \phi_0)$$

Reimplemented from GW.

Definition at line 229 of file LISACODE-GWBinary.cpp.

References Ac, forb, and phi0.

10.10.3.14 double GWBinary::hp (double *t*) `[virtual]`

Gives h_plus polarisation component depending on *t* input time.

$$\text{Returned value} = A_p \cdot cos(4 \cdot \pi \cdot f_{orb} \cdot t + \phi_0)$$

Reimplemented from GW.

Definition at line 217 of file LISACODE-GWBinary.cpp.

References Ap, forb, and phi0.

10.10.3.15 void GWBinary::init ()

Sets polarized amplitudes attributes Ap and Ac.

Returns:

Gravitational wave polarized amplitudes :

$$A_p = A \cdot (1 + cos^2(inc))$$

$$A_c = -2 \cdot A \cdot cos(inc)$$

where binary components orbital separation is :

$$R = (\frac{G \cdot (M_1 + M_2)}{(2 \cdot \pi \cdot f_{orb})^2})^{\frac{1}{3}} \text{ m}$$

and gravitational wave intrinsic amplitude is:

$$A = \frac{2 \cdot G^2 \cdot M_1 \cdot M_2}{C^4 \cdot r \cdot R}$$

Definition at line 192 of file LISACODE-GWBinary.cpp.

References Ac, Ap, c_SI, forb, G_SI, inc, M1, M2, and r.

Referenced by GWBinary().

10.10.3.16 void GW::setAnglPol (double *AnglPol_n*) [inherited]

Sets AnglPol attribute.

Input is checked:

$$AnglPol_n \in [0, 2 \cdot \pi]$$

Reimplemented in GWMono.

Definition at line 168 of file LISACODE-GW.cpp.

References GW::AnglPol.

10.10.3.17 void GW::setBeta (double *Beta_n*) [inherited]

Sets Beta and DirProp attributes.

Input is checked:

$$\beta_n \in [-\frac{\pi}{2}, -\frac{\pi}{2}]$$

CalculDirProp method is called to fill DirProp attribute

Definition at line 100 of file LISACODE-GW.cpp.

References GW::Beta, and GW::CalculDirProp().

10.10.3.18 void GW::setDirProp (vector< double > *DirProp_n*) [inherited]

Sets DirProp attribute.

Input is checked: *DirProp_n* must be a 3 components vector.

Computations: $DirProp_{norm}$ is normalized DirProp_n

- $$\beta = asin(DirProp_{norm}[2])$$
- $$\lambda = mod(atan(\frac{-DirProp_{norm}[1]}{-DirProp_{norm}[0]}), 2 \cdot \pi)$$

Definition at line 133 of file LISACODE-GW.cpp.

References GW::Beta, GW::DirProp, GW::Lambda, and PRECISION.

10.10.3.19 void GWBinary::setDistance (double *r_n*)

Sets r attribute.

Input is checked:

$$r_n \geq 0$$

Definition at line 171 of file LISACODE-GWBinary.cpp.

References r.

10.10.3.20 void GWBinary::setForb (double *forb_n*)

Sets forb attribute.

Input is checked:

$$forb_n \geq 0$$

Definition at line 138 of file LISACODE-GWBinary.cpp.

References forb.

10.10.3.21 void GWBinary::setInc (double *inc_n*)

Sets inc attribute.

Input is checked:

$$0 \leq inc_n \leq 2 \cdot \pi$$

Definition at line 149 of file LISACODE-GWBinary.cpp.

References inc.

10.10.3.22 void GW::setLambda (double *Lambda_n*) `[inherited]`

Sets Lambda attribute.

Input is checked:

$$\lambda_n \in [0, 2 \cdot \pi]$$

CalculDirProp method is called to set DirProp attribute

Definition at line 115 of file LISACODE-GW.cpp.

References GW::CalculDirProp(), and GW::Lambda.

10.10.3.23 void GWBinary::setM1 (double *M1_n*)

Sets M1 attribute.

Input is checked:

$$M1_n \geq 0$$

Definition at line 116 of file LISACODE-GWBinary.cpp.

References M1.

10.10.3.24 void GWBinary::setM2 (double *M2_n*)

Sets M2 attribute.

Input is checked:

$$M2_n \geq 0$$

Definition at line 127 of file LISACODE-GWBinary.cpp.

References M2.

10.10.3.25 void GWBinary::setPhi0 (double *phi0_n*)

Sets phi0 attribute.

Input is checked:

$$0 \leq phi0_n \leq 2 \cdot \pi$$

Definition at line 160 of file LISACODE-GWBinary.cpp.

References phi0.

10.10.4 Member Data Documentation

10.10.4.1 GWBinary::Ac [protected]

Polarized amplitude.

Referenced by hbin(), hc(), and init().

10.10.4.2 double GW::AnglPol [protected, inherited]

Polarisation angle.

Reimplemented in GWMono, and GWPeriGate.

Definition at line 48 of file LISACODE-GW.h.

Referenced by GW::getAnglPol(), GW::GW(), GWBinary(), GWNewton2::GWNewton2(), and GW::setAnglPol().

10.10.4.3 GWBinary::Ap [protected]

Polarized amplitude.

Referenced by hbin(), hp(), and init().

10.10.4.4 GW::Beta [protected, inherited]

Source direction angle (in the heliocentric reference frame) in radians.

Referenced by GW::CalculDirProp(), GW::getBeta(), GW::GW(), GWBinary(), GWNewton2::GWNewton2(), GW::setBeta(), and GW::setDirProp().

10.10.4.5 vector<double> GW::DirProp [protected, inherited]

Source direction unit vector (in the heliocentric reference frame).

Definition at line 46 of file LISACODE-GW.h.

Referenced by GW::CalculDirProp(), GW::getDirProp(), GW::GW(), and GW::setDirProp().

10.10.4.6 double GWBinary::forb [protected]

orbital frequency (Hertz)

Definition at line 43 of file LISACODE-GWBinary.h.

Referenced by getForb(), GWBinary(), hbin(), hc(), hp(), init(), and setForb().

10.10.4.7 GWBinary::inc [protected]

inclination angle ($[0, 2 \cdot \pi[$)

Referenced by getInc(), GWBinary(), init(), and setInc().

10.10.4.8 GW::Lambda [protected, inherited]

Source direction angle (in the heliocentric reference frame) in radians.

Referenced by GW::CalculDirProp(), GW::getLambda(), GW::GW(), GWBinary(), GWNewton2::GWNewton2(), GW::setDirProp(), and GW::setLambda().

10.10.4.9 GWBinary::M1 [protected]

First star mass (in solar masses).

Referenced by getM1(), GWBinary(), init(), and setM1().

10.10.4.10 GWBinary::M2 [protected]

Second star mass (in solar masses).

Referenced by getM2(), GWBinary(), init(), and setM2().

10.10.4.11 GWBinary::phi0 [protected]

initial phase ($[0, 2 \cdot \pi[$)

Referenced by getPhi0(), GWBinary(), hbin(), hc(), hp(), and setPhi0().

10.10.4.12 double GWBinary::r [protected]

Distance to the source in kpc (kiloparsec).

Definition at line 50 of file LISACODE-GWBinary.h.

Referenced by getDistance(), GWBinary(), init(), and setDistance().

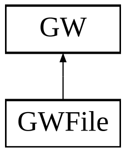
The documentation for this class was generated from the following files:

- LISACODE-GWBinary.h
- LISACODE-GWBinary.cpp

10.11 GWFile Class Reference

```
#include <LISACODE-GWFile.h>
```

Inheritance diagram for GWFile::



10.11.1 Detailed Description

Gravitational Waves file management.

Temporal polarization constraints are read from file.

Definition at line 38 of file LISACODE-GWFile.h.

Public Member Functions

- GWFile ()

  Constructs an instance and initializes it with default values.

- GWFile (double Beta_n, double Lambda_n, double AnglPol_n)

  Constructs an instance and initializes it with default values and inputs.

- GWFile (double Beta_n, double Lambda_n, double AnglPol_n, char ∗FileName)

  Constructs an instance and initializes it with default values and inputs.

- ∼GWFile ()

  Destructor.

- unsigned int getNbStored ()

  Returns size of TimeList attribute.

- void ReadFile (char ∗FileName)

  Reads file specified in argument.

- double Interpol (double t, int type)

  Makes Lagrange order 3 interpolation at tinput time t*, in list of hList specified by* type *(1 for h_plus (hp) or 2 for h_cross (hc)).*

- double hp (double t)

  Gives h_plus polarisation component depending on t *input time.*

- double hc (double t)

  Gives h_cross polarisation component depending on t *input time.*

- double getBeta () const

  Returns Beta attribute.

- void setBeta (double Beta_n)

  Sets Beta and DirProp attributes.

- double getLambda () const

  Returns Lambda attribute.

- void setLambda (double Lambda_n)

  Sets Lambda attribute.

- vector< double > getDirProp () const

  Returns DirProp attribute.

- void setDirProp (vector< double > DirProp_n)

  Sets DirProp attribute.

- double getAnglPol ()

  Returns AnglPol attribute.

- void setAnglPol (double AnglPol_n)

  Sets AnglPol attribute.

- void CalculDirProp ()

  Computes DirProp attribute components, depending on Lambda and Beta attributes.

Protected Attributes

- vector< double > TimeList

  Vector of time values associated to components samples.

- vector< Couple > hList

  List of (h_plus,h_cross)=(hp,hc) polarisation components.

- int LastUsedBin

  Last bin read.

- double Beta

  Source direction angle (in the heliocentric reference frame) in radians.

- double Lambda

  Source direction angle (in the heliocentric reference frame) in radians.

- vector< double > DirProp

  Source direction unit vector (in the heliocentric reference frame).

- double AnglPol

  Polarisation angle.

10.11.2 Constructor & Destructor Documentation

10.11.2.1 GWFile::GWFile ()

Constructs an instance and initializes it with default values.

Inherited attributes from GW are set to default values.

- TimeList = empty
- hList = empty
- LastUsedBin = 0

Definition at line 24 of file LISACODE-GWFile.cpp.

References hList, LastUsedBin, and TimeList.

10.11.2.2 GWFile::GWFile (double *Beta_n*, double *Lambda_n*, double *AnglPol_n*)

Constructs an instance and initializes it with default values and inputs.

GW constructor is called using *Beta_n*, *Lambda_n* and *AnglPol_n* inputs

- TimeList = empty
- hList = empty
- LastUsedBin = 0

Definition at line 39 of file LISACODE-GWFile.cpp.

References hList, LastUsedBin, and TimeList.

10.11.2.3 GWFile::GWFile (double *Beta_n*, double *Lambda_n*, double *AnglPol_n*, char ∗ *FileName*)

Constructs an instance and initializes it with default values and inputs.

GW constructor is called using *Beta_n*, *Lambda_n* and *AnglPol_n* inputs as arguments

- TimeList = empty
- hList = empty
- LastUsedBin = 0 TimeList and hList attributes are filled using ReadFile method with *FileName* input as argument

Definition at line 56 of file LISACODE-GWFile.cpp.

References hList, LastUsedBin, ReadFile(), and TimeList.

10.11.2.4 GWFile::∼GWFile ()

Destructor.

Definition at line 67 of file LISACODE-GWFile.cpp.

References hList, and TimeList.

10.11.3 Member Function Documentation

10.11.3.1 void GW::CalculDirProp () [inherited]

Computes DirProp attribute components, depending on Lambda and Beta attributes.

$$DirProp = \left(\begin{array}{l} -cos(\lambda) \cdot cos(\beta) \\ -sin(\lambda) \cdot cos(\beta) \\ -sin(\beta) \end{array} \right)$$

Definition at line 203 of file LISACODE-GW.cpp.

References GW::Beta, GW::DirProp, and GW::Lambda.

Referenced by GW::GW(), GWBinary::GWBinary(), GW::setBeta(), and GW::setLambda().

10.11.3.2 double GW::getAnglPol () [inline, inherited]

Returns AnglPol attribute.

Definition at line 70 of file LISACODE-GW.h.

References GW::AnglPol.

10.11.3.3 double GW::getBeta () const [inline, inherited]

Returns Beta attribute.

Definition at line 59 of file LISACODE-GW.h.

References GW::Beta.

10.11.3.4 vector<double> GW::getDirProp () const [inline, inherited]

Returns DirProp attribute.

Definition at line 66 of file LISACODE-GW.h.

References GW::DirProp.

10.11.3.5 double GW::getLambda () const [inline, inherited]

Returns Lambda attribute.

Definition at line 62 of file LISACODE-GW.h.

References GW::Lambda.

10.11.3.6 unsigned int GWFile::getNbStored () [inline]

Returns size of TimeList attribute.

Definition at line 57 of file LISACODE-GWFile.h.

References TimeList.

10.11.3.7 double GWFile::hc (double *t*) [virtual]

Gives h_cross polarisation component depending on *t* input time.

Returns:

Result of Interpol method with *t* input time and 2 as arguments.

Reimplemented from GW.

Definition at line 193 of file LISACODE-GWFile.cpp.

References Interpol().

10.11.3.8 double GWFile::hp (double *t*) [virtual]

Gives h_plus polarisation component depending on *t* input time.

returned value = result of Interpol method with t input time and 1 as arguments

Reimplemented from GW.

Definition at line 183 of file LISACODE-GWFile.cpp.

References Interpol().

10.11.3.9 double GWFile::Interpol (double *t*, int *type*)

Makes Lagrange order 3 interpolation at tinput time *t*, in list of hList specified by *type* (1 for h_plus (hp) or 2 for h_cross (hc)).

Returns:

$$returnedvalue = \begin{cases} \sum_{k=kmin}^{kmax} hList[k].x \cdot P_k & \text{if type=1} \\ \sum_{k=kmin}^{kmax} hList[k].y \cdot P_k & \text{if type=2} \end{cases}$$

$$\text{where } P_k = \prod_{j=kmin,j\neq k}^{kmax} \frac{t - TimeList[j]}{TimeList[k] - TimeList[j]}$$

LastUsedBin is updated, so that

$$TimeList[LastUsedBin] \leq t \leq TimeList[LastUsedBin + 1]$$

$$ordermin = floor(\frac{order + 1}{2}) \text{ where } order = 3$$

$$kmin = min(0, LastUsedBin - ordermin + 1)$$

$$kmax = max(LastUsedBin + order + 1 - ordermin, sizeof(TimeList) - 1)$$

Definition at line 125 of file LISACODE-GWFile.cpp.

References hList, LastUsedBin, and TimeList.

Referenced by hc(), and hp().

10.11.3.10 void GWFile::ReadFile (char * *FileName*)

Reads file specified in argument.

Header (lines begining with "#" charected) are ignored. For each line, time and 2 components are read and pushed back into TimeList and hList attributes.

Definition at line 81 of file LISACODE-GWFile.cpp.

References hList, ReadFile(), TimeList, Couple::x, and Couple::y.

Referenced by GWFile(), and ReadFile().

10.11.3.11 void GW::setAnglPol (double *AnglPol_n*) `[inherited]`

Sets AnglPol attribute.

Input is checked:

$$AnglPol_n \in [0, 2 \cdot \pi]$$

Reimplemented in GWMono.

Definition at line 168 of file LISACODE-GW.cpp.

References GW::AnglPol.

10.11.3.12 void GW::setBeta (double *Beta_n*) `[inherited]`

Sets Beta and DirProp attributes.

Input is checked:

$$\beta_n \in [-\frac{\pi}{2}, -\frac{\pi}{2}]$$

CalculDirProp method is called to fill DirProp attribute

Definition at line 100 of file LISACODE-GW.cpp.

References GW::Beta, and GW::CalculDirProp().

10.11.3.13 void GW::setDirProp (vector< double > *DirProp_n*) `[inherited]`

Sets DirProp attribute.

Input is checked: *DirProp_n* must be a 3 components vector.

Computations: $DirProp_{norm}$ is normalized DirProp_n

- $$\beta = asin(DirProp_{norm}[2])$$
- $$\lambda = mod(atan(\frac{-DirProp_{norm}[1]}{-DirProp_{norm}[0]}), 2 \cdot \pi)$$

Definition at line 133 of file LISACODE-GW.cpp.

References GW::Beta, GW::DirProp, GW::Lambda, and PRECISION.

10.11.3.14 void GW::setLambda (double *Lambda_n*) [inherited]

Sets Lambda attribute.

Input is checked:

$$\lambda_n \in [0, 2 \cdot \pi]$$

CalculDirProp method is called to set DirProp attribute

Definition at line 115 of file LISACODE-GW.cpp.

References GW::CalculDirProp(), and GW::Lambda.

10.11.4 Member Data Documentation

10.11.4.1 double GW::AnglPol [protected, inherited]

Polarisation angle.

Reimplemented in GWMono, and GWPeriGate.

Definition at line 48 of file LISACODE-GW.h.

Referenced by GW::getAnglPol(), GW::GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), and GW::setAnglPol().

10.11.4.2 GW::Beta [protected, inherited]

Source direction angle (in the heliocentric reference frame) in radians.

Referenced by GW::CalculDirProp(), GW::getBeta(), GW::GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), GW::setBeta(), and GW::setDirProp().

10.11.4.3 vector<double> GW::DirProp [protected, inherited]

Source direction unit vector (in the heliocentric reference frame).

Definition at line 46 of file LISACODE-GW.h.

Referenced by GW::CalculDirProp(), GW::getDirProp(), GW::GW(), and GW::setDirProp().

10.11.4.4 vector<Couple> GWFile::hList [protected]

List of (h_plus,h_cross)=(hp,hc) polarisation components.

Definition at line 44 of file LISACODE-GWFile.h.

Referenced by GWFile(), Interpol(), ReadFile(), and ~GWFile().

10.11.4.5 GW::Lambda [protected, inherited]

Source direction angle (in the heliocentric reference frame) in radians.

Referenced by GW::CalculDirProp(), GW::getLambda(), GW::GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), GW::setDirProp(), and GW::setLambda().

10.11.4.6 int GWFile::LastUsedBin [protected]

Last bin read.

Definition at line 46 of file LISACODE-GWFile.h.

Referenced by GWFile(), and Interpol().

10.11.4.7 vector<double> GWFile::TimeList [protected]

Vector of time values associated to components samples.

Definition at line 42 of file LISACODE-GWFile.h.

Referenced by getNbStored(), GWFile(), Interpol(), ReadFile(), and ∼GWFile().

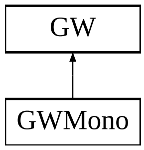
The documentation for this class was generated from the following files:

- LISACODE-GWFile.h
- LISACODE-GWFile.cpp

10.12 GWMono Class Reference

```
#include <LISACODE-GWMono.h>
```

Inheritance diagram for GWMono::



10.12.1 Detailed Description

Gravitational Waves instantaneous parameters h_plus and h_cross are described in this class.

Definition at line 35 of file LISACODE-GWMono.h.

Public Member Functions

- GWMono ()

  Constructs an instance and initializes it with default values.

- GWMono (double Beta_n, double Lambda_n, double AnglPol_n, double Freq_n, double Amplhp_n, double Amplhc_n)

  Constructs an instance and initializes it with default values and inputs.

- GWMono (double Beta_n, double Lambda_n, double AnglPol_n, double Freq_n, double Amplhp_n, double Amplhc_n, double Phi0hp_n, double Phi0hc_n)

  Constructs an instance and initializes it with inputs.

- double getFreq () const

  Returns Freq attribute.

- void setFreq (double Freq_n)

  Sets Freq attribute.

- double getAmplhp () const

  Returns Amplhp attribute.

- void setAmplhp (double Amplhp_n)

  Sets Amplhp attribute.

- double getAmplhc () const

  Returns Amplhc attribute.

- void setAmplhc (double Amplhc_n)

  Sets Amplhc attribute.

- double getAnglPol () const

Returns AnglPol attribute.

- void setAnglPol (double AnglPol_n)

  Sets AnglPol attribute.

- double getPhi0hp () const

  Returns Phi0hp attribute.

- void setPhi0hp (double Phi0hp_n)

  Sets Phi0hp attribute.

- double getPhi0hc () const

  Returns Phi0hc attribute.

- void setPhi0hc (double Phi0hc_n)

  Sets Phi0hc attribute.

- double hp (double t)

  Gives h_plus polarisation component depending on t *input time.*

- double hc (double t)

  Gives h_cross polarisation component depending on t *input time.*

- double getBeta () const

  Returns Beta attribute.

- void setBeta (double Beta_n)

  Sets Beta and DirProp attributes.

- double getLambda () const

  Returns Lambda attribute.

- void setLambda (double Lambda_n)

  Sets Lambda attribute.

- vector< double > getDirProp () const

  Returns DirProp attribute.

- void setDirProp (vector< double > DirProp_n)

  Sets DirProp attribute.

- double getAnglPol ()

  Returns AnglPol attribute.

- void CalculDirProp ()

  Computes DirProp attribute components, depending on Lambda and Beta attributes.

Protected Attributes

- double Freq

  Frequency.

- double Amplhp

  h_plus polarisation component amplitude

- double Amplhc

  h_cross polarisation component amplitude

- double AnglPol

  Polarisation angle (rad).

- double Phi0hp

  Initial phase of h_plus polarisation component.

- double Phi0hc

  Initial phase of h_cross polarisation component.

- double Beta

  Source direction angle (in the heliocentric reference frame) in radians.

- double Lambda

  Source direction angle (in the heliocentric reference frame) in radians.

- vector< double > DirProp

  Source direction unit vector (in the heliocentric reference frame).

10.12.2 Constructor & Destructor Documentation

10.12.2.1 GWMono::GWMono ()

Constructs an instance and initializes it with default values.

- Freq = 0
- Amplhp = 0
- Amplhc = 0
- AnglPol = 0
- Phi0hp = 0
- Phi0hc = 0

Definition at line 26 of file LISACODE-GWMono.cpp.

References Amplhc, Amplhp, AnglPol, Freq, Phi0hc, and Phi0hp.

10.12.2.2 GWMono::GWMono (double *Beta_n*, double *Lambda_n*, double *AnglPol_n*, double *Freq_n*, double *Amplhp_n*, double *Amplhc_n*)

Constructs an instance and initializes it with default values and inputs.

Inputs are checked :*Freq_n* must be positive or null.

GW constructor is called using *Beta_n*, *Lambda_n* and *AnglPol_n* inputs.

- Freq = Freq_n
- Amplhp = Amplhp_n
- Amplhc = Amplhc_n
- Phi0hp = 0
- Phi0hc = 0

Definition at line 46 of file LISACODE-GWMono.cpp.

References Amplhc, Amplhp, Freq, Phi0hc, and Phi0hp.

10.12.2.3 GWMono::GWMono (double *Beta_n*, double *Lambda_n*, double *AnglPol_n*, double *Freq_n*, double *Amplhp_n*, double *Amplhc_n*, double *Phi0hp_n*, double *Phi0hc_n*)

Constructs an instance and initializes it with inputs.

Inputs are checked :*Freq_n* must be positive or null.

GW constructor is called using *Beta_n*, *Lambda_n* and *AnglPol_n* inputs.

- Freq = Freq_n
- Amplhp = Amplhp_n
- Amplhc = Amplhc_n
- Phi0hp = Phi0hp_n
- Phi0hc = Phi0hc_n

Definition at line 78 of file LISACODE-GWMono.cpp.

References Amplhc, Amplhp, Freq, Phi0hc, and Phi0hp.

10.12.3 Member Function Documentation

10.12.3.1 void GW::CalculDirProp () `[inherited]`

Computes DirProp attribute components, depending on Lambda and Beta attributes.

$$DirProp = \begin{pmatrix} -cos(\lambda) \cdot cos(\beta) \\ -sin(\lambda) \cdot cos(\beta) \\ -sin(\beta) \end{pmatrix}$$

Definition at line 203 of file LISACODE-GW.cpp.

References GW::Beta, GW::DirProp, and GW::Lambda.

Referenced by GW::GW(), GWBinary::GWBinary(), GW::setBeta(), and GW::setLambda().

10.12.3.2 double GWMono::getAmplhc () const [inline]

Returns Amplhc attribute.

Definition at line 82 of file LISACODE-GWMono.h.

References Amplhc.

10.12.3.3 double GWMono::getAmplhp () const [inline]

Returns Amplhp attribute.

Definition at line 79 of file LISACODE-GWMono.h.

References Amplhp.

10.12.3.4 double GW::getAnglPol () [inline, inherited]

Returns AnglPol attribute.

Definition at line 70 of file LISACODE-GW.h.

References GW::AnglPol.

10.12.3.5 double GWMono::getAnglPol () const [inline]

Returns AnglPol attribute.

Definition at line 85 of file LISACODE-GWMono.h.

References AnglPol.

10.12.3.6 double GW::getBeta () const [inline, inherited]

Returns Beta attribute.

Definition at line 59 of file LISACODE-GW.h.

References GW::Beta.

10.12.3.7 vector<double> GW::getDirProp () const [inline, inherited]

Returns DirProp attribute.

Definition at line 66 of file LISACODE-GW.h.

References GW::DirProp.

10.12.3.8 double GWMono::getFreq () const [inline]

Returns Freq attribute.

Definition at line 76 of file LISACODE-GWMono.h.

References Freq.

10.12.3.9 double GW::getLambda () const [inline, inherited]

Returns Lambda attribute.

Definition at line 62 of file LISACODE-GW.h.

References GW::Lambda.

10.12.3.10 double GWMono::getPhi0hc () const [inline]

Returns Phi0hc attribute.

Definition at line 92 of file LISACODE-GWMono.h.

References Phi0hc.

10.12.3.11 double GWMono::getPhi0hp () const [inline]

Returns Phi0hp attribute.

Definition at line 88 of file LISACODE-GWMono.h.

References Phi0hp.

10.12.3.12 double GWMono::hc (double *t*) [virtual]

Gives h_cross polarisation component depending on *t* input time.

$$\text{returned value} = Amplhc \cdot sin(2 \cdot \pi \cdot Freq \cdot t + Phi0hc)$$

Reimplemented from GW.

Definition at line 152 of file LISACODE-GWMono.cpp.

References Amplhc, Freq, and Phi0hc.

10.12.3.13 double GWMono::hp (double *t*) [virtual]

Gives h_plus polarisation component depending on *t* input time.

$$\text{returned value} = Amplhp \cdot sin(2 \cdot \pi \cdot Freq \cdot t + Phi0hp)$$

Reimplemented from GW.

Definition at line 142 of file LISACODE-GWMono.cpp.

References Amplhp, Freq, and Phi0hp.

10.12.3.14 void GWMono::setAmplhc (double *Amplhc_n*)

Sets Amplhc attribute.

Definition at line 127 of file LISACODE-GWMono.cpp.

References Amplhc.

10.12.3.15 void GWMono::setAmplhp (double *Amplhp_n*)

Sets Amplhp attribute.

Definition at line 118 of file LISACODE-GWMono.cpp.

References Amplhp.

10.12.3.16 void GWMono::setAnglPol (double *AnglPol_n*)

Sets AnglPol attribute.

Input is checked:

$$AnglPol_n \in [0, 2 \cdot \pi]$$

Reimplemented from GW.

10.12.3.17 void GW::setBeta (double *Beta_n*) [inherited]

Sets Beta and DirProp attributes.

Input is checked:

$$\beta_n \in [-\frac{\pi}{2}, -\frac{\pi}{2}]$$

CalculDirProp method is called to fill DirProp attribute

Definition at line 100 of file LISACODE-GW.cpp.

References GW::Beta, and GW::CalculDirProp().

10.12.3.18 void GW::setDirProp (vector< double > *DirProp_n*) [inherited]

Sets DirProp attribute.

Input is checked: *DirProp_n* must be a 3 components vector.

Computations: $DirProp_{norm}$ is normalized DirProp_n

- $$\beta = asin(DirProp_{norm}[2])$$
- $$\lambda = mod(atan(\frac{-DirProp_{norm}[1]}{-DirProp_{norm}[0]}), 2 \cdot \pi)$$

Definition at line 133 of file LISACODE-GW.cpp.

References GW::Beta, GW::DirProp, GW::Lambda, and PRECISION.

10.12.3.19 void GWMono::setFreq (double *Freq_n*)

Sets Freq attribute.

Input is checked:*Freq_n* must be positive or null

Definition at line 109 of file LISACODE-GWMono.cpp.

References Freq.

10.12.3.20 void GW::setLambda (double *Lambda_n*) [inherited]

Sets Lambda attribute.

Input is checked:

$$\lambda_n \in [0, 2 \cdot \pi]$$

CalculDirProp method is called to set DirProp attribute

Definition at line 115 of file LISACODE-GW.cpp.

References GW::CalculDirProp(), and GW::Lambda.

10.12.3.21 void GWMono::setPhi0hc (double *Phi0hc_n*) [inline]

Sets Phi0hc attribute.

Definition at line 94 of file LISACODE-GWMono.h.

References Phi0hc.

10.12.3.22 void GWMono::setPhi0hp (double *Phi0hp_n*) [inline]

Sets Phi0hp attribute.

Definition at line 90 of file LISACODE-GWMono.h.

References Phi0hp.

10.12.4 Member Data Documentation

10.12.4.1 double GWMono::Amplhc [protected]

h_cross polarisation component amplitude

Definition at line 43 of file LISACODE-GWMono.h.

Referenced by getAmplhc(), GWMono(), hc(), and setAmplhc().

10.12.4.2 double GWMono::Amplhp [protected]

h_plus polarisation component amplitude

Definition at line 41 of file LISACODE-GWMono.h.

Referenced by getAmplhp(), GWMono(), hp(), and setAmplhp().

10.12.4.3 double GWMono::AnglPol [protected]

Polarisation angle (rad).

Angle between the projection of x (vernal point direction) in the wave frame and the polarisation vector

Reimplemented from GW.

Definition at line 49 of file LISACODE-GWMono.h.

Referenced by getAnglPol(), and GWMono().

10.12.4.4 GW::Beta [protected, inherited]

Source direction angle (in the heliocentric reference frame) in radians.

Referenced by GW::CalculDirProp(), GW::getBeta(), GW::GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), GW::setBeta(), and GW::setDirProp().

10.12.4.5 vector<double> GW::DirProp [protected, inherited]

Source direction unit vector (in the heliocentric reference frame).

Definition at line 46 of file LISACODE-GW.h.

Referenced by GW::CalculDirProp(), GW::getDirProp(), GW::GW(), and GW::setDirProp().

10.12.4.6 double GWMono::Freq [protected]

Frequency.

Definition at line 39 of file LISACODE-GWMono.h.

Referenced by getFreq(), GWMono(), hc(), hp(), and setFreq().

10.12.4.7 GW::Lambda [protected, inherited]

Source direction angle (in the heliocentric reference frame) in radians.

Referenced by GW::CalculDirProp(), GW::getLambda(), GW::GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), GW::setDirProp(), and GW::setLambda().

10.12.4.8 double GWMono::Phi0hc [protected]

Initial phase of h_cross polarisation component.

Definition at line 54 of file LISACODE-GWMono.h.

Referenced by getPhi0hc(), GWMono(), hc(), and setPhi0hc().

10.12.4.9 double GWMono::Phi0hp [protected]

Initial phase of h_plus polarisation component.

Definition at line 52 of file LISACODE-GWMono.h.

Referenced by getPhi0hp(), GWMono(), hp(), and setPhi0hp().

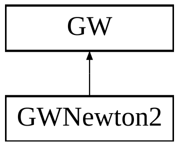
The documentation for this class was generated from the following files:

- LISACODE-GWMono.h
- LISACODE-GWMono.cpp

10.13 GWNewton2 Class Reference

```
#include <LISACODE-GWNewton2.h>
```

Inheritance diagram for GWNewton2::



10.13.1 Detailed Description

Gravitational Waves binary system parameters computation.

h_plus and h_cross polarisation components are computed at 1 or 2.5 Post-Newtonian approximation order.

Definition at line 38 of file LISACODE-GWNewton2.h.

Public Member Functions

- GWNewton2 ()

  Constructs an instance and initializes it with default values.

- GWNewton2 (double Beta_n, double Lambda_n, double AnglPol_n, int ttype, double mm1, double mm2, double ttcoal, double iinc, double pphcoal, double rrdist, double ttaud0, double oomega0, double ggw)

  Constructs an instance and initializes it with default values and inputs.

- ~GWNewton2 ()

  Destructor.

- double getM1 ()

  Returns m1 attribute.

- double getM2 ()

  Returns m2 attribute.

- double getTcoal ()

  Returns tcoal attribute.

- double getInc ()

  Returns inc attribute.

- double getPhCoal ()

  Returns phcoal attribute.

- double getDistance ()

  Returns rdist attribute.

- void setM1 (double m1_n)

  Sets m1 attribute.

- void setM2 (double m2_n)

  Sets m2 attribute.

- void setTcoal (double tcoal_n)

  Sets tcoal attribute.

- void setInc (double inc_n)

  Qets inc attribute.

- void setPhCoal (double phcoal_n)

  Sets phcoal attribute.

- void setDistance (double rdist_n)

  Sets rdist attribute.

- double hp (double t)

  Gives hp component depending on temps *input time (s).*

- double hc (double t)

  Gives hc component depending on temps input time (s).

- Couple hbin (double t)

  Gives (hp,hc) components depending on t *input time (s).*

- double fe (double t)

  Returns frequency depending on temps *input time.*

- double phase (double t)

  Returns phase depending on temps *input time.*

- double getBeta () const

  Returns Beta attribute.

- void setBeta (double Beta_n)

  Sets Beta and DirProp attributes.

- double getLambda () const

  Returns Lambda attribute.

- void setLambda (double Lambda_n)

  Sets Lambda attribute.

- vector< double > getDirProp () const

  Returns DirProp attribute.

- void setDirProp (vector< double > DirProp_n)

Sets DirProp attribute.

- double getAnglPol ()

  Returns AnglPol attribute.

- void setAnglPol (double AnglPol_n)

  Sets AnglPol attribute.

- void CalculDirProp ()

  Computes DirProp attribute components, depending on Lambda and Beta attributes.

Protected Member Functions

- void commun (double ttime)

  sets phi, omega, hint and time_encour attributes depending on temps *input time.*

Protected Attributes

- int type

  Computation order type.

- double m1

  First star mass (in solar masses).

- double m2

  Second star mass (in solar masses).

- double tcoal

  Time before the coalescence.

- double inc

  inclination angle ($[0, 2 \cdot \pi[$)

- double phcoal

  Phase before the coalescence.

- double rdist

  Distance to the source in kpc(kiloparsec).

- double mtot

  Total mass.

- double deltam

  Mass difference.

- double mu

  Reduced mass.

- double nu

  Mass ratio.

- double ci

  cos(inc)

- double si

  sin(inc)

- double cmass

  Chirp mass.

- double time_encour

  Current computation time.

- double hint

  1PN amplitude (depending on time)

- double omega

  Pulsation (depending on time).

- double phi

  Phase (depending on time).

- double teta

  System constant.

- double lambda

  System constant.

- double taud

  2.5 PN approximation parameter.

- double taud0

  System parameter.

- double omega0

  System parameter.

- double gw

  System parameter.

- double psi

  2.5 PN parameter (depending on time).

- double a1

  2.5 PN approximation parameter.

- double a2

2.5 PN approximation parameter.

- double a3

  2.5 PN approximation parameter.

- double a4

  2.5 PN approximation parameter.

- double a5

  2.5 PN approximation parameter.

- double a6

  2.5 PN approximation parameter.

- double b1

  2.5 PN approximation parameter.

- double b2

  2.5 PN approximation parameter.

- double b3

  2.5 PN approximation parameter.

- double b4

  2.5 PN approximation parameter.

- double a11

  2.5 PN approximation parameter.

- double a22

  2.5 PN approximation parameter.

- double b11

  2.5 PN approximation parameter.

- double c1

  2.5 PN approximation parameter.

- double c2

  2.5 PN approximation parameter.

- double c3

  2.5 PN approximation parameter.

- double d1

  2.5 PN approximation parameter.

- double d2

  2.5 PN approximation parameter.

- double e1

  2.5 PN approximation parameter.

- double e2

  2.5 PN approximation parameter.

- double e3

  2.5 PN approximation parameter.

- double e4

  2.5 PN approximation parameter.

- double e5

  2.5 PN approximation parameter.

- double a7

  2.5 PN approximation parameter.

- double f1

  2.5 PN approximation parameter.

- double f3

  2.5 PN approximation parameter.

- double f5

  2.5 PN approximation parameter.

- double f6

  2.5 PN approximation parameter.

- double f7

  2.5 PN approximation parameter.

- double f8

  2.5 PN approximation parameter.

- double f9

  2.5 PN approximation parameter.

- double f10

  2.5 PN approximation parameter.

- double a1x

  2.5 PN approximation parameter.

- double b1x

  2.5 PN approximation parameter.

- double c1x

  2.5 PN approximation parameter.

- double c2x

  2.5 PN approximation parameter.

- double d1x

  2.5 PN approximation parameter.

- double d2x

  2.5 PN approximation parameter.

- double d3x

  2.5 PN approximation parameter.

- double d4x

  2.5 PN approximation parameter.

- double d5x

  2.5 PN approximation parameter.

- double e1x

  2.5 PN approximation parameter.

- double e2x

  2.5 PN approximation parameter.

- double e3x

  2.5 PN approximation parameter.

- double e4x

  2.5 PN approximation parameter.

- double e5x

  2.5 PN approximation parameter.

- double e6x

  2.5 PN approximation parameter.

- double e7x

  2.5 PN approximation parameter.

- double e8x

  2.5 PN approximation parameter.

- double g1

  2.5 PN approximation parameter.

- double Beta

  Source direction angle (in the heliocentric reference frame) in radians.

- double Lambda

Source direction angle (in the heliocentric reference frame) in radians.

- vector< double > DirProp

  Source direction unit vector (in the heliocentric reference frame).

- double AnglPol

  Polarisation angle.

10.13.2 Constructor & Destructor Documentation

10.13.2.1 GWNewton2::GWNewton2 ()

Constructs an instance and initializes it with default values.

$$\beta = 0.917311 - \frac{\pi}{2} \text{ rad}$$

$$\lambda = 2.97378 - \pi \text{ rad}$$

$$AnglPol = 2.46531 \text{ rad}$$

$$type = 1$$

$$m_1 = 10^5 \cdot M_{Sun} \text{ kg}$$

$$m_2 = 10^5 \cdot M_{Sun} \text{ kg, where } M_{Sun} = 1.9889 \cdot 10^{30} \text{ kg}$$

$$rdist = 10^6 \cdot kpc_m \text{ m, where } kpc_m = 3.0856675807 \cdot 10^{19} \text{ m}$$

$$\phi_{coal} = 0 \text{ rad}$$

$$\tau_{d0} = 0$$

$$\omega_0 = 0$$

$$gw = 1$$

$$inc = \frac{\pi}{2} \text{ rad}$$

$$t_{coal} = 60000 \text{ s}$$

$$m_{tot} = m1 + m2$$

$$deltam = m1 - m2$$

$$\mu = \frac{m_1 * m_2}{m_1 + m_2}$$

$$\nu = \frac{\mu}{m_{tot}}$$

$$ci = cos(inc)$$

$$si = sin(inc)$$

$$c_{mass} = \mu^{\frac{3}{5}} \cdot m_{tot}^{\frac{2}{5}}$$

$$time_{encour} = -1000$$

- If (type=2)

$$\theta = -\frac{11831}{9240}$$

$$\lambda = \frac{3}{7} \cdot (\theta - \frac{1039}{4620})$$

$$a1 = \frac{55 \cdot \nu}{96} + \frac{3715}{8064}$$

$$a2 = -\frac{3 \cdot \pi}{4}$$

$$a3 = \frac{9275495}{14450688} + \frac{284875}{258048} \cdot \nu + \frac{1855}{2048}) \cdot \nu^2$$

$$a4 = (-\frac{38645}{172032} + \frac{65}{2048} \cdot \nu) \cdot \pi$$

$$a5 = \frac{831032450749357}{57682522275840} - \frac{53}{40} \cdot \pi^2 - \frac{107}{56} \cdot CE_{RG} + \frac{107}{448} \text{ where } CE_{RG} \text{ is Euler constant}$$

$$a6 = (-\frac{123292747421}{4161798144} + \frac{2255}{2048} \cdot \pi^2 + \frac{385}{48} \cdot \lambda - \frac{55}{16} \cdot \theta) \cdot \nu + \frac{154565}{1835008} \cdot \nu^2 - \frac{1179625}{1769472} \cdot \nu^3$$

$$a7 = (\frac{188516689}{173408256} + \frac{488825}{516096} \cdot \nu - \frac{141769}{516096} \cdot \nu^2) \cdot \pi$$

$$b1 = \frac{C^3}{8 \cdot G \cdot mtot}$$

$$b2 = \frac{743}{2688} + \frac{11}{32} \cdot \nu$$

$$b3 = -\frac{3 \cdot \pi}{10}$$

$$b4 = \frac{1855099}{14450688} + \frac{56975}{258048} \cdot \nu + \frac{371}{2048} \cdot \nu^2$$

$$a11 = \frac{2 \cdot G \cdot mtot \cdot \nu}{rdist \cdot C^2}$$

$$a22 = \frac{G \cdot mtot}{C^3}$$

$$b11 = -(1 + ci^2)$$

$$c1 = -\frac{si}{8} \cdot \frac{deltam}{mtot}$$

$$c2 = 5 + ci^2$$

$$c3 = 9 \cdot (1 + ci^2)$$

$$d1 = \frac{1}{6} \cdot (19 + 9 \cdot ci^2 - 2 \cdot ci^4 - \nu \cdot (19 - 11 \cdot ci^2 - 6 \cdot ci^4))$$

$$d2 = \frac{4}{3} \cdot si^2 \cdot (1 + ci^2) \cdot (1 - 3 \cdot \nu)$$

$$e1 = \frac{si}{192} \cdot \frac{deltam}{mtot}$$

$$e2 = (57 + 60 \cdot ci^2 - ci^4) - 2 \cdot \nu \cdot (49 - 12 \cdot ci^2 - ci^4)$$

$$e3 = \frac{27}{2} \cdot ((73 + 40 \cdot ci^2 - 9 \cdot ci^4) - 2 \cdot \nu \cdot (25 - 8 \cdot ci^2 - 9 \cdot ci^4))$$

$$e4 = \frac{625}{2} \cdot (1 - 2 \cdot \nu) \cdot si^2 \cdot (1 + ci^2)$$

$$e5 = 2 \cdot \pi \cdot (1 + ci^2)$$
$$f1 = \frac{1}{120} \cdot (22+396 \cdot ci^2+145 \cdot ci^4-5 \cdot ci^6+\frac{5}{3} \cdot \nu \cdot (706-216 \cdot ci^2-251 \cdot ci^4+15 \cdot ci^6)-5 \cdot \nu^2 \cdot (98-108 \cdot ci^2+7 \cdot ci^4+5 \cdot ci^6))$$
$$f3 = \frac{2}{15} \cdot si^2 \cdot ((59+35 \cdot ci^2-8 \cdot ci^4)-\frac{5}{3} \cdot \nu \cdot (131+59 \cdot ci^2-24 \cdot ci^4)+5 \cdot \nu^2 \cdot (21-3 \cdot ci^2-8 \cdot ci^4))$$
$$f5 = \frac{81}{40} \cdot (1 - 5 \cdot \nu + 5 \cdot \nu^2) \cdot si^4 \cdot (1 + ci^2)$$
$$f6 = \frac{si}{40} \cdot \frac{deltam}{mtot}$$
$$f7 = 11 + 7 \cdot ci^2 + 10 \cdot (5 + ci^2) \cdot log(2)$$
$$f8 = 5 \cdot \pi \cdot (5 + ci^2)$$
$$f9 = 27 \cdot (7 - 10 \cdot log(\frac{3}{2})) \cdot (1 + ci^2)$$
$$f10 = 135 \cdot \pi \cdot (1 + ci^2)$$
$$a1x = -2 \cdot ci$$
$$b1x = -\frac{3}{4} \cdot si \cdot ci \cdot \frac{deltam}{mtot}$$
$$c1x = \frac{ci}{3} \cdot ((17 - 4 \cdot ci^2) - \cdot \nu \cdot (13 - 12 \cdot ci^2))$$
$$c2x = -\frac{8}{3} \cdot (1 - 3 \cdot \nu) \cdot ci \cdot si^2$$
$$d1x = \frac{si \cdot ci}{96} \cdot \frac{deltam}{mtot}$$
$$d2x = 63 - 5 \cdot ci^2 - 2 \cdot \nu \cdot (23 - 5 \cdot ci^2)$$
$$d3x = -\frac{27}{2} \cdot (67 - 15 \cdot ci^2 - 2 \cdot \nu \cdot (19 - 15 \cdot ci^2))$$
$$d4x = \frac{625}{2} \cdot (1 - 2 \cdot \nu) \cdot si^2$$
$$d5x = -4 \cdot \pi \cdot ci$$
$$e1x = \frac{ci}{60} \cdot (68+226 \cdot ci^2-15 \cdot ci^4+\frac{5}{3} \cdot \nu \cdot (572-490 \cdot ci^2+45 \cdot ci^4)-5 \cdot \nu^2 \cdot (56-70 \cdot ci^2+15 \cdot ci^4))$$
$$e2x = \frac{4}{15} \cdot ci \cdot si^2 \cdot (55 - 12 \cdot ci^2 - \frac{5}{3} \cdot \nu \cdot (119 - 36 \cdot ci^2) + 5 \cdot \nu^2 \cdot (17 - 12 \cdot ci^2))$$
$$e3x = -\frac{81}{20} \cdot (1 - 5 \cdot \nu + 5 \cdot \nu^2) \cdot ci \cdot si^4$$
$$e4x = -\frac{3}{20} \cdot si \cdot ci \cdot \frac{deltam}{mtot}$$
$$e5x = 3 + 10 \cdot log(2)$$
$$e6x = 5 \cdot \pi$$
$$e7x = -9 \cdot (7 - 10 \cdot log(\frac{3}{2}))$$
$$e8x = -45 \cdot \pi$$
$$g1 = \frac{\cdot G \cdot mtot}{C^3}$$

Definition at line 107 of file Ondes_Gravit/SRC/LISACODE-GWNewton2.cpp.

References a1, a11, a1x, a2, a22, a3, a4, a5, a6, a7, GW::AnglPol, b1, b11, b1x, b2, b3, b4, GW::Beta, c1, c1x, c2, c2x, c3, c_SI, CE_RG, ci, cmass, d1, d1x, d2, d2x, d3x, d4x, d5x, deltam, e1, e1x, e2, e2x, e3, e3x, e4, e4x, e5, e5x, e6x, e7x, e8x, f1, f10, f3, f5, f6, f7, f8, f9, g1, G_SI, gw, inc, kpc_m, lambda, GW::Lambda, m1, m2, MS_SI, mtot, mu, nu, omega0, phcoal, rdist, si, taud0, tcoal, teta, time_encour, and type.

10.13.2.2 GWNewton2::GWNewton2 (double *Beta_n*, double *Lambda_n*, double *AnglPol_n*, int *ttype*, double *mm1*, double *mm2*, double *ttcoal*, double *iinc*, double *pphcoal*, double *rrdist*, double *ttaud0*, double *oomega0*, double *ggw*)

Constructs an instance and initializes it with default values and inputs.

- Beta= Beta_n (using GW constructor)
- Lambda = Lambda_n (using GW constructor)
- AnglPol = AnglPol_n (using GW constructor)
- type = ttype

$$m1 = mm1 \cdot M_{Sun} \text{ kg, where } M_{Sun} = 1.9889 \cdot 10^{30} \text{ kg}$$

$$m2 = mm2 \cdot M_{Sun} \text{ kg, where } M_{Sun} = 1.9889 \cdot 10^{30} \text{ kg}$$

$$rdist = \cdot kpc_m \text{ m, where } kpc_m = 3.0856675807 \cdot 10^{19} \text{ m}$$

- phcoal = pphcoal
- taud0 = ttaud0
- omega0 = oomega0
- gw = ggw
- inc = iinc
- tcoal = ttcoal

$$m_{tot} = m1 + m2$$

$$deltam = m1 - m2$$

$$\mu = \frac{m_1 * m_2}{m_1 + m_2}$$

$$\nu = \frac{\mu}{m_{tot}}$$

$$ci = cos(inc)$$

$$si = sin(inc)$$

$$c_{mass} = \mu^{\frac{3}{5}} \cdot m_{tot}^{\frac{2}{5}}$$

$$time_encour = -1000$$

- If (type=2)

$$\theta = -\frac{11831}{9240}$$

$$\lambda = \frac{3}{7} \cdot (\theta - \frac{1039}{4620})$$

$$a1 = \frac{55 \cdot \nu}{96} + \frac{3715}{8064}$$

$$a2 = -\frac{3 \cdot \pi}{4}$$

$$a3 = \frac{9275495}{14450688} + \frac{284875}{258048} \cdot \nu + \frac{1855}{2048}) \cdot \nu^2$$

$$a4 = (-\frac{38645}{172032} + \frac{65}{2048} \cdot \nu) \cdot \pi$$

$$a5 = \frac{831032450749357}{57682522275840} - \frac{53}{40} \cdot \pi^2 - \frac{107}{56} \cdot CE_{RG} + \frac{107}{448} \text{ where } CE_{RG} \text{ is Euler constant}$$

$$a6 = (-\frac{123292747421}{4161798144} + \frac{2255}{2048} \cdot \pi^2 + \frac{385}{48} \cdot \lambda - \frac{55}{16} \cdot \theta) \cdot \nu + \frac{154565}{1835008} \cdot \nu^2 - \frac{1179625}{1769472} \cdot \nu^3$$

$$a7 = (\frac{188516689}{173408256} + \frac{488825}{516096} \cdot \nu - \frac{141769}{516096} \cdot \nu^2) \cdot \pi$$

$$b1 = \frac{C^3}{8 \cdot G \cdot mtot}$$

$$b2 = \frac{743}{2688} + \frac{11}{32} \cdot \nu$$

$$b3 = -\frac{3 \cdot \pi}{10}$$

$$b4 = \frac{1855099}{14450688} + \frac{56975}{258048} \cdot \nu + \frac{371}{2048} \cdot \nu^2$$

$$a11 = \frac{2 \cdot G \cdot mtot \cdot \nu}{rdist \cdot C^2}$$

$$a22 = \frac{G \cdot mtot}{C^3}$$

$$b11 = -(1 + ci^2)$$

$$c1 = -\frac{si}{8} \cdot \frac{deltam}{mtot}$$

$$c2 = 5 + ci^2$$

$$c3 = 9 \cdot (1 + ci^2)$$

$$d1 = \frac{1}{6} \cdot (19 + 9 \cdot ci^2 - 2 \cdot ci^4 - \nu \cdot (19 - 11 \cdot ci^2 - 6 \cdot ci^4))$$

$$d2 = \frac{4}{3} \cdot si^2 \cdot (1 + ci^2) \cdot (1 - 3 \cdot \nu)$$

$$e1 = \frac{si}{192} \cdot \frac{deltam}{mtot}$$

$$e2 = (57 + 60 \cdot ci^2 - ci^4) - 2 \cdot \nu \cdot (49 - 12 \cdot ci^2 - ci^4)$$

$$e3 = \frac{27}{2} \cdot ((73 + 40 \cdot ci^2 - 9 \cdot ci^4) - 2 \cdot \nu \cdot (25 - 8 \cdot ci^2 - 9 \cdot ci^4))$$

$$e4 = \frac{625}{2} \cdot (1 - 2 \cdot \nu) \cdot si^2 \cdot (1 + ci^2)$$

$$e5 = 2 \cdot \pi \cdot (1 + ci^2)$$

$$f1 = \frac{1}{120} \cdot (22 + 396 \cdot ci^2 + 145 \cdot ci^4 - 5 \cdot ci^6 + \frac{5}{3} \cdot \nu \cdot (706 - 216 \cdot ci^2 - 251 \cdot ci^4 + 15 \cdot ci^6) - 5 \cdot \nu^2 \cdot (98 - 108 \cdot ci^2 + 7 \cdot ci^4 + 5 \cdot ci^6))$$

$$f3 = \frac{2}{15} \cdot si^2 \cdot ((59 + 35 \cdot ci^2 - 8 \cdot ci^4) - \frac{5}{3} \cdot \nu \cdot (131 + 59 \cdot ci^2 - 24 \cdot ci^4) + 5 \cdot \nu^2 \cdot (21 - 3 \cdot ci^2 - 8 \cdot ci^4))$$

$$f5 = \frac{81}{40} \cdot (1 - 5 \cdot \nu + 5 \cdot \nu^2) \cdot si^4 \cdot (1 + ci^2)$$

$$f6 = \frac{si}{40} \cdot \frac{deltam}{mtot}$$

$$f7 = 11 + 7 \cdot ci^2 + 10 \cdot (5 + ci^2) \cdot log(2)$$

$$f8 = 5 \cdot \pi \cdot (5 + ci^2)$$

$$f9 = 27 \cdot (7 - 10 \cdot log(\frac{3}{2})) \cdot (1 + ci^2)$$

$$f10 = 135 \cdot \pi \cdot (1 + ci^2)$$

$$a1x = -2 \cdot ci$$

$$b1x = -\frac{3}{4} \cdot si \cdot ci \cdot \frac{deltam}{mtot}$$

$$c1x = \frac{ci}{3} \cdot ((17 - 4 \cdot ci^2) - \cdot \nu \cdot (13 - 12 \cdot ci^2))$$

$$c2x = -\frac{8}{3} \cdot (1 - 3 \cdot \nu) \cdot ci \cdot si^2$$

$$d1x = \frac{si \cdot ci}{96} \cdot \frac{deltam}{mtot}$$

$$d2x = 63 - 5 \cdot ci^2 - 2 \cdot \nu \cdot (23 - 5 \cdot ci^2)$$

$$d3x = -\frac{27}{2} \cdot (67 - 15 \cdot ci^2 - 2 \cdot \nu \cdot (19 - 15 \cdot ci^2))$$

$$d4x = \frac{625}{2} \cdot (1 - 2 \cdot \nu) \cdot si^2$$

$$d5x = -4 \cdot \pi \cdot ci$$

$$e1x = \frac{ci}{60} \cdot (68 + 226 \cdot ci^2 - 15 \cdot ci^4 + \frac{5}{3} \cdot \nu \cdot (572 - 490 \cdot ci^2 + 45 \cdot ci^4) - 5 \cdot \nu^2 \cdot (56 - 70 \cdot ci^2 + 15 \cdot ci^4))$$

$$e2x = \frac{4}{15} \cdot ci \cdot si^2 \cdot (55 - 12 \cdot ci^2 - \frac{5}{3} \cdot \nu \cdot (119 - 36 \cdot ci^2) + 5 \cdot \nu^2 \cdot (17 - 12 \cdot ci^2))$$

$$e3x = -\frac{81}{20} \cdot (1 - 5 \cdot \nu + 5 \cdot \nu^2) \cdot ci \cdot si^4$$

$$e4x = -\frac{3}{20} \cdot si \cdot ci \cdot \frac{deltam}{mtot}$$

$$e5x = 3 + 10 \cdot log(2)$$

$$e6x = 5 \cdot \pi$$

$$e7x = -9 \cdot (7 - 10 \cdot log(\frac{3}{2}))$$

$$e8x = -45 \cdot \pi$$

$$g1 = \frac{\cdot G \cdot mtot}{C^3}$$

Definition at line 303 of file Ondes_Gravit/SRC/LISACODE-GWNewton2.cpp.

References a1, a11, a1x, a2, a22, a3, a4, a5, a6, a7, b1, b11, b1x, b2, b3, b4, c1, c1x, c2, c2x, c3, c_SI, CE_RG, ci, cmass, d1, d1x, d2, d2x, d3x, d4x, d5x, deltam, e1, e1x, e2, e2x, e3, e3x, e4, e4x, e5, e5x, e6x, e7x, e8x, f1, f10, f3, f5, f6, f7, f8, f9, g1, G_SI, gw, inc, kpc_m, lambda, m1, m2, MS_SI, mtot, mu, nu, omega0, phcoal, rdist, si, taud0, tcoal, teta, time_encour, and type.

10.13.2.3 GWNewton2::∼GWNewton2 ()

Destructor.

Definition at line 420 of file Ondes_Gravit/SRC/LISACODE-GWNewton2.cpp.

10.13.3 Member Function Documentation

10.13.3.1 void GW::CalculDirProp () [inherited]

Computes DirProp attribute components, depending on Lambda and Beta attributes.

$$DirProp = \begin{pmatrix} -cos(\lambda) \cdot cos(\beta) \\ -sin(\lambda) \cdot cos(\beta) \\ -sin(\beta) \end{pmatrix}$$

Definition at line 203 of file LISACODE-GW.cpp.

References GW::Beta, GW::DirProp, and GW::Lambda.

Referenced by GW::GW(), GWBinary::GWBinary(), GW::setBeta(), and GW::setLambda().

10.13.3.2 void GWNewton2::commun (double *temps*) [protected]

sets phi, omega, hint and time_encour attributes depending on *temps* input time.

If *temps* $\neq$ time_encour , time dependant attributes must be updated and time_encour is set to *temps* input :

- if type=1

$$\phi = \phi_{coal} - \left(C^3 * \frac{t_{coal} - temps}{5 \cdot G \cdot cmass} \right)^{\frac{5}{8}}$$

$$\omega = \frac{C^3}{8 \cdot G \cdot cmass} \cdot * \left(\frac{t_{coal} - temps}{5 \cdot G \cdot cmass} \right)^{-\frac{3}{8}}$$

$$hint = -\frac{2 \cdot G \cdot mtot \cdot \nu}{C^2 \cdot rdist} \cdot \left(\frac{G \cdot mtot \cdot \omega}{C^3} \right)^{\frac{2}{3}}$$

- if type=2

$$\tau_d = \nu \cdot C^3 \cdot \frac{t_{coal} - temps}{5 \cdot G \cdot mtot}$$

$$\phi = \frac{-1}{\nu} \left(\tau_d^{\frac{5}{8}} + a1 \cdot \tau_d^{\frac{3}{8}} + a2 \cdot \tau_d^{\frac{1}{4}} + a3 \cdot \tau_d^{\frac{1}{8}} + a4 \cdot log(\frac{\tau_d}{\tau_{d0}}) + \left(a5 \cdot log(\frac{\tau_d}{256}) + a6 \right) \cdot \tau_d^{-\frac{1}{4}} + a7 \cdot \tau_d^{-\frac{1}{4}} \right)$$

$$\omega = b1 \cdot (\tau_d^{-\frac{3}{8}} + b2 \cdot \tau_d^{-\frac{5}{8}} + b3 \cdot \tau_d^{-\frac{4}{8}} + b4 \cdot \tau_d^{-\frac{7}{8}})$$

$$\psi = \phi - \frac{2 \cdot G \cdot mtot \cdot \omega}{C^3} \cdot log(\frac{\omega}{\omega_0})$$

Definition at line 513 of file Ondes_Gravit/SRC/LISACODE-GWNewton2.cpp.

References a1, a2, a3, a4, a5, a6, a7, b1, b2, b3, b4, c_SI, cmass, G_SI, hint, mtot, nu, omega, omega0, phcoal, phi, PRECISION, psi, rdist, taud, taud0, tcoal, time_encour, and type.

Referenced by fe(), hbin(), hc(), hp(), and phase().

10.13.3.3 double GWNewton2::fe (double *temps*)

Returns frequency depending on *temps* input time.

commun method is called to update time depending attributes

If *temps* $>$ tcoal -100, fe=0, else :

$$fe = \frac{\omega}{\pi}$$

Definition at line 690 of file Ondes_Gravit/SRC/LISACODE-GWNewton2.cpp.

References commun(), fe(), omega, and tcoal.

Referenced by fe().

10.13.3.4 double GW::getAnglPol () `[inline, inherited]`

Returns AnglPol attribute.

Definition at line 70 of file LISACODE-GW.h.

References GW::AnglPol.

10.13.3.5 double GW::getBeta () const `[inline, inherited]`

Returns Beta attribute.

Definition at line 59 of file LISACODE-GW.h.

References GW::Beta.

10.13.3.6 vector<double> GW::getDirProp () const `[inline, inherited]`

Returns DirProp attribute.

Definition at line 66 of file LISACODE-GW.h.

References GW::DirProp.

10.13.3.7 double GWNewton2::getDistance () `[inline]`

Returns rdist attribute.

Definition at line 246 of file LISACODE-GWNewton2.h.

References rdist.

10.13.3.8 double GWNewton2::getInc () `[inline]`

Returns inc attribute.

Definition at line 242 of file LISACODE-GWNewton2.h.

References inc.

10.13.3.9 double GW::getLambda () const `[inline, inherited]`

Returns Lambda attribute.

Definition at line 62 of file LISACODE-GW.h.

References GW::Lambda.

10.13.3.10 double GWNewton2::getM1 () [inline]

Returns m1 attribute.

Definition at line 236 of file LISACODE-GWNewton2.h.

References m1.

10.13.3.11 double GWNewton2::getM2 () [inline]

Returns m2 attribute.

Definition at line 238 of file LISACODE-GWNewton2.h.

References m2.

10.13.3.12 double GWNewton2::getPhCoal () [inline]

Returns phcoal attribute.

Definition at line 244 of file LISACODE-GWNewton2.h.

References phcoal.

10.13.3.13 double GWNewton2::getTcoal () [inline]

Returns tcoal attribute.

Definition at line 240 of file LISACODE-GWNewton2.h.

References tcoal.

10.13.3.14 Couple GWNewton2::hbin (double *t*)

Gives (hp,hc) components depending on *t* input time (s).

hp and hc methods are called.

Returned value is (hp,hc) couple

Definition at line 672 of file Ondes_Gravit/SRC/LISACODE-GWNewton2.cpp.

References commun(), hbin(), hc(), hp(), Couple::x, and Couple::y.

Referenced by hbin().

10.13.3.15 double GWNewton2::hc (double *temps*) [virtual]

Gives hc component depending on temps input time (s).

commun method is called to update time depending attributes

if temps $>$ tcoal -100, hc=0, else :

- if type=1

$$hc = hint \cdot 2 \cdot ci \cdot \sin(2 \cdot \phi)$$

- if type=2

$$
\begin{gathered}
hc = a11 \cdot (a22 \cdot \omega)^{\frac{2}{3}} \cdot \\
(a1x \cdot \sin(2 \cdot \psi) \\
+\left(\frac{g1 \cdot \omega}{gw}\right)^{\frac{1}{3}} (b1x \cdot (\sin(\psi) - 3 \cdot \sin(3 \cdot \psi))) \\
+\left(\frac{g1 \cdot \omega}{gw}\right)^{\frac{2}{3}} \cdot (c1x \cdot \sin(2 \cdot \psi) + c2x \cdot \sin(4 \cdot \psi)) \\
+\frac{g1 \cdot \omega}{gw} (d1x \cdot (d2x \cdot \sin(\psi) + d3x \cdot \sin(3 \cdot \psi) + d4x \cdot \sin(5 \cdot \psi)) + d5x \cdot \sin(2 \cdot \psi)) \\
+\left(\frac{g1 \cdot \omega}{gw}\right)^{\frac{4}{3}} \cdot (e1x \sin(2 \cdot \psi) + e2x \sin(4 \cdot \psi) + e3x \sin(6 \cdot \psi) \\
+e4x \cdot ((e5x \cdot \cos(\psi) + e6x \cdot \sin(\psi) + e7x) \cdots \sin(3 \cdot \psi) - e8x \sin(2 \cdot \psi))))
\end{gathered}
$$

Reimplemented from GW.

Definition at line 628 of file Ondes_Gravit/SRC/LISACODE-GWNewton2.cpp.

References a11, a1x, a22, b1x, c1x, c2x, ci, commun(), d1x, d2x, d3x, d4x, d5x, e1x, e2x, e3x, e4x, e5x, e6x, e7x, e8x, g1, gw, hc(), hint, omega, omega, phi, psi, tcoal, and type.

Referenced by hbin(), and hc().

10.13.3.16 double GWNewton2::hp (double *temps*) `[virtual]`

Gives hp component depending on *temps* input time (s).

commun method is called to update time depending attributes

if temps > tcoal -100, hp=0, else :

- if type=1

$$hp = hint \cdot (1 + ci^2) \cos(2 \cdot \phi)$$

- if type=2

$$
\begin{gathered}
hp = a11 \cdot (a22 \cdot \omega)^{\frac{2}{3}} \cdot (b11 \cdot \cos(2 \cdot \psi) \\
+c1 \cdot \left(\frac{g1 \cdot \omega}{gw}\right)^{\frac{1}{3}} \cdot (c2 \cdot \cos(\psi) - c3 \cdot \cos(3 \cdot \psi)) \\
+\left(\frac{g1 \cdot \omega}{gw}\right)^{\frac{2}{3}} \cdot (d1 \cdot \cos(2. \cdot \psi) - d2 \cdot \cos(4 \cdot \psi)) \\
+\frac{g1 \cdot \omega}{gw} \cdot (e1\,(e2 \cdot cos(\cdot \psi) - e3 \cdot cos(3 \cdot \psi) + e4 \cdot \cos(5 \cdot \psi)) - e5 \cdot \cos(2 \cdot \psi)) \\
+\left(\frac{g1 \cdot \omega}{gw}\right)^{\frac{2}{3}} \cdot (f1 \cdot \cos(2 \cdot \psi) + f3 \cdot \cos(4 \cdot \psi) - f5 \cdot \cos(6 \cdot \psi) \\
+f6 \cdot (f7 \sin(\psi) + f8 \cdot \cos(\psi) - f9 \cdot \sin(3 \cdot \psi) + f10 \cdot \cos(3 \cdot \psi))))
\end{gathered}
$$

Reimplemented from GW.

Definition at line 560 of file Ondes_Gravit/SRC/LISACODE-GWNewton2.cpp.

References a11, a22, b11, c1, c2, c3, ci, commun(), d1, d2, e1, e2, e3, e4, e5, f1, f10, f3, f5, f6, f7, f8, f9, g1, gw, hint, hp(), omega, omega, phi, psi, tcoal, and type.

Referenced by hbin(), and hp().

10.13.3.17 double GWNewton2::phase (double *temps*)

Returns phase depending on *temps* input time.

commun method is called to update time depending attributes

If *temps* > tcoal -100, fe=0, else :

$$fe = \frac{\omega}{\pi}$$

Definition at line 712 of file Ondes_Gravit/SRC/LISACODE-GWNewton2.cpp.

References commun(), phase(), psi, and tcoal.

Referenced by phase().

10.13.3.18 void GW::setAnglPol (double *AnglPol_n*) `[inherited]`

Sets AnglPol attribute.

Input is checked:

$$AnglPol_n \in [0, 2 \cdot \pi]$$

Reimplemented in GWMono.

Definition at line 168 of file LISACODE-GW.cpp.

References GW::AnglPol.

10.13.3.19 void GW::setBeta (double *Beta_n*) `[inherited]`

Sets Beta and DirProp attributes.

Input is checked:

$$\beta_n \in [-\frac{\pi}{2}, -\frac{\pi}{2}]$$

CalculDirProp method is called to fill DirProp attribute

Definition at line 100 of file LISACODE-GW.cpp.

References GW::Beta, and GW::CalculDirProp().

10.13.3.20 void GW::setDirProp (vector< double > *DirProp_n*) `[inherited]`

Sets DirProp attribute.

Input is checked: *DirProp_n* must be a 3 components vector.

Computations: $DirProp_{norm}$ is normalized DirProp_n

- $$\beta = asin(DirProp_{norm}[2])$$
- $$\lambda = mod(atan(\frac{-DirProp_{norm}[1]}{-DirProp_{norm}[0]}), 2 \cdot \pi)$$

Definition at line 133 of file LISACODE-GW.cpp.

References GW::Beta, GW::DirProp, GW::Lambda, and PRECISION.

10.13.3.21 void GWNewton2::setDistance (double *rdist_n*)

Sets rdist attribute.

Input is checked:

$$rdist_n \geq 0$$

Definition at line 486 of file Ondes_Gravit/SRC/LISACODE-GWNewton2.cpp.

References rdist.

10.13.3.22 void GWNewton2::setInc (double *inc_n*)

Qets inc attribute.

Input is checked:

$$0 \leq inc_n \leq 1 \cdot \pi$$

Definition at line 464 of file Ondes_Gravit/SRC/LISACODE-GWNewton2.cpp.

References inc.

10.13.3.23 void GW::setLambda (double *Lambda_n*) `[inherited]`

Sets Lambda attribute.

Input is checked:

$$\lambda_n \in [0, 2 \cdot \pi]$$

CalculDirProp method is called to set DirProp attribute

Definition at line 115 of file LISACODE-GW.cpp.

References GW::CalculDirProp(), and GW::Lambda.

10.13.3.24 void GWNewton2::setM1 (double *M1_n*)

Sets m1 attribute.

Input is checked:

$$M1_n \geq 0$$

Definition at line 431 of file Ondes_Gravit/SRC/LISACODE-GWNewton2.cpp.

References m1.

10.13.3.25 void GWNewton2::setM2 (double *M2_n*)

Sets m2 attribute.

Input is checked:

$$M2_n \geq 0$$

Definition at line 442 of file Ondes_Gravit/SRC/LISACODE-GWNewton2.cpp.

References m2.

10.13.3.26 void GWNewton2::setPhCoal (double *phcoal_n*)

Sets phcoal attribute.

Input is checked:

$$0 \leq phcoal_n \leq 1 \cdot \pi$$

Definition at line 475 of file Ondes_Gravit/SRC/LISACODE-GWNewton2.cpp.

References phcoal.

10.13.3.27 void GWNewton2::setTcoal (double *tcoal_n*)

Sets tcoal attribute.

Input is checked:

$$tcoal_n \geq 0$$

Definition at line 453 of file Ondes_Gravit/SRC/LISACODE-GWNewton2.cpp.

References tcoal.

10.13.4 Member Data Documentation

10.13.4.1 GWNewton2::a1 `[protected]`

2.5 PN approximation parameter.

Referenced by commun(), and GWNewton2().

10.13.4.2 GWNewton2::a11 `[protected]`

2.5 PN approximation parameter.

Referenced by GWNewton2(), hc(), and hp().

10.13.4.3 GWNewton2::a1x `[protected]`

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hc().

10.13.4.4 GWNewton2::a2 `[protected]`

2.5 PN approximation parameter.

Referenced by commun(), and GWNewton2().

10.13.4.5 GWNewton2::a22 `[protected]`

2.5 PN approximation parameter.

Referenced by GWNewton2(), hc(), and hp().

10.13.4.6 GWNewton2::a3 [protected]

2.5 PN approximation parameter.

Referenced by commun(), and GWNewton2().

10.13.4.7 GWNewton2::a4 [protected]

2.5 PN approximation parameter.

Referenced by commun(), and GWNewton2().

10.13.4.8 GWNewton2::a5 [protected]

2.5 PN approximation parameter.

Referenced by commun(), and GWNewton2().

10.13.4.9 GWNewton2::a6 [protected]

2.5 PN approximation parameter.

Referenced by commun(), and GWNewton2().

10.13.4.10 GWNewton2::a7 [protected]

2.5 PN approximation parameter.

Referenced by commun(), and GWNewton2().

10.13.4.11 double GW::AnglPol [protected, inherited]

Polarisation angle.

Reimplemented in GWMono, and GWPeriGate.

Definition at line 48 of file LISACODE-GW.h.

Referenced by GW::getAnglPol(), GW::GW(), GWBinary::GWBinary(), GWNewton2(), and GW::setAnglPol().

10.13.4.12 GWNewton2::b1 [protected]

2.5 PN approximation parameter.

Referenced by commun(), and GWNewton2().

10.13.4.13 GWNewton2::b11 [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hp().

10.13.4.14 GWNewton2::b1x [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hc().

10.13.4.15 GWNewton2::b2 [protected]

2.5 PN approximation parameter.

Referenced by commun(), and GWNewton2().

10.13.4.16 GWNewton2::b3 [protected]

2.5 PN approximation parameter.

Referenced by commun(), and GWNewton2().

10.13.4.17 GWNewton2::b4 [protected]

2.5 PN approximation parameter.

Referenced by commun(), and GWNewton2().

10.13.4.18 GW::Beta [protected, inherited]

Source direction angle (in the heliocentric reference frame) in radians.

Referenced by GW::CalculDirProp(), GW::getBeta(), GW::GW(), GWBinary::GWBinary(), GWNewton2(), GW::setBeta(), and GW::setDirProp().

10.13.4.19 GWNewton2::c1 [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hp().

10.13.4.20 GWNewton2::c1x [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hc().

10.13.4.21 GWNewton2::c2 [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hp().

10.13.4.22 GWNewton2::c2x [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hc().

10.13.4.23 GWNewton2::c3 [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hp().

10.13.4.24 double GWNewton2::ci [protected]

cos(inc)

Definition at line 77 of file LISACODE-GWNewton2.h.

Referenced by GWNewton2(), hc(), and hp().

10.13.4.25 double GWNewton2::cmass [protected]

Chirp mass.

Definition at line 81 of file LISACODE-GWNewton2.h.

Referenced by commun(), and GWNewton2().

10.13.4.26 GWNewton2::d1 [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hp().

10.13.4.27 GWNewton2::d1x [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hc().

10.13.4.28 GWNewton2::d2 [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hp().

10.13.4.29 GWNewton2::d2x [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hc().

10.13.4.30 GWNewton2::d3x [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hc().

10.13.4.31 GWNewton2::d4x [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hc().

10.13.4.32 GWNewton2::d5x [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hc().

10.13.4.33 double GWNewton2::deltam [protected]

Mass difference.

Definition at line 71 of file LISACODE-GWNewton2.h.

Referenced by GWNewton2().

10.13.4.34 vector<double> GW::DirProp [protected, inherited]

Source direction unit vector (in the heliocentric reference frame).

Definition at line 46 of file LISACODE-GW.h.

Referenced by GW::CalculDirProp(), GW::getDirProp(), GW::GW(), and GW::setDirProp().

10.13.4.35 GWNewton2::e1 [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hp().

10.13.4.36 GWNewton2::e1x [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hc().

10.13.4.37 GWNewton2::e2 [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hp().

10.13.4.38 GWNewton2::e2x [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hc().

10.13.4.39 GWNewton2::e3 `[protected]`

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hp().

10.13.4.40 GWNewton2::e3x `[protected]`

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hc().

10.13.4.41 GWNewton2::e4 `[protected]`

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hp().

10.13.4.42 GWNewton2::e4x `[protected]`

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hc().

10.13.4.43 GWNewton2::e5 `[protected]`

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hp().

10.13.4.44 GWNewton2::e5x `[protected]`

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hc().

10.13.4.45 GWNewton2::e6x `[protected]`

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hc().

10.13.4.46 GWNewton2::e7x `[protected]`

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hc().

10.13.4.47 GWNewton2::e8x `[protected]`

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hc().

10.13.4.48 GWNewton2::f1 [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hp().

10.13.4.49 GWNewton2::f10 [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hp().

10.13.4.50 GWNewton2::f3 [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hp().

10.13.4.51 GWNewton2::f5 [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hp().

10.13.4.52 GWNewton2::f6 [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hp().

10.13.4.53 GWNewton2::f7 [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hp().

10.13.4.54 GWNewton2::f8 [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hp().

10.13.4.55 GWNewton2::f9 [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), and hp().

10.13.4.56 GWNewton2::g1 [protected]

2.5 PN approximation parameter.

Referenced by GWNewton2(), hc(), and hp().

10.13.4.57 GWNewton2::gw [protected]

System parameter.

Referenced by GWNewton2(), hc(), and hp().

10.13.4.58 double GWNewton2::hint [protected]

1PN amplitude (depending on time)

Definition at line 85 of file LISACODE-GWNewton2.h.

Referenced by commun(), hc(), and hp().

10.13.4.59 double GWNewton2::inc [protected]

inclination angle ($[0, 2 \cdot \pi[$)

Definition at line 63 of file LISACODE-GWNewton2.h.

Referenced by getInc(), GWNewton2(), and setInc().

10.13.4.60 GW::Lambda [protected, inherited]

Source direction angle (in the heliocentric reference frame) in radians.

Referenced by GW::CalculDirProp(), GW::getLambda(), GW::GW(), GWBinary::GWBinary(), GWNewton2(), GW::setDirProp(), and GW::setLambda().

10.13.4.61 GWNewton2::lambda [protected]

System constant.

Referenced by GWNewton2().

10.13.4.62 double GWNewton2::m1 [protected]

First star mass (in solar masses).

Definition at line 57 of file LISACODE-GWNewton2.h.

Referenced by getM1(), GWNewton2(), and setM1().

10.13.4.63 double GWNewton2::m2 [protected]

Second star mass (in solar masses).

Definition at line 59 of file LISACODE-GWNewton2.h.

Referenced by getM2(), GWNewton2(), and setM2().

10.13.4.64 double GWNewton2::mtot [protected]

Total mass.

Definition at line 69 of file LISACODE-GWNewton2.h.

Referenced by commun(), and GWNewton2().

10.13.4.65 double GWNewton2::mu [protected]

Reduced mass.

Definition at line 73 of file LISACODE-GWNewton2.h.

Referenced by GWNewton2().

10.13.4.66 double GWNewton2::nu [protected]

Mass ratio.

Definition at line 75 of file LISACODE-GWNewton2.h.

Referenced by commun(), and GWNewton2().

10.13.4.67 double GWNewton2::omega [protected]

Pulsation (depending on time).

Definition at line 87 of file LISACODE-GWNewton2.h.

Referenced by commun(), fe(), hc(), and hp().

10.13.4.68 GWNewton2::omega0 [protected]

System parameter.

Referenced by commun(), and GWNewton2().

10.13.4.69 double GWNewton2::phcoal [protected]

Phase before the coalescence.

Definition at line 65 of file LISACODE-GWNewton2.h.

Referenced by commun(), getPhCoal(), GWNewton2(), and setPhCoal().

10.13.4.70 double GWNewton2::phi [protected]

Phase (depending on time).

Definition at line 89 of file LISACODE-GWNewton2.h.

Referenced by commun(), hc(), and hp().

10.13.4.71 GWNewton2::psi [protected]

2.5 PN parameter (depending on time).

Referenced by commun(), hc(), hp(), and phase().

10.13.4.72 double GWNewton2::rdist [protected]

Distance to the source in kpc(kiloparsec).

Definition at line 67 of file LISACODE-GWNewton2.h.

Referenced by commun(), getDistance(), GWNewton2(), and setDistance().

10.13.4.73 double GWNewton2::si [protected]

sin(inc)

Definition at line 79 of file LISACODE-GWNewton2.h.

Referenced by GWNewton2().

10.13.4.74 GWNewton2::taud [protected]

2.5 PN approximation parameter.

Referenced by commun().

10.13.4.75 GWNewton2::taud0 [protected]

System parameter.

Referenced by commun(), and GWNewton2().

10.13.4.76 double GWNewton2::tcoal [protected]

Time before the coalescence.

Definition at line 61 of file LISACODE-GWNewton2.h.

Referenced by commun(), fe(), getTcoal(), GWNewton2(), hc(), hp(), phase(), and setTcoal().

10.13.4.77 GWNewton2::teta [protected]

System constant.

Referenced by GWNewton2().

10.13.4.78 double GWNewton2::time_encour [protected]

Current computation time.

Definition at line 83 of file LISACODE-GWNewton2.h.

Referenced by commun(), and GWNewton2().

10.13.4.79 int GWNewton2::type [protected]

Computation order type.

If type=1, order=1; if type=2, order=2

Definition at line 55 of file LISACODE-GWNewton2.h.

Referenced by commun(), GWNewton2(), hc(), and hp().

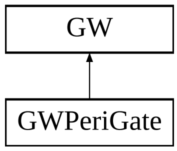
The documentation for this class was generated from the following files:

- LISACODE-GWNewton2.h
- Ondes_Gravit/SRC/LISACODE-GWNewton2.cpp
- Orbitographie/SRC/LISACODE-GWNewton2.cpp

10.14 GWPeriGate Class Reference

```
#include <LISACODE-GWPeriGate.h>
```

Inheritance diagram for GWPeriGate::



10.14.1 Detailed Description

Gravitational Waves periodic gate signal.

h_plus and h_cross polarisation componenrs are modelised as periodic gate signals.

Definition at line 36 of file LISACODE-GWPeriGate.h.

Public Member Functions

- GWPeriGate ()

  Constructs an instance and initializes it with default values.

- GWPeriGate (double Beta_n, double Lambda_n, double AnglPol_n, double Freq_n, double Amplhp_n, double Amplhc_n)

  Constructs an instance and initializes it with default values and inputs.

- double getFreq () const

  Returns Freq attribute.

- void setFreq (double Freq_n)

  Sets Freq attribute.

- double getAmplhp () const

  Returns Amplhp attribute.

- void setAmplhp (double Amplhp_n)

  Sets Amplhp attribute.

- double getAmplhc () const

  Returns Amplhc attribute.

- void setAmplhc (double Amplhc_n)

  Sets Amplhc attribute.

- double hp (double t)

  Gives h_plus polarisation component depending on t *input time.*

- double hc (double t)

  Gives h_cross polarisation component depending on t *input time.*

- double getBeta () const

  Returns Beta attribute.

- void setBeta (double Beta_n)

  Sets Beta and DirProp attributes.

- double getLambda () const

  Returns Lambda attribute.

- void setLambda (double Lambda_n)

  Sets Lambda attribute.

- vector< double > getDirProp () const

  Returns DirProp attribute.

- void setDirProp (vector< double > DirProp_n)

  Sets DirProp attribute.

- double getAnglPol ()

  Returns AnglPol attribute.

- void setAnglPol (double AnglPol_n)

  Sets AnglPol attribute.

- void CalculDirProp ()

  Computes DirProp attribute components, depending on Lambda and Beta attributes.

Protected Attributes

- double Freq

  Frequency.

- double Amplhp

  hp component amplitude

- double Amplhc

  hc component amplitude

- double AnglPol

  Polarisation angle (rad).

- double Beta

  Source direction angle (in the heliocentric reference frame) in radians.

- double Lambda

Source direction angle (in the heliocentric reference frame) in radians.

- vector< double > DirProp

  Source direction unit vector (in the heliocentric reference frame).

10.14.2 Constructor & Destructor Documentation

10.14.2.1 GWPeriGate::GWPeriGate ()

Constructs an instance and initializes it with default values.

- Freq = 0
- Amplhp = 0
- Amplhc = 0
- AnglPol = 0

Definition at line 24 of file LISACODE-GWPeriGate.cpp.

References Amplhc, Amplhp, AnglPol, and Freq.

10.14.2.2 GWPeriGate::GWPeriGate (double *Beta_n*, double *Lambda_n*, double *AnglPol_n*, double *Freq_n*, double *Amplhp_n*, double *Amplhc_n*)

Constructs an instance and initializes it with default values and inputs.

Inputs are checked :*Freq_n*, *Amplhp_n* and *Amplhc_n* must be positive or null.

- Beta= Beta_n (using GW constructor)
- Lambda = Lambda_n (using GW constructor)
- AnglPol = AnglPol_n (using GW constructor)
- Freq = Freq_n
- Amplhp = Amplhp_n
- Amplhc = Amplhc_n

Definition at line 43 of file LISACODE-GWPeriGate.cpp.

References Amplhc, Amplhp, and Freq.

10.14.3 Member Function Documentation

10.14.3.1 void GW::CalculDirProp () `[inherited]`

Computes DirProp attribute components, depending on Lambda and Beta attributes.

$$DirProp = \left(\begin{array}{c} -cos(\lambda) \cdot cos(\beta) \\ -sin(\lambda) \cdot cos(\beta) \\ -sin(\beta) \end{array} \right)$$

Definition at line 203 of file LISACODE-GW.cpp.

References GW::Beta, GW::DirProp, and GW::Lambda.

Referenced by GW::GW(), GWBinary::GWBinary(), GW::setBeta(), and GW::setLambda().

10.14.3.2 double GWPeriGate::getAmplhc () const `[inline]`

Returns Amplhc attribute.

Definition at line 66 of file LISACODE-GWPeriGate.h.

References Amplhc.

10.14.3.3 double GWPeriGate::getAmplhp () const `[inline]`

Returns Amplhp attribute.

Definition at line 63 of file LISACODE-GWPeriGate.h.

References Amplhp.

10.14.3.4 double GW::getAnglPol () `[inline, inherited]`

Returns AnglPol attribute.

Definition at line 70 of file LISACODE-GW.h.

References GW::AnglPol.

10.14.3.5 double GW::getBeta () const `[inline, inherited]`

Returns Beta attribute.

Definition at line 59 of file LISACODE-GW.h.

References GW::Beta.

10.14.3.6 vector<double> GW::getDirProp () const `[inline, inherited]`

Returns DirProp attribute.

Definition at line 66 of file LISACODE-GW.h.

References GW::DirProp.

10.14.3.7 double GWPeriGate::getFreq () const `[inline]`

Returns Freq attribute.

Definition at line 60 of file LISACODE-GWPeriGate.h.

References Freq.

10.14.3.8 double GW::getLambda () const [inline, inherited]

Returns Lambda attribute.

Definition at line 62 of file LISACODE-GW.h.

References GW::Lambda.

10.14.3.9 double GWPeriGate::hc (double *t*) [virtual]

Gives h_cross polarisation component depending on *t* input time.

$$\text{returned value} = \begin{cases} Amplhc & \text{if } \frac{Freq \cdot t}{1} - ceil(\frac{Freq \cdot t}{1}) < 0.5 \\ 0 & else \end{cases}$$

Reimplemented from GW.

Definition at line 117 of file LISACODE-GWPeriGate.cpp.

References Amplhc, and Freq.

10.14.3.10 double GWPeriGate::hp (double *t*) [virtual]

Gives h_plus polarisation component depending on *t* input time.

$$\text{returned value} = \begin{cases} Amplhp & \text{if } \frac{Freq \cdot t}{1} - ceil(\frac{Freq \cdot t}{1}) < 0.5 \\ 0 & else \end{cases}$$

Reimplemented from GW.

Definition at line 104 of file LISACODE-GWPeriGate.cpp.

References Amplhp, and Freq.

10.14.3.11 void GWPeriGate::setAmplhc (double *Amplhc_n*)

Sets Amplhc attribute.

Input is checked: Amplhc_n must be positive or null.

Definition at line 87 of file LISACODE-GWPeriGate.cpp.

References Amplhc.

10.14.3.12 void GWPeriGate::setAmplhp (double *Amplhp_n*)

Sets Amplhp attribute.

Input is checked: Amplhp_n must be positive or null.

Definition at line 76 of file LISACODE-GWPeriGate.cpp.

References Amplhp.

10.14.3.13 void GW::setAnglPol (double *AnglPol_n*) [inherited]

Sets AnglPol attribute.

Input is checked:

$$AnglPol_n \in [0, 2 \cdot \pi]$$

Reimplemented in GWMono.

Definition at line 168 of file LISACODE-GW.cpp.

References GW::AnglPol.

10.14.3.14 void GW::setBeta (double *Beta_n*) [inherited]

Sets Beta and DirProp attributes.

Input is checked:

$$\beta_n \in [-\frac{\pi}{2}, -\frac{\pi}{2}]$$

CalculDirProp method is called to fill DirProp attribute

Definition at line 100 of file LISACODE-GW.cpp.

References GW::Beta, and GW::CalculDirProp().

10.14.3.15 void GW::setDirProp (vector< double > *DirProp_n*) [inherited]

Sets DirProp attribute.

Input is checked: *DirProp_n* must be a 3 components vector.

Computations: $DirProp_{norm}$ is normalized DirProp_n

- $$\beta = asin(DirProp_{norm}[2])$$

- $$\lambda = mod(atan(\frac{-DirProp_{norm}[1]}{-DirProp_{norm}[0]}), 2 \cdot \pi)$$

Definition at line 133 of file LISACODE-GW.cpp.

References GW::Beta, GW::DirProp, GW::Lambda, and PRECISION.

10.14.3.16 void GWPeriGate::setFreq (double *Freq_n*)

Sets Freq attribute.

Input is checked: *Freq_n* must be positive or null

Definition at line 65 of file LISACODE-GWPeriGate.cpp.

References Freq.

10.14.3.17 void GW::setLambda (double *Lambda_n*) [inherited]

Sets Lambda attribute.

Input is checked:

$$\lambda_n \in [0, 2 \cdot \pi]$$

CalculDirProp method is called to set DirProp attribute

Definition at line 115 of file LISACODE-GW.cpp.

References GW::CalculDirProp(), and GW::Lambda.

10.14.4 Member Data Documentation

10.14.4.1 double GWPeriGate::Amplhc [protected]

hc component amplitude

Definition at line 44 of file LISACODE-GWPeriGate.h.

Referenced by getAmplhc(), GWPeriGate(), hc(), and setAmplhc().

10.14.4.2 double GWPeriGate::Amplhp [protected]

hp component amplitude

Definition at line 42 of file LISACODE-GWPeriGate.h.

Referenced by getAmplhp(), GWPeriGate(), hp(), and setAmplhp().

10.14.4.3 double GWPeriGate::AnglPol [protected]

Polarisation angle (rad).

Angle between the projection of x (vernal point direction) in the wave frame and the polarisation vector

Reimplemented from GW.

Definition at line 50 of file LISACODE-GWPeriGate.h.

Referenced by GWPeriGate().

10.14.4.4 GW::Beta [protected, inherited]

Source direction angle (in the heliocentric reference frame) in radians.

Referenced by GW::CalculDirProp(), GW::getBeta(), GW::GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), GW::setBeta(), and GW::setDirProp().

10.14.4.5 vector<double> GW::DirProp [protected, inherited]

Source direction unit vector (in the heliocentric reference frame).

Definition at line 46 of file LISACODE-GW.h.

Referenced by GW::CalculDirProp(), GW::getDirProp(), GW::GW(), and GW::setDirProp().

10.14.4.6 double GWPeriGate::Freq [protected]

Frequency.

Definition at line 40 of file LISACODE-GWPeriGate.h.

Referenced by getFreq(), GWPeriGate(), hc(), hp(), and setFreq().

10.14.4.7 GW::Lambda [protected, inherited]

Source direction angle (in the heliocentric reference frame) in radians.

Referenced by GW::CalculDirProp(), GW::getLambda(), GW::GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), GW::setDirProp(), and GW::setLambda().

The documentation for this class was generated from the following files:

- LISACODE-GWPeriGate.h
- LISACODE-GWPeriGate.cpp

10.15 LISA Class Reference

```
#include <LISACODE-LISA.h>
```

10.15.1 Detailed Description

This class contains and manages all the elements necessary to LISA satellites simulation.

It contains the set of noises (laser noise, optical bench noise and inertial mass noise, see NoisePointers), the delays, the satelites position (SCPos), the set of photodiodes-phasemeters (PhotoDetects) and the gravitational waves transfert fonstion (sGW). It is related to memory object to which data are sent (RecordPDPM).

Definition at line 58 of file LISACODE-LISA.h.

Public Member Functions

- LISA ()

  Constructs an instance and initializes it with default values.

- LISA (double tStepPhy_n, double tStepMes_n, double tMemNoiseFirst_n, double tMemNoiseLast_n, double tMemRAM_n, double PSDLaser, double PSDOptBench, double PSDInertialMass, vector< Memory ∗ > ∗RecordPDPM_n, vector< GW ∗ > ∗GW_n, Background ∗GWB_n)

  Constructs an instance and initializes it with default values and inputs.

- LISA (ConfigSim ∗ConfigLISA, vector< Memory ∗ > ∗RecordPDPM_n)

  Constructs an instance and initializes it with default values and ConfigLISA *and* RecordPDPM_n *inputs.*

- ∼LISA ()

  Destructor.

- double gDelayT (int i, int IndirectDir, double t)

  Gives delay depending on inputs : i *index,* IndirectDir*, and* t *time.*

- double gArmLength (int i, int IndirectDir, double t)

  Gives arm length depending for on input i index, IndirectDir, and t time.

- Vect gPosSC (int i, double t)

  Gives satellite position depending at a given time t.

- void Stabilization ()

  It does phasemeters stabilization (signal and noise) for a given LISA geometry.

- void MakeOneStepOfTime (double t)

  Makes one step in time after noises loading.

- vector< double > PresentMeanNoise (double t)

  Returns the list of noises mean values, depending on t time.

Protected Attributes

- Geometry $*$ SCPos

  Geometry pointer : spacecraft orbitography structure.

- vector$<$ Noise $*$ $>$ NoisePointers

  Vector of noise pointers.

- vector$<$ PhoDetPhaMet $>$ PhotoDetects

  Vector of photodetector-phasemeters.

- vector$<$ Memory $*$ $>$ $*$ RecordPDPM

  Vector of memory objects where the photodetector-phasemeters signals are stored.

- TrFctGW sGW

  Transfer fonction to compute the LISA answer to gravitationnals waves.

- Background $*$ GWB

  Background pointer : gravitational background noise (small mass binaries signal received by phasemeters).

- double tStepPhy

  Time step to simulate continuous physical process.

- double tMemRAM

  Duration (time) of photodetector-phasemeters signals stored.

- double tStepMes

  Time step for phasemeters measurement.

- vector$<$ USOClock $>$ USOs

  Vector of LISA USO clocks.

10.15.2 Constructor & Destructor Documentation

10.15.2.1 LISA::LISA ()

Constructs an instance and initializes it with default values.

- tStepPhy = 0.01
- tStepMes = 1.0
- tMemRAM = 60.0
- SCPos = Geometry instance with default values
- sGW = TrFctGW initialized with default values
- GWB = NULL
- RecordPDPM = 3 Memory elements intitalized with tMemRAM and tStepMes attributes

- USOs = 3 USOClock elements intitalized with 0.0 value
- NoisePointers = 18 NULL elements
- PhotoDetects = 12 elements (2 directions, 3 spacecrafts, 2 interference types : 6 first are S interference type, 6 last are TAU interference type)

Definition at line 30 of file LISACODE-LISA.cpp.

References GWB, TrFctGW::init(), NoisePointers, PhotoDetects, RecordPDPM, S, SCPos, sGW, Stabilization(), TAU, tMemRAM, tStepMes, tStepPhy, and USOs.

10.15.2.2 LISA::LISA (double *tStepPhy_n*, double *tStepMes_n*, double *tMemNoiseFirst_n*, double *tMemNoiseLast_n*, double *tMemRAM_n*, double *PSDLaser*, double *PSDOptBench*, double *PSDInertialMass*, vector< Memory ∗ > ∗ *RecordPDPM_n*, vector< GW ∗ > ∗ *GW_n*, Background ∗ *GWB_n*)

Constructs an instance and initializes it with default values and inputs.

- tStepPhy = *tStepPhy_n*
- tStepMes = *tStepMes_n*
- tMemRAM = *tMemRAM_n*
- SCPos = Geometry instance with default values
- sGW = initialized using *GW_n* and SCPos parameters
- GWB = *GWB_n* if not NULL, else it is initialized by Background::setGeometry method, using SCPos attribute
- RecordPDPM = *RecordDPDM_n*
- USOs = 3 USOClock elements intitalized with 0.0 value
- NoisePointers = 6 white noises (NULL if *PSDLaser* input = 0.0), 6 opticals benchs noises (NULL if *PSDOptBench* input = 0.0), 6 inertials masses noises (NULL if *PSDInertialMass* input = 0.0).
- PhotoDetects = 12 elements (2 directions, 3 spacecrafts, 2 interference types : 6 first are S interference type, 6 last are TAU interference type). All noises are initialized using NoiseWhite (ex. NoiseWhite(tStepPhy, tStepMes, *tMemNoiseFirst_n*, *tMemNoiseLast_n*, {PSDLaser,PSDOptBench,PSDInertialMass}).

Definition at line 123 of file LISACODE-LISA.cpp.

References GWB, TrFctGW::init(), NoisePointers, PhotoDetects, RecordPDPM, S, SCPos, Background::setGeometry(), sGW, Stabilization(), TAU, tMemRAM, tStepMes, tStepPhy, and USOs.

10.15.2.3 LISA::LISA (ConfigSim ∗ *ConfigLISA*, vector< Memory ∗ > ∗ *RecordPDPM_n*)

Constructs an instance and initializes it with default values and *ConfigLISA* and *RecordPDPM_n* inputs.

- tStepPhy = extracted from *ConfigLISA* input
- tStepMes = extracted from *ConfigLISA* input

- tMemRAM = extracted from *ConfigLISA* input
- RecordPDPM = RecordPDPM_n input
- SCPos is initialized with inputs extracted from *ConfigLISA*
- sGW is initialized with inputs extracted from *ConfigLISA*
- GWB = extracted from *ConfigLISA* input
- USOs = 3 USOClock elements intitalized with *ConfigLISA* input value
- NoisePointers = extracted from *ConfigLISA* input
- PhotoDetects = 12 elements (2 directions, 3 spacecrafts, 2 interference types : 6 first are S interference type, 6 last are TAU interference type)

Definition at line 252 of file LISACODE-LISA.cpp.

References ConfigSim::getArmlength(), ConfigSim::getGWB(), ConfigSim::getGWs(), ConfigSim::getNoises(), ConfigSim::getNoNoise(), ConfigSim::getOrbInitRot(), ConfigSim::getOrbMove(), ConfigSim::getOrbOrder(), ConfigSim::getOrbStartTime(), ConfigSim::getPhaMetFilter(), ConfigSim::getPhaMetFilterParam(), ConfigSim::gettMemSig(), ConfigSim::gettStepMes(), ConfigSim::gettStepPhy(), ConfigSim::getUSOs(), GWB, TrFctGW::init(), NoisePointers, PhotoDetects, RecordPDPM, S, SCPos, Background::setGeometry(), sGW, Stabilization(), TAU, tMemRAM, tStepMes, tStepPhy, and USOs.

10.15.2.4 LISA::∼LISA ()

Destructor.

Definition at line 338 of file LISACODE-LISA.cpp.

10.15.3 Member Function Documentation

10.15.3.1 double LISA::gArmLength (int *i*, int *IndirectDir*, double *t*)

Gives arm length depending for on input i index, IndirectDir, and t time.

Parameters:

i arm index from 1 to 3

IndirectDir optical bench direction (0 if the optical bench is in the direct direction, 1 else)

t time

i (arm index), *IndirectDir*, emitter and receiver satellites are related as follows:

- from satellite 1 to 2: arm i=3, IndirectDir=1
- from satellite 2 to 3: arm i=1, IndirectDir=1
- from satellite 3 to 1: arm i=2, IndirectDir=1
- from satellite 2 to 1: arm i=3, IndirectDir=0
- from satellite 3 to 2: arm i=1, IndirectDir=0
- from satellite 1 to 3: arm i=2, IndirectDir=0

So, emitter and receiver index depend on *IndirectDir* and arm index *i*:

- if (IndirectDir = 0) em = mod(i+2,3) and rec = mod(i+1,3)
- else em = mod(i+1,3) and rec = mod(i+2,3)

Geometry::gtdelay method is called, with order = 2 returned value is opposite of Geometry::gtdelay result, multiplied by c_SI.

Definition at line 404 of file LISACODE-LISA.cpp.

References c_SI, Geometry::gtdelay(), and SCPos.

10.15.3.2 double LISA::gDelayT (int *i*, int *IndirectDir*, double *t*)

Gives delay depending on inputs : *i* index, *IndirectDir*, and *t* time.

i (arm index), *IndirectDir*, emitter and receiver satellites are related as follows:

- from satellite 1 to 2: arm i=3, IndirectDir=1
- from satellite 2 to 3: arm i=1, IndirectDir=1
- from satellite 3 to 1: arm i=2, IndirectDir=1
- from satellite 2 to 1: arm i=3, IndirectDir=0
- from satellite 3 to 2: arm i=1, IndirectDir=0
- from satellite 1 to 3: arm i=2, IndirectDir=0

So, emitter and receiver index depend on *IndirectDir* and arm index *i*:

- if (IndirectDir = 0) em = mod(i+2,3) and rec = mod(i+1,3)
- else em = mod(i+1,3) and rec = mod(i+2,3)

Geometry::gtdelay method is called with *em*, *rec* index, order = 2 and *t* input. Returned value the is opposite of Geometry::gtdelay result.

Definition at line 364 of file LISACODE-LISA.cpp.

References Geometry::gtdelay(), and SCPos.

Referenced by main().

10.15.3.3 Vect LISA::gPosSC (int *i*, double *t*)

Gives satellite position depending at a given time *t*.

Parameters:

i: spacecraft number [1,3]

t: time. It calls Geometry::gposition with inpute *i* and *t*.

Definition at line 425 of file LISACODE-LISA.cpp.

References Geometry::gposition(), and SCPos.

Referenced by main().

10.15.3.4 void LISA::MakeOneStepOfTime (double *t*)

Makes one step in time after noises loading.

For all elements in NoisePointers (i index):

- if NoisePointers[i] is NULL, a noise vector is added to NoisePointers[i], using Noise::addNoise method.

For all elements in PhotoDetects:

- photodetector signal is stored in memory, using PhoDetPhaMet::IntegrateSignal method.

For all spacecrafts:

- photodetector-phasemeters signal (stored in RecordPDPM) is recorded (Memory::RecordAccData is called, using tStepMes attribute, and time extracted from USOs).

Definition at line 491 of file LISACODE-LISA.cpp.

References NoisePointers, PhotoDetects, tStepMes, and USOs.

Referenced by main().

10.15.3.5 vector< double > LISA::PresentMeanNoise (double *t*)

Returns the list of noises mean values, depending on t time.

For all elements in NoisePointers (i index):

- if NoisePointers[i] is not NULL, noise mean value is computed and pushed back in returned value.

Definition at line 520 of file LISACODE-LISA.cpp.

References NoisePointers, and tStepPhy.

10.15.3.6 void LISA::Stabilization ()

It does phasemeters stabilization (signal and noise) for a given LISA geometry.

While current time is lower than stabilization time (progressively incremented by with tStepMes) :

- For all elements in NoisePointers (i index): if NoisePointers[i] is not NULL, a noise vector is added to NoisePointers[i], using Noise::addNoise method
- For all elements in PhotoDetects: photodetector signal is stored in memory, using PhoDetPhaMet::IntegrateSignal method

Definition at line 453 of file LISACODE-LISA.cpp.

References NoisePointers, PhotoDetects, and tStepMes.

Referenced by LISA().

10.15.4 Member Data Documentation

10.15.4.1 Background* LISA::GWB [protected]

Background pointer : gravitational background noise (small mass binaries signal received by phasemeters).

Definition at line 77 of file LISACODE-LISA.h.

Referenced by LISA().

10.15.4.2 vector<Noise *> LISA::NoisePointers [protected]

Vector of noise pointers.

It contains the addresses of all LISA related noises in next order: noises of lasers, noises of optical bench, noises of inertial mass and others noises.

Definition at line 68 of file LISACODE-LISA.h.

Referenced by LISA(), MakeOneStepOfTime(), PresentMeanNoise(), and Stabilization().

10.15.4.3 vector<PhoDetPhaMet> LISA::PhotoDetects [protected]

Vector of photodetector-phasemeters.

Definition at line 71 of file LISACODE-LISA.h.

Referenced by LISA(), MakeOneStepOfTime(), and Stabilization().

10.15.4.4 vector<Memory *>* LISA::RecordPDPM [protected]

Vector of memory objects where the photodetector-phasemeters signals are stored.

Definition at line 73 of file LISACODE-LISA.h.

Referenced by LISA().

10.15.4.5 Geometry* LISA::SCPos [protected]

Geometry pointer : spacecraft orbitography structure.

Definition at line 62 of file LISACODE-LISA.h.

Referenced by gArmLength(), gDelayT(), gPosSC(), and LISA().

10.15.4.6 TrFctGW LISA::sGW [protected]

Transfer fonction to compute the LISA answer to gravitationnals waves.

Definition at line 75 of file LISACODE-LISA.h.

Referenced by LISA().

10.15.4.7 double LISA::tMemRAM [protected]

Duration (time) of photodetector-phasemeters signals stored.

Definition at line 81 of file LISACODE-LISA.h.

Referenced by LISA().

10.15.4.8 double LISA::tStepMes [protected]

Time step for phasemeters measurement.

Definition at line 83 of file LISACODE-LISA.h.

Referenced by LISA(), MakeOneStepOfTime(), and Stabilization().

10.15.4.9 double LISA::tStepPhy [protected]

Time step to simulate continuous physical process.

Definition at line 79 of file LISACODE-LISA.h.

Referenced by LISA(), and PresentMeanNoise().

10.15.4.10 vector<USOClock> LISA::USOs [protected]

Vector of LISA USO clocks.

There is one USO clock by spacecraft.

Definition at line 88 of file LISACODE-LISA.h.

Referenced by LISA(), and MakeOneStepOfTime().

The documentation for this class was generated from the following files:

- LISACODE-LISA.h
- LISACODE-LISA.cpp

10.16 Mat Class Reference

`#include <LISACODE-Mat.h>`

10.16.1 Detailed Description

(3x3) matrix management class.

Definition at line 33 of file LISACODE-Mat.h.

Public Member Functions

- Mat ()

  Constructs an instance and initializes it with default value.

- Mat (double[3][3])

  Constructs an instance and initializes it with A (3,3) matrix input.

- ∼Mat ()

  Destructor.

- void display ()

  Displays matrix components.

Public Attributes

- double p [3][3]

  (3x3) components

Friends

- Mat operator+ (Mat, Mat)

  Matrices addition. It returns matrix A+B.

- Mat operator- (Mat, Mat)

  Matrices subtraction. It returns matrix A-B.

- Mat operator ∗ (double, Mat)

  Product between a scalar and a matrix. It returns matrix: f.A.

- Vect operator ∗ (Mat, Vect)

  Product between a matrix and vector. It returns vector A.v.

10.16.2 Constructor & Destructor Documentation

10.16.2.1 Mat::Mat ()

Constructs an instance and initializes it with default value.

$$Mat = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Definition at line 25 of file LISACODE-Mat.cpp.

References p.

10.16.2.2 Mat::Mat (double *A*[3][3])

Constructs an instance and initializes it with A (3,3) matrix input.

Definition at line 34 of file LISACODE-Mat.cpp.

References p.

10.16.2.3 Mat::∼Mat ()

Destructor.

Definition at line 44 of file LISACODE-Mat.cpp.

10.16.3 Member Function Documentation

10.16.3.1 void Mat::display ()

Displays matrix components.

Definition at line 52 of file LISACODE-Mat.cpp.

References p.

10.16.4 Friends And Related Function Documentation

10.16.4.1 Vect operator ∗ (Mat *A*, Vect *u*) `[friend]`

Product between a matrix and vector. It returns vector A.v.

Definition at line 111 of file LISACODE-Mat.cpp.

10.16.4.2 Mat operator ∗ (double *f*, Mat *A*) `[friend]`

Product between a scalar and a matrix. It returns matrix: $f.A$.

Definition at line 96 of file LISACODE-Mat.cpp.

10.16.4.3 Mat operator+ (Mat *A*, Mat *B*) `[friend]`

Matrices addition. It returns matrix A+B.

Definition at line 70 of file LISACODE-Mat.cpp.

10.16.4.4 Mat operator- (Mat *A*, Mat *B*) `[friend]`

Matrices subtraction. It returns matrix A-B.

Definition at line 82 of file LISACODE-Mat.cpp.

10.16.5 Member Data Documentation

10.16.5.1 double Mat::p[3][3]

(3x3) components

Definition at line 37 of file LISACODE-Mat.h.

Referenced by display(), Mat(), operator $*$(), operator+(), and operator-().

The documentation for this class was generated from the following files:

- LISACODE-Mat.h
- LISACODE-Mat.cpp

10.17 MathUtils Class Reference

`#include <LISACODE-MathUtils.h>`

10.17.1 Detailed Description

Angle conversion class.

Definition at line 35 of file LISACODE-MathUtils.h.

Static Public Member Functions

- double deg2rad (double angle_)

    Angle conversion (from degrees to radians).

- double rad2deg (double angle_)

    Angle conversion (from radians to degrees).

10.17.2 Member Function Documentation

10.17.2.1 double MathUtils::deg2rad (double *angle_*) `[inline, static]`

Angle conversion (from degrees to radians).

Definition at line 40 of file LISACODE-MathUtils.h.

Referenced by ConfigSim::gXMLAngle(), and ConfigSim::ReadASCIIFile().

10.17.2.2 double MathUtils::rad2deg (double *angle_*) `[inline, static]`

Angle conversion (from radians to degrees).

Definition at line 47 of file LISACODE-MathUtils.h.

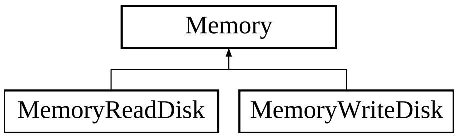
The documentation for this class was generated from the following file:

- LISACODE-MathUtils.h

10.18 Memory Class Reference

`#include <LISACODE-Memory.h>`

Inheritance diagram for Memory::



10.18.1 Detailed Description

Memory management class.

Definition at line 42 of file LISACODE-Memory.h.

Public Member Functions

- Memory ()

  Constructs an instance and initializes it with default values.

- Memory (double tStoreData_n, double tStepRecord_n)

  Constructs an instance and initializes it with tStoreData_n *and* tStepRecord_n *inputs.*

- virtual ∼Memory ()

  Destructor.

- int getNbSerie ()

  Gets the number of the serie.

- double gettStoreData ()

  Gets tStoreData attribute.

- void settStoreData (double tStoreData_n)

  Sets tStoreData attribute using tStoreData_n *input.*

- double gettStepRecord ()

  Gets tStepRecord attribute.

- void settStepRecord (double tStepRecord_n)

  Sets tStepRecord attribute using tStepRecord_n *input.*

- virtual double gettMax ()

  Gets maximum time, tStoreData attribute.

- virtual void AddSerieData (int SerieNumber, char ∗TypeName, int IndirectDirName, int iSCName)

  Adds series in ListTmpData and updates AlreadyRecDat set to false for added series.

- virtual void MakeTitles (char *FileNameHead="")

  Empty method.

- void ReceiveData (int SerieNumber, double data)

  Sets first value of ListTmpData[SerieNumber] to data *input value and sets AlreadyRecDat[SerieNumber] to TRUE.*

- virtual void RecordAccData (double tStep, double t)

  Records received data (make it between each step of time).

- double gData (int SerieNumber, double delay) const

  *Returns the data of specified serie (*SerieNumber*) delayed by* tDelay *delay.*

- double gData (int SerieNumber, double delay, INTERP InterpolType, double InterpUtilValue) const

  *Returns the data of specified serie (*SerieNumber*), delayed by* tDelay *and interpolated using interpolation parameters (*InterpolType*,* InterpUtilValue*).*

- int unusable (double tSinceFirstReception) const

  Returns 0 if there are enough stored data to use them.

Protected Attributes

- vector< Serie > ListTmpData

  List of stored data series (RAM).

- vector< bool > AlreadyRecDat

  Vector elements are set to 1 if data are already received for the corresponding serie.

- double tStoreData

  Memorization time (RAM) Time during which the data are preserved.

- double tStepRecord

  Time step for recording data.

10.18.2 Constructor & Destructor Documentation

10.18.2.1 Memory::Memory ()

Constructs an instance and initializes it with default values.

- tStoreData = 200.0
- tStepRecord = 0.01

Definition at line 21 of file LISACODE-Memory.cpp.

References tStepRecord, and tStoreData.

10.18.2.2 Memory::Memory (double *tStoreData_n*, double *tStepRecord_n*)

Constructs an instance and initializes it with *tStoreData_n* and *tStepRecord_n* inputs.

- tStoreData =*tStoreData_n*
- tStepRecord =*tStepRecord_n*

Definition at line 33 of file LISACODE-Memory.cpp.

References tStepRecord, and tStoreData.

10.18.2.3 Memory::~Memory () `[virtual]`

Destructor.

Definition at line 41 of file LISACODE-Memory.cpp.

10.18.3 Member Function Documentation

10.18.3.1 void Memory::AddSerieData (int *SerieNumber*, char ∗ *TypeName*, int *IndirectDirName*, int *iSCName*) `[virtual]`

Adds series in ListTmpData and updates AlreadyRecDat set to false for added series.

New series are added to ListTmpData while ListTmpData size is lower than SerieNumber input.

Corresponding FALSE flags are added into AlreadyRecDat vector.

Additionnal series are initialized with 0.0 start value and tStepRecord time step. Only *SerieNumber* input is used (*TypeName*,*IndirectDirName*, *iSCName* are unused).

Virtual unused method.

Reimplemented in MemoryReadDisk, and MemoryWriteDisk.

Definition at line 88 of file LISACODE-Memory.cpp.

References AlreadyRecDat, ListTmpData, and tStepRecord.

Referenced by PhoDetPhaMet::init(), and main().

10.18.3.2 double Memory::gData (int *SerieNumber*, double *tDelay*, INTERP *InterpolType*, double *InterpUtilValue*) const

Returns the data of specified serie (*SerieNumber*), delayed by *tDelay* and interpolated using interpolation parameters (*InterpolType*, *InterpUtilValue*).

SerieNumber input is checked : it must be positive or null, and lower than ListTmpData size.

Calls Serie::gData with *tDelay*, *InterpolType* and *InterpUtilValue* inputs.

Definition at line 181 of file LISACODE-Memory.cpp.

References ListTmpData.

10.18.3.3 double Memory::gData (int *SerieNumber*, double *tDelay*) const

Returns the data of specified serie (*SerieNumber*) delayed by *tDelay* delay.

SerieNumber input is checked : it must be positive or null, and lower than ListTmpData size

Definition at line 152 of file LISACODE-Memory.cpp.

References LAG, and ListTmpData.

Referenced by TDI::Compute(), TDI_InterData::ComputeEta(), and TDITools::RefreshDelay().

10.18.3.4 int Memory::getNbSerie () [inline]

Gets the number of the serie.

Definition at line 65 of file LISACODE-Memory.h.

References ListTmpData.

10.18.3.5 double Memory::gettMax () [virtual]

Gets maximum time, tStoreData attribute.

Reimplemented in MemoryReadDisk, and MemoryWriteDisk.

Definition at line 72 of file LISACODE-Memory.cpp.

References tStoreData.

10.18.3.6 double Memory::gettStepRecord () [inline]

Gets tStepRecord attribute.

Definition at line 71 of file LISACODE-Memory.h.

References tStepRecord.

10.18.3.7 double Memory::gettStoreData () [inline]

Gets tStoreData attribute.

Definition at line 67 of file LISACODE-Memory.h.

References tStoreData.

10.18.3.8 void Memory::MakeTitles (char ∗ *FileNameHead* = " ") [virtual]

Empty method.

Reimplemented in MemoryReadDisk, and MemoryWriteDisk.

Definition at line 100 of file LISACODE-Memory.cpp.

Referenced by main().

10.18.3.9 void Memory::ReceiveData (int *SerieNumber*, double *data*)

Sets first value of ListTmpData[SerieNumber] to *data* input value and sets AlreadyRecDat[SerieNumber] to TRUE.

SerieNumber input is checked : it must be positive or null, and lower than ListTmpData size

Reimplemented in MemoryReadDisk.

Definition at line 111 of file LISACODE-Memory.cpp.

References AlreadyRecDat, and ListTmpData.

Referenced by PhoDetPhaMet::IntegrateSignal(), and main().

10.18.3.10 void Memory::RecordAccData (double *tStep*, double *t*) [virtual]

Records received data (make it between each step of time).

tStep and *t* inputs are unused.

AlreadyRecDat attributes are checked : it must be TRUE (for all series).

Last data of ListTmpData are deleted for all series.

Virtual unused method.

Reimplemented in MemoryReadDisk, and MemoryWriteDisk.

Definition at line 131 of file LISACODE-Memory.cpp.

References AlreadyRecDat, ListTmpData, and tStoreData.

Referenced by main().

10.18.3.11 void Memory::settStepRecord (double *tStepRecord_n*)

Sets tStepRecord attribute using *tStepRecord_n* input.

tStepRecord_n input is checked : it is expected to be positive or null.

Definition at line 63 of file LISACODE-Memory.cpp.

References tStepRecord.

10.18.3.12 void Memory::settStoreData (double *tStoreData_n*)

Sets tStoreData attribute using *tStoreData_n* input.

tStoreData_n input is checked : it is expected to be positive or null.

Definition at line 52 of file LISACODE-Memory.cpp.

References tStoreData.

10.18.3.13 int Memory::unusable (double *tSinceFirstReception*) const

Returns 0 if there are enough stored data to use them.

Returns:

0 if *tSinceFirstReception* input is greater than tStoreData attribute , else 1.

Definition at line 209 of file LISACODE-Memory.cpp.

References tStoreData.

10.18.4 Member Data Documentation

10.18.4.1 vector<bool> Memory::AlreadyRecDat [protected]

Vector elements are set to 1 if data are already received for the corresponding serie.

Definition at line 48 of file LISACODE-Memory.h.

Referenced by MemoryWriteDisk::AddSerieData(), MemoryReadDisk::AddSerieData(), AddSerieData(), ReceiveData(), MemoryWriteDisk::RecordAccData(), and RecordAccData().

10.18.4.2 vector<Serie> Memory::ListTmpData [protected]

List of stored data series (RAM).

Definition at line 46 of file LISACODE-Memory.h.

Referenced by MemoryWriteDisk::AddSerieData(), MemoryReadDisk::AddSerieData(), AddSerieData(), gData(), getNbSerie(), ReceiveData(), MemoryWriteDisk::RecordAccData(), MemoryReadDisk::RecordAccData(), and RecordAccData().

10.18.4.3 double Memory::tStepRecord [protected]

Time step for recording data.

Definition at line 55 of file LISACODE-Memory.h.

Referenced by MemoryWriteDisk::AddSerieData(), MemoryReadDisk::AddSerieData(), AddSerieData(), MemoryReadDisk::gettMax(), gettStepRecord(), Memory(), MemoryReadDisk::MemoryReadDisk(), and settStepRecord().

10.18.4.4 double Memory::tStoreData [protected]

Memorization time (RAM) Time during which the data are preserved.

Definition at line 53 of file LISACODE-Memory.h.

Referenced by MemoryWriteDisk::gettMax(), gettMax(), gettStoreData(), Memory(), MemoryWriteDisk::RecordAccData(), MemoryReadDisk::RecordAccData(), RecordAccData(), settStoreData(), and unusable().

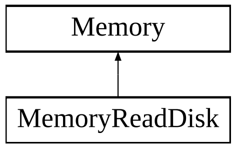
The documentation for this class was generated from the following files:

- LISACODE-Memory.h
- LISACODE-Memory.cpp

10.19 MemoryReadDisk Class Reference

```
#include <LISACODE-MemoryReadDisk.h>
```

Inheritance diagram for MemoryReadDisk::



10.19.1 Detailed Description

Class to manage disk reading.

Definition at line 35 of file LISACODE-MemoryReadDisk.h.

Public Member Functions

- MemoryReadDisk ()

  Constructs an empty instance.

- MemoryReadDisk (double tStoreData_n, double tStepRecord_n, char ∗NomFichMem_n)

  Constructs an instance and initializes it with default values and tStoreData_n, tStepRecord_n and NomFichMem_n inputs.

- ∼MemoryReadDisk ()

  Destructor.

- double gettMax ()

  Gets maximum time.

- void AddSerieData (int SerieNumber, char ∗TypeName, int IndirectDirName, int iSCName)

  Adds series in ListTmpData attribute, updates AlreadyRecDat attribute, fills IndexInReadData[SerieNumber] attribute.

- void MakeTitles (char ∗FileNameHead="")

  Empty method.

- void ReceiveData (int SerieNumber, double Data)

  Sets first value of ListTmpData[SerieNumber] to data *input value and sets AlreadyRecDat[SerieNumber] to TRUE.*

- void RecordAccData (double tStep, double t)

  Records received data.

- int getNbSerie ()

  Gets the number of the serie.

- double gettStoreData ()

  Gets tStoreData attribute.

- void settStoreData (double tStoreData_n)

  Sets tStoreData attribute using tStoreData_n *input.*

- double gettStepRecord ()

  Gets tStepRecord attribute.

- void settStepRecord (double tStepRecord_n)

  Sets tStepRecord attribute using tStepRecord_n *input.*

- double gData (int SerieNumber, double delay) const

  *Returns the data of specified serie (*SerieNumber*) delayed by* tDelay *delay.*

- double gData (int SerieNumber, double delay, INTERP InterpolType, double InterpUtilValue) const

  *Returns the data of specified serie (*SerieNumber*), delayed by* tDelay *and interpolated using interpolation parameters (*InterpolType, InterpUtilValue*).*

- int unusable (double tSinceFirstReception) const

  Returns 0 if there are enough stored data to use them.

Protected Attributes

- char * NomFichMem

  File name on which data are read.

- ifstream FichMem

  File object managing reading file NomFichMem.

- vector< vector< double > > ReadData

  Data read from file.

- vector< string > TitlesReadData

  Titles of read series.

- vector< int > IndexInReadData

  Index in read data.

- vector< Serie > ListTmpData

  List of stored data series (RAM).

- vector< bool > AlreadyRecDat

  Vector elements are set to 1 if data are already received for the corresponding serie.

- double tStoreData

  Memorization time (RAM) Time during which the data are preserved.

- double tStepRecord

  Time step for recording data.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 MemoryReadDisk::MemoryReadDisk ()

Constructs an empty instance.

Definition at line 17 of file LISACODE-MemoryReadDisk.cpp.

10.19.2.2 MemoryReadDisk::MemoryReadDisk (double *tStoreData_n*, double *tStepRecord_n*, char ∗ *NomFichMem_n*)

Constructs an instance and initializes it with default values and tStoreData_n, tStepRecord_n and NomFichMem_n inputs.

Memory constructor is called with tStoreData_n and tStepRecord_n inputs.

Others attributes are set using data read in NomFichMem_n file :

- NbColumns is read in the first line
- TitlesReadData are read (NbColumns elements)
- ReadData are read (NbColumns elements)

Definition at line 33 of file LISACODE-MemoryReadDisk.cpp.

References FichMem, NomFichMem, ReadData, TitlesReadData, and Memory::tStepRecord.

10.19.2.3 MemoryReadDisk::∼MemoryReadDisk ()

Destructor.

Definition at line 118 of file LISACODE-MemoryReadDisk.cpp.

10.19.3 Member Function Documentation

10.19.3.1 void MemoryReadDisk::AddSerieData (int *SerieNumber*, char ∗ *TypeName*, int *IndirectDirName*, int *iSCName*) [virtual]

Adds series in ListTmpData attribute, updates AlreadyRecDat attribute, fills IndexInReadData[SerieNumber] attribute.

New series are added in ListTmpData to have *SerieNumber* series in input ListTmpData. Corresponding FALSE flags are added into AlreadyRecDat attribute.

Additionnal series are initialized with 0.0 start value and tStepRecord time step.

IndexInReadData[SerieNumber] attribute is filled with index found by comparing TitlesReadData[SerieNumber] to RequiredTitle build with *IndirectDirName* and *iSCName* inputs.

Reimplemented from Memory.

Definition at line 144 of file LISACODE-MemoryReadDisk.cpp.

References Memory::AlreadyRecDat, IndexInReadData, Memory::ListTmpData, TitlesReadData, and Memory::tStepRecord.

10.19.3.2 double Memory::gData (int *SerieNumber*, double *tDelay*, INTERP *InterpolType*, double *InterpUtilValue*) const `[inherited]`

Returns the data of specified serie (*SerieNumber*), delayed by *tDelay* and interpolated using interpolation parameters (*InterpolType*, *InterpUtilValue*).

SerieNumber input is checked : it must be positive or null, and lower than ListTmpData size.

Calls Serie::gData with *tDelay*, *InterpolType* and *InterpUtilValue* inputs.

Definition at line 181 of file LISACODE-Memory.cpp.

References Memory::ListTmpData.

10.19.3.3 double Memory::gData (int *SerieNumber*, double *tDelay*) const `[inherited]`

Returns the data of specified serie (*SerieNumber*) delayed by *tDelay* delay.

SerieNumber input is checked : it must be positive or null, and lower than ListTmpData size

Definition at line 152 of file LISACODE-Memory.cpp.

References LAG, and Memory::ListTmpData.

Referenced by TDI::Compute(), TDI_InterData::ComputeEta(), and TDITools::RefreshDelay().

10.19.3.4 int Memory::getNbSerie () `[inline, inherited]`

Gets the number of the serie.

Definition at line 65 of file LISACODE-Memory.h.

References Memory::ListTmpData.

10.19.3.5 double MemoryReadDisk::gettMax () `[virtual]`

Gets maximum time.

Maximum time is number of data read (ReadData[0].size()-1) multiplied by tStepRecord.

Reimplemented from Memory.

Definition at line 129 of file LISACODE-MemoryReadDisk.cpp.

References ReadData, and Memory::tStepRecord.

10.19.3.6 double Memory::gettStepRecord () `[inline, inherited]`

Gets tStepRecord attribute.

Definition at line 71 of file LISACODE-Memory.h.

References Memory::tStepRecord.

10.19.3.7 double Memory::gettStoreData () [inline, inherited]

Gets tStoreData attribute.

Definition at line 67 of file LISACODE-Memory.h.

References Memory::tStoreData.

10.19.3.8 void MemoryReadDisk::MakeTitles (char ∗ *FileNameHead* = " ") [virtual]

Empty method.

Reimplemented from Memory.

Definition at line 174 of file LISACODE-MemoryReadDisk.cpp.

10.19.3.9 void MemoryReadDisk::ReceiveData (int *SerieNumber*, double *Data*)

Sets first value of ListTmpData[SerieNumber] to *data* input value and sets AlreadyRecDat[SerieNumber] to TRUE.

SerieNumber input is checked : it must be positive or null, and lower than ListTmpData size

Reimplemented from Memory.

10.19.3.10 void MemoryReadDisk::RecordAccData (double *tStep*, double *t*) [virtual]

Records received data.

tStep and t inputs are unused.

AlreadyRecDat attributes are checked : it must be TRUE (for all series).

ListTmpData last data are removed (for all series).

AlreadyRecDat attributes are set to FALSE (for all series).

Reimplemented from Memory.

Definition at line 189 of file LISACODE-MemoryReadDisk.cpp.

References IndexInReadData, Memory::ListTmpData, ReadData, and Memory::tStoreData.

10.19.3.11 void Memory::settStepRecord (double *tStepRecord_n*) [inherited]

Sets tStepRecord attribute using *tStepRecord_n* input.

tStepRecord_n input is checked : it is expected to be positive or null.

Definition at line 63 of file LISACODE-Memory.cpp.

References Memory::tStepRecord.

10.19.3.12 void Memory::settStoreData (double *tStoreData_n*) [inherited]

Sets tStoreData attribute using *tStoreData_n* input.

tStoreData_n input is checked : it is expected to be positive or null.

Definition at line 52 of file LISACODE-Memory.cpp.

References Memory::tStoreData.

10.19.3.13 int Memory::unusable (double *tSinceFirstReception*) const [inherited]

Returns 0 if there are enough stored data to use them.

Returns:

0 if *tSinceFirstReception* input is greater than tStoreData attribute , else 1.

Definition at line 209 of file LISACODE-Memory.cpp.

References Memory::tStoreData.

10.19.4 Member Data Documentation

10.19.4.1 vector<bool> Memory::AlreadyRecDat [protected, inherited]

Vector elements are set to 1 if data are already received for the corresponding serie.

Definition at line 48 of file LISACODE-Memory.h.

Referenced by MemoryWriteDisk::AddSerieData(), AddSerieData(), Memory::AddSerieData(), Memory::ReceiveData(), MemoryWriteDisk::RecordAccData(), and Memory::RecordAccData().

10.19.4.2 ifstream MemoryReadDisk::FichMem [protected]

File object managing reading file NomFichMem.

Definition at line 41 of file LISACODE-MemoryReadDisk.h.

Referenced by MemoryReadDisk().

10.19.4.3 vector<int> MemoryReadDisk::IndexInReadData [protected]

Index in read data.

Definition at line 47 of file LISACODE-MemoryReadDisk.h.

Referenced by AddSerieData(), and RecordAccData().

10.19.4.4 vector<Serie> Memory::ListTmpData [protected, inherited]

List of stored data series (RAM).

Definition at line 46 of file LISACODE-Memory.h.

Referenced by MemoryWriteDisk::AddSerieData(), AddSerieData(), Memory::AddSerieData(), Memory::gData(), Memory::getNbSerie(), Memory::ReceiveData(), MemoryWriteDisk::RecordAcc-Data(), RecordAccData(), and Memory::RecordAccData().

10.19.4.5 char* MemoryReadDisk::NomFichMem [protected]

File name on which data are read.

Definition at line 39 of file LISACODE-MemoryReadDisk.h.

Referenced by MemoryReadDisk().

10.19.4.6 vector< vector<double> > MemoryReadDisk::ReadData [protected]

Data read from file.

Definition at line 43 of file LISACODE-MemoryReadDisk.h.

Referenced by gettMax(), MemoryReadDisk(), and RecordAccData().

10.19.4.7 vector<string> MemoryReadDisk::TitlesReadData [protected]

Titles of read series.

Definition at line 45 of file LISACODE-MemoryReadDisk.h.

Referenced by AddSerieData(), and MemoryReadDisk().

10.19.4.8 double Memory::tStepRecord [protected, inherited]

Time step for recording data.

Definition at line 55 of file LISACODE-Memory.h.

Referenced by MemoryWriteDisk::AddSerieData(), AddSerieData(), Memory::AddSerieData(), gettMax(), Memory::gettStepRecord(), Memory::Memory(), MemoryReadDisk(), and Memory::settStepRecord().

10.19.4.9 double Memory::tStoreData [protected, inherited]

Memorization time (RAM) Time during which the data are preserved.

Definition at line 53 of file LISACODE-Memory.h.

Referenced by MemoryWriteDisk::gettMax(), Memory::gettMax(), Memory::gettStoreData(), Memory::Memory(), MemoryWriteDisk::RecordAccData(), RecordAccData(), Memory::RecordAccData(), Memory::settStoreData(), and Memory::unusable().

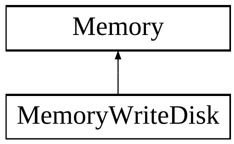
The documentation for this class was generated from the following files:

- LISACODE-MemoryReadDisk.h
- LISACODE-MemoryReadDisk.cpp

10.20 MemoryWriteDisk Class Reference

```
#include <LISACODE-MemoryWriteDisk.h>
```

Inheritance diagram for MemoryWriteDisk::



10.20.1 Detailed Description

Class to manage disk writting.

Definition at line 38 of file LISACODE-MemoryWriteDisk.h.

Public Member Functions

- MemoryWriteDisk ()

  Constructs an empty instance.

- MemoryWriteDisk (double tStoreData_n, double tStepRecord_n, char ∗NomFichMem_n)

  Constructs an instance and initializes it with default values and tStoreData_n, tStepRecord_n and NomFichMem_n inputs.

- ∼MemoryWriteDisk ()

  Destructor.

- double gettMax ()

  Gets maximum time.

- void AddSerieData (int SerieNumber, char ∗TypeName, int IndirectDirName, int iSCName)

  Adds series in ListTmpData, updates corresponding elements in AlreadyRecDat, fills TitleSerie[SerieNumber] attribute and sets SCSerie[SerieNumber].

- void MakeTitles (char ∗FileNameHead="")

  Copies header from FileNameHead *file to FichMem.*

- void RecordAccData (double tStep, double t)

  Records received data in file FichMem (make it between each step of time).

- void CloseFile ()

  Closes FichMem.

- int getNbSerie ()

  Gets the number of the serie.

- double gettStoreData ()

Gets tStoreData attribute.

- void settStoreData (double tStoreData_n)

  Sets tStoreData attribute using tStoreData_n *input.*

- double gettStepRecord ()

  Gets tStepRecord attribute.

- void settStepRecord (double tStepRecord_n)

  Sets tStepRecord attribute using tStepRecord_n *input.*

- void ReceiveData (int SerieNumber, double data)

  Sets first value of ListTmpData[SerieNumber] to data *input value and sets AlreadyRecDat[SerieNumber] to TRUE.*

- double gData (int SerieNumber, double delay) const

  *Returns the data of specified serie (*SerieNumber*) delayed by* tDelay *delay.*

- double gData (int SerieNumber, double delay, INTERP InterpolType, double InterpUtilValue) const

  *Returns the data of specified serie (*SerieNumber*), delayed by* tDelay *and interpolated using interpolation parameters (*InterpolType, InterpUtilValue*).*

- int unusable (double tSinceFirstReception) const

  Returns 0 if there are enough stored data to use them.

Protected Attributes

- char ∗ NomFichMem

  File name on which data are recorded.

- ofstream FichMem

  File object managing recording file NomFichMem.

- vector< int > SCSerie

  Spacecraft index for data series.

- vector< string > TitleSerie

  Vector of series titles.

- vector< Serie > ListTmpData

  List of stored data series (RAM).

- vector< bool > AlreadyRecDat

  Vector elements are set to 1 if data are already received for the corresponding serie.

- double tStoreData

  Memorization time (RAM) Time during which the data are preserved.

- double tStepRecord

  Time step for recording data.

10.20.2 Constructor & Destructor Documentation

10.20.2.1 MemoryWriteDisk::MemoryWriteDisk ()

Constructs an empty instance.

Definition at line 17 of file LISACODE-MemoryWriteDisk.cpp.

10.20.2.2 MemoryWriteDisk::MemoryWriteDisk (double *tStoreData_n*, double *tStepRecord_n*, char ∗ *NomFichMem_n*)

Constructs an instance and initializes it with default values and tStoreData_n, tStepRecord_n and NomFichMem_n inputs.

Memory constructor is called with tStoreData_n and tStepRecord_n inputs.

Others attributes are set using data read in NomFichMem_n file :

- NomFichMem = NomFichMem_n
- FichMem = manages data of *NomFichMem* file
- SCSerie = empty
- TitleSerie = empty

Definition at line 33 of file LISACODE-MemoryWriteDisk.cpp.

References FichMem, NomFichMem, SCSerie, and TitleSerie.

10.20.2.3 MemoryWriteDisk::∼MemoryWriteDisk ()

Destructor.

Definition at line 43 of file LISACODE-MemoryWriteDisk.cpp.

References FichMem.

10.20.3 Member Function Documentation

10.20.3.1 void MemoryWriteDisk::AddSerieData (int *SerieNumber*, char ∗ *TypeName*, int *IndirectDirName*, int *iSCName*) `[virtual]`

Adds series in ListTmpData, updates corresponding elements in AlreadyRecDat, fills TitleSerie[SerieNumber] attribute and sets SCSerie[SerieNumber].

New series are added in ListTmpData to have *SerieNumber* series in input ListTmpData. Corresponding FALSE flags are added into AlreadyRecDat attribute.

Additionnal series are initialized with 0.0 start value and tStepRecord time step.

TitleSerie[*SerieNumber*] is set using *TypeName* and SCSerie[*SerieNumber*]=*iSCName*.

Reimplemented from Memory.

Definition at line 69 of file LISACODE-MemoryWriteDisk.cpp.

References Memory::AlreadyRecDat, Memory::ListTmpData, SCSerie, TitleSerie, and Memory::tStep-Record.

10.20.3.2 void MemoryWriteDisk::CloseFile ()

Closes FichMem.

Definition at line 167 of file LISACODE-MemoryWriteDisk.cpp.

References FichMem.

10.20.3.3 double Memory::gData (int *SerieNumber*, double *tDelay*, INTERP *InterpolType*, double *InterpUtilValue*) const `[inherited]`

Returns the data of specified serie (*SerieNumber*), delayed by *tDelay* and interpolated using interpolation parameters (*InterpolType*, *InterpUtilValue*).

SerieNumber input is checked : it must be positive or null, and lower than ListTmpData size.

Calls Serie::gData with *tDelay*, *InterpolType* and *InterpUtilValue* inputs.

Definition at line 181 of file LISACODE-Memory.cpp.

References Memory::ListTmpData.

10.20.3.4 double Memory::gData (int *SerieNumber*, double *tDelay*) const `[inherited]`

Returns the data of specified serie (*SerieNumber*) delayed by *tDelay* delay.

SerieNumber input is checked : it must be positive or null, and lower than ListTmpData size

Definition at line 152 of file LISACODE-Memory.cpp.

References LAG, and Memory::ListTmpData.

Referenced by TDI::Compute(), TDI_InterData::ComputeEta(), and TDITools::RefreshDelay().

10.20.3.5 int Memory::getNbSerie () `[inline, inherited]`

Gets the number of the serie.

Definition at line 65 of file LISACODE-Memory.h.

References Memory::ListTmpData.

10.20.3.6 double MemoryWriteDisk::gettMax () `[virtual]`

Gets maximum time.

returned value = tStoreData attribute

Reimplemented from Memory.

Definition at line 55 of file LISACODE-MemoryWriteDisk.cpp.

References Memory::tStoreData.

10.20.3.7 double Memory::gettStepRecord () [inline, inherited]

Gets tStepRecord attribute.

Definition at line 71 of file LISACODE-Memory.h.

References Memory::tStepRecord.

10.20.3.8 double Memory::gettStoreData () [inline, inherited]

Gets tStoreData attribute.

Definition at line 67 of file LISACODE-Memory.h.

References Memory::tStoreData.

10.20.3.9 void MemoryWriteDisk::MakeTitles (char * *FileNameHead* = " ") [virtual]

Copies header from *FileNameHead* file to FichMem.

For all series (index i) in TitleSerie, writes corresponding TitleSerie[i] and SCSerie[i] into FichMem.

Reimplemented from Memory.

Definition at line 97 of file LISACODE-MemoryWriteDisk.cpp.

References FichMem, LISACodeVersion, SCSerie, and TitleSerie.

10.20.3.10 void Memory::ReceiveData (int *SerieNumber*, double *data*) [inherited]

Sets first value of ListTmpData[SerieNumber] to *data* input value and sets AlreadyRecDat[SerieNumber] to TRUE.

SerieNumber input is checked : it must be positive or null, and lower than ListTmpData size

Reimplemented in MemoryReadDisk.

Definition at line 111 of file LISACODE-Memory.cpp.

References Memory::AlreadyRecDat, and Memory::ListTmpData.

Referenced by PhoDetPhaMet::IntegrateSignal(), and main().

10.20.3.11 void MemoryWriteDisk::RecordAccData (double *tStep*, double *t*) [virtual]

Records received data in file FichMem (make it between each step of time).

AlreadyRecDat attributes are checked : it must be TRUE (for all series).

ListTmpData data are written into FichMem (for all series) and last ones are removed by Serie::delLastData.

AlreadyRecDat attributes are set to FALSE (for all series).

tStep input is unused. Time *t* is written into FichMem.

Reimplemented from Memory.

Definition at line 142 of file LISACODE-MemoryWriteDisk.cpp.

References Memory::AlreadyRecDat, FichMem, Memory::ListTmpData, and Memory::tStoreData.

10.20.3.12 void Memory::settStepRecord (double *tStepRecord_n*) `[inherited]`

Sets tStepRecord attribute using *tStepRecord_n* input.

tStepRecord_n input is checked : it is expected to be positive or null.

Definition at line 63 of file LISACODE-Memory.cpp.

References Memory::tStepRecord.

10.20.3.13 void Memory::settStoreData (double *tStoreData_n*) `[inherited]`

Sets tStoreData attribute using *tStoreData_n* input.

tStoreData_n input is checked : it is expected to be positive or null.

Definition at line 52 of file LISACODE-Memory.cpp.

References Memory::tStoreData.

10.20.3.14 int Memory::unusable (double *tSinceFirstReception*) const `[inherited]`

Returns 0 if there are enough stored data to use them.

Returns:

0 if *tSinceFirstReception* input is greater than tStoreData attribute , else 1.

Definition at line 209 of file LISACODE-Memory.cpp.

References Memory::tStoreData.

10.20.4 Member Data Documentation

10.20.4.1 vector<bool> Memory::AlreadyRecDat `[protected, inherited]`

Vector elements are set to 1 if data are already received for the corresponding serie.

Definition at line 48 of file LISACODE-Memory.h.

Referenced by AddSerieData(), MemoryReadDisk::AddSerieData(), Memory::AddSerieData(), Memory::ReceiveData(), RecordAccData(), and Memory::RecordAccData().

10.20.4.2 ofstream MemoryWriteDisk::FichMem `[protected]`

File object managing recording file NomFichMem.

Definition at line 44 of file LISACODE-MemoryWriteDisk.h.

Referenced by CloseFile(), MakeTitles(), MemoryWriteDisk(), RecordAccData(), and ∼MemoryWriteDisk().

10.20.4.3 vector<Serie> Memory::ListTmpData `[protected, inherited]`

List of stored data series (RAM).

Definition at line 46 of file LISACODE-Memory.h.

Referenced by AddSerieData(), MemoryReadDisk::AddSerieData(), Memory::AddSerieData(), Memory::gData(), Memory::getNbSerie(), Memory::ReceiveData(), RecordAccData(), MemoryReadDisk::RecordAccData(), and Memory::RecordAccData().

10.20.4.4 char* MemoryWriteDisk::NomFichMem [protected]

File name on which data are recorded.

Definition at line 42 of file LISACODE-MemoryWriteDisk.h.

Referenced by MemoryWriteDisk().

10.20.4.5 vector<int> MemoryWriteDisk::SCSerie [protected]

Spacecraft index for data series.

Definition at line 46 of file LISACODE-MemoryWriteDisk.h.

Referenced by AddSerieData(), MakeTitles(), and MemoryWriteDisk().

10.20.4.6 vector<string> MemoryWriteDisk::TitleSerie [protected]

Vector of series titles.

Definition at line 48 of file LISACODE-MemoryWriteDisk.h.

Referenced by AddSerieData(), MakeTitles(), and MemoryWriteDisk().

10.20.4.7 double Memory::tStepRecord [protected, inherited]

Time step for recording data.

Definition at line 55 of file LISACODE-Memory.h.

Referenced by AddSerieData(), MemoryReadDisk::AddSerieData(), Memory::AddSerieData(), MemoryReadDisk::gettMax(), Memory::gettStepRecord(), Memory::Memory(), MemoryReadDisk::MemoryReadDisk(), and Memory::settStepRecord().

10.20.4.8 double Memory::tStoreData [protected, inherited]

Memorization time (RAM) Time during which the data are preserved.

Definition at line 53 of file LISACODE-Memory.h.

Referenced by gettMax(), Memory::gettMax(), Memory::gettStoreData(), Memory::Memory(), RecordAccData(), MemoryReadDisk::RecordAccData(), Memory::RecordAccData(), Memory::settStoreData(), and Memory::unusable().

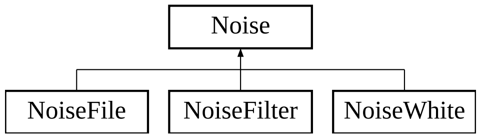
The documentation for this class was generated from the following files:

- LISACODE-MemoryWriteDisk.h
- LISACODE-MemoryWriteDisk.cpp

10.21 Noise Class Reference

```
#include <LISACODE-Noise.h>
```

Inheritance diagram for Noise::



10.21.1 Detailed Description

Noise base class.

It modelises and stores in a memory object any noise (white noise, filtered noise, noise read from a file, etc). Noise samples could be eventually delayed (tDurAdd). The storage is done using a given a physical time step (tStep).

Definition at line 46 of file LISACODE-Noise.h.

Public Member Functions

- Noise ()

  Base constructor.

- Noise (double tStep_n, double tDurAdd_n)

  Constructor.

- Noise (double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n)

  Constructor.

- virtual ∼Noise ()

  Destructor.

- bool TestType (char ∗SubmitType)

  It verifies the type of the noise (NoiseType).

- double gettStep () const

  It returns the time step between two saved data, that is the value of tStep attribute.

- void settStep (double tStep_n)

  It sets tStep value.

- double gettDurAdd () const

  It returns the duration for a noise addition, that is the value of tDurAdd attribute.

- void settDurAdd (double tDurAdd_n)

  It sets tDurAdd value and NbBinAdd.

- double gettFirst () const

  It returns the delay of the first data in data vector, that is the value of tFirst attribute.

- void settFirst (double tFirst_n)

  It sets tFirst.

- double gettLast () const

  It returns the delay of the last data in data vector, that is the value of tLast attribute.

- void settLast (double tLast_n)

  It sets tLast. It verifies that the input argument is a positif factor of tStep, otherwise it shows an error message.

- int getNbBinAdd () const

  It returns the number of bins added, that is the value of NbBinAdd attribute.

- virtual void loadNoise ()

  It initializes noise vector NoiseData.

- virtual void generNoise (int StartBin)

  Default noise generation.

- void addNoise ()

  Appends null noise data corresponding to one measurement into NoiseData attribute.

- double getNoise (double tDelay) const

  It returns the noise value for the specified delay.

Protected Attributes

- double tStep

  Time step in seconds between two saved data. It is used to simulate the continuous signal.

- double tDurAdd

  Noise computation time step (for each measurement).

- double tFirst

  Time of the first data in data vector.

- double tLast

  Time of the last data in data vector.

- int NbBinAdd

  Number of bins (for each measurement).

- int NbData

  Nominal number of data in the noise data vector NoiseData.

- vector< double > NoiseData

  Vector of noise data.

- char NoiseType [30]

  String to describe the noise type.

10.21.2 Constructor & Destructor Documentation

10.21.2.1 Noise::Noise ()

Base constructor.

It sets default values for class attributes and initializes noise vector.

- tStep = 0.01
- tDurAdd = 1.0
- tFirst = 5.0
- tLast = -20.0
- NoiseType = Generic

Todo

Make a function to compute a int from a double. Is there a reason to no call rint ?

Definition at line 33 of file LISACODE-Noise.cpp.

References loadNoise(), NbData, NoiseType, PRECISION, tDurAdd, tFirst, tLast, and tStep.

10.21.2.2 Noise::Noise (double *tStep_n*, double *tDurAdd_n*)

Constructor.

It sets tStep and tDurAdd attributes. The other attributes are set to default values and noise vector is initialized such it is done by the base constructor (Noise::Noise()).

Parameters:

tStep_n Value of tStep.

tDurAdd_n Value of tDurAdd.

Todo

Make a function to compute a int from a double. Is there a reason to no call rint ?

Definition at line 51 of file LISACODE-Noise.cpp.

References loadNoise(), NbData, NoiseType, PRECISION, settDurAdd(), settStep(), tFirst, tLast, and t-Step.

10.21.2.3 Noise::Noise (double *tStep_n*, double *tDurAdd_n*, double *tFirst_n*, double *tLast_n*)

Constructor.

It sets values for class attributes and initializes noise vector such it is done by the base constructor (Noise::Noise()).

NoiseType attribute is set to `Generic`.

Parameters:

tStep_n Value of tStep.

tDurAdd_n Value of tDurAdd.

tFirst_n Value of tFirst.

tLast_n Value of tLast.

Todo

Make a function to compute a int from a double. Is there a reason to no call rint ?

Definition at line 72 of file LISACODE-Noise.cpp.

References loadNoise(), NbData, NoiseType, PRECISION, settDurAdd(), settFirst(), settLast(), settStep(), tFirst, tLast, and tStep.

10.21.2.4 Noise::~Noise () `[virtual]`

Destructor.

It does not any particular action.

Definition at line 94 of file LISACODE-Noise.cpp.

10.21.3 Member Function Documentation

10.21.3.1 void Noise::addNoise ()

Appends null noise data corresponding to one measurement into NoiseData attribute.

NbBinAdd (corresponding to tDurAdd) zeros are inserted at the begining of NoiseData noise vector.

Definition at line 206 of file LISACODE-Noise.cpp.

References generNoise(), NbBinAdd, NbData, and NoiseData.

10.21.3.2 void Noise::generNoise (int *StartBin*) `[virtual]`

Default noise generation.

It sets to zero all the samples before `StartBin`.

Parameters:

StartBin Index before which the values will be set to 0.

Reimplemented in NoiseFile, NoiseFilter, and NoiseWhite.

Definition at line 193 of file LISACODE-Noise.cpp.

References NoiseData.

Referenced by addNoise().

10.21.3.3 int Noise::getNbBinAdd () const `[inline]`

It returns the number of bins added, that is the value of NbBinAdd attribute.

Definition at line 97 of file LISACODE-Noise.h.

References NbBinAdd.

10.21.3.4 double Noise::getNoise (double *tDelay*) const

It returns the noise value for the specified delay.

It transforms `tDelay` value into an index value in NoiseData.

The noise is extracted directly from noise vector (NoiseData) if the specified delay corresponds to a bin. Otherwise, the noise value is interpolated by a Lagrange 7th-order interpolation.

Parameters:

tDelay Delay between noise computation time and current time in seconds (must be negative).

Returns:

If `tDelay` is out of the time range or the values needed for the interpolation are not present, an error message is shown.

WARNING : tDelay must be inferior than tFirst

Todo

Make a function to compute a int from a double. Is there a reason to no call rint ?
Make a function to compute a int from a double. Is there a reason to no call rint ?
Replace code by call to InterLagrange or its optimised function

Definition at line 225 of file LISACODE-Noise.cpp.

References NoiseData, PRECISION, tFirst, and tStep.

10.21.3.5 double Noise::gettDurAdd () const `[inline]`

It returns the duration for a noise addition, that is the value of tDurAdd attribute.

Definition at line 85 of file LISACODE-Noise.h.

References tDurAdd.

10.21.3.6 double Noise::gettFirst () const `[inline]`

It returns the delay of the first data in data vector, that is the value of tFirst attribute.

Definition at line 90 of file LISACODE-Noise.h.

References tFirst.

10.21.3.7 double Noise::gettLast () const [inline]

It returns the delay of the last data in data vector, that is the value of tLast attribute.

Definition at line 94 of file LISACODE-Noise.h.

References tLast.

10.21.3.8 double Noise::gettStep () const [inline]

It returns the time step between two saved data, that is the value of tStep attribute.

Definition at line 81 of file LISACODE-Noise.h.

References tStep.

10.21.3.9 void Noise::loadNoise () [virtual]

It initializes noise vector NoiseData.

It inserts a vector of NbData zeros at the beginig of NoiseData.

Reimplemented in NoiseFile, NoiseFilter, and NoiseWhite.

Definition at line 183 of file LISACODE-Noise.cpp.

References NbData, and NoiseData.

Referenced by Noise().

10.21.3.10 void Noise::settDurAdd (double *tDurAdd_n*)

It sets tDurAdd value and NbBinAdd.

It verifies that the input argument is a positif factor of tStep, otherwise it shows an error message.

Todo

Make a function to verify that a value is an integer. and replace code here after.
Make a function to compute a int from a double. Is there a reason to no call rint ?

Definition at line 127 of file LISACODE-Noise.cpp.

References NbBinAdd, PRECISION, tDurAdd, and tStep.

Referenced by Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), and NoiseWhite::NoiseWhite().

10.21.3.11 void Noise::settFirst (double *tFirst_n*)

It sets tFirst.

It verifies that the input argument is a positif factor of tStep, otherwise it shows an error message.

Todo

Make a function to verify that a value is an integer. and replace code here after.

Definition at line 148 of file LISACODE-Noise.cpp.

References PRECISION, tFirst, and tStep.

Referenced by Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), and NoiseWhite::NoiseWhite().

10.21.3.12 void Noise::settLast (double *tLast_n*)

It sets tLast. It verifies that the input argument is a positif factor of tStep, otherwise it shows an error message.

Todo

Make a function to verifie that a value is an integer. and replace code here after.

Definition at line 163 of file LISACODE-Noise.cpp.

References PRECISION, tLast, and tStep.

Referenced by Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), and NoiseWhite::NoiseWhite().

10.21.3.13 void Noise::settStep (double *tStep_n*)

It sets tStep value.

It verifies that input argument is positive, otherwise it shows an error message.

Definition at line 116 of file LISACODE-Noise.cpp.

References tStep.

Referenced by Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), and NoiseWhite::NoiseWhite().

10.21.3.14 bool Noise::TestType (char * *SubmitType*)

It verifies the type of the noise (NoiseType).

Parameters:

SubmitType String contaning the expected noise type.

Returns:

It returns `true` if the Noise object has the expected type. `false` otherwise.

Definition at line 105 of file LISACODE-Noise.cpp.

References NoiseType.

10.21.4 Member Data Documentation

10.21.4.1 int Noise::NbBinAdd [protected]

Number of bins (for each measurement).

Definition at line 58 of file LISACODE-Noise.h.

Referenced by addNoise(), NoiseFilter::generNoise(), getNbBinAdd(), and settDurAdd().

10.21.4.2 int Noise::NbData [protected]

Nominal number of data in the noise data vector NoiseData.

Definition at line 60 of file LISACODE-Noise.h.

Referenced by addNoise(), NoiseFilter::generNoise(), NoiseWhite::loadNoise(), NoiseFilter::loadNoise(), NoiseFile::loadNoise(), loadNoise(), Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), and NoiseWhite::NoiseWhite().

10.21.4.3 vector<double> Noise::NoiseData [protected]

Vector of noise data.

Definition at line 62 of file LISACODE-Noise.h.

Referenced by addNoise(), NoiseWhite::generNoise(), NoiseFilter::generNoise(), NoiseFile::generNoise(), generNoise(), getNoise(), NoiseWhite::loadNoise(), NoiseFilter::loadNoise(), NoiseFile::loadNoise(), and loadNoise().

10.21.4.4 char Noise::NoiseType[30] [protected]

String to describe the noise type.

Definition at line 64 of file LISACODE-Noise.h.

Referenced by Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseWhite::NoiseWhite(), and TestType().

10.21.4.5 double Noise::tDurAdd [protected]

Noise computation time step (for each measurement).

Definition at line 52 of file LISACODE-Noise.h.

Referenced by gettDurAdd(), Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseWhite::NoiseWhite(), and settDurAdd().

10.21.4.6 double Noise::tFirst [protected]

Time of the first data in data vector.

Definition at line 54 of file LISACODE-Noise.h.

Referenced by getNoise(), gettFirst(), Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseWhite::NoiseWhite(), and settFirst().

10.21.4.7 double Noise::tLast [protected]

Time of the last data in data vector.

Definition at line 56 of file LISACODE-Noise.h.

Referenced by gettLast(), Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseWhite::NoiseWhite(), and settLast().

10.21.4.8 double Noise::tStep [protected]

Time step in seconds between two saved data. It is used to simulate the continuous signal.

Definition at line 50 of file LISACODE-Noise.h.

Referenced by NoiseFilter::generNoise(), getNoise(), NoiseWhite::getPSD(), NoiseWhite::getSqPSD(), gettStep(), NoiseFilter::loadNoise(), NoiseFile::loadNoise(), Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseWhite::NoiseWhite(), NoiseWhite::setSqPSD(), settDurAdd(), settFirst(), settLast(), and settStep().

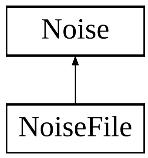
The documentation for this class was generated from the following files:

- LISACODE-Noise.h
- LISACODE-Noise.cpp

10.22 NoiseFile Class Reference

```
#include <LISACODE-NoiseFile.h>
```

Inheritance diagram for NoiseFile::



10.22.1 Detailed Description

Noise derived class to treat files with noise data.

Definition at line 40 of file LISACODE-NoiseFile.h.

Public Member Functions

- NoiseFile ()

  Base constructor.

- NoiseFile (double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, char ∗FileName_-n)

  Constructor.

- char ∗ getFileName ()

  It returns the name of the file where the noise is read, FileName.

- void setFileName (char ∗FileName_n)

  It sets the name of the file where the noise is read, FileName.

- int getNbDataStored ()

  It returns NbDataStored, the number of data stored in the noise data list StoredData.

- void loadNoise ()

  Initializes instance using data read from FileName file.

- void generNoise (int StartBin)

  Noise generation (for one measurement), using Startbin input as beginning index.

- bool TestType (char ∗SubmitType)

  It verifies the type of the noise (NoiseType).

- double gettStep () const

  It returns the time step between two saved data, that is the value of tStep attribute.

- void settStep (double tStep_n)

  It sets tStep value.

- double gettDurAdd () const

  It returns the duration for a noise addition, that is the value of tDurAdd attribute.

- void settDurAdd (double tDurAdd_n)

  It sets tDurAdd value and NbBinAdd.

- double gettFirst () const

  It returns the delay of the first data in data vector, that is the value of tFirst attribute.

- void settFirst (double tFirst_n)

  It sets tFirst.

- double gettLast () const

  It returns the delay of the last data in data vector, that is the value of tLast attribute.

- void settLast (double tLast_n)

  It sets tLast. It verifies that the input argument is a positif factor of tStep, otherwise it shows an error message.

- int getNbBinAdd () const

  It returns the number of bins added, that is the value of NbBinAdd attribute.

- void addNoise ()

  Appends null noise data corresponding to one measurement into NoiseData attribute.

- double getNoise (double tDelay) const

  It returns the noise value for the specified delay.

Protected Attributes

- char ∗ FileName

  Name of the file where the noise is read.

- vector< double > StoredData

  List of noise date read in the file.

- int NbDataStored

  Number of data stored in the noise data list StoredData.

- int ReadBin

  Index of last bin read in StoredData.

- double tStep

  Time step in seconds between two saved data. It is used to simulate the continuous signal.

- double tDurAdd

  Noise computation time step (for each measurement).

- double tFirst

  Time of the first data in data vector.

- double tLast

  Time of the last data in data vector.

- int NbBinAdd

  Number of bins (for each measurement).

- int NbData

  Nominal number of data in the noise data vector NoiseData.

- vector< double > NoiseData

  Vector of noise data.

- char NoiseType [30]

  String to describe the noise type.

10.22.2 Constructor & Destructor Documentation

10.22.2.1 NoiseFile::NoiseFile ()

Base constructor.

It sets default values for class attributes and reads noise data from the file. ?Update sentence in relation to loadNoise understanding : Data read are stored in StoredData and write in Noise::NoiseData.

- tStep = 0.01
- tDurAdd = 1.0
- tFirst = 5.0
- tLast = -20.0
- NoiseType = File
- FileName = DefaultNoise

Todo

Make a function to compute a int from a double. Is there a reason to no call rint ?

Definition at line 28 of file LISACODE-NoiseFile.cpp.

References FileName, loadNoise(), Noise::NbData, Noise::NoiseType, PRECISION, StoredData, Noise::t-DurAdd, Noise::tFirst, Noise::tLast, and Noise::tStep.

10.22.2.2 NoiseFile::NoiseFile (double *tStep_n*, double *tDurAdd_n*, double *tFirst_n*, double *tLast_n*, char ∗ *FileName_n*)

Constructor.

It sets values for class attributes and reads et stores data such it is done by the default constructor (Noise-File::NoiseFile()).

NoiseType attribute is set to `File`.

Parameters:

tStep_n Value of tStep.

tDurAdd_n Value of tDurAdd.

tFirst_n Value of tFirst.

tLast_n Value of tLast.

FileName_n Value of FileName.

Todo

Make a function to compute a int from a double. Is there a reason to no call rint ?

Definition at line 55 of file LISACODE-NoiseFile.cpp.

References FileName, loadNoise(), Noise::NbData, Noise::NoiseType, PRECISION, Noise::settDurAdd(), Noise::settFirst(), Noise::settLast(), Noise::settStep(), StoredData, Noise::tFirst, Noise::tLast, and Noise::t-Step.

10.22.3 Member Function Documentation

10.22.3.1 void Noise::addNoise () `[inherited]`

Appends null noise data corresponding to one measurement into NoiseData attribute.

NbBinAdd (corresponding to tDurAdd) zeros are inserted at the begining of NoiseData noise vector.

Definition at line 206 of file LISACODE-Noise.cpp.

References Noise::generNoise(), Noise::NbBinAdd, Noise::NbData, and Noise::NoiseData.

10.22.3.2 void NoiseFile::generNoise (int *StartBin*) `[virtual]`

Noise generation (for one measurement), using Startbin input as beginning index.

NoiseData StartBin first data are set.

for i=0,…,Startbin $NoiseData[StartBin - i] = \begin{cases} StoredData[i] & \text{if } i \leq NbDataStored \\ StoredData[0] & \text{else} \end{cases}$

Reimplemented from Noise.

Definition at line 188 of file LISACODE-NoiseFile.cpp.

References NbDataStored, Noise::NoiseData, ReadBin, and StoredData.

10.22.3.3 char ∗ NoiseFile::getFileName ()

It returns the name of the file where the noise is read, FileName.

Definition at line 77 of file LISACODE-NoiseFile.cpp.

References FileName.

10.22.3.4 int Noise::getNbBinAdd () const [inline, inherited]

It returns the number of bins added, that is the value of NbBinAdd attribute.

Definition at line 97 of file LISACODE-Noise.h.

References Noise::NbBinAdd.

10.22.3.5 int NoiseFile::getNbDataStored ()

It returns NbDataStored, the number of data stored in the noise data list StoredData.

Definition at line 89 of file LISACODE-NoiseFile.cpp.

References NbDataStored.

10.22.3.6 double Noise::getNoise (double *tDelay*) const [inherited]

It returns the noise value for the specified delay.

It transforms `tDelay` value into an index value in NoiseData.

The noise is extracted directly from noise vector (NoiseData) if the specified delay corresponds to a bin. Otherwise, the noise value is interpolated by a Lagrange 7th-order interpolation.

Parameters:

tDelay Delay between noise computation time and current time in seconds (must be negative).

Returns:

If `tDelay` is out of the time range or the values needed for the interpolation are not present, an error message is shown.

WARNING : tDelay must be inferior than tFirst

Todo

Make a function to compute a int from a double. Is there a reason to no call rint ?
Make a function to compute a int from a double. Is there a reason to no call rint ?
Replace code by call to InterLagrange or its optimised function

Definition at line 225 of file LISACODE-Noise.cpp.

References Noise::NoiseData, PRECISION, Noise::tFirst, and Noise::tStep.

10.22.3.7 double Noise::gettDurAdd () const [inline, inherited]

It returns the duration for a noise addition, that is the value of tDurAdd attribute.

Definition at line 85 of file LISACODE-Noise.h.

References Noise::tDurAdd.

10.22.3.8 double Noise::gettFirst () const [inline, inherited]

It returns the delay of the first data in data vector, that is the value of tFirst attribute.

Definition at line 90 of file LISACODE-Noise.h.

References Noise::tFirst.

10.22.3.9 double Noise::gettLast () const [inline, inherited]

It returns the delay of the last data in data vector, that is the value of tLast attribute.

Definition at line 94 of file LISACODE-Noise.h.

References Noise::tLast.

10.22.3.10 double Noise::gettStep () const [inline, inherited]

It returns the time step between two saved data, that is the value of tStep attribute.

Definition at line 81 of file LISACODE-Noise.h.

References Noise::tStep.

10.22.3.11 void NoiseFile::loadNoise () [virtual]

Initializes instance using data read from FileName file.

Noise computation time step is checked : it must be equal to physical time step.

Noises read from FileName file are stored. Read data size is checked : it must be equal to NbDataStored.

Todo

Make a function to compute a int from a double. Is there a reason to no call rint ?

Reimplemented from Noise.

Definition at line 102 of file LISACODE-NoiseFile.cpp.

References FileName, Noise::NbData, NbDataStored, Noise::NoiseData, PRECISION, ReadBin, StoredData, and Noise::tStep.

Referenced by NoiseFile().

10.22.3.12 void NoiseFile::setFileName (char * *FileName_n*)

It sets the name of the file where the noise is read, FileName.

Definition at line 83 of file LISACODE-NoiseFile.cpp.

References FileName.

10.22.3.13 void Noise::settDurAdd (double *tDurAdd_n*) [inherited]

It sets tDurAdd value and NbBinAdd.

It verifies that the input argument is a positif factor of tStep, otherwise it shows an error message.

Todo

Make a function to verify that a value is an integer. and replace code here after.
Make a function to compute a int from a double. Is there a reason to no call rint ?

Definition at line 127 of file LISACODE-Noise.cpp.

References Noise::NbBinAdd, PRECISION, Noise::tDurAdd, and Noise::tStep.

Referenced by Noise::Noise(), NoiseFile(), NoiseFilter::NoiseFilter(), and NoiseWhite::NoiseWhite().

10.22.3.14 void Noise::settFirst (double *tFirst_n*) [inherited]

It sets tFirst.

It verifies that the input argument is a positif factor of tStep, otherwise it shows an error message.

Todo

Make a function to verify that a value is an integer. and replace code here after.

Definition at line 148 of file LISACODE-Noise.cpp.

References PRECISION, Noise::tFirst, and Noise::tStep.

Referenced by Noise::Noise(), NoiseFile(), NoiseFilter::NoiseFilter(), and NoiseWhite::NoiseWhite().

10.22.3.15 void Noise::settLast (double *tLast_n*) [inherited]

It sets tLast. It verifies that the input argument is a positif factor of tStep, otherwise it shows an error message.

Todo

Make a function to verifie that a value is an integer. and replace code here after.

Definition at line 163 of file LISACODE-Noise.cpp.

References PRECISION, Noise::tLast, and Noise::tStep.

Referenced by Noise::Noise(), NoiseFile(), NoiseFilter::NoiseFilter(), and NoiseWhite::NoiseWhite().

10.22.3.16 void Noise::settStep (double *tStep_n*) [inherited]

It sets tStep value.

It verifies that input argument is positive, otherwise it shows an error message.

Definition at line 116 of file LISACODE-Noise.cpp.

References Noise::tStep.

Referenced by Noise::Noise(), NoiseFile(), NoiseFilter::NoiseFilter(), and NoiseWhite::NoiseWhite().

10.22.3.17 bool Noise::TestType (char * *SubmitType*) [inherited]

It verifies the type of the noise (NoiseType).

Parameters:

SubmitType String contaning the expected noise type.

Returns:

It returns `true` if the Noise object has the expected type. `false` otherwise.

Definition at line 105 of file LISACODE-Noise.cpp.

References Noise::NoiseType.

10.22.4 Member Data Documentation

10.22.4.1 char* NoiseFile::FileName [protected]

Name of the file where the noise is read.

Definition at line 44 of file LISACODE-NoiseFile.h.

Referenced by getFileName(), loadNoise(), NoiseFile(), and setFileName().

10.22.4.2 int Noise::NbBinAdd [protected, inherited]

Number of bins (for each measurement).

Definition at line 58 of file LISACODE-Noise.h.

Referenced by Noise::addNoise(), NoiseFilter::generNoise(), Noise::getNbBinAdd(), and Noise::settDurAdd().

10.22.4.3 int Noise::NbData [protected, inherited]

Nominal number of data in the noise data vector NoiseData.

Definition at line 60 of file LISACODE-Noise.h.

Referenced by Noise::addNoise(), NoiseFilter::generNoise(), NoiseWhite::loadNoise(), NoiseFilter::loadNoise(), loadNoise(), Noise::loadNoise(), Noise::Noise(), NoiseFile(), NoiseFilter::NoiseFilter(), and NoiseWhite::NoiseWhite().

10.22.4.4 int NoiseFile::NbDataStored [protected]

Number of data stored in the noise data list StoredData.

Definition at line 49 of file LISACODE-NoiseFile.h.

Referenced by generNoise(), getNbDataStored(), and loadNoise().

10.22.4.5 vector<double> Noise::NoiseData [protected, inherited]

Vector of noise data.

Definition at line 62 of file LISACODE-Noise.h.

Referenced by Noise::addNoise(), NoiseWhite::generNoise(), NoiseFilter::generNoise(), generNoise(), Noise::generNoise(), Noise::getNoise(), NoiseWhite::loadNoise(), NoiseFilter::loadNoise(), loadNoise(), and Noise::loadNoise().

10.22.4.6 char Noise::NoiseType[30] [protected, inherited]

String to describe the noise type.

Definition at line 64 of file LISACODE-Noise.h.

Referenced by Noise::Noise(), NoiseFile(), NoiseFilter::NoiseFilter(), NoiseWhite::NoiseWhite(), and Noise::TestType().

10.22.4.7 int NoiseFile::ReadBin [protected]

Index of last bin read in StoredData.

Definition at line 51 of file LISACODE-NoiseFile.h.

Referenced by generNoise(), and loadNoise().

10.22.4.8 vector<double> NoiseFile::StoredData [protected]

List of noise date read in the file.

Definition at line 47 of file LISACODE-NoiseFile.h.

Referenced by generNoise(), loadNoise(), and NoiseFile().

10.22.4.9 double Noise::tDurAdd [protected, inherited]

Noise computation time step (for each measurement).

Definition at line 52 of file LISACODE-Noise.h.

Referenced by Noise::gettDurAdd(), Noise::Noise(), NoiseFile(), NoiseFilter::NoiseFilter(), NoiseWhite::NoiseWhite(), and Noise::settDurAdd().

10.22.4.10 double Noise::tFirst [protected, inherited]

Time of the first data in data vector.

Definition at line 54 of file LISACODE-Noise.h.

Referenced by Noise::getNoise(), Noise::gettFirst(), Noise::Noise(), NoiseFile(), NoiseFilter::NoiseFilter(), NoiseWhite::NoiseWhite(), and Noise::settFirst().

10.22.4.11 double Noise::tLast [protected, inherited]

Time of the last data in data vector.

Definition at line 56 of file LISACODE-Noise.h.

Referenced by Noise::gettLast(), Noise::Noise(), NoiseFile(), NoiseFilter::NoiseFilter(), NoiseWhite::NoiseWhite(), and Noise::settLast().

10.22.4.12 double Noise::tStep [protected, inherited]

Time step in seconds between two saved data. It is used to simulate the continuous signal.

Definition at line 50 of file LISACODE-Noise.h.

Referenced by NoiseFilter::generNoise(), Noise::getNoise(), NoiseWhite::getPSD(), NoiseWhite::getSqPSD(), Noise::gettStep(), NoiseFilter::loadNoise(), loadNoise(), Noise::Noise(), NoiseFile(), NoiseFilter::NoiseFilter(), NoiseWhite::NoiseWhite(), NoiseWhite::setSqPSD(), Noise::settDurAdd(), Noise::settFirst(), Noise::settLast(), and Noise::settStep().

The documentation for this class was generated from the following files:

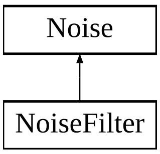
- LISACODE-NoiseFile.h

- LISACODE-NoiseFile.cpp

10.23 NoiseFilter Class Reference

```
#include <LISACODE-NoiseFilter.h>
```

Inheritance diagram for NoiseFilter::



10.23.1 Detailed Description

Noise derived class to treat noise filters.

It creates a noise signal from a filtererd white noise. The filter is given by the next expression:

$$y[n] = \sum_{k=1}^{N_a} alpha[k]y[n-k] + \sum_{k=0}^{N_b} beta[k]x[n-k]$$

where N_a is the number of poles and N_b is the number of zeros in the Z-transform.

Definition at line 47 of file LISACODE-NoiseFilter.h.

Public Member Functions

- NoiseFilter ()

  Base constructor.

- NoiseFilter (double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n)

  Constructor.

- NoiseFilter (double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, Filter NFilter_n)

  Constructor.

- NoiseFilter (double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, vector< vector< double > > FilterAlpha_n, vector< vector< double > > FilterBeta_n)

  Constructor.

- NoiseFilter (double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, vector< vector< double > > FilterAlpha_n, vector< vector< double > > FilterBeta_n, int FilterNbDataStab_n)

  Constructor.

- vector< vector< double > > getFilterAlpha ()

  It returns the list of the alpha parameters (see Filter::alpha) of the filter NFilter.

- vector< vector< double > > getFilterBeta ()

  It returns the list of the beta parameters (see Filter::beta) of the filter NFilter.

- void loadNoise ()

Initialization.

- void generNoise (int StartBin)

  Noise generation (for one measurement), using Startbin input as beginning index.

- bool TestType (char ∗SubmitType)

  It verifies the type of the noise (NoiseType).

- double gettStep () const

  It returns the time step between two saved data, that is the value of tStep attribute.

- void settStep (double tStep_n)

  It sets tStep value.

- double gettDurAdd () const

  It returns the duration for a noise addition, that is the value of tDurAdd attribute.

- void settDurAdd (double tDurAdd_n)

  It sets tDurAdd value and NbBinAdd.

- double gettFirst () const

  It returns the delay of the first data in data vector, that is the value of tFirst attribute.

- void settFirst (double tFirst_n)

  It sets tFirst.

- double gettLast () const

  It returns the delay of the last data in data vector, that is the value of tLast attribute.

- void settLast (double tLast_n)

  It sets tLast. It verifies that the input argument is a positif factor of tStep, otherwise it shows an error message.

- int getNbBinAdd () const

  It returns the number of bins added, that is the value of NbBinAdd attribute.

- void addNoise ()

  Appends null noise data corresponding to one measurement into NoiseData attribute.

- double getNoise (double tDelay) const

  It returns the noise value for the specified delay.

Protected Attributes

- vector< double > WhiteData

  Vector of raw data before filtering.

- Filter NFilter

  Filter for the white noise used to generate the final noise.

- double tStep

  Time step in seconds between two saved data. It is used to simulate the continuous signal.

- double tDurAdd

  Noise computation time step (for each measurement).

- double tFirst

  Time of the first data in data vector.

- double tLast

  Time of the last data in data vector.

- int NbBinAdd

  Number of bins (for each measurement).

- int NbData

  Nominal number of data in the noise data vector NoiseData.

- vector< double > NoiseData

  Vector of noise data.

- char NoiseType [30]

  String to describe the noise type.

10.23.2 Constructor & Destructor Documentation

10.23.2.1 NoiseFilter::NoiseFilter ()

Base constructor.

It sets default values for class attributes. It initializes the filter and applies it to a white noise to generate the final noise (loadNoise).

- tStep = 0.01
- tDurAdd = 1.0
- tFirst = 5.0
- tLast = -20.0
- NoiseType = Filter
- alpha coefficients of NFilter = 0 (see Filter::alpha)
- beta parameter of NFilter = 1 (see Filter::beta)
- NbDataStab parameter of NFilter = 0 (see Filter::NbDataStab)

Todo

Make a function to compute a int from a double. Is there a reason to no call rint ?

Definition at line 32 of file LISACODE-NoiseFilter.cpp.

References Filter::init(), loadNoise(), Noise::NbData, NFilter, Noise::NoiseType, PRECISION, Noise::tDurAdd, Noise::tFirst, Noise::tLast, and Noise::tStep.

10.23.2.2 NoiseFilter::NoiseFilter (double *tStep_n*, double *tDurAdd_n*, double *tFirst_n*, double *tLast_n*)

Constructor.

It sets some class attributes. The other attributes are set to default values (see NoiseFilter::NoiseFilter()). It acts like the base constructor (NoiseFilter::NoiseFilter()).

Parameters:

tStep_n Value of tStep.

tDurAdd_n Value of tDurAdd.

tFirst_n Value of tFirst.

tLast_n Value of tLast.

Todo

Make a function to compute a int from a double. Is there a reason to no call rint ?

cout << " - WhiteData size = " << WhiteData.size() << endl; cout << " - NoiseData size = " << NoiseData.size() << endl; for(int i=0; i<10; i++) cout << " WhiteData[i] = " << WhiteData[i] << " - NoiseData[i] = " << NoiseData[i] << endl;

Definition at line 62 of file LISACODE-NoiseFilter.cpp.

References Filter::init(), loadNoise(), Noise::NbData, NFilter, Noise::NoiseType, PRECISION, Noise::settDurAdd(), Noise::settFirst(), Noise::settLast(), Noise::settStep(), Noise::tFirst, Noise::tLast, and Noise::tStep.

10.23.2.3 NoiseFilter::NoiseFilter (double *tStep_n*, double *tDurAdd_n*, double *tFirst_n*, double *tLast_n*, Filter *NFilter_n*)

Constructor.

It sets class attributes including the noise filter. It applies the filter to a white noise to generate the final noise (see loadNoise)). NoiseType attribute is set to `File`.

Parameters:

tStep_n Value of tStep.

tDurAdd_n Value of tDurAdd.

tFirst_n Value of tFirst.

tLast_n Value of tLast.

NFilter_n Filter for NFilter.

Definition at line 95 of file LISACODE-NoiseFilter.cpp.

References loadNoise(), Noise::NbData, NFilter, Noise::NoiseType, PRECISION, Noise::settDurAdd(), Noise::settFirst(), Noise::settLast(), Noise::settStep(), Noise::tFirst, Noise::tLast, and Noise::tStep.

10.23.2.4 NoiseFilter::NoiseFilter (double *tStep_n*, double *tDurAdd_n*, double *tFirst_n*, double *tLast_n*, vector< vector< double > > *FilterAlpha_n*, vector< vector< double > > *FilterBeta_n*)

Constructor.

It sets some class attributes and provides the alpha (Filter::alpha) and beta (Filter::beta) parameters of the noise filter (NFilter). It initializes the filter using alpha and applies it to a white noise to generate the final noise (loadNoise). NoiseType attribute is set to `File`. NbDataStab parameter of NFilter is set to 0 (see Filter::NbDataStab).

Parameters:

tStep_n Value of tStep.

tDurAdd_n Value of tDurAdd.

tFirst_n Value of tFirst.

tLast_n Value of tLast.

FilterAlpha_n Vector of alpha paramaters for the NFilter (see Filter::alpha).

FilterBeta_n Vector of beta paramaters for the NFilter (see Filter::beta).

Todo

Make a function to compute a int from a double. Is there a reason to no call rint ?

Definition at line 126 of file LISACODE-NoiseFilter.cpp.

References Filter::init(), loadNoise(), Noise::NbData, NFilter, Noise::NoiseType, PRECISION, Noise::settDurAdd(), Noise::settFirst(), Noise::settLast(), Noise::settStep(), Noise::tFirst, Noise::tLast, and Noise::tStep.

10.23.2.5 NoiseFilter::NoiseFilter (double *tStep_n*, double *tDurAdd_n*, double *tFirst_n*, double *tLast_n*, vector< vector< double > > *FilterAlpha_n*, vector< vector< double > > *FilterBeta_n*, int *FilterNbDataStab_n*)

Constructor.

It sets some class attributes and provides the parameters of the noise filter (NFilter). It initializes the filter using alpha, beta and FilterNbDataStab and applies it to a white noise to generate the final noise (loadNoise). NoiseType attribute is set to `File`.

Parameters:

tStep_n Value of tStep.

tDurAdd_n Value of tDurAdd.

tFirst_n Value of tFirst.

tLast_n Value of tLast.

FilterAlpha_n Vector of alpha paramaters for the NFilter (see Filter::alpha).

FilterBeta_n Vector of beta paramaters for the NFilter (see Filter::beta).

FilterNbDataStab_n Paramater for the NFilter (see Filter::NbDataStab).

Todo

Make a function to compute a int from a double. Is there a reason to no call rint ?

Definition at line 158 of file LISACODE-NoiseFilter.cpp.

References Filter::init(), loadNoise(), Noise::NbData, NFilter, Noise::NoiseType, PRECISION, Noise::settDurAdd(), Noise::settFirst(), Noise::settLast(), Noise::settStep(), Noise::tFirst, Noise::tLast, and Noise::tStep.

10.23.3 Member Function Documentation

10.23.3.1 void Noise::addNoise () [inherited]

Appends null noise data corresponding to one measurement into NoiseData attribute.

NbBinAdd (corresponding to tDurAdd) zeros are inserted at the begining of NoiseData noise vector.

Definition at line 206 of file LISACODE-Noise.cpp.

References Noise::generNoise(), Noise::NbBinAdd, Noise::NbData, and Noise::NoiseData.

10.23.3.2 void NoiseFilter::generNoise (int *StartBin*) [virtual]

Noise generation (for one measurement), using Startbin input as beginning index.

NbBinAdd data are inserted in WhiteData.

WhiteData is generated as a standard gaussian :

$$\text{for i=0,\ldots,Startbin } WhiteData[i] = \sqrt{-\frac{log(r1)}{tStep}} \cdot cos(2 \cdot \pi \cdot r1)$$

where r1 and r2 are random values between 0 and 1 (using genunf).

Noise data are generated using Filter::App method with StartBin, WhiteData, and NoiseData arguments.

Then last data are deleted : WhiteData size and NoiseData size are set to NbData.

Reimplemented from Noise.

Definition at line 243 of file LISACODE-NoiseFilter.cpp.

References Filter::App(), genunf(), Noise::NbBinAdd, Noise::NbData, NFilter, Noise::NoiseData, Noise::tStep, and WhiteData.

10.23.3.3 vector< vector <double> > NoiseFilter::getFilterAlpha () [inline]

It returns the list of the alpha parameters (see Filter::alpha) of the filter NFilter.

Definition at line 91 of file LISACODE-NoiseFilter.h.

References Filter::getAlpha(), and NFilter.

10.23.3.4 vector< vector <double> > NoiseFilter::getFilterBeta () [inline]

It returns the list of the beta parameters (see Filter::beta) of the filter NFilter.

Definition at line 93 of file LISACODE-NoiseFilter.h.

References Filter::getBeta(), and NFilter.

10.23.3.5 int Noise::getNbBinAdd () const [inline, inherited]

It returns the number of bins added, that is the value of NbBinAdd attribute.

Definition at line 97 of file LISACODE-Noise.h.

References Noise::NbBinAdd.

10.23.3.6 double Noise::getNoise (double *tDelay*) const `[inherited]`

It returns the noise value for the specified delay.

It transforms `tDelay` value into an index value in NoiseData.

The noise is extracted directly from noise vector (NoiseData) if the specified delay corresponds to a bin. Otherwise, the noise value is interpolated by a Lagrange 7th-order interpolation.

Parameters:

tDelay Delay between noise computation time and current time in seconds (must be negative).

Returns:

If `tDelay` is out of the time range or the values needed for the interpolation are not present, an error message is shown.

WARNING : tDelay must be inferior than tFirst

Todo

Make a function to compute a int from a double. Is there a reason to no call rint ?
Make a function to compute a int from a double. Is there a reason to no call rint ?
Replace code by call to InterLagrange or its optimised function

Definition at line 225 of file LISACODE-Noise.cpp.

References Noise::NoiseData, PRECISION, Noise::tFirst, and Noise::tStep.

10.23.3.7 double Noise::gettDurAdd () const `[inline, inherited]`

It returns the duration for a noise addition, that is the value of tDurAdd attribute.

Definition at line 85 of file LISACODE-Noise.h.

References Noise::tDurAdd.

10.23.3.8 double Noise::gettFirst () const `[inline, inherited]`

It returns the delay of the first data in data vector, that is the value of tFirst attribute.

Definition at line 90 of file LISACODE-Noise.h.

References Noise::tFirst.

10.23.3.9 double Noise::gettLast () const `[inline, inherited]`

It returns the delay of the last data in data vector, that is the value of tLast attribute.

Definition at line 94 of file LISACODE-Noise.h.

References Noise::tLast.

10.23.3.10 double Noise::gettStep () const `[inline, inherited]`

It returns the time step between two saved data, that is the value of tStep attribute.

Definition at line 81 of file LISACODE-Noise.h.

References Noise::tStep.

10.23.3.11 void NoiseFilter::loadNoise () [virtual]

Initialization.

Number of bins must be adjusted depending on filter stabilization :

$$NbDataStab = 2 \cdot NbData + NbDataStab_{NFilter} + Max(size(\alpha_{NFilter}), size(\beta_{NFilter}))$$

WhiteData size and NoiseData size are set to NbDataStab.

WhiteData is generated as a standard gaussian :

for i=0,...,Startbin $WhiteData[i] = \sqrt{-\frac{log(r1)}{tStep}} \cdot cos(2 \cdot \pi \cdot r1)$

where r1 and r2 are random values between 0 and 1 (using genunf).

Noise data are generated using Filter::App method with StartBin, WhiteData, and NoiseData arguments.

Then last data are deleted : WhiteData size and NoiseData size are set to NbData.

Reimplemented from Noise.

Definition at line 197 of file LISACODE-NoiseFilter.cpp.

References Filter::App(), genunf(), Filter::getDepth(), Filter::getNbDataStab(), Noise::NbData, NFilter, Noise::NoiseData, Noise::tStep, and WhiteData.

Referenced by NoiseFilter().

10.23.3.12 void Noise::settDurAdd (double *tDurAdd_n*) [inherited]

It sets tDurAdd value and NbBinAdd.

It verifies that the input argument is a positif factor of tStep, otherwise it shows an error message.

Todo

Make a function to verify that a value is an integer. and replace code here after.
Make a function to compute a int from a double. Is there a reason to no call rint ?

Definition at line 127 of file LISACODE-Noise.cpp.

References Noise::NbBinAdd, PRECISION, Noise::tDurAdd, and Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter(), and NoiseWhite::NoiseWhite().

10.23.3.13 void Noise::settFirst (double *tFirst_n*) [inherited]

It sets tFirst.

It verifies that the input argument is a positif factor of tStep, otherwise it shows an error message.

Todo

Make a function to verify that a value is an integer. and replace code here after.

Definition at line 148 of file LISACODE-Noise.cpp.

References PRECISION, Noise::tFirst, and Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter(), and NoiseWhite::NoiseWhite().

10.23.3.14 void Noise::settLast (double *tLast_n*) [inherited]

It sets tLast. It verifies that the input argument is a positif factor of tStep, otherwise it shows an error message.

Todo

Make a function to verifie that a value is an integer. and replace code here after.

Definition at line 163 of file LISACODE-Noise.cpp.

References PRECISION, Noise::tLast, and Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter(), and NoiseWhite::NoiseWhite().

10.23.3.15 void Noise::settStep (double *tStep_n*) [inherited]

It sets tStep value.

It verifies that input argument is positive, otherwise it shows an error message.

Definition at line 116 of file LISACODE-Noise.cpp.

References Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter(), and NoiseWhite::NoiseWhite().

10.23.3.16 bool Noise::TestType (char * *SubmitType*) [inherited]

It verifies the type of the noise (NoiseType).

Parameters:

SubmitType String contaning the expected noise type.

Returns:

It returns `true` if the Noise object has the expected type. `false` otherwise.

Definition at line 105 of file LISACODE-Noise.cpp.

References Noise::NoiseType.

10.23.4 Member Data Documentation

10.23.4.1 int Noise::NbBinAdd [protected, inherited]

Number of bins (for each measurement).

Definition at line 58 of file LISACODE-Noise.h.

Referenced by Noise::addNoise(), generNoise(), Noise::getNbBinAdd(), and Noise::settDurAdd().

10.23.4.2 int Noise::NbData [protected, inherited]

Nominal number of data in the noise data vector NoiseData.

Definition at line 60 of file LISACODE-Noise.h.

Referenced by Noise::addNoise(), generNoise(), NoiseWhite::loadNoise(), loadNoise(), NoiseFile::loadNoise(), Noise::loadNoise(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter(), and NoiseWhite::NoiseWhite().

10.23.4.3 Filter NoiseFilter::NFilter [protected]

Filter for the white noise used to generate the final noise.

Definition at line 55 of file LISACODE-NoiseFilter.h.

Referenced by generNoise(), getFilterAlpha(), getFilterBeta(), loadNoise(), and NoiseFilter().

10.23.4.4 vector<double> Noise::NoiseData [protected, inherited]

Vector of noise data.

Definition at line 62 of file LISACODE-Noise.h.

Referenced by Noise::addNoise(), NoiseWhite::generNoise(), generNoise(), NoiseFile::generNoise(), Noise::generNoise(), Noise::getNoise(), NoiseWhite::loadNoise(), loadNoise(), NoiseFile::loadNoise(), and Noise::loadNoise().

10.23.4.5 char Noise::NoiseType[30] [protected, inherited]

String to describe the noise type.

Definition at line 64 of file LISACODE-Noise.h.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter(), NoiseWhite::NoiseWhite(), and Noise::TestType().

10.23.4.6 double Noise::tDurAdd [protected, inherited]

Noise computation time step (for each measurement).

Definition at line 52 of file LISACODE-Noise.h.

Referenced by Noise::gettDurAdd(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter(), NoiseWhite::NoiseWhite(), and Noise::settDurAdd().

10.23.4.7 double Noise::tFirst [protected, inherited]

Time of the first data in data vector.

Definition at line 54 of file LISACODE-Noise.h.

Referenced by Noise::getNoise(), Noise::gettFirst(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter(), NoiseWhite::NoiseWhite(), and Noise::settFirst().

10.23.4.8 double Noise::tLast [protected, inherited]

Time of the last data in data vector.

Definition at line 56 of file LISACODE-Noise.h.

Referenced by Noise::gettLast(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter(), NoiseWhite::NoiseWhite(), and Noise::settLast().

10.23.4.9 double Noise::tStep [protected, inherited]

Time step in seconds between two saved data. It is used to simulate the continuous signal.

Definition at line 50 of file LISACODE-Noise.h.

Referenced by generNoise(), Noise::getNoise(), NoiseWhite::getPSD(), NoiseWhite::getSqPSD(), Noise::gettStep(), loadNoise(), NoiseFile::loadNoise(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter(), NoiseWhite::NoiseWhite(), NoiseWhite::setSqPSD(), Noise::settDurAdd(), Noise::settFirst(), Noise::settLast(), and Noise::settStep().

10.23.4.10 vector<double> NoiseFilter::WhiteData [protected]

Vector of raw data before filtering.

White noise data are generated with a standard gaussian.

Definition at line 53 of file LISACODE-NoiseFilter.h.

Referenced by generNoise(), and loadNoise().

The documentation for this class was generated from the following files:

- LISACODE-NoiseFilter.h
- LISACODE-NoiseFilter.cpp

10.24 NoiseSpec Struct Reference

```
#include <LISACODE-ConfigSim.h>
```

10.24.1 Detailed Description

Noise specification structure.

Definition at line 57 of file LISACODE-ConfigSim.h.

Public Attributes

- int NType
  Type of Noise.
- double NVal0
  Noise level (used if NType=4,5 or 6).
- vector< double > NVal1
  α recursive coefficients (used if NType=3)
- vector< double > NVal2
  β recursive coefficients (used if NType=3)
- char NStr [256]
  Filename (used if NType=2).

10.24.2 Member Data Documentation

10.24.2.1 char NoiseSpec::NStr[256]

Filename (used if NType=2).

Definition at line 77 of file LISACODE-ConfigSim.h.

Referenced by ConfigSim::NoisesCreation(), ConfigSim::ReadASCIIFile(), and ConfigSim::ReadXMLFile().

10.24.2.2 int NoiseSpec::NType

Type of Noise.

The possible values for the nois type are:

- 0: No noise.
- 1: White noise.
- 2: Noise read from a file.
- 3: Noise generated by filtering a white noise (undefined filter : parameters are given).

- 4: Noise generated by filtering a white noise (filter parameters are computed for 1/frequency noise; only noise level is given).
- 5: Noise generated by filtering a white noise (filter parameters are computed for frequency noise; only noise level is given).
- 6: Noise generated by filtering a white noise (filter parameters are computed for frequency noise; only noise level is given; noise is proportional to arms length and square root power too).

Definition at line 69 of file LISACODE-ConfigSim.h.

Referenced by ConfigSim::NoisesCreation(), ConfigSim::ReadASCIIFile(), and ConfigSim::ReadXMLFile().

10.24.2.3 double NoiseSpec::NVal0

Noise level (used if NType=4,5 or 6).

Definition at line 71 of file LISACODE-ConfigSim.h.

Referenced by ConfigSim::NoisesCreation(), ConfigSim::ReadASCIIFile(), and ConfigSim::ReadXMLFile().

10.24.2.4 vector<double> NoiseSpec::NVal1

α recursive coefficients (used if NType=3)

Definition at line 73 of file LISACODE-ConfigSim.h.

Referenced by ConfigSim::NoisesCreation(), ConfigSim::ReadASCIIFile(), and ConfigSim::ReadXMLFile().

10.24.2.5 vector<double> NoiseSpec::NVal2

β recursive coefficients (used if NType=3)

Definition at line 75 of file LISACODE-ConfigSim.h.

Referenced by ConfigSim::NoisesCreation(), ConfigSim::ReadASCIIFile(), and ConfigSim::ReadXMLFile().

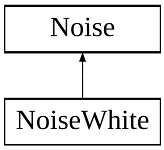
The documentation for this struct was generated from the following file:

- LISACODE-ConfigSim.h

10.25 NoiseWhite Class Reference

`#include <LISACODE-NoiseWhite.h>`

Inheritance diagram for NoiseWhite::



10.25.1 Detailed Description

Noise derived class to treat white noise.

It creates a white noise using a given σ. The σ value is obtained from the power spectral density following the next expression:

$$\sigma = \sqrt{\frac{PSD}{2 \cdot tStep}}$$

where PSD is the power spectral density and $tStep$ is the time between two samples.

Definition at line 46 of file LISACODE-NoiseWhite.h.

Public Member Functions

- NoiseWhite ()

    Base constructor.

- NoiseWhite (double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n)

    Constructor.

- NoiseWhite (double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, double SqPSD)

    Constructor.

- double getPSD ()
- double getSqPSD ()
- void setSqPSD (double SqPSD)
- void loadNoise ()

    Initialization.

- void generNoise (int StartBin)

    Noise generation (for one measurement), using Startbin input as beginning index.

- bool TestType (char ∗SubmitType)

    It verifies the type of the noise (NoiseType).

- double gettStep () const

    It returns the time step between two saved data, that is the value of tStep attribute.

- void settStep (double tStep_n)

  It sets tStep value.

- double gettDurAdd () const

  It returns the duration for a noise addition, that is the value of tDurAdd attribute.

- void settDurAdd (double tDurAdd_n)

  It sets tDurAdd value and NbBinAdd.

- double gettFirst () const

  It returns the delay of the first data in data vector, that is the value of tFirst attribute.

- void settFirst (double tFirst_n)

  It sets tFirst.

- double gettLast () const

  It returns the delay of the last data in data vector, that is the value of tLast attribute.

- void settLast (double tLast_n)

  It sets tLast. It verifies that the input argument is a positif factor of tStep, otherwise it shows an error message.

- int getNbBinAdd () const

  It returns the number of bins added, that is the value of NbBinAdd attribute.

- void addNoise ()

  Appends null noise data corresponding to one measurement into NoiseData attribute.

- double getNoise (double tDelay) const

  It returns the noise value for the specified delay.

Protected Attributes

- double Sigma

  White noise standard deviation.

- double tStep

  Time step in seconds between two saved data. It is used to simulate the continuous signal.

- double tDurAdd

  Noise computation time step (for each measurement).

- double tFirst

  Time of the first data in data vector.

- double tLast

  Time of the last data in data vector.

- int NbBinAdd

Number of bins (for each measurement).

- int NbData

  Nominal number of data in the noise data vector NoiseData.

- vector< double > NoiseData

  Vector of noise data.

- char NoiseType [30]

  String to describe the noise type.

10.25.2 Constructor & Destructor Documentation

10.25.2.1 NoiseWhite::NoiseWhite ()

Base constructor.

It sets default values for class attributes and generate white noise vector (see loadNoise).

Sigma(σ) is computed by setSqPSD from the root square of the PSD set to 1.0e-13.

- tStep = 0.01
- tDurAdd = 1.0
- tFirst = 5.0
- tLast = -20.0
- NoiseType = White

Todo

Make a function to compute a int from a double. Is there a reason to no call rint ?

Definition at line 27 of file LISACODE-NoiseWhite.cpp.

References loadNoise(), Noise::NbData, Noise::NoiseType, PRECISION, setSqPSD(), Noise::tDurAdd, Noise::tFirst, Noise::tLast, and Noise::tStep.

10.25.2.2 NoiseWhite::NoiseWhite (double *tStep_n*, double *tDurAdd_n*, double *tFirst_n*, double *tLast_n*)

Constructor.

It sets values for class attributes and computes a white noise vector such it is done by the base constructor (NoiseWhite::NoiseWhite()). Sigma(σ) is computed from a by default PSD value.

NoiseType attribute is set to `White`.

Sigma(σ) is computed by setSqPSD from the root square of the PSD set to 1.0e-13.

Parameters:

tStep_n Value of tStep.

tDurAdd_n Value of tDurAdd.

tFirst_n Value of tFirst.

tLast_n Value of tLast.

Todo

Make a function to compute a int from a double. Is there a reason to no call rint ?

Definition at line 52 of file LISACODE-NoiseWhite.cpp.

References loadNoise(), Noise::NbData, Noise::NoiseType, PRECISION, setSqPSD(), Noise::settDurAdd(), Noise::settFirst(), Noise::settLast(), Noise::settStep(), Noise::tFirst, Noise::tLast, and Noise::tStep.

10.25.2.3 NoiseWhite::NoiseWhite (double *tStep_n*, double *tDurAdd_n*, double *tFirst_n*, double *tLast_n*, double *SqPSD*)

Constructor.

It sets values for class attributes and computes a white noise vector such it is done by the base constructor (NoiseWhite::NoiseWhite()).

NoiseType attribute is set to `White`.

Parameters:

tStep_n Value of tStep.

tDurAdd_n Value of tDurAdd.

tFirst_n Value of tFirst.

tLast_n Value of tLast.

SqPSD Value of root square to compute Sigma (see setSqPSD()).

Todo

Make a function to compute a int from a double. Is there a reason to no call rint ?

Definition at line 81 of file LISACODE-NoiseWhite.cpp.

References loadNoise(), Noise::NbData, Noise::NoiseType, PRECISION, setSqPSD(), Noise::settDurAdd(), Noise::settFirst(), Noise::settLast(), Noise::settStep(), Noise::tFirst, Noise::tLast, and Noise::tStep.

10.25.3 Member Function Documentation

10.25.3.1 void Noise::addNoise () `[inherited]`

Appends null noise data corresponding to one measurement into NoiseData attribute.

NbBinAdd (corresponding to tDurAdd) zeros are inserted at the begining of NoiseData noise vector.

Definition at line 206 of file LISACODE-Noise.cpp.

References Noise::generNoise(), Noise::NbBinAdd, Noise::NbData, and Noise::NoiseData.

10.25.3.2 void NoiseWhite::generNoise (int *StartBin*) `[virtual]`

Noise generation (for one measurement), using Startbin input as beginning index.

NoiseData size is set to NbData.

NoiseData is generated as below :

$$\text{for i=0,\ldots,StartBin } WhiteData[i] = \sigma \cdot \sqrt{-2 \cdot log(r2)} \cdot cos(2 \cdot \pi \cdot r1)$$

where r1 and r2 are random values between 0 and 1 (using genunf).

Reimplemented from Noise.

Definition at line 170 of file LISACODE-NoiseWhite.cpp.

References genunf(), Noise::NoiseData, and Sigma.

10.25.3.3 int Noise::getNbBinAdd () const `[inline, inherited]`

It returns the number of bins added, that is the value of NbBinAdd attribute.

Definition at line 97 of file LISACODE-Noise.h.

References Noise::NbBinAdd.

10.25.3.4 double Noise::getNoise (double *tDelay*) const `[inherited]`

It returns the noise value for the specified delay.

It transforms `tDelay` value into an index value in NoiseData.

The noise is extracted directly from noise vector (NoiseData) if the specified delay corresponds to a bin. Otherwise, the noise value is interpolated by a Lagrange 7th-order interpolation.

Parameters:

tDelay Delay between noise computation time and current time in seconds (must be negative).

Returns:

If `tDelay` is out of the time range or the values needed for the interpolation are not present, an error message is shown.

WARNING : tDelay must be inferior than tFirst

Todo

Make a function to compute a int from a double. Is there a reason to no call rint ?
Make a function to compute a int from a double. Is there a reason to no call rint ?
Replace code by call to InterLagrange or its optimised function

Definition at line 225 of file LISACODE-Noise.cpp.

References Noise::NoiseData, PRECISION, Noise::tFirst, and Noise::tStep.

10.25.3.5 double NoiseWhite::getPSD ()

It returns the power spectrum density (PSD) from Sigma(σ) value. The equation used to calculate the PSD is:

$$PSD = \sigma^2 \cdot 2.0 \cdot tStep$$

where $tStep$ is the time between to samples.

Definition at line 105 of file LISACODE-NoiseWhite.cpp.

References Sigma, and Noise::tStep.

10.25.3.6 double NoiseWhite::getSqPSD ()

It returns the root square of power spectrum density (SqPSD). It is computed from Sigma(σ) value using the equation:

$$SqPSD = \sigma \cdot \sqrt{2.0 \cdot tStep}$$

where $tStep$ is the time between to samples.

Definition at line 116 of file LISACODE-NoiseWhite.cpp.

References Sigma, and Noise::tStep.

10.25.3.7 double Noise::gettDurAdd () const `[inline, inherited]`

It returns the duration for a noise addition, that is the value of tDurAdd attribute.

Definition at line 85 of file LISACODE-Noise.h.

References Noise::tDurAdd.

10.25.3.8 double Noise::gettFirst () const `[inline, inherited]`

It returns the delay of the first data in data vector, that is the value of tFirst attribute.

Definition at line 90 of file LISACODE-Noise.h.

References Noise::tFirst.

10.25.3.9 double Noise::gettLast () const `[inline, inherited]`

It returns the delay of the last data in data vector, that is the value of tLast attribute.

Definition at line 94 of file LISACODE-Noise.h.

References Noise::tLast.

10.25.3.10 double Noise::gettStep () const `[inline, inherited]`

It returns the time step between two saved data, that is the value of tStep attribute.

Definition at line 81 of file LISACODE-Noise.h.

References Noise::tStep.

10.25.3.11 void NoiseWhite::loadNoise () `[virtual]`

Initialization.

NoiseData size is set to NbData.

NoiseData is generated as below :

for i=0,…,NbData $WhiteData[i] = \sigma \cdot \sqrt{-2 \cdot log(r2)} \cdot cos(2 \cdot \pi \cdot r1)$

where r1 and r2 are random values between 0 and 1 (using genunf).

Reimplemented from Noise.

Definition at line 143 of file LISACODE-NoiseWhite.cpp.

References genunf(), Noise::NbData, Noise::NoiseData, and Sigma.

Referenced by NoiseWhite().

10.25.3.12 void NoiseWhite::setSqPSD (double *SqPSD*)

It sets Sigma(σ) by giving the root square of power spectrum density (PSD). Sigma(σ) is computed by using the equation:

$$\sigma = \frac{SqPSD}{\sqrt{2.0 \cdot tStep}}$$

where $tStep$ tStep is the time between to samples. If $SqPSD$ is negatif a message is shown.

Definition at line 126 of file LISACODE-NoiseWhite.cpp.

References Sigma, and Noise::tStep.

Referenced by NoiseWhite().

10.25.3.13 void Noise::settDurAdd (double *tDurAdd_n*) `[inherited]`

It sets tDurAdd value and NbBinAdd.

It verifies that the input argument is a positif factor of tStep, otherwise it shows an error message.

Todo

Make a function to verify that a value is an integer. and replace code here after.
Make a function to compute a int from a double. Is there a reason to no call rint ?

Definition at line 127 of file LISACODE-Noise.cpp.

References Noise::NbBinAdd, PRECISION, Noise::tDurAdd, and Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), and NoiseWhite().

10.25.3.14 void Noise::settFirst (double *tFirst_n*) `[inherited]`

It sets tFirst.

It verifies that the input argument is a positif factor of tStep, otherwise it shows an error message.

Todo

Make a function to verify that a value is an integer. and replace code here after.

Definition at line 148 of file LISACODE-Noise.cpp.

References PRECISION, Noise::tFirst, and Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), and NoiseWhite().

10.25.3.15 void Noise::settLast (double *tLast_n*) `[inherited]`

It sets tLast. It verifies that the input argument is a positif factor of tStep, otherwise it shows an error message.

Todo

Make a function to verifie that a value is an integer. and replace code here after.

Definition at line 163 of file LISACODE-Noise.cpp.

References PRECISION, Noise::tLast, and Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), and NoiseWhite().

10.25.3.16 void Noise::settStep (double *tStep_n*) `[inherited]`

It sets tStep value.

It verifies that input argument is positive, otherwise it shows an error message.

Definition at line 116 of file LISACODE-Noise.cpp.

References Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), and NoiseWhite().

10.25.3.17 bool Noise::TestType (char * *SubmitType*) `[inherited]`

It verifies the type of the noise (NoiseType).

Parameters:

SubmitType String contaning the expected noise type.

Returns:

It returns `true` if the Noise object has the expected type. `false` otherwise.

Definition at line 105 of file LISACODE-Noise.cpp.

References Noise::NoiseType.

10.25.4 Member Data Documentation

10.25.4.1 int Noise::NbBinAdd `[protected, inherited]`

Number of bins (for each measurement).

Definition at line 58 of file LISACODE-Noise.h.

Referenced by Noise::addNoise(), NoiseFilter::generNoise(), Noise::getNbBinAdd(), and Noise::settDurAdd().

10.25.4.2 int Noise::NbData `[protected, inherited]`

Nominal number of data in the noise data vector NoiseData.

Definition at line 60 of file LISACODE-Noise.h.

Referenced by Noise::addNoise(), NoiseFilter::generNoise(), loadNoise(), NoiseFilter::loadNoise(), NoiseFile::loadNoise(), Noise::loadNoise(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), and NoiseWhite().

10.25.4.3 vector<double> Noise::NoiseData [protected, inherited]

Vector of noise data.

Definition at line 62 of file LISACODE-Noise.h.

Referenced by Noise::addNoise(), generNoise(), NoiseFilter::generNoise(), NoiseFile::generNoise(), Noise::generNoise(), Noise::getNoise(), loadNoise(), NoiseFilter::loadNoise(), NoiseFile::loadNoise(), and Noise::loadNoise().

10.25.4.4 char Noise::NoiseType[30] [protected, inherited]

String to describe the noise type.

Definition at line 64 of file LISACODE-Noise.h.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseWhite(), and Noise::TestType().

10.25.4.5 double NoiseWhite::Sigma [protected]

White noise standard deviation.

Definition at line 51 of file LISACODE-NoiseWhite.h.

Referenced by generNoise(), getPSD(), getSqPSD(), loadNoise(), and setSqPSD().

10.25.4.6 double Noise::tDurAdd [protected, inherited]

Noise computation time step (for each measurement).

Definition at line 52 of file LISACODE-Noise.h.

Referenced by Noise::gettDurAdd(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseWhite(), and Noise::settDurAdd().

10.25.4.7 double Noise::tFirst [protected, inherited]

Time of the first data in data vector.

Definition at line 54 of file LISACODE-Noise.h.

Referenced by Noise::getNoise(), Noise::gettFirst(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseWhite(), and Noise::settFirst().

10.25.4.8 double Noise::tLast [protected, inherited]

Time of the last data in data vector.

Definition at line 56 of file LISACODE-Noise.h.

Referenced by Noise::gettLast(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseWhite(), and Noise::settLast().

10.25.4.9 double Noise::tStep [protected, inherited]

Time step in seconds between two saved data. It is used to simulate the continuous signal.

Definition at line 50 of file LISACODE-Noise.h.

Referenced by NoiseFilter::generNoise(), Noise::getNoise(), getPSD(), getSqPSD(), Noise::gettStep(), NoiseFilter::loadNoise(), NoiseFile::loadNoise(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseWhite(), setSqPSD(), Noise::settDurAdd(), Noise::settFirst(), Noise::settLast(), and Noise::settStep().

The documentation for this class was generated from the following files:

- LISACODE-NoiseWhite.h
- LISACODE-NoiseWhite.cpp

10.26 PhoDetPhaMet Class Reference

`#include <LISACODE-PhoDetPhaMet.h>`

10.26.1 Detailed Description

Phasemeter photodiode class.

All noises are phase differences.

Noises are combined (addition, substraction, delay).

Gravitational Wave contribution is added.

Noises sampling is equal to physical time step.

Phasemeter output is phase difference between 2 beams received by photodiode; its sampling is equal to measurement time step.

Low pass filter keeps only frequencies lower than half measurement frequency.

Simulator time step is equal to measurement step, so that 1 phasemeter signal data is stored in memory when 1 signal data is received.

Definition at line 77 of file LISACODE-PhoDetPhaMet.h.

Public Member Functions

- PhoDetPhaMet ()

  Constructs an instance and initializes it with default values.

- PhoDetPhaMet (PDPMINTERF InterfType_n, int IndirectDir_n, int iSC_n, Geometry ∗SCPos_n, vector< Noise ∗ > ∗NPs_n, USOClock ∗USO_n, Memory ∗RecordData_n, double tStepPhy_n, double tStepMes_n)

  Constructs an instance and initializes it with default values and inputs.

- PhoDetPhaMet (PDPMINTERF InterfType_n, int IndirectDir_n, int iSC_n, Geometry ∗SCPos_n, vector< Noise ∗ > ∗NPs_n, USOClock ∗USO_n, Memory ∗RecordData_n, TrFctGW ∗sGW_n, Background ∗GWB_n, double tStepPhy_n, double tStepMes_n)

  *Constructs an instance and initializes it with default values and inputs. It is like previous constructor with added inputs for the transfer function (*sGW_n*) and the background (*GWB_n*).*

- PhoDetPhaMet (PDPMINTERF InterfType_n, int IndirectDir_n, int iSC_n, Geometry ∗SCPos_n, vector< Noise ∗ > ∗NPs_n, USOClock ∗USO_n, Memory ∗RecordData_n, TrFctGW ∗sGW_n, Background ∗GWB_n, double tStepPhy_n, double tStepMes_n, bool FilterON_n, vector< double > FilterParam_n)

  Constructs an instance and initializes it with inputs. It is like previous constructor with added inputs for filter description.

- ∼PhoDetPhaMet ()

  Destructor.

- void init (PDPMINTERF InterfType_n, int IndirectDir_n, int iSC_n, Geometry ∗SCPos_n, vector< Noise ∗ > ∗NPs_n, USOClock ∗USO_n, Memory ∗RecordData_n, TrFctGW ∗sGW_n, Background ∗GWB_n, double tStepPhy_n, double tStepMes_n, bool FilterON_n, vector< double > FilterParam_n)

Initializes an instance with default values and inputs.

- double getIndirectDir () const

  Returns IndirectDir attribute.

- double getiSC () const

  Returns iSC attribute.

- void DisplayStoredData ()

  Displays stored data.

- double gettStab ()

  Gets stabilization time.

- double gN (NOISEORIG OrigN, int iSC, int IndirectDir, double tDelay)

  Returns value of specified noise after delay computation.

- double gGWB (int iSC, int IndirectDir, double t)

  Returns value of Gravitationnal Wave Background (iSC={1,2,3}, IndirectDir={0,1}).

- void ReceiveSignal (double t)

  Computes the signal received by photodetector-phasemeters.

- void IntegrateSignal (double t)

  Stores the result in memory (one measurement).

- bool getNoNoise ()

  Indicates if noises are present or not. It returns true if there are no noises.

Protected Attributes

- PDPMINTERF InterfType

  Type of interferences made by the photodetector-phasemeter.

- int IndirectDir

  Direction flag : 0 if the optical bench is in the direct direction, else 1.

- int iSC

  Spacecraft index corresponding to the photodetector-phasemeter.

- Geometry $*$ SCPos

  Pointer to LISA geometry.

- vector$<$ Noise $*$ $>$ $*$ NPs

  Noise pointers vector.

- USOClock $*$ USO

  Pointers to spacecrafts USO clocks.

- Memory ∗ RecordData

  Pointer to the storage memory of the photodetector-phasemeters signal.

- TrFctGW ∗ sGW

  Pointer to the transfer function.

- Background ∗ GWB

  Background pointer : confusion whites dwarfs background.

- double tStepPhy

  Physical simulation time step.

- double tStepMes

  Measurement time step.

- vector< double > InterfPhyData

  Physical data vector.

- vector< double > FilterPhyData

  Filtered physical data vector.

- int NbDataStored

  Number of data stored (in InterfPhyData and FilterPhyData).

- int NbDataAdd

  Number of data added (in InterfPhyData and FilterPhyData) for each measurement.

- Filter ∗ PBFilter

  Filter pointer to a low pass filter.

- bool FilterON

  Filter flag : If true, the filter is applied.

- vector< double > FilterParam

  Filter parameters : attenuation [dB], oscillations in bandwidth [dB], low transition frequency divided by measurement frequency, high transition frequency divided by measurement frequency.

- bool NoNoise

  Noise flag : if true, there is no noise.

10.26.2 Constructor & Destructor Documentation

10.26.2.1 PhoDetPhaMet::PhoDetPhaMet ()

Constructs an instance and initializes it with default values.

init method is called with the following arguments :

- Interf_type_n = S

- IndirectDir_n = 0
- iSC_n = 1
- SCPos_n = empty
- NPs_n = 6 empty Noise vectors
- USO_n = empty
- RecordData_n = empty
- sGW_n = empty
- GWB_n = empty
- tStepPhy_n = 0.01
- tStepMes_n = 1.0
- FilterON_n = FALSE
- attenuation : FilterParam_n[0] = 140.0 dB
- oscillations in bandwidth FilterParam_n[1] = 0.1 dB
- low transition frequency / measurement frequency : FilterParam_n[2] = 0.1
- high transition frequency / measurement frequency : FilterParam_n[3] = 0.3

Definition at line 38 of file LISACODE-PhoDetPhaMet.cpp.

References init(), and S.

10.26.2.2 PhoDetPhaMet::PhoDetPhaMet (PDPMINTERF *InterfType_n*, int *IndirectDir_n*, int *iSC_n*, Geometry ∗ *SCPos_n*, vector< Noise ∗ > ∗ *NPs_n*, USOClock ∗ *USO_n*, Memory ∗ *RecordData_n*, double *tStepPhy_n*, double *tStepMes_n*)

Constructs an instance and initializes it with default values and inputs.

init method is called with the following arguments :

- Interf_type_n input
- IndirectDir_n input
- iSC_n input
- SCPos_n input
- NPs_n input
- USO_n input
- RecordData_n input
- sGW_n = empty
- GWB_n = empty
- tStepPhy_n input

- tStepMes_n input
- FilterON_n = TRUE
- attenuation : FilterParam_n[0] = 140.0 dB
- oscillations in bandwidth FilterParam_n[1] = 0.1 dB
- low transition frequency / measurement frequency : FilterParam_n[2] = 0.1
- high transition frequency / measurement frequency : FilterParam_n[3] = 0.3

Definition at line 79 of file LISACODE-PhoDetPhaMet.cpp.

References init().

10.26.2.3 PhoDetPhaMet::PhoDetPhaMet (PDPMINTERF *InterfType_n*, int *IndirectDir_n*, int *iSC_n*, Geometry ∗ *SCPos_n*, vector< Noise ∗ > ∗ *NPs_n*, USOClock ∗ *USO_n*, Memory ∗ *RecordData_n*, TrFctGW ∗ *sGW_n*, Background ∗ *GWB_n*, double *tStepPhy_n*, double *tStepMes_n*)

Constructs an instance and initializes it with default values and inputs. It is like previous constructor with added inputs for the transfer function (*sGW_n*) and the background (*GWB_n*).

init method is called with the following arguments :

- Interf_type_n input
- IndirectDir_n input
- iSC_n input
- SCPos_n input
- NPs_n input
- USO_n input
- RecordData_n input
- sGW_n input
- GWB_n input
- tStepPhy_n input
- tStepMes_n input
- FilterON_n = TRUE
- attenuation : FilterParam_n[0] = 140.0 dB
- oscillations in bandwidth FilterParam_n[1] = 0.1 dB
- low transition frequency / measurement frequency : FilterParam_n[2] = 0.1
- high transition frequency / measurement frequency : FilterParam_n[3] = 0.3

Definition at line 120 of file LISACODE-PhoDetPhaMet.cpp.

References init().

10.26.2.4 PhoDetPhaMet::PhoDetPhaMet (PDPMINTERF *InterfType_n*, int *IndirectDir_n*, int *iSC_n*, Geometry $*$ *SCPos_n*, vector$<$ Noise $* > *$ *NPs_n*, USOClock $*$ *USO_n*, Memory $*$ *RecordData_n*, TrFctGW $*$ *sGW_n*, Background $*$ *GWB_n*, double *tStepPhy_n*, double *tStepMes_n*, bool *FilterON_n*, vector$<$ double $>$ *FilterParam_n*)

Constructs an instance and initializes it with inputs. It is like previous constructor with added inputs for filter description.

init method is called with the following arguments :

- Interf_type_n input
- IndirectDir_n input
- iSC_n input
- SCPos_n input
- NPs_n input
- USO_n input
- RecordData_n input
- sGW_n input
- GWB_n input
- tStepPhy_n input
- tStepMes_n input
- FilterON_n input
- attenuation : FilterParam_n[0] input
- oscillations in bandwidth FilterParam_n[1] input
- low transition frequency / measurement frequency : FilterParam_n[2] input
- high transition frequency / measurement frequency : FilterParam_n[3] input

Definition at line 162 of file LISACODE-PhoDetPhaMet.cpp.

References init().

10.26.2.5 PhoDetPhaMet::∼PhoDetPhaMet ()

Destructor.

Definition at line 183 of file LISACODE-PhoDetPhaMet.cpp.

10.26.3 Member Function Documentation

10.26.3.1 void PhoDetPhaMet::DisplayStoredData ()

Displays stored data.

Displayed data are :

- Phasemeter informations : InterfType, iSC and IndirectDir
- physical data InterfPhyData and filtered physical data FilterPhyData if present (depending on FilterON)

Definition at line 696 of file LISACODE-PhoDetPhaMet.cpp.

References FilterON, FilterPhyData, IndirectDir, InterfPhyData, InterfType, and iSC.

10.26.3.2 double PhoDetPhaMet::getIndirectDir () const `[inline]`

Returns IndirectDir attribute.

Definition at line 178 of file LISACODE-PhoDetPhaMet.h.

References IndirectDir.

10.26.3.3 double PhoDetPhaMet::getiSC () const `[inline]`

Returns iSC attribute.

Definition at line 180 of file LISACODE-PhoDetPhaMet.h.

References iSC.

10.26.3.4 bool PhoDetPhaMet::getNoNoise ()

Indicates if noises are present or not. It returns true if there are no noises.

Checks all elements of NPs attribute.

If all vectors are NULL, returned value is TRUE, else FALSE.

Definition at line 1016 of file LISACODE-PhoDetPhaMet.cpp.

Referenced by init().

10.26.3.5 double PhoDetPhaMet::gettStab ()

Gets stabilization time.

Returns:

Product between low pass filter PBFilter stabilization data number and physical simulation time step tStepPhy

Definition at line 710 of file LISACODE-PhoDetPhaMet.cpp.

References FilterON, Filter::getNbDataStab(), PBFilter, and tStepPhy.

10.26.3.6 double PhoDetPhaMet::gGWB (int *iSC*, int *IndirectDir*, double *t*)

Returns value of Gravitationnal Wave Background (iSC={1,2,3}, IndirectDir={0,1}).

Parameters:

iSC = {1,2,3} is the spacecraft index

IndirectDir = {0,1}

t = time.

Returns:

If GWB pointer is NULL, returned value is 0, else
it is result of Background::deltanu method, called with *iSC*, *IndirectDir* and *t* inputs.

Definition at line 791 of file LISACODE-PhoDetPhaMet.cpp.

References Background::deltanu(), and GWB.

Referenced by ReceiveSignal().

10.26.3.7 double PhoDetPhaMet::gN (NOISEORIG *OrigN*, int *iSC*, int *IndirectDir*, double *tDelay*)

Returns value of specified noise after delay computation.

Parameters:

OrigN noise origin

IndirectDir optical bench direction (0: direct ; 1 : indirect)

iSC space craft number (1,2,3)

tDelay delay time

Inputs are checked : OrigN expected values are

- LA (laser noise),
- OB (optical bench),
- IM (inertial mass),
- OP (optical paths noise).

First, index is computed :

$$indexNPs = \begin{cases} 3 \cdot IndirectDir + iSC - 1 & \text{if OrigN=LA} \\ 3 \cdot IndirectDir + iSC - 1 + 6 & \text{if OrigN=OB} \\ 3 \cdot IndirectDir + iSC - 1 + 12 & \text{if OrigN=IM} \\ 3 \cdot IndirectDir + iSC - 1 + 18 & \text{if OrigN=OP} \\ 24 & \text{else} \end{cases}$$

Then, noise corresponding to that index and tDelay time is given by Noise::getNoise method and its value is returned.

Definition at line 741 of file LISACODE-PhoDetPhaMet.cpp.

References IM, LA, OB, and OP.

Referenced by ReceiveSignal().

10.26.3.8 void PhoDetPhaMet::init (PDPMINTERF *InterfType_n*, int *IndirectDir_n*, int *iSC_n*, Geometry ∗ *SCPos_n*, vector< Noise ∗ > ∗ *NPs_n*, USOClock ∗ *USO_n*, Memory ∗ *RecordData_n*, TrFctGW ∗ *sGW_n*, Background ∗ *GWB_n*, double *tStepPhy_n*, double *tStepMes_n*, bool *FilterON_n*, vector< double > *FilterParam_n*)

Initializes an instance with default values and inputs.

Inputs are checked :

- travel direction IndirectDir_n must have to be 0 and 1
- Spacecraft index iSC_n expected values are 1, 2 and 3
- Physical time step tStepPhy_n must be positive or null
- Measurement time step tStepMes_n must be positive or null

Set attributes are :

- InterfType = InterfType_n
- IndirectDir = IndirectDir_n
- iSC = iSC_n
- SCPos = SCPos_n
- NPs = NPs_n
- USO = USO_n
- RecordData = RecordData_n + additinal serie data (using Memory::AddSerieData method, with IndirectDir, InterfType and iSC information)
- sGW = sGW_n
- GWB = GWB_n
- tStepPhy = tStepPhy_n
- tStepMes = tStepMes_n
- FilterON = FilterON_n
- FilterParam = FilterParam_n
- NoNoise is informed using getNoNoise method

InterfPhyData = NbDataStored null elements, with $NbDataAdd = 2 \cdot (int)(\frac{tStepMes}{tStepPhy} + \frac{1}{2})$

$$\text{if (FilterON)} \begin{cases} \text{PBFilter = new Filter(1/tStepPhy, FilterParam[0], FilterParam[1], FilterParam[2]/tStepMes, FilterParam[} \\ \text{FilterPhyData = NbDataStored null elements} \end{cases}$$

Definition at line 219 of file LISACODE-PhoDetPhaMet.cpp.

References Memory::AddSerieData(), FilterON, FilterParam, FilterPhyData, Filter::getDepth(), Filter::getNbDataStab(), getNoNoise(), GWB, IndirectDir, InterfPhyData, InterfType, iSC, MAX, NbDataAdd, NbDataStored, NoNoise, NPs, PBFilter, RecordData, S, SCPos, sGW, TAU, tStepMes, tStepPhy, and USO.

Referenced by PhoDetPhaMet().

10.26.3.9 void PhoDetPhaMet::IntegrateSignal (double *t*)

Stores the result in memory (one measurement).

InterfType attribute is checked : expected values are S and TAU.

Steps :

List of physical values (InterfPhyData) and filtered data (FilterPhyData) (if FilterON) are made.

Signal received by photodetector-phasemeters is computed.

Filtering is applied (if FilterON).

Last data are deleted.

Returns:

If (FilterON) PhaMetResult = FilterPhyData first value, else InterfPhyData first value.
Result is stored in memory, with serieNumber=IndirectDir if InterfType is S, 2+IndirectDir if InterfType is TAU.

Definition at line 961 of file LISACODE-PhoDetPhaMet.cpp.

References Filter::App(), FilterON, FilterPhyData, IndirectDir, InterfPhyData, InterfType, NbDataAdd, NbDataStored, PBFilter, Memory::ReceiveData(), ReceiveSignal(), RecordData, S, and TAU.

10.26.3.10 void PhoDetPhaMet::ReceiveSignal (double *t*)

Computes the signal received by photodetector-phasemeters.

Parameters:

t time

InterfType attribute is checked : expected values are S and TAU.

Computations :

- if (InterfType=S)

$GWSignal = sGW- > deltanu(iSC, modulo(iSC + 1 + IndirectDir, 3), 2, t)$
$+gGWB(iSC, IndirectDir, t)$
for $i = 0, \ldots, NbDataAdd - 1,$
$$\left\{ \begin{array}{ll} \text{if (NoNoise)} & InterfPhyData[i] = GWSignal \\ \text{else} & InterfPhyData[i] = GWSignal + noise_i \end{array} \right.$$
$noise_i = gN(OP, iSC, IndirectDir, tDPhy_i)$
$+gN(LA, iSCpe1, 1 - IndirectDir, tDPhy_i$
$+(*SCPos).gtdelay(modulo(iSC + 1 + IndirectDir, 3), iSC, 2, t + tDPhy_i))$
$+gN(OB, iSCpe1, 1 - IndirectDir, tDPhy_i$
$+(*SCPos).gtdelay(modulo(iSC + 1 + IndirectDir, 3), iSC, 2, t + tDPhy_i))$
$-2 \cdot gN(IM, iSC, 0, tDPhy_i)$
$-gN(LA, iSC, IndirectDir, tDPhy_i)$
$-gN(OB, iSC, IndirectDir, tDPhy_i)$
$tDPhy_i = USO- > gGap(t, tStepPhy) - i \cdot tStepPhy$

- if (InterfType=TAU)
  for $i = 0, \ldots, NbDataAdd - 1$
  $InterfPhyData[i] = gN(LA, iSC, 1, tDPhy_i)$
  $-2 \cdot gN(IM, iSC, 1 - IndirectDir, tDPhy_i)$
  $-2 \cdot gN(OB, iSC, 1 - IndirectDir, tDPhy_i)$
  $-gN(LA, iSC, IndirectDir, tDPhy_i)$

PhoDetPhaMet::gN , TrFctGW::deltanu USOClock::gGap and Geometry::gtdelay methods are called.

Definition at line 840 of file LISACODE-PhoDetPhaMet.cpp.

References TrFctGW::deltanu(), USOClock::gGap(), gGWB(), gN(), IM, IndirectDir, InterfPhyData, InterfType, iSC, LA, NbDataAdd, NoNoise, OB, OP, S, sGW, TAU, tStepPhy, and USO.

Referenced by IntegrateSignal().

10.26.4 Member Data Documentation

10.26.4.1 bool PhoDetPhaMet::FilterON `[protected]`

Filter flag : If true, the filter is applied.

Definition at line 113 of file LISACODE-PhoDetPhaMet.h.

Referenced by DisplayStoredData(), gettStab(), init(), and IntegrateSignal().

10.26.4.2 vector<double> PhoDetPhaMet::FilterParam `[protected]`

Filter parameters : attenuation [dB], oscillations in bandwidth [dB], low transition frequency divided by measurement frequency, high transition frequency divided by measurement frequency.

Definition at line 119 of file LISACODE-PhoDetPhaMet.h.

Referenced by init().

10.26.4.3 vector<double> PhoDetPhaMet::FilterPhyData `[protected]`

Filtered physical data vector.

Definition at line 105 of file LISACODE-PhoDetPhaMet.h.

Referenced by DisplayStoredData(), init(), and IntegrateSignal().

10.26.4.4 Background* PhoDetPhaMet::GWB `[protected]`

Background pointer : confusion whites dwarfs background.

Definition at line 97 of file LISACODE-PhoDetPhaMet.h.

Referenced by gGWB(), and init().

10.26.4.5 int PhoDetPhaMet::IndirectDir `[protected]`

Direction flag : 0 if the optical bench is in the direct direction, else 1.

Definition at line 83 of file LISACODE-PhoDetPhaMet.h.

Referenced by DisplayStoredData(), getIndirectDir(), init(), IntegrateSignal(), and ReceiveSignal().

10.26.4.6 vector<double> PhoDetPhaMet::InterfPhyData [protected]

Physical data vector.

Definition at line 103 of file LISACODE-PhoDetPhaMet.h.

Referenced by DisplayStoredData(), init(), IntegrateSignal(), and ReceiveSignal().

10.26.4.7 PDPMINTERF PhoDetPhaMet::InterfType [protected]

Type of interferences made by the photodetector-phasemeter.

Definition at line 81 of file LISACODE-PhoDetPhaMet.h.

Referenced by DisplayStoredData(), init(), IntegrateSignal(), and ReceiveSignal().

10.26.4.8 int PhoDetPhaMet::iSC [protected]

Spacecraft index corresponding to the photodetector-phasemeter.

Definition at line 85 of file LISACODE-PhoDetPhaMet.h.

Referenced by DisplayStoredData(), getiSC(), init(), and ReceiveSignal().

10.26.4.9 int PhoDetPhaMet::NbDataAdd [protected]

Number of data added (in InterfPhyData and FilterPhyData) for each measurement.

Definition at line 109 of file LISACODE-PhoDetPhaMet.h.

Referenced by init(), IntegrateSignal(), and ReceiveSignal().

10.26.4.10 int PhoDetPhaMet::NbDataStored [protected]

Number of data stored (in InterfPhyData and FilterPhyData).

Definition at line 107 of file LISACODE-PhoDetPhaMet.h.

Referenced by init(), and IntegrateSignal().

10.26.4.11 bool PhoDetPhaMet::NoNoise [protected]

Noise flag : if true, there is no noise.

Definition at line 121 of file LISACODE-PhoDetPhaMet.h.

Referenced by init(), and ReceiveSignal().

10.26.4.12 vector<Noise ∗>∗ PhoDetPhaMet::NPs [protected]

Noise pointers vector.

Definition at line 89 of file LISACODE-PhoDetPhaMet.h.

Referenced by init().

10.26.4.13 Filter* PhoDetPhaMet::PBFilter [protected]

Filter pointer to a low pass filter.

Definition at line 111 of file LISACODE-PhoDetPhaMet.h.

Referenced by gettStab(), init(), and IntegrateSignal().

10.26.4.14 Memory* PhoDetPhaMet::RecordData [protected]

Pointer to the storage memory of the photodetector-phasemeters signal.

Definition at line 93 of file LISACODE-PhoDetPhaMet.h.

Referenced by init(), and IntegrateSignal().

10.26.4.15 Geometry* PhoDetPhaMet::SCPos [protected]

Pointer to LISA geometry.

Definition at line 87 of file LISACODE-PhoDetPhaMet.h.

Referenced by init().

10.26.4.16 TrFctGW* PhoDetPhaMet::sGW [protected]

Pointer to the transfer function.

Definition at line 95 of file LISACODE-PhoDetPhaMet.h.

Referenced by init(), and ReceiveSignal().

10.26.4.17 double PhoDetPhaMet::tStepMes [protected]

Measurement time step.

Definition at line 101 of file LISACODE-PhoDetPhaMet.h.

Referenced by init().

10.26.4.18 double PhoDetPhaMet::tStepPhy [protected]

Physical simulation time step.

Definition at line 99 of file LISACODE-PhoDetPhaMet.h.

Referenced by gettStab(), init(), and ReceiveSignal().

10.26.4.19 USOClock* PhoDetPhaMet::USO [protected]

Pointers to spacecrafts USO clocks.

Definition at line 91 of file LISACODE-PhoDetPhaMet.h.

Referenced by init(), and ReceiveSignal().

The documentation for this class was generated from the following files:

- LISACODE-PhoDetPhaMet.h
- LISACODE-PhoDetPhaMet.cpp

10.27 QuadCell Struct Reference

```
#include <LISACODE-EllipticFilter.h>
```

10.27.1 Detailed Description

Elliptic cell structure.

Definition at line 24 of file LISACODE-EllipticFilter.h.

Public Attributes

- double a0

  Scale factor.

- double a1

  First direct coefficient divided by scale factor.

- double b1

  First recursive coefficient.

- double b2

  Second recursive coefficient.

- complex< double > zero

  Complex value corresponding to zero (0).

- complex< double > pole

  Complex value corresponding to pole.

- double u [2]

  Memory.

10.27.2 Member Data Documentation

10.27.2.1 QuadCell::a0

Scale factor.

Referenced by AbsRespFunctQuadCell(), FilterQuadCell(), KmQuadCell(), and TransfZQuadCell().

10.27.2.2 QuadCell::a1

First direct coefficient divided by scale factor.

Referenced by AbsRespFunctQuadCell(), FilterQuadCell(), KmQuadCell(), and TransfZQuadCell().

10.27.2.3 QuadCell::b1

First recursive coefficient.

Referenced by AbsRespFunctQuadCell(), CalcEllipticFilter4LISACode(), FilterQuadCell(), HmQuadCell(), KmQuadCell(), and TransfZQuadCell().

10.27.2.4 QuadCell::b2

Second recursive coefficient.

Referenced by AbsRespFunctQuadCell(), FilterQuadCell(), HmQuadCell(), KmQuadCell(), and TransfZQuadCell().

10.27.2.5 QuadCell::pole

Complex value corresponding to pole.

10.27.2.6 QuadCell::u

Memory.

Referenced by CalcEllipticFilter(), and FilterQuadCell().

10.27.2.7 QuadCell::zero

Complex value corresponding to zero (0).

The documentation for this struct was generated from the following file:

- LISACODE-EllipticFilter.h

10.28 RandomMT Class Reference

`#include <LISACODE-Random.h>`

10.28.1 Detailed Description

Mersenne twister random generator class.

Definition at line 47 of file LISACODE-Random.h.

Public Member Functions

- RandomMT ()
- RandomMT (int32)
- ~RandomMT ()
- double UniformMT (double, double)
- Serie UniformMTSerie (double, double, int, double, double)
- SerieC UniformMTSerieC (double, double, int, double, double)
- double NormalMT (double, double)
- Serie NormalMTSerie (double, double, int, double, double)
- SerieC NormalMTSerieC (double, double, int, double, double)

Protected Attributes

- int32 seedMT

  seed

10.28.2 Constructor & Destructor Documentation

10.28.2.1 RandomMT::RandomMT ()

Definition at line 16 of file LISACODE-Random.cpp.

References seedMT.

10.28.2.2 RandomMT::RandomMT (int32)

Definition at line 21 of file LISACODE-Random.cpp.

References seedMT.

10.28.2.3 RandomMT::~RandomMT ()

Definition at line 27 of file LISACODE-Random.cpp.

10.28.3 Member Function Documentation

10.28.3.1 double RandomMT::NormalMT (double, double)

Definition at line 98 of file LISACODE-Random.cpp.

References seedMT.

10.28.3.2 Serie RandomMT::NormalMTSerie (double, double, int, double, double)

Definition at line 107 of file LISACODE-Random.cpp.

References seedMT, and Serie::setBinValue().

10.28.3.3 SerieC RandomMT::NormalMTSerieC (double, double, int, double, double)

Definition at line 122 of file LISACODE-Random.cpp.

References seedMT, and SerieC::setBinValueC().

10.28.3.4 double RandomMT::UniformMT (double, double)

Definition at line 32 of file LISACODE-Random.cpp.

References seedMT.

10.28.3.5 Serie RandomMT::UniformMTSerie (double, double, int, double, double)

Definition at line 43 of file LISACODE-Random.cpp.

References seedMT, and Serie::setBinValue().

10.28.3.6 SerieC RandomMT::UniformMTSerieC (double, double, int, double, double)

Definition at line 68 of file LISACODE-Random.cpp.

References seedMT, and SerieC::setBinValueC().

10.28.4 Member Data Documentation

10.28.4.1 int32 RandomMT::seedMT `[protected]`

seed

Definition at line 53 of file LISACODE-Random.h.

Referenced by NormalMT(), NormalMTSerie(), NormalMTSerieC(), RandomMT(), UniformMT(), UniformMTSerie(), and UniformMTSerieC().

The documentation for this class was generated from the following files:

- LISACODE-Random.h
- LISACODE-Random.cpp

10.29 Serie Class Reference

```
#include <LISACODE-Serie.h>
```

10.29.1 Detailed Description

Serie interpolation class.

Definition at line 44 of file LISACODE-Serie.h.

Public Member Functions

- Serie ()

  Constructs an instance and initializes it with default values.

- Serie (double start, double delta)

  Constructs an instance and initializes it with inputs and default values.

- Serie (double start, double delta, int length)

  Constructs an instance and initializes it with inputs and default values.

- Serie (double start, double delta, vector< double > ys_n)

  Constructs an instance and initializes it with inputs.

- Serie (char *fname)

  Constructs an instance and initializes it with data read in fname input file.

- ~Serie ()

  Destructor.

- int getNbVal () const

  Returns number of values (size of ys attribute).

- void setNbVal (int lenght)

  Sets ys attribute size to lenght input.

- void setRefStart (double start)

  Sets x0 attribute size to start input.

- double getRefStep () const

  Returns dx attribute.

- void setRefStep (double delta)

  Sets dx attribute size to delta input.

- double getRef (int bin) const

  Gets reference value corresponding to bin input.

- double getBinValue (int bin) const

Gets reference y value corresponding to bin input.

- void setBinValue (int bin, double x)

  Sets reference y value corresponding to bin and x inputs.

- void addData (double y)

  Adds data at the begining of the serie.

- void delLastData ()

  Deletes the last data of the serie.

- void delLastData (double xMax)

  Deletes the last elements of the serie ys, between index associated to xMax *and the end of the serie.*

- void wfile (char ∗name)

  Writes the serie as two columns (X,Y) into fname input file.

- void rfile (char ∗name)

  Reads the serie as two columns (X,Y) from fname input file.

- double gData (double x, INTERP InterpolType, double InterpUtilValue) const

  Returns the exact value if x is a multiple of dx, else interpolated value.

- double TruncVal (double x) const

  Returns truncated value.

- double InterLinear (double x) const

  Returns linear interpolation result.

- double InterCubic (double x) const

  Returns cubic interpolation result.

- double InterHermite (double x, double tension, double bias) const

  Returns hermite interpolation result, depending on x, tension and bias inputs.

- double InterLagrange (double x, int order) const

  Returns Lagrange interpolation result.

Protected Attributes

- vector< double > ys

  Reference serie data.

- double x0

  Reference serie start value.

- double dx

  Reference serie step.

10.29.2 Constructor & Destructor Documentation

10.29.2.1 Serie::Serie ()

Constructs an instance and initializes it with default values.

- x0 = 0
- dx =1

Definition at line 28 of file LISACODE-Serie.cpp.

References dx, and x0.

10.29.2.2 Serie::Serie (double *start*, double *delta*)

Constructs an instance and initializes it with inputs and default values.

- x0 = start
- dx = delta

Definition at line 39 of file LISACODE-Serie.cpp.

References dx, and x0.

10.29.2.3 Serie::Serie (double *start*, double *delta*, int *length*)

Constructs an instance and initializes it with inputs and default values.

- x0 = start
- dx = delta
- ys = length elements, with 0 values

Definition at line 52 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

10.29.2.4 Serie::Serie (double *start*, double *delta*, vector< double > *ys_n*)

Constructs an instance and initializes it with inputs.

- x0 = start
- dx = delta
- ys = ys_n input

Definition at line 68 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

10.29.2.5 Serie::Serie (char ∗ *fname*)

Constructs an instance and initializes it with data read in fname input file.

Input file must have at least 2 elements.

Attributes are set :

- x0 = first read element
- dx = difference betwwen second and first read elements
- ys = read elements

Definition at line 84 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

10.29.2.6 Serie::∼Serie ()

Destructor.

Definition at line 112 of file LISACODE-Serie.cpp.

References ys.

10.29.3 Member Function Documentation

10.29.3.1 void Serie::addData (double *y*)

Adds data at the begining of the serie.

Definition at line 183 of file LISACODE-Serie.cpp.

References ys.

10.29.3.2 void Serie::delLastData (double *xMax*)

Deletes the last elements of the serie ys, between index associated to *xMax* and the end of the serie.

Definition at line 198 of file LISACODE-Serie.cpp.

References dx, and ys.

10.29.3.3 void Serie::delLastData ()

Deletes the last data of the serie.

Definition at line 190 of file LISACODE-Serie.cpp.

References ys.

10.29.3.4 double Serie::gData (double *x*, INTERP *InterpolType*, double *InterpUtilValue*) const

Returns the exact value if x is a multiple of dx, else interpolated value.

Interpolation method depends on InterpolType input.

It is checked. Its expected values are TRU, LIN, CUB and LAG.

Returns:

$$\begin{cases} TruncVal(x) & \text{if } modulo(\frac{x-x_0}{dx},1) \approx 0 \\ TruncVal(x) & \text{if } InterpolType = TRU \\ InterLinear(x) & \text{if } InterpolType = LIN \\ InterCubic(x) & \text{if } InterpolType = CUB \\ InterLagrange(x, int(InterpUtilValue)) & \text{if } InterpolType = LAG \end{cases}$$

Definition at line 259 of file LISACODE-Serie.cpp.

References CUB, dx, InterCubic(), InterLagrange(), InterLinear(), LAG, LIN, PRECISION, TRU, TruncVal(), and x0.

10.29.3.5 double Serie::getBinValue (int *bin*) const

Gets reference y value corresponding to bin input.

Input is checked ; it must be positive or null, and lower than ys attribute size.

Returns:

ys[bin]

Definition at line 158 of file LISACODE-Serie.cpp.

References ys.

10.29.3.6 int Serie::getNbVal () const [inline]

Returns number of values (size of ys attribute).

Definition at line 67 of file LISACODE-Serie.h.

References ys.

10.29.3.7 double Serie::getRef (int *bin*) const

Gets reference value corresponding to bin input.

Input is checked ; it must be positive or null, and lower than ys attribute size.

Returns:

$x0 + dx \cdot bin$

Definition at line 145 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

10.29.3.8 double Serie::getRefStep () const [inline]

Returns dx attribute.

Definition at line 71 of file LISACODE-Serie.h.

References dx.

10.29.3.9 double Serie::InterCubic (double *x*) const

Returns cubic interpolation result.

Indices are computed :

$bin_1 = floor(\frac{x-x_0}{dx})$

$bin_2 = bin_1 + 1$

$bin_0 = bin_1 - 1$

$bin_3 = bin_2 + 1$

Indices are checked ; bin_0 must be positive or null, and bin_3 must be lower than ys attribute size.

Returns:

$\mu^3 \cdot (ys[bin_3] - ys[bin_2] - ys[bin_0] + ys[bin_1])$
$+\mu^2 \cdot (2 \cdot ys[bin_0] - 2 \cdot ys[bin_1] + ys[bin_2] - ys[bin_3])$
$+\mu \cdot (ys[bin_2] - ys[bin_0])$
$+ys[bin_3]$,
where :
$\mu = \frac{x-x_0}{dx} - floor(\frac{x-x_0}{dx})$

Definition at line 349 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

Referenced by gData().

10.29.3.10 double Serie::InterHermite (double *x*, double *tension*, double *bias*) const

Returns hermite interpolation result, depending on x, tension and bias inputs.

Indices are computed :

$bin_1 = floor(\frac{x-x_0}{dx})$

$bin_2 = bin_1 + 1$

$bin_0 = bin_1 - 1$

$bin_3 = bin_2 + 1$

Indices are checked ; bin_0 must be positive or null, and bin_3 must be lower than ys attribute size.

Returns:

$(\mu^3 - 2 \cdot \mu^2 + \mu) \cdot ys[bin_1] + (\mu^3 - 2 \cdot \mu^2 + \mu) \cdot ((ys[bin_1] - ys[bin_0]) \cdot (1 + bias) \cdot \frac{1-tension}{2} + (ys[bin_2] - ys[bin_1]) \cdot (1 - bias) \cdot \frac{1-tension}{2})$,
where :
$\mu = \frac{x-x_0}{dx} - floor(\frac{x-x_0}{dx})$

Definition at line 394 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

10.29.3.11 double Serie::InterLagrange (double *x*, int *order*) const

Returns Lagrange interpolation result.

Indices are computed :

$bin = floor(\frac{x-x_0}{dx})$

$kmin = bin - ordermin + 1$, where $ordermin = floor(\frac{order+1}{2})$

$kmax = bin + order + 1 - ordermin$

Indices are checked :

- bin must be positive or null, and lower than (ys attribute size -1)
- kmin must be positive or null
- kmax must be and lower than (ys attribute size -1)

Returns:

$\sum_{k=kmin}^{kmax} ys[k] \cdot P_k$, where $P_k = \prod_{j=kmin,j\neq k}^{kmax} \frac{x-x_0-j\cdot dx}{(k-j)\cdot dx}$

Definition at line 445 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

Referenced by gData().

10.29.3.12 double Serie::InterLinear (double *x*) const

Returns linear interpolation result.

First, bin index is computed :

$$bin = floor(\frac{x-x_0}{dx})$$

Then bin is checked ; it must be positive or null, and lower than ys attribute size.

Returns:

$$(bin + 1 - \frac{x-x_0}{dx}) \cdot ys[bin] + \frac{x-x_0}{dx - bin} \cdot ys[bin+1]$$

Definition at line 321 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

Referenced by gData().

10.29.3.13 void Serie::rfile (char ∗ *fname*)

Reads the serie as two columns (X,Y) from fname input file.

Definition at line 226 of file LISACODE-Serie.cpp.

References ys.

10.29.3.14 void Serie::setBinValue (int *bin*, double *x*)

Sets reference y value corresponding to bin and x inputs.

Input is checked ; it must be positive or null, and lower than ys attribute size.

Then ys is filled : ys[bin]=x .

Definition at line 171 of file LISACODE-Serie.cpp.

References ys.

Referenced by RandomMT::NormalMTSerie(), and RandomMT::UniformMTSerie().

10.29.3.15 void Serie::setNbVal (int *lenght*)

Sets ys attribute size to lenght input.

Definition at line 121 of file LISACODE-Serie.cpp.

References ys.

10.29.3.16 void Serie::setRefStart (double *start*)

Sets x0 attribute size to start input.

Definition at line 127 of file LISACODE-Serie.cpp.

References x0.

10.29.3.17 void Serie::setRefStep (double *delta*)

Sets dx attribute size to delta input.

Definition at line 133 of file LISACODE-Serie.cpp.

References dx.

10.29.3.18 double Serie::TruncVal (double *x*) const

Returns truncated value.

First, bin index is computed :

$$bin = floor(\frac{x - x_0}{dx})$$

Then bin is checked ; it must be positive or null, and lower than ys attribute size.

Returns:

$$ys[bin]$$

Definition at line 300 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

Referenced by gData().

10.29.3.19 void Serie::wfile (char * *fname*)

Writes the serie as two columns (X,Y) into fname input file.

Definition at line 209 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

10.29.4 Member Data Documentation

10.29.4.1 double Serie::dx [protected]

Reference serie step.

Definition at line 52 of file LISACODE-Serie.h.

Referenced by delLastData(), gData(), getRef(), getRefStep(), InterCubic(), InterHermite(), InterLagrange(), InterLinear(), Serie(), setRefStep(), TruncVal(), and wfile().

10.29.4.2 double Serie::x0 [protected]

Reference serie start value.

Definition at line 50 of file LISACODE-Serie.h.

Referenced by gData(), getRef(), InterCubic(), InterHermite(), InterLagrange(), InterLinear(), Serie(), setRefStart(), TruncVal(), and wfile().

10.29.4.3 vector<double> Serie::ys [protected]

Reference serie data.

Definition at line 48 of file LISACODE-Serie.h.

Referenced by addData(), delLastData(), getBinValue(), getNbVal(), getRef(), InterCubic(), InterHermite(), InterLagrange(), InterLinear(), rfile(), Serie(), setBinValue(), setNbVal(), TruncVal(), wfile(), and ~Serie().

The documentation for this class was generated from the following files:

- LISACODE-Serie.h
- LISACODE-Serie.cpp

10.30 SerieC Class Reference

`#include <LISACODE-Serie.h>`

10.30.1 Detailed Description

complex serie interpolation class.

Definition at line 99 of file LISACODE-Serie.h.

Public Member Functions

- SerieC ()

  Constructs an instance and initializes it with default values.

- SerieC (double start, double delta)

  Constructs an instance and initializes it with inputs and default values.

- SerieC (double start, double delta, int length)

  Constructs an instance and initializes it with inputs and default values.

- SerieC (double start, double delta, vector< complex< double > > ys_n)

  Constructs an instance and initializes it with inputs.

- SerieC (char ∗fname)

  Constructs an instance and initializes it with data read in fname input file.

- ∼SerieC ()

  Destructor.

- int getNbValC () const

  Returns number of values (size of ys attribute).

- void setNbValC (int lenght)

  Sets ys attribute size to lenght input.

- void setRefStartC (double start)

  Sets x0 attribute size to start input.

- double getRefStepC () const

  Returns dx attribute.

- void setRefStepC (double delta)

  Sets dx attribute size to delta input.

- double getRefC (int bin) const

  Gets reference value corresponding to bin input.

- complex< double > getBinValueC (int bin) const

Gets reference y value corresponding to bin input.

- void setBinValueC (int bin, complex< double > x)

  Sets reference y value corresponding to bin and x inputs.

- void addDataC (complex< double > y)

  Adds data at the begining of the serie.

- void delLastDataC ()

  Deletes the last data of the serie.

- void delLastDataC (double xMax)

  Deletes the last data of the serie, while x reference is greater than xMax input.

- void wfileC (char ∗name)

  Writes the serie as 3 columns (X,Y.real,Y.imag) into fname input file.

- void rfileC (char ∗name)

  Writes the serie as 3 columns (X,Y.real,Y.imag) from fname input file.

Protected Attributes

- vector< complex< double > > ys

  Reference serie data.

- double x0

  Reference serie start value.

- double dx

  Reference serie step.

10.30.2 Constructor & Destructor Documentation

10.30.2.1 SerieC::SerieC ()

Constructs an instance and initializes it with default values.

- x0 = 0
- dx =1

Definition at line 510 of file LISACODE-Serie.cpp.

References dx, and x0.

10.30.2.2 SerieC::SerieC (double *start*, double *delta*)

Constructs an instance and initializes it with inputs and default values.

- x0 = start
- dx = delta

Definition at line 521 of file LISACODE-Serie.cpp.

References dx, and x0.

10.30.2.3 SerieC::SerieC (double *start*, double *delta*, int *lenght*)

Constructs an instance and initializes it with inputs and default values.

- x0 = start
- dx = delta
- ys = lenght elements, with 0 values

Definition at line 534 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

10.30.2.4 SerieC::SerieC (double *start*, double *delta*, vector< complex< double > > *ys_n*)

Constructs an instance and initializes it with inputs.

- x0 = start
- dx = delta
- ys = ys_n input

Definition at line 552 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

10.30.2.5 SerieC::SerieC (char ∗ *fname*)

Constructs an instance and initializes it with data read in fname input file.

Input file must have at least 2 elements.

Attributes are set :

- x0 = first read element
- dx = difference betwwen second and first read elements
- ys = read elements

Definition at line 568 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

10.30.2.6 SerieC::~SerieC ()

Destructor.

Definition at line 600 of file LISACODE-Serie.cpp.

References ys.

10.30.3 Member Function Documentation

10.30.3.1 void SerieC::addDataC (complex< double > *y*)

Adds data at the begining of the serie.

Definition at line 671 of file LISACODE-Serie.cpp.

References ys.

10.30.3.2 void SerieC::delLastDataC (double *xMax*)

Deletes the last data of the serie, while x reference is greater than xMax input.

Definition at line 685 of file LISACODE-Serie.cpp.

References dx, and ys.

10.30.3.3 void SerieC::delLastDataC ()

Deletes the last data of the serie.

Definition at line 678 of file LISACODE-Serie.cpp.

References ys.

10.30.3.4 complex< double > SerieC::getBinValueC (int *bin*) const

Gets reference y value corresponding to bin input.

Input is checked ; it must be positive or null, and lower than ys attribute size.

Returns:

ys[bin]

Definition at line 646 of file LISACODE-Serie.cpp.

References ys.

10.30.3.5 int SerieC::getNbValC () const [inline]

Returns number of values (size of ys attribute).

Definition at line 122 of file LISACODE-Serie.h.

References ys.

10.30.3.6 double SerieC::getRefC (int *bin*) const

Gets reference value corresponding to bin input.

Input is checked ; it must be positive or null, and lower than ys attribute size.

Returns:

$x0 + dx \cdot bin$

Definition at line 633 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

10.30.3.7 double SerieC::getRefStepC () const [inline]

Returns dx attribute.

Definition at line 126 of file LISACODE-Serie.h.

References dx.

10.30.3.8 void SerieC::rfileC (char $*$ *fname*)

Writes the serie as 3 columns (X,Y.real,Y.imag) from fname input file.

Definition at line 713 of file LISACODE-Serie.cpp.

References ys.

10.30.3.9 void SerieC::setBinValueC (int *bin*, complex$<$ double $>$ *x*)

Sets reference y value corresponding to bin and x inputs.

Input is checked ; it must be positive or null, and lower than ys attribute size.

Returns:

ys[bin]=x

Definition at line 659 of file LISACODE-Serie.cpp.

References ys.

Referenced by RandomMT::NormalMTSerieC(), and RandomMT::UniformMTSerieC().

10.30.3.10 void SerieC::setNbValC (int *lenght*)

Sets ys attribute size to lenght input.

Definition at line 609 of file LISACODE-Serie.cpp.

References ys.

10.30.3.11 void SerieC::setRefStartC (double *start*)

Sets x0 attribute size to start input.

Definition at line 615 of file LISACODE-Serie.cpp.

References x0.

10.30.3.12 void SerieC::setRefStepC (double *delta*)

Sets dx attribute size to delta input.

Definition at line 621 of file LISACODE-Serie.cpp.

References dx.

10.30.3.13 void SerieC::wfileC (char ∗ *fname*)

Writes the serie as 3 columns (X,Y.real,Y.imag) into fname input file.

Definition at line 696 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

10.30.4 Member Data Documentation

10.30.4.1 double SerieC::dx `[protected]`

Reference serie step.

Definition at line 107 of file LISACODE-Serie.h.

Referenced by delLastDataC(), getRefC(), getRefStepC(), SerieC(), setRefStepC(), and wfileC().

10.30.4.2 double SerieC::x0 `[protected]`

Reference serie start value.

Definition at line 105 of file LISACODE-Serie.h.

Referenced by getRefC(), SerieC(), setRefStartC(), and wfileC().

10.30.4.3 vector<complex<double> > SerieC::ys `[protected]`

Reference serie data.

Definition at line 103 of file LISACODE-Serie.h.

Referenced by addDataC(), delLastDataC(), getBinValueC(), getNbValC(), getRefC(), rfileC(), SerieC(), setBinValueC(), setNbValC(), wfileC(), and ∼SerieC().

The documentation for this class was generated from the following files:

- LISACODE-Serie.h
- LISACODE-Serie.cpp

10.31 TDI Class Reference

`#include <LISACODE-TDI.h>`

10.31.1 Detailed Description

Time Delay Interferometry combinaison class.

Definition at line 46 of file LISACODE-TDI.h.

Public Member Functions

- TDI ()

  Constructs an instance and initializes it with zero value for all attributes.

- TDI (Memory ∗TDelay_n, TDI_InterData ∗Eta_n, ofstream ∗OutFile_n, int iSerie_n)

  Constructs an instance and initializes it using TDelay_n, Eta_n, ofstream and iSerie_n inputs.

- TDI (Memory ∗TDelay_n, TDI_InterData ∗Eta_n, ofstream ∗OutFile_n, int iSerie_n, vector< int > Sign_n, vector< int > IndexEta_n, vector< vector< int > > IndexDelay_n, TDITools ∗TDIQuickMod_n)

  Constructs an instance and initializes it using TDelay_n, Eta_n, OutFile_n, iSerie_n, Sign_n, IndexEta_n, IndexDelay_n and TDIQuickMod_n inputs.

- TDI (Memory ∗TDelay_n, TDI_InterData ∗Eta_n, ofstream ∗OutFile_n, int iSerie_n, vector< int > SignEtaDelays, TDITools ∗TDIQuickMod_n)

  Constructs an instance and initializes it using TDelay_n, Eta_n, OutFile_n, iSerie_n, SignEtaDelays and TDIQuickMod_n inputs.

- virtual ∼TDI ()

  Destructor.

- int getCountInterDelay ()

  Returns tmpCountInterDelay attribute.

- int getCountInterEta ()

  Returns tmpCountInterEta attribute.

- void ReadSignEtaDelays (vector< int > SignEtaDelays)

  Reads lists of packs that are in the following form : 1231 for D1 D2 D3 Eta1.

- double Compute (double tComputeDelay)

  Computes the result of generator.

- void RecordResult (double tComputeDelay)

  Records the result of generator.

- double RecordAndReturnResult (double tComputeDelay)

  Records the result of generator and returns the result.

- int NbDelayMax ()

  Gives maximum number of delays.

Protected Attributes

- Memory ∗ TDelay

  Pointer to the list of delay's lengths.

- TDI_InterData ∗ Eta

  Memory where the signals Eta are stored.

- ofstream ∗ OutFile

  File where TDI result are recorded.

- int iSerie

  Index of the serie in memory RecordMem.

- vector< int > Sign

  List of sign of the pack (1 for + and -1 for -).

- vector< int > IndexEta

  List of index of signal Eta where the pack work (value=[1,6]).

- vector< vector< int > > IndexDelay

  List of list of delay's index for each pack (value=[1,6]).

- TDITools ∗ TDIQuickMod

  TDI access tools.

- int tmpCountInterDelay

  Temporary data (unused).

- int tmpCountInterEta

  Temporary data (unused).

10.31.2 Constructor & Destructor Documentation

10.31.2.1 TDI::TDI ()

Constructs an instance and initializes it with zero value for all attributes.

Attributes are :

- TDelay = empty
- Eta = empty
- OutFile = "DefTDIGen.txt" opened file

- iSerie = 0
- Sign = empty
- IndexEta = empty
- IndexDelay = empty
- TDIQuickMod = empty

Definition at line 29 of file LISACODE-TDI.cpp.

References Eta, IndexDelay, IndexEta, iSerie, OutFile, Sign, TDelay, and TDIQuickMod.

10.31.2.2 TDI::TDI (Memory ∗ *TDelay_n*, TDI_InterData ∗ *Eta_n*, ofstream ∗ *OutFile_n*, int *iSerie_n*)

Constructs an instance and initializes it using TDelay_n, Eta_n, ofstream and iSerie_n inputs.

Attributes are :

- TDelay = TDelay_n
- Eta = Eta_n
- OutFile = OutFile_n
- iSerie = iSerie_n
- Sign = empty
- IndexEta = empty
- IndexDelay = empty
- TDIQuickMod = empty

Definition at line 58 of file LISACODE-TDI.cpp.

References Eta, IndexDelay, IndexEta, iSerie, OutFile, Sign, TDelay, and TDIQuickMod.

10.31.2.3 TDI::TDI (Memory ∗ *TDelay_n*, TDI_InterData ∗ *Eta_n*, ofstream ∗ *OutFile_n*, int *iSerie_n*, vector< int > *Sign_n*, vector< int > *IndexEta_n*, vector< vector< int > > *IndexDelay_n*, TDITools ∗ *TDIQuickMod_n*)

Constructs an instance and initializes it using TDelay_n, Eta_n, OutFile_n, iSerie_n, Sign_n, IndexEta_n, IndexDelay_n and TDIQuickMod_n inputs.

Attributes are :

- TDelay = TDelay_n
- Eta = Eta_n
- OutFile = OutFile_n
- iSerie = iSerie_n
- Sign = Sign_n

- IndexEta = IndexEta_n
- IndexDelay = IndexDelay_n
- TDIQuickMod = TDIQuickMod_n
- tmpCountInterDelay = 0
- tmpCountInterEta = 0

Definition at line 92 of file LISACODE-TDI.cpp.

References Eta, IndexDelay, IndexEta, iSerie, OutFile, Sign, TDelay, TDIQuickMod, tmpCountInterDelay, and tmpCountInterEta.

10.31.2.4 TDI::TDI (Memory ∗ *TDelay_n*, TDI_InterData ∗ *Eta_n*, ofstream ∗ *OutFile_n*, int *iSerie_n*, vector< int > *SignEtaDelays*, TDITools ∗ *TDIQuickMod_n*)

Constructs an instance and initializes it using TDelay_n, Eta_n, OutFile_n, iSerie_n, SignEtaDelays and TDIQuickMod_n inputs.

- TDelay = TDelay_n
- Eta = Eta_n
- OutFile = OutFile_n
- iSerie = iSerie_n
- Sign : set by ReadSignEtaDelays, using SignEtaDelays input
- IndexEta : set by ReadSignEtaDelays, using SignEtaDelays input
- IndexDelay : set by ReadSignEtaDelays, using SignEtaDelays input
- TDIQuickMod = TDIQuickMod_n
- tmpCountInterDelay = 0
- tmpCountInterEta = 0

Definition at line 135 of file LISACODE-TDI.cpp.

References Eta, IndexDelay, IndexEta, iSerie, OutFile, ReadSignEtaDelays(), Sign, TDelay, TDIQuickMod, tmpCountInterDelay, and tmpCountInterEta.

10.31.2.5 TDI::∼TDI () `[virtual]`

Destructor.

Definition at line 163 of file LISACODE-TDI.cpp.

10.31.3 Member Function Documentation

10.31.3.1 double TDI::Compute (double *tComputeDelay*)

Computes the result of generator.

Computes delay using tComputeDelay input and class attributes.

If approximations are done in order to compute delays more quickly

$$TotalDelay = \sum_{iPack=0}^{size(IndexEta)-1} \sum_{iDelay=0}^{size(IndexDelay[iPack])-1}$$

$$TDIQuickMod- > getDelay((IndexDelay[iPack])[iDelay])$$

else

$$TotalDelay = \sum_{iPack=0}^{size(IndexEta)-1} \sum_{iDelay=0}^{size(IndexDelay[iPack])-1}$$

$$TDelay- > gData((IndexDelay[iPack])[iDelay] - 1, TotalDelay, LAG, 6)$$

TDITools::getDelay and Memory::gData methods are called.

Returns:

if enough data are stored

$$\sum_{iPack=0}^{size(IndexEta)-1} \sum_{iDelay=0}^{size(IndexDelay[iPack])-1}$$

$$Sign[iPack] \cdot Eta- > gData(IndexEta[iPack], TotalDelay)$$

else 0

Definition at line 270 of file LISACODE-TDI.cpp.

References Eta, TDI_InterData::gData(), Memory::gData(), TDITools::getDelay(), TDITools::getRapidOption(), TDI_InterData::getUsable(), IndexDelay, IndexEta, LAG, Sign, TDelay, and TDIQuickMod.

Referenced by RecordAndReturnResult(), and RecordResult().

10.31.3.2 int TDI::getCountInterDelay () [inline]

Returns tmpCountInterDelay attribute.

Definition at line 109 of file LISACODE-TDI.h.

References tmpCountInterDelay.

10.31.3.3 int TDI::getCountInterEta () [inline]

Returns tmpCountInterEta attribute.

Definition at line 111 of file LISACODE-TDI.h.

References tmpCountInterEta.

10.31.3.4 int TDI::NbDelayMax ()

Gives maximum number of delays.

Maximum number of delays is size of IndexDelay attribute.

Definition at line 343 of file LISACODE-TDI.cpp.

References IndexDelay.

10.31.3.5 void TDI::ReadSignEtaDelays (vector< int > *SignEtaDelays*)

Reads lists of packs that are in the following form : 1231 for D1 D2 D3 Eta1.

Reads TDI combinaisons and push them in Sign IndexEta and IndexDelay attributes.

- for $iPack = 0, \dots, size(SignEtaDelays) - 1$:

  SignEtaDelays[iPack] is checked : $SignEtaDelays[iPack] \neq 0$

  $$\begin{cases} \text{if } (SignEtaDelays[iPack] > 0) & +1 \text{ is pushed back in Sign attribute} \\ \text{if } (SignEtaDelays[iPack] < 0) & -1 \text{ is pushed back in Sign attribute} \end{cases}$$

  $$TmpInfo = abs(SignEtaDelays[iPack])$$

  $$TmpIndexEta = ceil(10 \cdot (\frac{TmpInfo}{10} - floor(\frac{TmpInfo}{10}) + 10^-6))$$

  TmpIndexEta is checked : $1 \leq TmpIndexEta \leq 6$

  TmpIndexEta is pushed back in IndexEta attribute

- $TmpInfo = ceil(\frac{TmpInfo}{10})$; while $tmpInfo \neq 0$:

  $$TmpIndexDelay = ceil(10 \cdot (\frac{TmpInfo}{10} - floor(\frac{TmpInfo}{10}) + 10^-6))$$

  TmpIndexDelay is checked :$1 \leq TmpIndexDelay \leq 6$

  TmpIndexDelay is pushed back in IndexDelay attribute

  $$TmpInfo = ceil(\frac{TmpInfo}{10})$$

Definition at line 193 of file LISACODE-TDI.cpp.

References IndexDelay, IndexEta, and Sign.

Referenced by TDI().

10.31.3.6 double TDI::RecordAndReturnResult (double *tComputeDelay*)

Records the result of generator and returns the result.

Computes delay using tComputeDelay input, writes result into OutFile and returns it.

Definition at line 330 of file LISACODE-TDI.cpp.

References Compute().

10.31.3.7 void TDI::RecordResult (double *tComputeDelay*)

Records the result of generator.

Writes tComputeDelay input into OutFile.

Definition at line 320 of file LISACODE-TDI.cpp.

References Compute().

10.31.4 Member Data Documentation

10.31.4.1 TDI_InterData* TDI::Eta `[protected]`

Memory where the signals Eta are stored.

Definition at line 52 of file LISACODE-TDI.h.

Referenced by Compute(), and TDI().

10.31.4.2 vector< vector<int> > TDI::IndexDelay `[protected]`

List of list of delay's index for each pack (value=[1,6]).

Definition at line 62 of file LISACODE-TDI.h.

Referenced by Compute(), NbDelayMax(), ReadSignEtaDelays(), and TDI().

10.31.4.3 vector<int> TDI::IndexEta `[protected]`

List of index of signal Eta where the pack work (value=[1,6]).

Definition at line 60 of file LISACODE-TDI.h.

Referenced by Compute(), ReadSignEtaDelays(), and TDI().

10.31.4.4 int TDI::iSerie `[protected]`

Index of the serie in memory RecordMem.

Definition at line 56 of file LISACODE-TDI.h.

Referenced by TDI().

10.31.4.5 ofstream* TDI::OutFile `[protected]`

File where TDI result are recorded.

Definition at line 54 of file LISACODE-TDI.h.

Referenced by TDI().

10.31.4.6 vector<int> TDI::Sign `[protected]`

List of sign of the pack (1 for + and -1 for -).

Definition at line 58 of file LISACODE-TDI.h.

Referenced by Compute(), ReadSignEtaDelays(), and TDI().

10.31.4.7 Memory* TDI::TDelay [protected]

Pointer to the list of delay's lengths.

Definition at line 50 of file LISACODE-TDI.h.

Referenced by Compute(), and TDI().

10.31.4.8 TDITools* TDI::TDIQuickMod [protected]

TDI access tools.

Definition at line 67 of file LISACODE-TDI.h.

Referenced by Compute(), and TDI().

10.31.4.9 int TDI::tmpCountInterDelay [protected]

Temporary data (unused).

Definition at line 69 of file LISACODE-TDI.h.

Referenced by getCountInterDelay(), and TDI().

10.31.4.10 int TDI::tmpCountInterEta [protected]

Temporary data (unused).

Definition at line 71 of file LISACODE-TDI.h.

Referenced by getCountInterEta(), and TDI().

The documentation for this class was generated from the following files:

- LISACODE-TDI.h
- LISACODE-TDI.cpp

10.32 TDI_InterData Class Reference

`#include <LISACODE-TDI_InterData.h>`

10.32.1 Detailed Description

Time Delay Interferometry interpolated signal class.

Definition at line 42 of file LISACODE-TDI_InterData.h.

Public Member Functions

- TDI_InterData ()

  Constructs an instance and initializes it with default values.

- TDI_InterData (Memory ∗TDelay_n, vector< Memory ∗ > ∗PDPMMem_n)

  Constructs an instance and initializes it with TDelay_n and PDPMMem_n inputs and default values.

- TDI_InterData (Memory ∗TDelay_n, vector< Memory ∗ > ∗PDPMMem_n, double TimeStore_n, double tShift_n, bool NoNoise, INTERP InterpType_n=LAG, double InterpUtilValue_n=6)

  Constructs an instance and initializes it with TDelay_n and PDPMMem_n inputs and default values.

- ∼TDI_InterData ()

  Desctrutor.

- double getTimeStore ()

  Returns TimeStore attribue .

- void setTimeStore (double TimeStored_n)

  Sets TimeStore attribute using TimeStore_n input.

- double gettStep ()

  Returns first Eta ref step attribue .

- bool getUsable ()

  Returns Usable attribute.

- double gettDelayCompute ()

  Returns tShift attribute.

- void ComputeEta ()

  Computes Eta using TimeStore and NoNoise attributes.

- double gData (int iSC, int IndirectDir, double Delay)

  Returns value interpolated for the delay (iSC=[1,3] and Indirect=[0,1].

- double gData (int iSerie, double Delay)

  Returns value interpolated for the delay (iSerie=[1,6]).

Protected Attributes

- Memory ∗ TDelay

  Pointer to the list of delay's lengths.

- vector< Memory ∗ > ∗ PDPMMem

  Memory where the signals of photodetector-phasemeter are stored.

- double TimeStore

  Time during which the values Eta are stored.

- vector< Serie > Eta

  List of the memory where Eta's values are stored.

- bool Usable

  True if enough data are stored.

- INTERP InterpType

  Type of interpolation used to obtain and to get data.

- double InterpUtilValue

  Value used for interpolation.

- double tShift

  Delay between the compute of Eta and the signals.

- bool NoNoise

  If true, there are no noise.

10.32.2 Constructor & Destructor Documentation

10.32.2.1 TDI_InterData::TDI_InterData ()

Constructs an instance and initializes it with default values.

Attributes are :

- TDelay = NULL
- PDPMMem = NULL
- TimeStore = 30.0
- tShift = 0.0
- Eta = 6 elements initialzed with tShift and 1.0 arguments
- Usable = false
- InterpType = LAG
- InterpUtilValue = 6

- NoNoise = false

Definition at line 29 of file LISACODE-TDI_InterData.cpp.

References Eta, InterpType, InterpUtilValue, LAG, NoNoise, PDPMMem, TDelay, TimeStore, tShift, and Usable.

10.32.2.2 TDI_InterData::TDI_InterData (Memory ∗ *TDelay_n*, vector< Memory ∗ > ∗ *PDPMMem_n*)

Constructs an instance and initializes it with TDelay_n and PDPMMem_n inputs and default values.

Attributes are :

- TDelay = TDelay_n
- PDPMMem = NULL
- TimeStore = 30.0
- tShift = 0.0
- Eta = 6 elements initialzed with tShift and 1.0 arguments
- Usable = false
- InterpType = LAG
- InterpUtilValue = 6
- NoNoise = false

Definition at line 59 of file LISACODE-TDI_InterData.cpp.

References Eta, InterpType, InterpUtilValue, LAG, NoNoise, PDPMMem, TDelay, TimeStore, tShift, and Usable.

10.32.2.3 TDI_InterData::TDI_InterData (Memory ∗ *TDelay_n*, vector< Memory ∗ > ∗ *PDPMMem_n*, double *TimeStore_n*, double *tShift_n*, bool *NoNoise_n*, INTERP *InterpType_n* = LAG, double *InterpUtilValue_n* = 6)

Constructs an instance and initializes it with TDelay_n and PDPMMem_n inputs and default values.

Attributes are :

- TDelay = TDelay_n
- PDPMMem = PDPMMem_n
- TimeStore = TimeStore_n (checked; expected positive or null value)
- tShift = tShift_n
- Eta = 6 elements initialzed with tShift and step (from PDPMMem) arguments
- Usable = false
- InterpType = InterpUtilValue_n

- InterpUtilValue = 6
- NoNoise = false

Definition at line 91 of file LISACODE-TDI_InterData.cpp.

References Eta, InterpType, InterpUtilValue, NoNoise, PDPMMem, TDelay, TimeStore, tShift, and Usable.

10.32.2.4 TDI_InterData::∼TDI_InterData ()

Desctrutor.

Definition at line 119 of file LISACODE-TDI_InterData.cpp.

10.32.3 Member Function Documentation

10.32.3.1 void TDI_InterData::ComputeEta ()

Computes Eta using TimeStore and NoNoise attributes.

for all spacecrafts (index iSC=1,2,3) :

- if there is no noise

  $Eta[iSC-1].addData((*PDPMMem)[iSC-1]->gData(0,tShift,InterpType,InterpUtilValue))$

  $Eta[iSC+2].addData((*PDPMMem)[iSC-1]->gData(1,tShift,InterpType,InterpUtilValue))$

  using Memory::gData method

- else the following value is added at the begining of Eta[iSC-1] serie, using Serie::addData method

$$
\begin{gathered}
(*PDPMMem)[iSC-1]->gData(0,tShift,InterpType,InterpUtilValue)\\
-\frac{1}{2}\cdot(*PDPMMem)[mod(iSC+1,3)]->gData(2,tShift+TDelay->gData(mod(iSC+2,3),tShift),\\
InterpType,InterpUtilValue)\\
-\frac{1}{2}\cdot(*PDPMMem)[mod(iSC+1,3)]->gData(3,tShift+TDelay->gData(mod(iSC+2,3),tShift),\\
InterpType,InterpUtilValue)
\end{gathered}
$$

  and the following value is added at the begining of Eta[iSC+2] serie, using Serie::addData method

$$
\begin{gathered}
(*PDPMMem)[iSC-1]->gData(1,tShift,InterpType,InterpUtilValue)\\
+\frac{1}{2}\cdot(*PDPMMem)[iSC-1]->gData(2,tShift,InterpType,InterpUtilValue)\\
-\frac{1}{2}\cdot(*PDPMMem)[iSC-1]->gData(3,tShift,InterpType,InterpUtilValue)
\end{gathered}
$$

Last data of series Eta[iSC+2] and Eta[iSC-1], while x reference is greater than TimeStore input, using Serie::delLastData method.

Definition at line 161 of file LISACODE-TDI_InterData.cpp.

References Eta, Memory::gData(), gData(), InterpType, InterpUtilValue, NoNoise, PDPMMem, TDelay, TimeStore, tShift, and Usable.

Referenced by main().

10.32.3.2 double TDI_InterData::gData (int *iSerie*, double *Delay*)

Returns value interpolated for the delay (iSerie=[1,6]).

Input is checked :

- Serie index iSerie must be 1, 2, 3, 4, 5 or 6

Eta index is computed : $iSerie - 1$.

Interpolation for Delay reference is computed by Serie::gData method, using InterpType and InterpUtilValue attributes

Definition at line 247 of file LISACODE-TDI_InterData.cpp.

References Eta, InterpType, and InterpUtilValue.

10.32.3.3 double TDI_InterData::gData (int *iSC*, int *IndirectDir*, double *Delay*)

Returns value interpolated for the delay (iSC=[1,3] and Indirect=[0,1].

Inputs are checked :

- Spacecraft index iSC must be 1, 2 or 3
- The travel direction IndirectDir must be 0 or 1

Eta index is computed : $iSC - 1 + 2 \cdot IndirectDir$.

Interpolation for Delay reference is computed by Serie::gData method, using InterpType and InterpUtilValue attributes.

Definition at line 224 of file LISACODE-TDI_InterData.cpp.

References Eta, InterpType, and InterpUtilValue.

Referenced by TDI::Compute(), and ComputeEta().

10.32.3.4 double TDI_InterData::gettDelayCompute () `[inline]`

Returns tShift attribute.

Definition at line 88 of file LISACODE-TDI_InterData.h.

References tShift.

10.32.3.5 double TDI_InterData::getTimeStore () `[inline]`

Returns TimeStore attribue .

Definition at line 81 of file LISACODE-TDI_InterData.h.

References TimeStore.

10.32.3.6 double TDI_InterData::gettStep () `[inline]`

Returns first Eta ref step attribue .

Definition at line 84 of file LISACODE-TDI_InterData.h.

References Eta.

10.32.3.7 bool TDI_InterData::getUsable () [inline]

Returns Usable attribute.

Definition at line 86 of file LISACODE-TDI_InterData.h.

References Usable.

Referenced by TDI::Compute().

10.32.3.8 void TDI_InterData::setTimeStore (double *TimeStore_n*)

Sets TimeStore attribute using TimeStore_n input.

TimeStore_n input is checked : it is expected to be positive or null.

Definition at line 130 of file LISACODE-TDI_InterData.cpp.

References TimeStore.

10.32.4 Member Data Documentation

10.32.4.1 vector<Serie> TDI_InterData::Eta [protected]

List of the memory where Eta's values are stored.

Definition at line 52 of file LISACODE-TDI_InterData.h.

Referenced by ComputeEta(), gData(), gettStep(), and TDI_InterData().

10.32.4.2 INTERP TDI_InterData::InterpType [protected]

Type of interpolation used to obtain and to get data.

Definition at line 56 of file LISACODE-TDI_InterData.h.

Referenced by ComputeEta(), gData(), and TDI_InterData().

10.32.4.3 double TDI_InterData::InterpUtilValue [protected]

Value used for interpolation.

Definition at line 58 of file LISACODE-TDI_InterData.h.

Referenced by ComputeEta(), gData(), and TDI_InterData().

10.32.4.4 bool TDI_InterData::NoNoise [protected]

If true, there are no noise.

Definition at line 62 of file LISACODE-TDI_InterData.h.

Referenced by ComputeEta(), and TDI_InterData().

10.32.4.5 vector<Memory *>* TDI_InterData::PDPMMem [protected]

Memory where the signals of photodetector-phasemeter are stored.

Definition at line 48 of file LISACODE-TDI_InterData.h.

Referenced by ComputeEta(), and TDI_InterData().

10.32.4.6 Memory* TDI_InterData::TDelay [protected]

Pointer to the list of delay's lengths.

Definition at line 46 of file LISACODE-TDI_InterData.h.

Referenced by ComputeEta(), and TDI_InterData().

10.32.4.7 double TDI_InterData::TimeStore [protected]

Time during which the values Eta are stored.

Definition at line 50 of file LISACODE-TDI_InterData.h.

Referenced by ComputeEta(), getTimeStore(), setTimeStore(), and TDI_InterData().

10.32.4.8 double TDI_InterData::tShift [protected]

Delay between the compute of Eta and the signals.

Definition at line 60 of file LISACODE-TDI_InterData.h.

Referenced by ComputeEta(), gettDelayCompute(), and TDI_InterData().

10.32.4.9 bool TDI_InterData::Usable [protected]

True if enough data are stored.

Definition at line 54 of file LISACODE-TDI_InterData.h.

Referenced by ComputeEta(), getUsable(), and TDI_InterData().

The documentation for this class was generated from the following files:

- LISACODE-TDI_InterData.h
- LISACODE-TDI_InterData.cpp

10.33 TDITools Class Reference

```
#include <LISACODE-TDITools.h>
```

10.33.1 Detailed Description

Time Delay Interferometry tools class.

Definition at line 38 of file LISACODE-TDITools.h.

Public Member Functions

- TDITools ()
  Constructs an instance and initializes it with default values.
- TDITools (Memory ∗TDelay_n, bool RapidOption_n)
  Constructs an instance and initializes it with inputs default values.
- virtual ∼TDITools ()
  Destructor.
- double getDelay (int IndexDelay)
  Returns DelayMem[mod(IndexDelay-1,3)] attribute.
- bool getRapidOption ()
  Returns RapidOption attribute.
- void RefreshDelay (double tComputeDelay)
  Updates DelayMem attribute.

Protected Attributes

- Memory ∗ TDelay
  Pointer to the list of delay's lengths.
- double DelayMem [3]
  Memory of the 3 delays for the current time.
- bool RapidOption
  If it's TRUE, approximations are done in order to compute delays more quickly.

10.33.2 Constructor & Destructor Documentation

10.33.2.1 TDITools::TDITools ()

Constructs an instance and initializes it with default values.

Attributes are :

- TDelay = empty
- DelayMem = (0,0,0)
- RapidOption = FALSE

Definition at line 23 of file LISACODE-TDITools.cpp.

References DelayMem, RapidOption, and TDelay.

10.33.2.2 TDITools::TDITools (Memory ∗ *TDelay_n*, bool *RapidOption_n*)

Constructs an instance and initializes it with inputs default values.

Attributes are :

- TDelay = TDelay_n input
- DelayMem = (0,0,0)
- RapidOption = RapidOption_n input

Definition at line 39 of file LISACODE-TDITools.cpp.

References DelayMem, RapidOption, and TDelay.

10.33.2.3 TDITools::∼TDITools () [virtual]

Destructor.

Definition at line 49 of file LISACODE-TDITools.cpp.

10.33.3 Member Function Documentation

10.33.3.1 double TDITools::getDelay (int *IndexDelay*)

Returns DelayMem[mod(IndexDelay-1,3)] attribute.

Definition at line 57 of file LISACODE-TDITools.cpp.

References DelayMem.

Referenced by TDI::Compute().

10.33.3.2 bool TDITools::getRapidOption () [inline]

Returns RapidOption attribute.

Definition at line 58 of file LISACODE-TDITools.h.

References RapidOption.

Referenced by TDI::Compute(), and main().

10.33.3.3 void TDITools::RefreshDelay (double *tComputeDelay*)

Updates DelayMem attribute.

For ech delay (0,1,2), Memory::gdata method is used with tComputeDelay input delay, truncated interpolation type, and zero InterpUtilValue.

Definition at line 73 of file LISACODE-TDITools.cpp.

References DelayMem, Memory::gData(), TDelay, and TRU.

Referenced by main().

10.33.4 Member Data Documentation

10.33.4.1 double TDITools::DelayMem[3] `[protected]`

Memory of the 3 delays for the current time.

Definition at line 44 of file LISACODE-TDITools.h.

Referenced by getDelay(), RefreshDelay(), and TDITools().

10.33.4.2 bool TDITools::RapidOption `[protected]`

If it's TRUE, approximations are done in order to compute delays more quickly.

Definition at line 46 of file LISACODE-TDITools.h.

Referenced by getRapidOption(), and TDITools().

10.33.4.3 Memory* TDITools::TDelay `[protected]`

Pointer to the list of delay's lengths.

Definition at line 42 of file LISACODE-TDITools.h.

Referenced by RefreshDelay(), and TDITools().

The documentation for this class was generated from the following files:

- LISACODE-TDITools.h
- LISACODE-TDITools.cpp

10.34 TrFctGW Class Reference

`#include <LISACODE-TrFctGW.h>`

10.34.1 Detailed Description

Gravitational Waves Transfer Function class.

Definition at line 33 of file LISACODE-TrFctGW.h.

Public Member Functions

- TrFctGW ()

  Constructs an instance and initializes it with default values.

- TrFctGW (vector< GW ∗ > ∗GWSources_n, Geometry ∗LISAGeo_n)

  Constructs an instance and initializes it with default values.

- ∼TrFctGW ()
- void init (vector< GW ∗ > ∗GWSources_n, Geometry ∗LISAGeo_n)

  Initializes an instance with default values and inputs.

- double deltanu (int rec, int em, int order, double trec)

  Returns the fluctuation frequency due to GW.

Protected Attributes

- vector< GW ∗ > ∗ GWSources

  Gravitational Waves sources.

- Geometry ∗ LISAGeo

  LISA's geometry.

- vector< Vect > u

  Unit transverse vector.

- vector< Vect > v

  Unit transverse vector.

- vector< Vect > k

  (k,u,v) is direct trihedron.

10.34.2 Constructor & Destructor Documentation

10.34.2.1 TrFctGW::TrFctGW ()

Constructs an instance and initializes it with default values.

init method is called

Definition at line 20 of file LISACODE-TrFctGW.cpp.

References init().

10.34.2.2 TrFctGW::TrFctGW (vector< GW * > * *GWSources_n*, Geometry * *LISAGeo_n*)

Constructs an instance and initializes it with default values.

init method is called

Definition at line 37 of file LISACODE-TrFctGW.cpp.

References init().

10.34.2.3 TrFctGW::∼TrFctGW ()

Definition at line 44 of file LISACODE-TrFctGW.cpp.

10.34.3 Member Function Documentation

10.34.3.1 double TrFctGW::deltanu (int *rec*, int *em*, int *order*, double *trec*)

Returns the fluctuation frequency due to GW.

$$tem = trec + LISAGeo- > gtdelay(em, rec, order, trec)$$

$$\overrightarrow{rrec} = \frac{LISAGeo- > gposition(rec, trec)}{C}$$

$$\overrightarrow{rem} = LISAGeo- > gposition(em, trec)$$

$$\overrightarrow{dr} = \overrightarrow{rrec} - \overrightarrow{rem}$$

$$\overrightarrow{n} = \frac{\overrightarrow{dr}}{||\overrightarrow{dr}||}$$

$$\delta_\nu = 0$$

For each source (iGWindex) in GWSources,

$$hpr = (*GWSources)[iGW]- > hp(trec - \overrightarrow{k[iGW]} \cdot \overrightarrow{rrec})$$

$$hpe = (*GWSources)[iGW]- > hp(tem - \overrightarrow{k[iGW]} \cdot \overrightarrow{rem})$$

$$hcr = (*GWSources)[iGW]- > hc(trec - \overrightarrow{k[iGW]} \cdot \overrightarrow{rrec})$$

$$hce = (*GWSources)[iGW]- > hc(tem - \overrightarrow{k[iGW]} \cdot \overrightarrow{rem})$$

$$\delta_\nu = \delta_\nu + \frac{(hpe - hpr) \cdot \frac{(\overrightarrow{u[iGW]} \cdot \overrightarrow{n})^2 - (\overrightarrow{v[iGW]} \cdot \overrightarrow{n})^2}{2} + (hce - hcr) \cdot (\overrightarrow{u[iGW]} \cdot \overrightarrow{n}) \cdot (\overrightarrow{v[iGW]} \cdot \overrightarrow{n})}{1 - \frac{\overrightarrow{k[iGW]} \cdot \overrightarrow{rrec} - \overrightarrow{k[iGW]} \cdot \overrightarrow{rem}}{\|\overrightarrow{dr}\|}}$$

Definition at line 148 of file LISACODE-TrFctGW.cpp.

References c_SI, Geometry::gposition(), Geometry::gtdelay(), GWSources, k, LISAGeo, Vect::norme(), u, and v.

Referenced by main(), and PhoDetPhaMet::ReceiveSignal().

10.34.3.2 void TrFctGW::init (vector< GW * > * *GWSources_n*, Geometry * *LISAGeo_n*)

Initializes an instance with default values and inputs.

GWSources attribute = GWSources_n input

LISAGeo attribute = LISAGeo_n input

size of u, v, and k attributes is set to GWSources_n input size

- For each source in GWSources (GW index) :
  (k,u,v) is direct,
  the ecliptic latitude is $\beta = \Pi - \theta$ and the ecliptic longitude $\phi = \lambda$.

$$\lambda = \lambda_{GW} + \pi$$

$$\beta = -\beta_{GW} + \pi$$

$$\psi = AnglPol_{GW}$$

$$k = \begin{pmatrix} cos(\lambda) \cdot cos(\beta) \\ sin(\lambda) \cdot cos(\beta) \\ sin(\beta) \end{pmatrix}$$

$$u = \begin{pmatrix} sin(\beta) \cdot cos(\lambda) \cdot cos(\psi) + sin(\lambda) \cdot sin(\psi) \\ sin(\beta) \cdot sin(\lambda) \cdot cos(\psi) - cos(\lambda) \cdot cos(\psi) \\ -cos(\beta) \cdot cos(\psi) \end{pmatrix}$$

$$v = \begin{pmatrix} sin(\beta) \cdot cos(\lambda) \cdot sin(\psi) - sin(\lambda) \cdot cos(\psi) \\ sin(\beta) \cdot sin(\lambda) \cdot sin(\psi) + cos(\lambda) \cdot cos(\psi) \\ -cos(\beta) \cdot sin(\psi) \end{pmatrix}$$

Definition at line 74 of file LISACODE-TrFctGW.cpp.

References GWSources, k, LISAGeo, u, and v.

Referenced by LISA::LISA(), and TrFctGW().

10.34.4 Member Data Documentation

10.34.4.1 vector<GW *>* TrFctGW::GWSources [protected]

Gravitational Waves sources.

Definition at line 38 of file LISACODE-TrFctGW.h.

Referenced by deltanu(), and init().

10.34.4.2 TrFctGW::k [protected]

(k,u,v) is direct trihedron.

Referenced by deltanu(), and init().

10.34.4.3 Geometry* TrFctGW::LISAGeo [protected]

LISA's geometry.

Definition at line 40 of file LISACODE-TrFctGW.h.

Referenced by deltanu(), and init().

10.34.4.4 TrFctGW::u [protected]

Unit transverse vector.

Referenced by deltanu(), and init().

10.34.4.5 TrFctGW::v [protected]

Unit transverse vector.

Referenced by deltanu(), and init().

The documentation for this class was generated from the following files:

- LISACODE-TrFctGW.h
- LISACODE-TrFctGW.cpp

10.35 USOClock Class Reference

`#include <LISACODE-USOClock.h>`

10.35.1 Detailed Description

Ultra Stable Oscillator based satellite time is defined in this class.

Definition at line 43 of file LISACODE-USOClock.h.

Public Member Functions

- USOClock ()
  Constructs an instance and initializes it with zero value for all attributes.
- USOClock (double Offset_n)
  Constructs an instance and initializes it using Offset_n input.
- USOClock (double Offset_n, double DerivLinearCoef_n, double SigmaNoise_n)
  Constructs an instance and initializes it using Offset_n, DerivLinearCoef_n and SigmaNoise_n inputs.
- ∼USOClock ()
  Destructor.
- void init (double Offset_n, double DerivLinearCoef_n, double SigmaNoise_n)
  Sets attributes using Offset_n, DerivLinearCoef_n and SigmaNoise_n inputs.
- double getOffset ()
  Returns Offset attribute.
- double getDeriv ()
  Returns DerivLinearCoef attribute.
- double getNoise ()
  Returns SigmaNoise attribute.
- double gGap (double t, double tStep)
  Computes gap using t and tStep inputs and attributes.
- double gTime (double t, double tStep)
  Computes time using t and tStep inputs and gGap method.

Protected Attributes

- double Offset
  Offset.
- double DerivLinearCoef

Slope.

- double SigmaNoise

  USO noise in second per second.

- bool USONoise

  FALSE if the is no noise, else TRUE.

10.35.2 Constructor & Destructor Documentation

10.35.2.1 USOClock::USOClock ()

Constructs an instance and initializes it with zero value for all attributes.

Attributes are :

- Offset = 0
- DerivLinearCoef = 0
- SigmaNoise = 0
- USONoise = FALSE

Definition at line 23 of file LISACODE-USOClock.cpp.

References init().

10.35.2.2 USOClock::USOClock (double *Offset_n*)

Constructs an instance and initializes it using Offset_n input.

Attributes are :

- Offset = Offset_n
- DerivLinearCoef = 0
- SigmaNoise = 0
- USONoise = FALSE

Definition at line 37 of file LISACODE-USOClock.cpp.

References init().

10.35.2.3 USOClock::USOClock (double *Offset_n*, double *DerivLinearCoef_n*, double *SigmaNoise_n*)

Constructs an instance and initializes it using Offset_n, DerivLinearCoef_n and SigmaNoise_n inputs.

Attributes are :

- Offset = Offset_n

- DerivLinearCoef = DerivLinearCoef_n
- SigmaNoise = SigmaNoise_n
- USONoise = TRUE if $SigmaNoise > 10^{-20}$, else FALSE

Definition at line 51 of file LISACODE-USOClock.cpp.

References init().

10.35.2.4 USOClock::~USOClock ()

Destructor.

Definition at line 58 of file LISACODE-USOClock.cpp.

10.35.3 Member Function Documentation

10.35.3.1 double USOClock::getDeriv () `[inline]`

Returns DerivLinearCoef attribute.

Definition at line 69 of file LISACODE-USOClock.h.

References DerivLinearCoef.

10.35.3.2 double USOClock::getNoise () `[inline]`

Returns SigmaNoise attribute.

Definition at line 71 of file LISACODE-USOClock.h.

References SigmaNoise.

10.35.3.3 double USOClock::getOffset () `[inline]`

Returns Offset attribute.

Definition at line 67 of file LISACODE-USOClock.h.

References Offset.

10.35.3.4 double USOClock::gGap (double *t*, double *tStep*)

Computes gap using t and tStep inputs and attributes.

$$gGap = Offset + DerivLinearCoef \cdot t$$

If there is noise, TimeNoise must be added

$$TimeNoise = (SigmaNoise \cdot tStep) \cdot \sqrt{(-2.0 \cdot log(r2)) \cdot cos(2 \cdot \pi \cdot r1))}$$

where $r1$ and $r2$ are random values between 0 and 1.

Returns:
gGap

Definition at line 95 of file LISACODE-USOClock.cpp.

References DerivLinearCoef, genunf(), Offset, SigmaNoise, and USONoise.

Referenced by gTime(), and PhoDetPhaMet::ReceiveSignal().

10.35.3.5 double USOClock::gTime (double *t*, double *tStep*)

Computes time using t and tStep inputs and gGap method.

Returns:

$$t + gGap(t, tStep)$$

Definition at line 117 of file LISACODE-USOClock.cpp.

References gGap().

10.35.3.6 void USOClock::init (double *Offset_n*, double *DerivLinearCoef_n*, double *SigmaNoise_n*)

Sets attributes using Offset_n, DerivLinearCoef_n and SigmaNoise_n inputs.

Set attributes are :

- Offset = Offset_n
- DerivLinearCoef = DerivLinearCoef_n
- SigmaNoise = SigmaNoise_n
- USONoise = TRUE if $SigmaNoise > 10^{-20}$, else FALSE

Definition at line 73 of file LISACODE-USOClock.cpp.

References DerivLinearCoef, Offset, SigmaNoise, and USONoise.

Referenced by USOClock().

10.35.4 Member Data Documentation

10.35.4.1 double USOClock::DerivLinearCoef `[protected]`

Slope.

Definition at line 49 of file LISACODE-USOClock.h.

Referenced by getDeriv(), gGap(), and init().

10.35.4.2 double USOClock::Offset `[protected]`

Offset.

Definition at line 47 of file LISACODE-USOClock.h.

Referenced by getOffset(), gGap(), and init().

10.35.4.3 double USOClock::SigmaNoise [protected]

USO noise in second per second.

Definition at line 51 of file LISACODE-USOClock.h.

Referenced by getNoise(), gGap(), and init().

10.35.4.4 bool USOClock::USONoise [protected]

FALSE if the is no noise, else TRUE.

Definition at line 53 of file LISACODE-USOClock.h.

Referenced by gGap(), and init().

The documentation for this class was generated from the following files:

- LISACODE-USOClock.h
- LISACODE-USOClock.cpp

10.36 Vect Class Reference

```
#include <LISACODE-Vect.h>
```

10.36.1 Detailed Description

3 components vector management class.

Definition at line 29 of file LISACODE-Vect.h.

Public Member Functions

- Vect ()

  Constructs an instance and initializes it with default values.

- Vect (double[3])

  Constructs an instance and initializes it with t input.

- ∼Vect ()

  Destructor.

- void display ()

  Displays vector components.

- double norme ()

  Returns vector norm.

- Vect unit ()

  Returns unit vector.

Public Attributes

- double p [3]

  3 components.

Friends

- Vect operator+ (Vect, Vect)

  Vectors addition : returns vector u+v.

- Vect operator- (Vect, Vect)

  Vectors subtraction : returns vector u-v.

- double operator ∗ (Vect, Vect)

  Vectors scalar product, returns a scalar.

- Vect operator ∗ (double, Vect)

Vector and scalar product, returns a vector.

- Vect operator $*$ (Vect, double)

  Vector and scalar product, returns a vector.

- Vect operator/ (Vect, double)

  Vector and scalar division, returns a vector.

10.36.2 Constructor & Destructor Documentation

10.36.2.1 Vect::Vect ()

Constructs an instance and initializes it with default values.

p attribute is set

$$p = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Definition at line 21 of file LISACODE-Vect.cpp.

References p.

10.36.2.2 Vect::Vect (double *t*[3])

Constructs an instance and initializes it with t input.

p attribute is set to t input

Definition at line 32 of file LISACODE-Vect.cpp.

References p.

10.36.2.3 Vect::$\sim$Vect ()

Destructor.

Definition at line 41 of file LISACODE-Vect.cpp.

10.36.3 Member Function Documentation

10.36.3.1 void Vect::display ()

Displays vector components.

Definition at line 49 of file LISACODE-Vect.cpp.

References p.

10.36.3.2 double Vect::norme ()

Returns vector norm.

$$\text{returned value} = \sqrt{(\overrightarrow{p} \cdot \overrightarrow{p})}$$

Definition at line 64 of file LISACODE-Vect.cpp.

References p.

Referenced by TrFctGW::deltanu(), Geometry::tdelay(), and Geometry::tdelayOrderContribution().

10.36.3.3 Vect Vect::unit ()

Returns unit vector.

norme method is called

$$\text{returned value} = \frac{\overrightarrow{p}}{\sqrt{(\overrightarrow{p} \cdot \overrightarrow{p})}}$$

Definition at line 82 of file LISACODE-Vect.cpp.

References p.

Referenced by Geometry::ArmVelocity(), Geometry::tdelay(), Geometry::tdelayOrderContribution(), and Geometry::VectNormal().

10.36.4 Friends And Related Function Documentation

10.36.4.1 Vect operator * (Vect *u*, double *a*) `[friend]`

Vector and scalar product, returns a vector.

$$\text{returned value} = a \cdot \overrightarrow{u}$$

Definition at line 157 of file LISACODE-Vect.cpp.

10.36.4.2 Vect operator * (double *a*, Vect *u*) `[friend]`

Vector and scalar product, returns a vector.

$$\text{returned value} = a \cdot \overrightarrow{u}$$

Definition at line 143 of file LISACODE-Vect.cpp.

10.36.4.3 double operator * (Vect *u*, Vect *v*) `[friend]`

Vectors scalar product, returns a scalar.

$$\text{returned value} = \overrightarrow{u} \cdot \overrightarrow{v}$$

Definition at line 128 of file LISACODE-Vect.cpp.

10.36.4.4 Vect operator+ (Vect *u*, Vect *v*) `[friend]`

Vectors addition : returns vector u+v.

Definition at line 101 of file LISACODE-Vect.cpp.

10.36.4.5 Vect operator- (Vect *u*, Vect *v*) `[friend]`

Vectors subtraction : returns vector u-v.

Definition at line 112 of file LISACODE-Vect.cpp.

10.36.4.6 Vect operator/ (Vect *u*, double *a*) `[friend]`

Vector and scalar division, returns a vector.

$$\text{returned value} = \frac{\overrightarrow{u}}{a}$$

Definition at line 171 of file LISACODE-Vect.cpp.

10.36.5 Member Data Documentation

10.36.5.1 double Vect::p[3]

3 components.

Definition at line 33 of file LISACODE-Vect.h.

Referenced by display(), main(), norme(), operator $*$(), operator+(), operator-(), operator/(), Geometry::position(), unit(), Vect(), Geometry::VectNormal(), and Geometry::velocity().

The documentation for this class was generated from the following files:

- LISACODE-Vect.h
- LISACODE-Vect.cpp

Chapter 11

LISACode File Documentation

11.1 com.c File Reference

```
#include "randlib.h"
#include <stdio.h>
#include <stdlib.h>
```

Defines

- #define numg 32L
- #define numg 32L
- #define numg 32L
- #define numg 32L
- #define numg 32L
- #define numg 32L
- #define numg 32L
- #define numg 32L

Functions

- void advnst (long k)
- void getsd (long ∗iseed1, long ∗iseed2)
- long ignlgi (void)
- void initgn (long isdtyp)
- void inrgcm (void)
- void setall (long iseed1, long iseed2)
- void setant (long qvalue)
- void setsd (long iseed1, long iseed2)

Variables

- long Xm1
- long Xm2

- long Xa1
- long Xa2
- long Xcg1 [32]
- long Xcg2 [32]
- long Xa1w
- long Xa2w
- long Xig1 [32]
- long Xig2 [32]
- long Xlg1 [32]
- long Xlg2 [32]
- long Xa1vw
- long Xa2vw
- long Xqanti [32]

11.1.1 Define Documentation

11.1.1.1 #define numg 32L

11.1.1.2 #define numg 32L

11.1.1.3 #define numg 32L

11.1.1.4 #define numg 32L

11.1.1.5 #define numg 32L

11.1.1.6 #define numg 32L

11.1.1.7 #define numg 32L

11.1.1.8 #define numg 32L

11.1.2 Function Documentation

11.1.2.1 void advnst (long *k*)

Definition at line 7 of file com.c.

References gscgn(), gsrgs(), mltmod(), setsd(), Xa1, Xa2, Xcg1, Xcg2, Xm1, and Xm2.

11.1.2.2 void getsd (long ∗ *iseed1*, long ∗ *iseed2*)

Definition at line 52 of file com.c.

References gscgn(), gsrgs(), Xcg1, and Xcg2.

Referenced by main().

11.1.2.3 long ignlgi (void)

Definition at line 89 of file com.c.

References gscgn(), gsrgs(), gssst(), ignlgi(), inrgcm(), setall(), Xa1, Xa2, Xcg1, Xcg2, Xm1, Xm2, and Xqanti.

Referenced by ignlgi(), ignuin(), and ranf().

11.1.2.4 void initgn (long *isdtyp*)

Definition at line 143 of file com.c.

References gscgn(), gsrgs(), mltmod(), Xa1w, Xa2w, Xcg1, Xcg2, Xig1, Xig2, Xlg1, Xlg2, Xm1, and Xm2.

Referenced by setall(), and setsd().

11.1.2.5 void inrgcm (void)

Definition at line 203 of file com.c.

References gsrgs(), Xa1, Xa1vw, Xa1w, Xa2, Xa2vw, Xa2w, Xm1, Xm2, and Xqanti.

Referenced by ignlgi(), and setall().

11.1.2.6 void setall (long *iseed1*, long *iseed2*)

Definition at line 245 of file com.c.

References gscgn(), gsrgs(), gssst(), initgn(), inrgcm(), mltmod(), Xa1vw, Xa2vw, Xig1, Xig2, Xm1, and Xm2.

Referenced by ignlgi(), and main().

11.1.2.7 void setant (long *qvalue*)

Definition at line 296 of file com.c.

References gscgn(), gsrgs(), and Xqanti.

11.1.2.8 void setsd (long *iseed1*, long *iseed2*)

Definition at line 337 of file com.c.

References gscgn(), gsrgs(), initgn(), Xig1, and Xig2.

Referenced by advnst().

11.1.3 Variable Documentation

11.1.3.1 long Xa1 `[static]`

Definition at line 4 of file com.c.

Referenced by advnst(), ignlgi(), and inrgcm().

11.1.3.2 long Xa1vw [static]

Definition at line 4 of file com.c.

Referenced by inrgcm(), and setall().

11.1.3.3 long Xa1w [static]

Definition at line 4 of file com.c.

Referenced by initgn(), and inrgcm().

11.1.3.4 long Xa2 [static]

Definition at line 4 of file com.c.

Referenced by advnst(), ignlgi(), and inrgcm().

11.1.3.5 long Xa2vw [static]

Definition at line 4 of file com.c.

Referenced by inrgcm(), and setall().

11.1.3.6 long Xa2w [static]

Definition at line 4 of file com.c.

Referenced by initgn(), and inrgcm().

11.1.3.7 long Xcg1[32] [static]

Definition at line 4 of file com.c.

Referenced by advnst(), getsd(), ignlgi(), and initgn().

11.1.3.8 long Xcg2[32] [static]

Definition at line 4 of file com.c.

Referenced by advnst(), getsd(), ignlgi(), and initgn().

11.1.3.9 long Xig1[32] [static]

Definition at line 4 of file com.c.

Referenced by initgn(), setall(), and setsd().

11.1.3.10 long Xig2[32] [static]

Definition at line 4 of file com.c.

Referenced by initgn(), setall(), and setsd().

11.1.3.11 long Xlg1[32] `[static]`

Definition at line 4 of file com.c.

Referenced by initgn().

11.1.3.12 long Xlg2[32] `[static]`

Definition at line 4 of file com.c.

Referenced by initgn().

11.1.3.13 long Xm1 `[static]`

Definition at line 4 of file com.c.

Referenced by advnst(), ignlgi(), initgn(), inrgcm(), and setall().

11.1.3.14 long Xm2 `[static]`

Definition at line 4 of file com.c.

Referenced by advnst(), ignlgi(), initgn(), inrgcm(), and setall().

11.1.3.15 long Xqanti[32] `[static]`

Definition at line 6 of file com.c.

Referenced by ignlgi(), inrgcm(), and setant().

11.2 Doxygen.bibliography File Reference

11.3 ezxml.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <ctype.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include "ezxml.h"
```

Classes

- struct ezxml_root

Defines

- #define EZXML_WS "\t\r\n "
- #define EZXML_ERRL 128

Typedefs

- typedef ezxml_root * ezxml_root_t

Functions

- ezxml_t ezxml_child (ezxml_t xml, const char *name)

  Returns the first child tag (one level deeper) with the given name or NULL if not found.

- ezxml_t ezxml_idx (ezxml_t xml, int idx)

  Returns the Nth tag with the same name in the same section at the same depth or NULL if not found. An index of 0 returns the tag given.

- const char * ezxml_attr (ezxml_t xml, const char *attr)

  Returns the value of the requested tag attribute, or NULL if not found.

- ezxml_t ezxml_vget (ezxml_t xml, va_list ap)
- ezxml_t ezxml_get (ezxml_t xml,...)

  Traverses the ezxml sturcture to retrieve a specific subtag.

- const char ** ezxml_pi (ezxml_t xml, const char *target)

  Returns a NULL terminated array of processing instructions for the given target.

- ezxml_t ezxml_err (ezxml_root_t root, char ∗s, const char ∗err,...)
- char ∗ ezxml_decode (char ∗s, char ∗∗ent, char t)
- void ezxml_open_tag (ezxml_root_t root, char ∗name, char ∗∗attr)
- void ezxml_char_content (ezxml_root_t root, char ∗s, size_t len, char t)
- ezxml_t ezxml_close_tag (ezxml_root_t root, char ∗name, char ∗s)
- int ezxml_ent_ok (char ∗name, char ∗s, char ∗∗ent)
- void ezxml_proc_inst (ezxml_root_t root, char ∗s, size_t len)
- short ezxml_internal_dtd (ezxml_root_t root, char ∗s, size_t len)
- char ∗ ezxml_str2utf8 (char ∗∗s, size_t ∗len)
- void ezxml_free_attr (char ∗∗attr)
- ezxml_t ezxml_parse_str (char ∗s, size_t len)

    Given a string of xml data and its length, parses it and creates an ezxml structure. For efficiency, modifies the data by adding null terminators and decoding ampersand sequences. If you don't want this, copy the data and pass in the copy. Returns NULL on failure.

- ezxml_t ezxml_parse_fp (FILE ∗fp)

    Wrapper for ezxml_parse_str() that accepts a file stream. Reads the entire stream into memory and then parses it. For xml files, use ezxml_parse_file() or ezxml_parse_fd().

- ezxml_t ezxml_parse_fd (int fd)

    A wrapper for ezxml_parse_str() that accepts a file descriptor. First attempts to mem map the file. Failing that, reads the file into memory. Returns NULL on failure.

- ezxml_t ezxml_parse_file (const char ∗file)

    a wrapper for ezxml_parse_fd() that accepts a file name

- char ∗ ezxml_ampencode (const char ∗s, size_t len, char ∗∗dst, size_t ∗dlen, size_t ∗max, short a)
- char ∗ ezxml_toxml_r (ezxml_t xml, char ∗∗s, size_t ∗len, size_t ∗max, size_t start, char ∗∗∗attr)
- char ∗ ezxml_toxml (ezxml_t xml)

    Converts an ezxml structure back to xml. Returns a string of xml data that must be freed.

- void ezxml_free (ezxml_t xml)

    Frees the memory allocated for an ezxml structure.

- const char ∗ ezxml_error (ezxml_t xml)

    Returns parser error message or empty string if none.

- ezxml_t ezxml_new (const char ∗name)

    Returns a new empty ezxml structure with the given root tag name.

- ezxml_t ezxml_add_child (ezxml_t xml, const char ∗name, size_t off)

    Adds a child tag. off is the offset of the child tag relative to the start of the parent tag's character content. Returns the child tag.

- ezxml_t ezxml_set_txt (ezxml_t xml, const char ∗txt)

    Sets the character content for the given tag and returns the tag.

- void ezxml_set_attr (ezxml_t xml, const char ∗name, const char ∗value)

    Sets the given tag attribute or adds a new attribute if not found. A value of NULL will remove the specified attribute.

- ezxml_t ezxml_set_flag (ezxml_t xml, short flag)

  Sets a flag for the given tag and returns the tag.

- void ezxml_remove (ezxml_t xml)

  Removes a tag along with all its subtags.

Variables

- char ∗ EZXML_NIL [ ] = { NULL }

11.3.1 Define Documentation

11.3.1.1 #define EZXML_ERRL 128

Definition at line 39 of file ezxml.c.

Referenced by ezxml_err().

11.3.1.2 #define EZXML_WS "\t\r\n "

Definition at line 38 of file ezxml.c.

Referenced by ezxml_internal_dtd(), ezxml_parse_str(), and ezxml_proc_inst().

11.3.2 Typedef Documentation

11.3.2.1 typedef struct ezxml_root∗ ezxml_root_t

Definition at line 41 of file ezxml.c.

Referenced by ezxml_attr(), ezxml_char_content(), ezxml_close_tag(), ezxml_err(), ezxml_error(), ezxml_free(), ezxml_internal_dtd(), ezxml_new(), ezxml_open_tag(), ezxml_parse_fd(), ezxml_parse_-fp(), ezxml_parse_str(), ezxml_pi(), ezxml_proc_inst(), and ezxml_toxml().

11.3.3 Function Documentation

11.3.3.1 ezxml_t ezxml_add_child (ezxml_t *xml*, const char ∗ *name*, size_t *off*)

Adds a child tag. off is the offset of the child tag relative to the start of the parent tag's character content. Returns the child tag.

Definition at line 850 of file ezxml.c.

References ezxml::attr, ezxml::child, EZXML_NIL, ezxml_t, ezxml::name, ezxml::next, ezxml::off, ezxml::ordered, ezxml::parent, ezxml::sibling, and ezxml::txt.

Referenced by ezxml_open_tag().

11.3.3.2 char* ezxml_ampencode (const char * *s*, size_t *len*, char ** *dst*, size_t * *dlen*, size_t * *max*, short *a*)

Definition at line 667 of file ezxml.c.

References EZXML_BUFSIZE, and max.

Referenced by ezxml_toxml_r().

11.3.3.3 const char* ezxml_attr (ezxml_t *xml*, const char * *attr*)

Returns the value of the requested tag attribute, or NULL if not found.

Definition at line 76 of file ezxml.c.

References ezxml_root::attr, ezxml::attr, ezxml_root_t, ezxml_t, ezxml::name, ezxml::parent, and ezxml_-root::xml.

Referenced by ezxml_toxml_r(), ConfigSim::gXMLAngle(), ConfigSim::gXMLAstroDistance(), ConfigSim::gXMLAstroMass(), ConfigSim::gXMLFrequency(), ConfigSim::gXMLTime(), Config-Sim::gXMLTimeSeries(), and ConfigSim::ReadXMLFile().

11.3.3.4 void ezxml_char_content (ezxml_root_t *root*, char * *s*, size_t *len*, char *t*)

Definition at line 233 of file ezxml.c.

References ezxml_root::cur, ezxml_root::ent, ezxml_decode(), ezxml_root_t, ezxml_set_flag(), ezxml_t, EZXML_TXTM, ezxml::flags, ezxml::name, and ezxml::txt.

Referenced by ezxml_parse_str().

11.3.3.5 ezxml_t ezxml_child (ezxml_t *xml*, const char * *name*)

Returns the first child tag (one level deeper) with the given name or NULL if not found.

Definition at line 60 of file ezxml.c.

References ezxml::child, ezxml_t, ezxml::name, and ezxml::sibling.

Referenced by ezxml_vget(), ConfigSim::gXMLTimeSeries(), and ConfigSim::ReadXMLFile().

11.3.3.6 ezxml_t ezxml_close_tag (ezxml_root_t *root*, char * *name*, char * *s*)

Definition at line 257 of file ezxml.c.

References ezxml_root::cur, ezxml_err(), ezxml_root_t, ezxml_t, ezxml::name, and ezxml::parent.

Referenced by ezxml_parse_str().

11.3.3.7 char* ezxml_decode (char * *s*, char ** *ent*, char *t*)

Definition at line 158 of file ezxml.c.

Referenced by ezxml_char_content(), ezxml_internal_dtd(), and ezxml_parse_str().

11.3.3.8 int ezxml_ent_ok (char ∗ *name*, char ∗ *s*, char ∗∗ *ent*)

Definition at line 268 of file ezxml.c.

Referenced by ezxml_internal_dtd().

11.3.3.9 ezxml_t ezxml_err (ezxml_root_t *root*, char ∗ *s*, const char ∗ *err*, ...)

Definition at line 136 of file ezxml.c.

References ezxml_root::err, EZXML_ERRL, ezxml_root_t, ezxml_t, ezxml_root::s, and ezxml_root::xml.

Referenced by ezxml_close_tag(), ezxml_internal_dtd(), and ezxml_parse_str().

11.3.3.10 const char∗ ezxml_error (ezxml_t *xml*)

Returns parser error message or empty string if none.

Definition at line 827 of file ezxml.c.

References ezxml_root_t, ezxml_t, and ezxml::parent.

11.3.3.11 void ezxml_free (ezxml_t *xml*)

Frees the memory allocated for an ezxml structure.

Definition at line 784 of file ezxml.c.

References ezxml::attr, ezxml_root::attr, ezxml::child, ezxml_root::e, ezxml_root::ent, ezxml_free_attr(), EZXML_NAMEM, ezxml_root_t, ezxml_t, EZXML_TXTM, ezxml::flags, ezxml_root::len, ezxml_-root::m, ezxml::name, ezxml::ordered, ezxml::parent, ezxml_root::pi, ezxml_root::s, ezxml::txt, and ezxml_root::u.

Referenced by ezxml_remove(), and ConfigSim::ReadXMLFile().

11.3.3.12 void ezxml_free_attr (char ∗∗ *attr*)

Definition at line 454 of file ezxml.c.

References EZXML_NAMEM, EZXML_NIL, and EZXML_TXTM.

Referenced by ezxml_free(), and ezxml_parse_str().

11.3.3.13 ezxml_t ezxml_get (ezxml_t *xml*, ...)

Traverses the ezxml sturcture to retrieve a specific subtag.

Takes a variable length list of tag names and indexes. The argument list must be terminated by either an index of -1 or an empty string tag name. Example: title = ezxml_get(library, "shelf", 0, "book", 2, "title", -1); This retrieves the title of the 3rd book on the 1st shelf of library. Returns NULL if not found.

Definition at line 111 of file ezxml.c.

References ezxml_t, and ezxml_vget().

11.3.3.14 ezxml_t ezxml_idx (ezxml_t *xml*, int *idx*)

Returns the Nth tag with the same name in the same section at the same depth or NULL if not found. An index of 0 returns the tag given.

Definition at line 69 of file ezxml.c.

References ezxml_t, and ezxml::next.

Referenced by ezxml_vget().

11.3.3.15 short ezxml_internal_dtd (ezxml_root_t *root*, char ∗ *s*, size_t *len*)

Definition at line 319 of file ezxml.c.

References ezxml_root::attr, ezxml_root::ent, ezxml_root::err, ezxml_decode(), ezxml_ent_ok(), ezxml_-err(), EZXML_NIL, ezxml_proc_inst(), ezxml_root_t, EZXML_WS, and ezxml_root::standalone.

Referenced by ezxml_parse_str().

11.3.3.16 ezxml_t ezxml_new (const char ∗ *name*)

Returns a new empty ezxml structure with the given root tag name.

Definition at line 834 of file ezxml.c.

References ezxml::attr, ezxml_root::attr, ezxml_root::cur, ezxml_root::ent, ezxml_root::err, EZXML_NIL, ezxml_root_t, ezxml_t, ezxml::name, ezxml_root::pi, ezxml::txt, and ezxml_root::xml.

Referenced by ezxml_parse_str().

11.3.3.17 void ezxml_open_tag (ezxml_root_t *root*, char ∗ *name*, char ∗∗ *attr*)

Definition at line 221 of file ezxml.c.

References ezxml::attr, ezxml_root::cur, ezxml_add_child(), ezxml_root_t, ezxml_t, ezxml::name, and ezxml::txt.

Referenced by ezxml_parse_str().

11.3.3.18 ezxml_t ezxml_parse_fd (int *fd*)

A wrapper for ezxml_parse_str() that accepts a file descriptor. First attempts to mem map the file. Failing that, reads the file into memory. Returns NULL on failure.

Definition at line 626 of file ezxml.c.

References ezxml_parse_str(), ezxml_root_t, ezxml_t, ezxml_root::len, and ezxml_root::xml.

Referenced by ezxml_parse_file().

11.3.3.19 ezxml_t ezxml_parse_file (const char ∗ *file*)

a wrapper for ezxml_parse_fd() that accepts a file name

Definition at line 656 of file ezxml.c.

References ezxml_parse_fd(), and ezxml_t.

Referenced by ConfigSim::ReadXMLFile().

11.3.3.20 ezxml_t ezxml_parse_fp (FILE * *fp*)

Wrapper for ezxml_parse_str() that accepts a file stream. Reads the entire stream into memory and then parses it. For xml files, use ezxml_parse_file() or ezxml_parse_fd().

Definition at line 605 of file ezxml.c.

References EZXML_BUFSIZE, ezxml_parse_str(), ezxml_root_t, ezxml_t, ezxml_root::len, and ezxml_-root::xml.

11.3.3.21 ezxml_t ezxml_parse_str (char * *s*, size_t *len*)

Given a string of xml data and its length, parses it and creates an ezxml structure. For efficiency, modifies the data by adding null terminators and decoding ampersand sequences. If you don't want this, copy the data and pass in the copy. Returns NULL on failure.

Definition at line 470 of file ezxml.c.

References ezxml_root::attr, ezxml_root::cur, ezxml_root::e, ezxml_root::ent, ezxml_char_content(), ezxml_close_tag(), ezxml_decode(), ezxml_err(), ezxml_free_attr(), ezxml_internal_dtd(), ezxml_-new(), EZXML_NIL, ezxml_open_tag(), ezxml_proc_inst(), ezxml_root_t, ezxml_str2utf8(), ezxml_t, EZXML_TXTM, EZXML_WS, ezxml_root::m, ezxml::name, ezxml_root::s, ezxml_root::u, and ezxml_-root::xml.

Referenced by ezxml_parse_fd(), and ezxml_parse_fp().

11.3.3.22 const char** ezxml_pi (ezxml_t *xml*, const char * *target*)

Returns a NULL terminated array of processing instructions for the given target.

Definition at line 124 of file ezxml.c.

References EZXML_NIL, ezxml_root_t, ezxml_t, ezxml::parent, ezxml_root::pi, and ezxml_root::xml.

11.3.3.23 void ezxml_proc_inst (ezxml_root_t *root*, char * *s*, size_t *len*)

Definition at line 282 of file ezxml.c.

References ezxml_root_t, EZXML_WS, ezxml::name, ezxml_root::pi, ezxml_root::standalone, and ezxml_root::xml.

Referenced by ezxml_internal_dtd(), and ezxml_parse_str().

11.3.3.24 void ezxml_remove (ezxml_t *xml*)

Removes a tag along with all its subtags.

Definition at line 954 of file ezxml.c.

References ezxml::child, ezxml_free(), ezxml_t, ezxml::name, ezxml::next, ezxml::ordered, ezxml::parent, and ezxml::sibling.

11.3.3.25 void ezxml_set_attr (ezxml_t *xml*, const char * *name*, const char * *value*)

Sets the given tag attribute or adds a new attribute if not found. A value of NULL will remove the specified attribute.

Definition at line 907 of file ezxml.c.

References ezxml::attr, EZXML_DUP, EZXML_NAMEM, EZXML_NIL, ezxml_t, EZXML_TXTM, and ezxml::flags.

11.3.3.26 ezxml_t ezxml_set_flag (ezxml_t *xml*, short *flag*)

Sets a flag for the given tag and returns the tag.

Definition at line 947 of file ezxml.c.

References ezxml_t, and ezxml::flags.

Referenced by ezxml_char_content().

11.3.3.27 ezxml_t ezxml_set_txt (ezxml_t *xml*, const char * *txt*)

Sets the character content for the given tag and returns the tag.

Definition at line 896 of file ezxml.c.

References ezxml_t, EZXML_TXTM, ezxml::flags, and ezxml::txt.

11.3.3.28 char* ezxml_str2utf8 (char ** *s*, size_t * *len*)

Definition at line 422 of file ezxml.c.

References EZXML_BUFSIZE, and max.

Referenced by ezxml_parse_str().

11.3.3.29 char* ezxml_toxml (ezxml_t *xml*)

Converts an ezxml structure back to xml. Returns a string of xml data that must be freed.

Definition at line 745 of file ezxml.c.

References ezxml_root::attr, EZXML_BUFSIZE, ezxml_root_t, ezxml_t, ezxml_toxml_r(), max, ezxml::name, ezxml::ordered, ezxml::parent, ezxml_root::pi, and ezxml_root::xml.

11.3.3.30 char* ezxml_toxml_r (ezxml_t *xml*, char ** *s*, size_t * *len*, size_t * *max*, size_t *start*, char *** *attr*)

Definition at line 693 of file ezxml.c.

References ezxml::attr, ezxml::child, ezxml_ampencode(), ezxml_attr(), EZXML_BUFSIZE, ezxml_t, max, ezxml::name, ezxml::off, ezxml::ordered, ezxml::parent, and ezxml::txt.

Referenced by ezxml_toxml().

11.3.3.31 ezxml_t ezxml_vget (ezxml_t *xml*, va_list *ap*)

Definition at line 93 of file ezxml.c.

References ezxml_child(), ezxml_idx(), and ezxml_t.

Referenced by ezxml_get().

11.3.4 Variable Documentation

11.3.4.1 char∗ EZXML_NIL[ ] = { NULL }

Definition at line 57 of file ezxml.c.

Referenced by ezxml_add_child(), ezxml_free_attr(), ezxml_internal_dtd(), ezxml_new(), ezxml_parse_-str(), ezxml_pi(), and ezxml_set_attr().

11.4 ezxml.h File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <fcntl.h>
```

Classes

- struct ezxml

Defines

- #define EZXML_BUFSIZE 1024

  size of internal memory buffers

- #define EZXML_NAMEM 0x80

  name is malloced

- #define EZXML_TXTM 0x40

  attribute name and value are strduped

- #define EZXML_DUP 0x20
- #define ezxml_next(xml) ((xml) ? xml → next : NULL)

  Returns the next tag of the same name in the same section and depth or NULL if not found.

- #define ezxml_name(xml) ((xml) ? xml → name : NULL)

  Returns the name of the given tag.

- #define ezxml_txt(xml) ((xml) ? xml → txt : "")

  Returns the given tag's character content or empty string if none.

- #define ezxml_new_d(name) ezxml_set_flag(ezxml_new(strdup(name)), EZXML_NAMEM)

  Wrapper for ezxml_new() that strdup()s names.

- #define ezxml_add_child_d(xml, name, off) ezxml_set_flag(ezxml_add_child(xml, strdup(name), off), EZXML_NAMEM)

  Xrapper for ezxml_add_child() that strdup()s name.

- #define ezxml_set_txt_d(xml, txt) ezxml_set_flag(ezxml_set_txt(xml, strdup(txt)), EZXML_-TXTM)

  Wrapper for ezxml_set_txt() that strdup()s txt.

- #define ezxml_set_attr_d(xml, name, value) ezxml_set_attr(ezxml_set_flag(xml, EZXML_DUP), strdup(name), strdup(value))

  Wrapper for ezxml_set_attr() that strdup()s name/value. Value cannot be NULL.

Typedefs

- typedef ezxml ∗ ezxml_t

    XML manipulation structure.

Functions

- ezxml_t ezxml_parse_str (char ∗s, size_t len)

    Given a string of xml data and its length, parses it and creates an ezxml structure. For efficiency, modifies the data by adding null terminators and decoding ampersand sequences. If you don't want this, copy the data and pass in the copy. Returns NULL on failure.

- ezxml_t ezxml_parse_fd (int fd)

    A wrapper for ezxml_parse_str() that accepts a file descriptor. First attempts to mem map the file. Failing that, reads the file into memory. Returns NULL on failure.

- ezxml_t ezxml_parse_file (const char ∗file)

    a wrapper for ezxml_parse_fd() that accepts a file name

- ezxml_t ezxml_parse_fp (FILE ∗fp)

    Wrapper for ezxml_parse_str() that accepts a file stream. Reads the entire stream into memory and then parses it. For xml files, use ezxml_parse_file() or ezxml_parse_fd().

- ezxml_t ezxml_child (ezxml_t xml, const char ∗name)

    Returns the first child tag (one level deeper) with the given name or NULL if not found.

- ezxml_t ezxml_idx (ezxml_t xml, int idx)

    Returns the Nth tag with the same name in the same section at the same depth or NULL if not found. An index of 0 returns the tag given.

- const char ∗ ezxml_attr (ezxml_t xml, const char ∗attr)

    Returns the value of the requested tag attribute, or NULL if not found.

- ezxml_t ezxml_get (ezxml_t xml,...)

    Traverses the ezxml sturcture to retrieve a specific subtag.

- char ∗ ezxml_toxml (ezxml_t xml)

    Converts an ezxml structure back to xml. Returns a string of xml data that must be freed.

- const char ∗∗ ezxml_pi (ezxml_t xml, const char ∗target)

    Returns a NULL terminated array of processing instructions for the given target.

- void ezxml_free (ezxml_t xml)

    Frees the memory allocated for an ezxml structure.

- const char ∗ ezxml_error (ezxml_t xml)

    Returns parser error message or empty string if none.

- ezxml_t ezxml_new (const char ∗name)

Returns a new empty ezxml structure with the given root tag name.

- ezxml_t ezxml_add_child (ezxml_t xml, const char ∗name, size_t off)

  Adds a child tag. off is the offset of the child tag relative to the start of the parent tag's character content. Returns the child tag.

- ezxml_t ezxml_set_txt (ezxml_t xml, const char ∗txt)

  Sets the character content for the given tag and returns the tag.

- void ezxml_set_attr (ezxml_t xml, const char ∗name, const char ∗value)

  Sets the given tag attribute or adds a new attribute if not found. A value of NULL will remove the specified attribute.

- ezxml_t ezxml_set_flag (ezxml_t xml, short flag)

  Sets a flag for the given tag and returns the tag.

- void ezxml_remove (ezxml_t xml)

  Removes a tag along with all its subtags.

11.4.1 Define Documentation

11.4.1.1 #define ezxml_add_child_d(xml, name, off) ezxml_set_flag(ezxml_add_child(xml, strdup(name), off), EZXML_NAMEM)

Xrapper for ezxml_add_child() that strdup()s name.

Definition at line 152 of file ezxml.h.

11.4.1.2 EZXML_BUFSIZE 1024

size of internal memory buffers

Definition at line 39 of file ezxml.h.

Referenced by ezxml_ampencode(), ezxml_parse_fp(), ezxml_str2utf8(), ezxml_toxml(), and ezxml_-toxml_r().

11.4.1.3 #define EZXML_DUP 0x20

Definition at line 48 of file ezxml.h.

Referenced by ezxml_set_attr().

11.4.1.4 #define ezxml_name(xml) ((xml) ? xml → name : NULL)

Returns the name of the given tag.

Definition at line 109 of file ezxml.h.

11.4.1.5 EZXML_NAMEM 0x80

name is malloced

Definition at line 42 of file ezxml.h.

Referenced by ezxml_free(), ezxml_free_attr(), and ezxml_set_attr().

11.4.1.6 #define ezxml_new_d(name) ezxml_set_flag(ezxml_new(strdup(name)), EZXML_NAMEM)

Wrapper for ezxml_new() that strdup()s names.

Definition at line 145 of file ezxml.h.

11.4.1.7 #define ezxml_next(xml) ((xml) ? xml → next : NULL)

Returns the next tag of the same name in the same section and depth or NULL if not found.

Definition at line 102 of file ezxml.h.

11.4.1.8 #define ezxml_set_attr_d(xml, name, value) ezxml_set_attr(ezxml_set_flag(xml, EZXML_DUP), strdup(name), strdup(value))

Wrapper for ezxml_set_attr() that strdup()s name/value. Value cannot be NULL.

Definition at line 167 of file ezxml.h.

11.4.1.9 #define ezxml_set_txt_d(xml, txt) ezxml_set_flag(ezxml_set_txt(xml, strdup(txt)), EZXML_TXTM)

Wrapper for ezxml_set_txt() that strdup()s txt.

Definition at line 159 of file ezxml.h.

11.4.1.10 #define ezxml_txt(xml) ((xml) ? xml → txt : "")

Returns the given tag's character content or empty string if none.

Definition at line 112 of file ezxml.h.

Referenced by ConfigSim::gXMLAngle(), ConfigSim::gXMLAstroDistance(), ConfigSim::gXMLAstroMass(), ConfigSim::gXMLFrequency(), ConfigSim::gXMLTime(), ConfigSim::gXMLTimeSeries(), and ConfigSim::ReadXMLFile().

11.4.1.11 EZXML_TXTM 0x40

attribute name and value are strduped

Definition at line 45 of file ezxml.h.

Referenced by ezxml_char_content(), ezxml_free(), ezxml_free_attr(), ezxml_parse_str(), ezxml_set_attr(), and ezxml_set_txt().

11.4.2 Typedef Documentation

11.4.2.1 typedef struct ezxml* ezxml_t

XML manipulation structure.

Definition at line 52 of file ezxml.h.

Referenced by ezxml_add_child(), ezxml_attr(), ezxml_char_content(), ezxml_child(), ezxml_close_-tag(), ezxml_err(), ezxml_error(), ezxml_free(), ezxml_get(), ezxml_idx(), ezxml_new(), ezxml_open_-tag(), ezxml_parse_fd(), ezxml_parse_file(), ezxml_parse_fp(), ezxml_parse_str(), ezxml_pi(), ezxml_-remove(), ezxml_set_attr(), ezxml_set_flag(), ezxml_set_txt(), ezxml_toxml(), ezxml_toxml_r(), ezxml_-vget(), ConfigSim::gXMLAngle(), ConfigSim::gXMLAstroDistance(), ConfigSim::gXMLAstroMass(), ConfigSim::gXMLFrequency(), ConfigSim::gXMLTime(), ConfigSim::gXMLTimeSeries(), and Config-Sim::ReadXMLFile().

11.4.3 Function Documentation

11.4.3.1 ezxml_t ezxml_add_child (ezxml_t *xml*, const char * *name*, size_t *off*)

Adds a child tag. off is the offset of the child tag relative to the start of the parent tag's character content. Returns the child tag.

Definition at line 850 of file ezxml.c.

References ezxml::attr, ezxml::child, EZXML_NIL, ezxml_t, ezxml::name, ezxml::next, ezxml::off, ezxml::ordered, ezxml::parent, ezxml::sibling, and ezxml::txt.

Referenced by ezxml_open_tag().

11.4.3.2 const char* ezxml_attr (ezxml_t *xml*, const char * *attr*)

Returns the value of the requested tag attribute, or NULL if not found.

Definition at line 76 of file ezxml.c.

References ezxml::attr, ezxml_root::attr, ezxml_root_t, ezxml_t, ezxml::name, ezxml::parent, and ezxml_-root::xml.

Referenced by ezxml_toxml_r(), ConfigSim::gXMLAngle(), ConfigSim::gXMLAstroDistance(), ConfigSim::gXMLAstroMass(), ConfigSim::gXMLFrequency(), ConfigSim::gXMLTime(), Config-Sim::gXMLTimeSeries(), and ConfigSim::ReadXMLFile().

11.4.3.3 ezxml_t ezxml_child (ezxml_t *xml*, const char * *name*)

Returns the first child tag (one level deeper) with the given name or NULL if not found.

Definition at line 60 of file ezxml.c.

References ezxml::child, ezxml_t, ezxml::name, and ezxml::sibling.

Referenced by ezxml_vget(), ConfigSim::gXMLTimeSeries(), and ConfigSim::ReadXMLFile().

11.4.3.4 const char* ezxml_error (ezxml_t *xml*)

Returns parser error message or empty string if none.

Definition at line 827 of file ezxml.c.

References ezxml_root_t, ezxml_t, and ezxml::parent.

11.4.3.5 void ezxml_free (ezxml_t *xml*)

Frees the memory allocated for an ezxml structure.

Definition at line 784 of file ezxml.c.

References ezxml_root::attr, ezxml::attr, ezxml::child, ezxml_root::e, ezxml_root::ent, ezxml_free_attr(), EZXML_NAMEM, ezxml_root_t, ezxml_t, EZXML_TXTM, ezxml::flags, ezxml_root::len, ezxml_-root::m, ezxml::name, ezxml::ordered, ezxml::parent, ezxml_root::pi, ezxml_root::s, ezxml::txt, and ezxml_root::u.

Referenced by ezxml_remove(), and ConfigSim::ReadXMLFile().

11.4.3.6 ezxml_t ezxml_get (ezxml_t *xml*, ...)

Traverses the ezxml sturcture to retrieve a specific subtag.

Takes a variable length list of tag names and indexes. The argument list must be terminated by either an index of -1 or an empty string tag name. Example: title = ezxml_get(library, "shelf", 0, "book", 2, "title", -1); This retrieves the title of the 3rd book on the 1st shelf of library. Returns NULL if not found.

Definition at line 111 of file ezxml.c.

References ezxml_t, and ezxml_vget().

11.4.3.7 ezxml_t ezxml_idx (ezxml_t *xml*, int *idx*)

Returns the Nth tag with the same name in the same section at the same depth or NULL if not found. An index of 0 returns the tag given.

Definition at line 69 of file ezxml.c.

References ezxml_t, and ezxml::next.

Referenced by ezxml_vget().

11.4.3.8 ezxml_t ezxml_new (const char * *name*)

Returns a new empty ezxml structure with the given root tag name.

Definition at line 834 of file ezxml.c.

References ezxml_root::attr, ezxml::attr, ezxml_root::cur, ezxml_root::ent, ezxml_root::err, EZXML_NIL, ezxml_root_t, ezxml_t, ezxml::name, ezxml_root::pi, ezxml::txt, and ezxml_root::xml.

Referenced by ezxml_parse_str().

11.4.3.9 ezxml_t ezxml_parse_fd (int *fd*)

A wrapper for ezxml_parse_str() that accepts a file descriptor. First attempts to mem map the file. Failing that, reads the file into memory. Returns NULL on failure.

Definition at line 626 of file ezxml.c.

References ezxml_parse_str(), ezxml_root_t, ezxml_t, ezxml_root::len, and ezxml_root::xml.

Referenced by ezxml_parse_file().

11.4.3.10 ezxml_t ezxml_parse_file (const char * *file*)

a wrapper for ezxml_parse_fd() that accepts a file name

Definition at line 656 of file ezxml.c.

References ezxml_parse_fd(), and ezxml_t.

Referenced by ConfigSim::ReadXMLFile().

11.4.3.11 ezxml_t ezxml_parse_fp (FILE * *fp*)

Wrapper for ezxml_parse_str() that accepts a file stream. Reads the entire stream into memory and then parses it. For xml files, use ezxml_parse_file() or ezxml_parse_fd().

Definition at line 605 of file ezxml.c.

References EZXML_BUFSIZE, ezxml_parse_str(), ezxml_root_t, ezxml_t, ezxml_root::len, and ezxml_root::xml.

11.4.3.12 ezxml_t ezxml_parse_str (char * *s*, size_t *len*)

Given a string of xml data and its length, parses it and creates an ezxml structure. For efficiency, modifies the data by adding null terminators and decoding ampersand sequences. If you don't want this, copy the data and pass in the copy. Returns NULL on failure.

Definition at line 470 of file ezxml.c.

References ezxml_root::attr, ezxml_root::cur, ezxml_root::e, ezxml_root::ent, ezxml_char_content(), ezxml_close_tag(), ezxml_decode(), ezxml_err(), ezxml_free_attr(), ezxml_internal_dtd(), ezxml_new(), EZXML_NIL, ezxml_open_tag(), ezxml_proc_inst(), ezxml_root_t, ezxml_str2utf8(), ezxml_t, EZXML_TXTM, EZXML_WS, ezxml_root::m, ezxml::name, ezxml_root::s, ezxml_root::u, and ezxml_root::xml.

Referenced by ezxml_parse_fd(), and ezxml_parse_fp().

11.4.3.13 const char** ezxml_pi (ezxml_t *xml*, const char * *target*)

Returns a NULL terminated array of processing instructions for the given target.

Definition at line 124 of file ezxml.c.

References EZXML_NIL, ezxml_root_t, ezxml_t, ezxml::parent, ezxml_root::pi, and ezxml_root::xml.

11.4.3.14 void ezxml_remove (ezxml_t *xml*)

Removes a tag along with all its subtags.

Definition at line 954 of file ezxml.c.

References ezxml::child, ezxml_free(), ezxml_t, ezxml::name, ezxml::next, ezxml::ordered, ezxml::parent, and ezxml::sibling.

11.4.3.15 void ezxml_set_attr (ezxml_t *xml*, const char ∗ *name*, const char ∗ *value*)

Sets the given tag attribute or adds a new attribute if not found. A value of NULL will remove the specified attribute.

Definition at line 907 of file ezxml.c.

References ezxml::attr, EZXML_DUP, EZXML_NAMEM, EZXML_NIL, ezxml_t, EZXML_TXTM, and ezxml::flags.

11.4.3.16 ezxml_t ezxml_set_flag (ezxml_t *xml*, short *flag*)

Sets a flag for the given tag and returns the tag.

Definition at line 947 of file ezxml.c.

References ezxml_t, and ezxml::flags.

Referenced by ezxml_char_content().

11.4.3.17 ezxml_t ezxml_set_txt (ezxml_t *xml*, const char ∗ *txt*)

Sets the character content for the given tag and returns the tag.

Definition at line 896 of file ezxml.c.

References ezxml_t, EZXML_TXTM, ezxml::flags, and ezxml::txt.

11.4.3.18 char∗ ezxml_toxml (ezxml_t *xml*)

Converts an ezxml structure back to xml. Returns a string of xml data that must be freed.

Definition at line 745 of file ezxml.c.

References ezxml_root::attr, EZXML_BUFSIZE, ezxml_root_t, ezxml_t, ezxml_toxml_r(), max, ezxml::name, ezxml::ordered, ezxml::parent, ezxml_root::pi, and ezxml_root::xml.

11.5 linpack.c File Reference

```
#include <math.h>
```

Functions

- float sdot (long n, float ∗sx, long incx, float ∗sy, long incy)
- void spofa (float ∗a, long lda, long n, long ∗info)

11.5.1 Function Documentation

11.5.1.1 float sdot (long *n*, float ∗ *sx*, long *incx*, float ∗ *sy*, long *incy*)

Definition at line 2 of file linpack.c.

References sdot().

Referenced by sdot(), and spofa().

11.5.1.2 void spofa (float ∗ *a*, long *lda*, long *n*, long ∗ *info*)

Definition at line 32 of file linpack.c.

References sdot().

Referenced by setgmn().

11.6 LISACODE-Background.cpp File Reference

```
#include "LISACODE-Background.h"
```

11.7 LISACODE-Background.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-Geometry.h"
```

Classes

- class Background

  Background signal received by phasemeters is described in this class.

11.8 LISACODE-BackgroundGalactic.cpp File Reference

```
#include "LISACODE-BackgroundGalactic.h"
```

11.9 LISACODE-BackgroundGalactic.h File Reference

```
#include <stdexcept>
#include <iostream.h>
#include <fstream>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-Background.h"
```

Classes

- class BackgroundGalactic

  Background Galactic signal received by phasemeters is described in this class.

11.10 LISACODE-ConfigSim.cpp File Reference

```
#include "LISACODE-ConfigSim.h"
```

11.11 LISACODE-ConfigSim.h File Reference

```
#include <iostream.h>
#include <stdexcept>
#include <math.h>
#include <fstream>
#include <sstream>
#include <string>
#include <iomanip.h>
#include "ezxml.h"
#include "LISACODE-LISAConstants.h"
#include "LISACODE-Serie.h"
#include "LISACODE-GW.h"
#include "LISACODE-GWMono.h"
#include "LISACODE-GWBinary.h"
#include "LISACODE-GWNewton2.h"
#include "LISACODE-GWFile.h"
#include "LISACODE-GWPeriGate.h"
#include "LISACODE-Background.h"
#include "LISACODE-BackgroundGalactic.h"
#include "LISACODE-Noise.h"
#include "LISACODE-NoiseWhite.h"
#include "LISACODE-NoiseFilter.h"
#include "LISACODE-NoiseFile.h"
#include "LISACODE-USOClock.h"
#include "LISACODE-Filter.h"
```

Classes

- class ConfigSim

  Class to configure LISA simulation, that is, LISACode execution.

- struct NoiseSpec

  Noise specification structure.

11.12 LISACODE-ConfigSim_s.cpp File Reference

```
#include "LISACODE-ConfigSim.h"
```

11.13 LISACODE-Couple.cpp File Reference

```
#include "LISACODE-Couple.h"
```

Functions

- Couple operator+ (Couple z1, Couple z2)

  2 couples addiction.

- Couple operator- (Couple z1, Couple z2)

  2 couples subtraction.

- Couple operator $*$ (Couple z1, Couple z2)

  ?? where operator$*$ (Couple,Couple) is defined?

- Couple operator $*$ (double a, Couple z1)

  Product of a couple by a scalar.

- Couple operator $*$ (Couple z1, double a)

  Product of a couple by a scalar.

- Couple operator/ (Couple z1, double a)

  Division of a couple by a scalar .

11.13.1 Function Documentation

11.13.1.1 Couple operator $*$ (Couple *z1*, double *a*)

Product of a couple by a scalar.

Definition at line 87 of file LISACODE-Couple.cpp.

References Couple::x, and Couple::y.

11.13.1.2 Couple operator $*$ (double *a*, Couple *z1*)

Product of a couple by a scalar.

Definition at line 78 of file LISACODE-Couple.cpp.

References Couple::x, and Couple::y.

11.13.1.3 Couple operator $*$ (Couple *z1*, Couple *z2*)

?? where operator$*$ (Couple,Couple) is defined?

Definition at line 65 of file LISACODE-Couple.cpp.

References Couple::x, and Couple::y.

11.13.1.4 Couple operator+ (Couple *z1*, Couple *z2*)

2 couples addiction.

Definition at line 47 of file LISACODE-Couple.cpp.

References Couple::x, and Couple::y.

11.13.1.5 Couple operator- (Couple *z1*, Couple *z2*)

2 couples subtraction.

Definition at line 56 of file LISACODE-Couple.cpp.

References Couple::x, and Couple::y.

11.13.1.6 Couple operator/ (Couple *z1*, double *a*)

Division of a couple by a scalar .

Definition at line 95 of file LISACODE-Couple.cpp.

References Couple::x, and Couple::y.

11.14 LISACODE-Couple.h File Reference

```
#include <stdexcept>
#include <iostream.h>
#include <math.h>
```

Classes

- class Couple

  Couple management class.

11.15 LISACODE-DnonGW.cpp File Reference

```
#include <stdexcept>
#include <iostream>
#include <fstream.h>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-LISAConstants.h"
#include "LISACODE-MathUtils.h"
#include "LISACODE-GWMono.h"
#include "LISACODE-GWFile.h"
#include "LISACODE-GWBinary.h"
#include "LISACODE-GWPeriGate.h"
#include "LISACODE-Geometry.h"
#include "LISACODE-TrFctGW.h"
#include "LISACODE-ConfigSim.h"
```

Functions

- int main (int argc, char *const argv[ ])

  LISA simulator.

  - *Initialization.*
    Random generator is initialized.
    Config is a ConfigSim instance created with data read from "ConfigRefBase" file.
    RecordPDPM is a Memory vector where spacecraft signals wil be recorded.
    LISACode is a LISA instance created with Config and RecordPDPM.
    Eta signals are created.
    TDI generators are created using approximative delay computation specified in Config.
  - *Data processing first step : time $t = 0, \ldots, tMemTDI + tTDIShift$ with tStepMes timsetep.*
    Signals are stored.
    LISA::MakeOneStepOfTime method is called.
    Delays are recorded.
    Positions are recorded.
  - *Data processing second step : when there are enough data, TDI is computed and results are stored in file, while time $t \leq tmax$ with tStepMes timsetep.*
    TDI is computed using TDI_InterData::ComputeEta method.
    Delays are recorded.
    Positions are recorded.
    .

11.16 LISACODE-EllipticFilter.cpp File Reference

```
#include "LISACODE-EllipticFilter.h"
```

Functions

- void elli (double eps, double A, double fa, double fb, double fe, int NCellMax, int *NCells, complex< double > poles[ ], complex< double > zeros[ ], double CoefA[ ], double CoefB[ ], double CoefC[ ], double CoefD[ ])

  Poles, zeros and elliptic cells coefficients computation.

- double ak (double y)

  Integral filter parameter computation.

- double cak (double x)

  Developped filter parameter computation.

- double sn (double y, double A, double ak1, double ak3)

  Recursive or direct coefficients computation.

- double FilterQuadCell (double xn, QuadCell *Cell)

  Elliptic cell filtering step, depending on xn and Cell (type QuadCell) inputs.

- double FilterQuadCellChain (double xn, int NCells, QuadCell Cell[ ])

  Elliptic cells chain filtering step, depending on xn, number of cells NCells and Cell (type QuadCell) inputs.

- complex< double > TransfZQuadCell (complex< double > Z, QuadCell Cell)

  Elliptic cell Z transform.

- complex< double > TransfZQuadCellChain (complex< double > Z, int NCells, QuadCell Cell[ ])

  Elliptic cells chain Z transform, depending on Z, number of cells NCells and Cell (type QuadCell) inputs.

- double AbsRespFunctQuadCell (double f, QuadCell Cell)

  Frequency response modulus, depending on f frequency and Cell (type QuadCell) inputs.

- double AbsRespFunctQuadCellChain (double f, int NCells, QuadCell Cell[ ])

  Elliptic cells chain frequency response modulus, depending on f frequency, number of cell NCells and Cell (type QuadCell) inputs.

- double HmQuadCell (QuadCell Cell)

  Returns max |1/D(w)|, where D(w) is the denominator of an ellitpic cell (type QuadCell).

- double KmQuadCell (QuadCell Cell)

  Returns max|Ell(w)|, where Ell(w) is the frequency response of an ellitpic cell (type QuadCell).

- void PoleMatching (int NCells, QuadCell Cell[ ])

  Matches nearest poles for a chain of elliptic cells.

- void OrderCellMaxNorm (int NCells, QuadCell Cell[ ])

Orders cells according to the the max of inf norm.

- double CalcScalingFact (int NCells, QuadCell Cell[ ])

  Scale factors computation for an elliptic cells chain : a0 attributes are updated and global factor is returned.

- double CalcEllipticFilter (double fe, double at, double bp, double fb, double fa, int NCellMax, QuadCell **FilterCellsOut, int *NCellsOut)

  Computes filter coefficients from user specifications and returns the global scale factor.

- void CalcEllipticFilter4LISACode (double fe, double at, double bp, double fb, double fa, int NCellMax, double CellsCoef[ ][5], int *NCellsOut)

  Computes filter coefficients from LISA Code user specifications.
  The global scale factor is included in the first cell.

11.17 LISACODE-EllipticFilter.h File Reference

```
#include <iostream.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <complex>
```

Classes

- struct QuadCell

  Elliptic cell structure.

Defines

- #define alog(A) log(A)
- #define alog10(A) log10(A)

Functions

- complex< double > I (0, 1)

  Pure imaginary=(0,1).

- void elli (double eps, double A, double wr, double wc, double fe, int NCellMax, int ∗NCells, complex< double > poles[ ], complex< double > zeros[ ], double CoefA[ ], double CoefB[ ], double CoefC[ ], double CoefD[ ])

  Poles, zeros and elliptic cells coefficients computation.

- double ak (double y)

  Integral filter parameter computation.

- double cak (double y)

  Developped filter parameter computation.

- double sn (double y, double A, double ak1, double ak3)

  Recursive or direct coefficients computation.

- double FilterQuadCell (double xn, QuadCell ∗Cell)

  Elliptic cell filtering step, depending on xn and Cell (type QuadCell) inputs.

- double FilterQuadCellChain (double xn, int NCells, QuadCell Cell[ ])

  Elliptic cells chain filtering step, depending on xn, number of cells NCells and Cell (type QuadCell) inputs.

- complex< double > TransfZQuadCell (complex< double > Z, QuadCell Cell)

  Elliptic cell Z transform.

- complex< double > TransfZQuadCellChain (complex< double > Z, int NCells, QuadCell Cell[ ])

  Elliptic cells chain Z transform, depending on Z, number of cells NCells and Cell (type QuadCell) inputs.

- double AbsRespFunctQuadCell (double f, QuadCell Cell)

  Frequency response modulus, depending on f frequency and Cell (type QuadCell) inputs.

- double AbsRespFunctQuadCellChain (double f, int NCells, QuadCell Cell[ ])

  Elliptic cells chain frequency response modulus, depending on f frequency, number of cell NCells and Cell (type QuadCell) inputs.

- double HmQuadCell (QuadCell Cell)

  Returns max |1/D(w)|, where D(w) is the denominator of an ellitpic cell (type QuadCell).

- double KmQuadCell (QuadCell Cell)

  Returns max|Ell(w)|, where Ell(w) is the frequency response of an ellitpic cell (type QuadCell).

- void PoleMatching (int NCells, QuadCell Cell[ ])

  Matches nearest poles for a chain of elliptic cells.

- void OrderCellMaxNorm (int NCells, QuadCell Cell[ ])

  Orders cells according to the the max of inf norm.

- double CalcScalingFact (int NCells, QuadCell Cell[ ])

  Scale factors computation for an elliptic cells chain : a0 attributes are updated and global factor is returned.

- double CalcEllipticFilter (double fe, double at, double bp, double fb, double fa, int NCellMax, QuadCell ∗∗FilterCellsOut, int ∗NCellsOut)

  Computes filter coefficients from user specifications and returns the global scale factor.

- void CalcEllipticFilter4LISACode (double fe, double at, double bp, double fb, double fa, int NCellMax, double CellsCoef[ ][5], int ∗NCellsOut)

  Computes filter coefficients from LISA Code user specifications.
  The global scale factor is included in the first cell.

11.18 LISACODE-Filter.cpp File Reference

```
#include "LISACODE-Filter.h"
```

11.19 LISACODE-Filter.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-MathUtils.h"
#include "LISACODE-LISAConstants.h"
#include "LISACODE-EllipticFilter.h"
```

Classes

- class Filter

  filter management class.

11.20 LISACODE-Geometry.cpp File Reference

```
#include "LISACODE-Geometry.h"
```

11.21 LISACODE-Geometry.h File Reference

```
#include <stdexcept>
#include <iostream.h>
#include <math.h>
#include <vector.h>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-LISAConstants.h"
#include "LISACODE-Couple.h"
#include "LISACODE-Vect.h"
```

Classes

- class Geometry

  Orbit geometry class.

11.22 LISACODE-Geometry_new.cpp File Reference

```
#include "LISACODE-Geometry.h"
```

11.23 LISACODE-GW.cpp File Reference

```
#include "LISACODE-GW.h"
```

11.24 LISACODE-GW.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "LISACODE-PhysicConstants.h"
```

Classes

- class GW

  Gravitational Waves parameters are described in this class.

11.25 LISACODE-GWBinary.cpp File Reference

```
#include "LISACODE-GWBinary.h"
```

11.26 LISACODE-GWBinary.h File Reference

```
#include <stdexcept>
#include <iostream.h>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-Couple.h"
#include "LISACODE-GW.h"
```

Classes

- class GWBinary

  Gravitational Waves parameters for a monochromatic binary system are defined in this class.

11.27 LISACODE-GWFile.cpp File Reference

```
#include "LISACODE-GWFile.h"
```

11.28 LISACODE-GWFile.h File Reference

```
#include <stdexcept>
#include <iostream.h>
#include <fstream>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-Couple.h"
#include "LISACODE-GW.h"
```

Classes

- class GWFile

  Gravitational Waves file management.

11.29 LISACODE-GWMono.cpp File Reference

```
#include "LISACODE-GWMono.h"
```

11.30 LISACODE-GWMono.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-GW.h"
```

Classes

- class GWMono
  Gravitational Waves instantaneous parameters h_plus and h_cross are described in this class.

11.31 LISACODE-GWNewton2.cpp File Reference

```
#include "LISACODE-GWNewton2.h"
```

11.32 LISACODE-GWNewton2.cpp File Reference

```
#include "LISACODE-GWNewton2.h"
```

11.33 LISACODE-GWNewton2.h File Reference

```
#include <stdexcept>
#include <iostream.h>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-Couple.h"
#include "LISACODE-GW.h"
```

Classes

- class GWNewton2

    Gravitational Waves binary system parameters computation.

11.34 LISACODE-GWPeriGate.cpp File Reference

```
#include "LISACODE-GWPeriGate.h"
```

11.35 LISACODE-GWPeriGate.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-GW.h"
```

Classes

- class GWPeriGate

  Gravitational Waves periodic gate signal.

11.36 LISACODE-LISA.cpp File Reference

```
#include "LISACODE-LISA.h"
```

11.37 LISACODE-LISA.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-MathUtils.h"
#include "LISACODE-LISAConstants.h"
#include "LISACODE-NoiseWhite.h"
#include "LISACODE-NoiseFilter.h"
#include "LISACODE-NoiseFile.h"
#include "LISACODE-TrFctGW.h"
#include "LISACODE-Geometry.h"
#include "LISACODE-Background.h"
#include "LISACODE-USOClock.h"
#include "LISACODE-PhoDetPhaMet.h"
#include "LISACODE-Memory.h"
#include "LISACODE-ConfigSim.h"
```

Classes

- class LISA

  This class contains and manages all the elements necessary to LISA satellites simulation.

11.38 LISACODE-LISACode.cpp File Reference

```
#include <stdexcept>
#include <iostream>
#include <fstream.h>
#include <stdlib.h>
#include "randlib.h"
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-MathUtils.h"
#include "LISACODE-LISAConstants.h"
#include "LISACODE-GW.h"
#include "LISACODE-GWMono.h"
#include "LISACODE-GWPeriGate.h"
#include "LISACODE-Memory.h"
#include "LISACODE-MemoryWriteDisk.h"
#include "LISACODE-MemoryReadDisk.h"
#include "LISACODE-TDI_InterData.h"
#include "LISACODE-TDITools.h"
#include "LISACODE-TDI.h"
#include "LISACODE-LISA.h"
#include "LISACODE-ConfigSim.h"
```

Functions

- int main (int argc, char ∗const argv[ ])

  LISA simulator.

  – *Initialization.*
    Random generator is initialized.
    Config is a ConfigSim instance created with data read from "ConfigRefBase" file.
    RecordPDPM is a Memory vector where spacecraft signals wil be recorded.
    LISACode is a LISA instance created with Config and RecordPDPM.
    Eta signals are created.
    TDI generators are created using approximative delay computation specified in Config.
  – *Data processing first step : time $t = 0, \ldots, tMemTDI + tTDIShift$ with tStepMes timsetep.*
    Signals are stored.
    LISA::MakeOneStepOfTime method is called.
    Delays are recorded.
    Positions are recorded.
  – *Data processing second step : when there are enough data, TDI is computed and results are stored in file, while time $t \leq tmax$ with tStepMes timsetep.*
    TDI is computed using TDI_InterData::ComputeEta method.
    Delays are recorded.
    Positions are recorded.
    .

11.38.1 Function Documentation

11.38.1.1 int main (int *argc*, char ∗const *argv*[ ])

LISA simulator.

- Initialization.

  Random generator is initialized.

  Config is a ConfigSim instance created with data read from "ConfigRefBase" file.

  RecordPDPM is a Memory vector where spacecraft signals wil be recorded.

  LISACode is a LISA instance created with Config and RecordPDPM.

  Eta signals are created.

  TDI generators are created using approximative delay computation specified in Config.

- Data processing first step : time $t = 0, \ldots, tMemTDI + tTDIShift$ with tStepMes timsetep.

  Signals are stored.

  LISA::MakeOneStepOfTime method is called.

  Delays are recorded.

  Positions are recorded.

- Data processing second step : when there are enough data, TDI is computed and results are stored in file, while time $t \leq tmax$ with tStepMes timsetep.

  TDI is computed using TDI_InterData::ComputeEta method.

  Delays are recorded.

  Positions are recorded.

  .

Definition at line 146 of file LISACODE-LISACode.cpp.

References Memory::AddSerieData(), TDI_InterData::ComputeEta(), LISA::gDelayT(), genunf(), ConfigSim::getFileNameDelays(), ConfigSim::getFileNamePositions(), ConfigSim::getFileNameSig(), ConfigSim::getFileNameTDI(), ConfigSim::getGenTDIPacks(), ConfigSim::getNameGenTDI(), ConfigSim::getNbMaxDelays(), ConfigSim::getNoNoise(), TDITools::getRapidOption(), getsd(), ConfigSim::gettDeltaTDIDelay(), ConfigSim::getTDIDelayApprox(), ConfigSim::getTDIInterp(), ConfigSim::getTDIInterpUtilVal(), ConfigSim::gettDisplay(), ConfigSim::gettMax(), ConfigSim::gettStepMes(), ConfigSim::gettStepPhy(), LISA::gPosSC(), LISACodeVersion, LISA::MakeOneStepOfTime(), Memory::MakeTitles(), MAX, ConfigSim::NbGenTDI(), Vect::p, Memory::ReceiveData(), Memory::RecordAccData(), TDITools::RefreshDelay(), setall(), ConfigSim::tMaxDelay(), ConfigSim::tMemNecInterpTDI(), and ConfigSim::tMinDelay().

11.39 LISACODE-LISAConstants.h File Reference

11.39.1 Detailed Description

Physical constants of LISA instrument.

Definition in file LISACODE-LISAConstants.h.

```
#include <math.h>
#include "LISACODE-PhysicConstants.h"
```

Variables

- const char LISACodeVersion [ ] = "LISACode v 1.3"

  Simulator Version.

- const double L0_m_default = 5.0e9

  Arms length (distance between every pair of satellites) in meters.

- const double Rgc = au_m

  Distance between the LISA barycenter and the Sun.

- const double omega = 2.*M_PI/Yr_SI

  Angular velocity.

- const double tRangeStorePos_default = 10.0

  Default time step for LISA geometry positions computation.

- const double tRangeStoreDelay_default = 10.0

  Default time step for LISA delays computation.

- const double la0Laser_m = 1.064e-6

  Nominal lasers wave length in meters.

- const double nu0Laser_Hz = c_SI/la0Laser_m

  Nominal lasers frequency in Hz.

- const double LaserPower_W_default = 1

  Lasers power in Watts.

11.39.2 Variable Documentation

11.39.2.1 const double L0_m_default = 5.0e9

Arms length (distance between every pair of satellites) in meters.

Definition at line 35 of file LISACODE-LISAConstants.h.

Referenced by ConfigSim::DefaultConfig(), Geometry::Geometry(), and ConfigSim::NoisesCreation().

11.39.2.2 const double la0Laser_m = 1.064e-6

Nominal lasers wave length in meters.

Definition at line 49 of file LISACODE-LISAConstants.h.

11.39.2.3 const double LaserPower_W_default = 1

Lasers power in Watts.

Definition at line 53 of file LISACODE-LISAConstants.h.

Referenced by ConfigSim::DefaultConfig(), and ConfigSim::NoisesCreation().

11.39.2.4 const char LISACodeVersion[ ] = "LISACode v 1.3"

Simulator Version.

Definition at line 31 of file LISACODE-LISAConstants.h.

Referenced by main(), and MemoryWriteDisk::MakeTitles().

11.39.2.5 const double nu0Laser_Hz = c_SI/la0Laser_m

Nominal lasers frequency in Hz.

Definition at line 51 of file LISACODE-LISAConstants.h.

11.39.2.6 const double omega = 2.*M_PI/Yr_SI

Angular velocity.

Definition at line 39 of file LISACODE-LISAConstants.h.

Referenced by elli(), Geometry::exanom(), GWNewton2::hc(), GWNewton2::hp(), ignpoi(), and Geometry::velocity().

11.39.2.7 const double Rgc = au_m

Distance between the LISA barycenter and the Sun.

Definition at line 37 of file LISACODE-LISAConstants.h.

Referenced by Geometry::init(), Geometry::position(), and Geometry::velocity().

11.39.2.8 const double tRangeStoreDelay_default = 10.0

Default time step for LISA delays computation.

Definition at line 45 of file LISACODE-LISAConstants.h.

Referenced by Geometry::init().

11.39.2.9 const double tRangeStorePos_default = 10.0

Default time step for LISA geometry positions computation.

Definition at line 42 of file LISACODE-LISAConstants.h.

Referenced by Geometry::init().

11.40 LISACODE-Mat.cpp File Reference

```
#include "LISACODE-Mat.h"
```

Functions

- Mat operator+ (Mat A, Mat B)
  Matrices addition. It returns matrix A+B.
- Mat operator- (Mat A, Mat B)
  Matrices subtraction. It returns matrix A-B.
- Mat operator * (double f, Mat A)
  Product between a scalar and a matrix. It returns matrix: f.A.
- Vect operator * (Mat A, Vect u)
  Product between a matrix and vector. It returns vector A.v.

11.40.1 Function Documentation

11.40.1.1 Vect operator * (Mat *A*, Vect *u*)

Product between a matrix and vector. It returns vector A.v.

Definition at line 111 of file LISACODE-Mat.cpp.

References Mat::p, and Vect::p.

11.40.1.2 Mat operator * (double *f*, Mat *A*)

Product between a scalar and a matrix. It returns matrix: $f.A$.

Definition at line 96 of file LISACODE-Mat.cpp.

References Mat::p.

11.40.1.3 Mat operator+ (Mat *A*, Mat *B*)

Matrices addition. It returns matrix A+B.

Definition at line 70 of file LISACODE-Mat.cpp.

References Mat::p.

11.40.1.4 Mat operator- (Mat *A*, Mat *B*)

Matrices subtraction. It returns matrix A-B.

Definition at line 82 of file LISACODE-Mat.cpp.

References Mat::p.

11.41 LISACODE-Mat.h File Reference

```
#include <stdexcept>
#include <iostream.h>
#include <math.h>
#include "LISACODE-Vect.h"
```

Classes

- class Mat

  (3x3) matrix management class.

11.42 LISACODE-MathUtils.h File Reference

```
#include <math.h>
#include <vector.h>
```

Classes

- class MathUtils

  Angle conversion class.

Defines

- #define SWAP(a, b) tempr=(a);(a)=(b);(b)=tempr
- #define MIN(a, b) (((a)<(b))?(a):(b))
- #define MAX(a, b) (((a)>(b))?(a):(b))

11.43 LISACODE-Memory.cpp File Reference

```
#include "LISACODE-Memory.h"
```

11.44 LISACODE-Memory.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <iomanip>
#include <vector.h>
#include <fstream.h>
#include <string>
#include <sstream>
#include "LISACODE-MathUtils.h"
#include "LISACODE-Serie.h"
#include "LISACODE-LISAConstants.h"
```

Classes

- class Memory

    Memory management class.

11.45 LISACODE-MemoryReadDisk.cpp File Reference

```
#include "LISACODE-MemoryReadDisk.h"
```

11.46 LISACODE-MemoryReadDisk.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <iomanip>
#include <vector.h>
#include <fstream.h>
#include <string>
#include <sstream>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-Serie.h"
#include "LISACODE-Memory.h"
```

Classes

- class MemoryReadDisk

  Class to manage disk reading.

11.47 LISACODE-MemoryWriteDisk.cpp File Reference

```
#include "LISACODE-MemoryWriteDisk.h"
```

11.48 LISACODE-MemoryWriteDisk.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <iomanip>
#include <vector.h>
#include <fstream.h>
#include <string>
#include <sstream>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-Serie.h"
#include "LISACODE-Memory.h"
```

Classes

- class MemoryWriteDisk

  Class to manage disk writting.

11.49 LISACODE-Noise.cpp File Reference

```
#include "LISACODE-Noise.h"
```

11.50 LISACODE-Noise.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-MathUtils.h"
#include "LISACODE-LISAConstants.h"
```

Namespaces

- namespace std

Classes

- class Noise

  Noise base class.

11.51 LISACODE-NoiseFile.cpp File Reference

```
#include "LISACODE-NoiseFile.h"
```

11.52 LISACODE-NoiseFile.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include <fstream.h>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-MathUtils.h"
#include "LISACODE-LISAConstants.h"
#include "LISACODE-Noise.h"
```

Classes

- class NoiseFile

  Noise derived class to treat files with noise data.

11.53 LISACODE-NoiseFilter.cpp File Reference

```
#include "LISACODE-NoiseFilter.h"
```

11.54 LISACODE-NoiseFilter.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "randlib.h"
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-MathUtils.h"
#include "LISACODE-LISAConstants.h"
#include "LISACODE-Noise.h"
#include "LISACODE-Filter.h"
```

Classes

- class NoiseFilter

  Noise derived class to treat noise filters.

11.55 LISACODE-NoiseWhite.cpp File Reference

```
#include "LISACODE-NoiseWhite.h"
```

11.56 LISACODE-NoiseWhite.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "randlib.h"
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-MathUtils.h"
#include "LISACODE-LISAConstants.h"
#include "LISACODE-Noise.h"
```

Classes

- class NoiseWhite

  Noise derived class to treat white noise.

11.57 LISACODE-PhoDetPhaMet.cpp File Reference

```
#include "LISACODE-PhoDetPhaMet.h"
```

11.58 LISACODE-PhoDetPhaMet.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-MathUtils.h"
#include "LISACODE-Geometry.h"
#include "LISACODE-Background.h"
#include "LISACODE-Noise.h"
#include "LISACODE-Filter.h"
#include "LISACODE-USOClock.h"
#include "LISACODE-TrFctGW.h"
#include "LISACODE-Memory.h"
```

Classes

- class PhoDetPhaMet

  Phasemeter photodiode class.

Enumerations

- enum NOISEORIG { LA, OB, IM, OP }
- enum PDPMINTERF { S, TAU }

  Photodetector-phasemeter interferences type.

11.59 LISACODE-PhysicConstants.h File Reference

11.59.1 Detailed Description

Physical constants, reference values and unit conversions.

Definition in file LISACODE-PhysicConstants.h.

`#include <math.h>`

`#include <float.h>`

Variables

- const double PRECISION = 100.0*DBL_EPSILON

  Acceptable precision error on doubles.

- const double c_SI = 299792458

  Light's speed in $m \cdot s^{-1}$.

- const double G_SI = 6.67259e-11

  Gravitational constant in $m^3 \cdot Kg^{-1} \cdot s^{-2}$.

- const double k_SI = 1.381e-23

  Boltzmann's constant in $joule \cdot kelvin^{-1}$.

- const double h_SI = 6.62620e-34

  Planck's constant in $joule \cdot second$.

- const double hb_SI = h_SI/(2*M_PI)

  Planck's constant divided by $2 \cdot \pi$ *in* $joule \cdot second$.

- const double S_SI = 5.671e-8

  Stefan's constant in $Watt \cdot meter^{-2} \cdot kelvin^{-1}$.

- const double mu0_SI = 4.e-7*M_PI

  Permeability of free space or magnetic constant μ_0 *in* $Henry \cdot meter^{-1}$.

- const double eps0_SI = 1.0/(4.e-7*M_PI*c_SI*c_SI)

  Permittivity of free space or permittivity ϵ_0 *in* $Farad \cdot meter^{-1}$.

- const double Na_SI = 6.02252e-27

  Avogadro's number in mol^{-1}.

- const double H0_cgs = 0.7*3.24e-18

  Hubble's constant in CGS.

- const double CE_RG = 0.577215664901532860606512090082402431042159335939994

  Euler's constant.

- const double me_SI = 9.1091e-31

Electron rest mass in Kg.

- const double mp_SI = 1.6726e-27

  Proton rest mass in Kg.

- const double mn_SI = 1.6748e-27

  Neutron rest mass in Kg.

- const double MS_SI = 1.9889e30

  Sun's mass in Kg.

- const double RS_SI = 6.95e8

  Sun's radius in meter.

- const double LS_SI = 3.83e26

  Sun's energy flux in Watt.

- const double Yr_SI = 3.15581498e7

  Sidereal year in seconds.

- const double Dy_SI = 24.0∗3600.0

  Day duration in seconds.

- const double RSchw = 1.47664e3

  Half of the Schwarzhild radius in $\frac{GM}{c^2}$.

- const double au_m = 1.49597870660e11

  Astronomical unit in meters.

- const double ly_m = c_SI∗365.25∗24.0∗3600.0

  Light year in meters ($9.460730472580800 \cdot 10^{15}$*).*

- const double ly_au = ly_m/au_m

  Light year in astronomical units (63240.17695575401*).*

- const double pc_au = M_PI/(3600.0∗180.0)

  Parsec in astronomical unit (206265*).*

- const double pc_m = pc_au∗au_m

  Parsec in meters ($3.086 \cdot 10^{16}$*).*

- const double pc_ly = pc_au/ly_au

  Parsec in light year (3.262*).*

- const double kpc_m = 3.0856675807e19

  Number of meters in a kiloparsec (kpc).

- const double gamma_u = 1.

  Post-Newtonian constant.

11.59.2 Variable Documentation

11.59.2.1 const double au_m = 1.49597870660e11

Astronomical unit in meters.

Definition at line 82 of file LISACODE-PhysicConstants.h.

11.59.2.2 const double c_SI = 299792458

Light's speed in $m \cdot s^{-1}$.

Definition at line 33 of file LISACODE-PhysicConstants.h.

Referenced by GWNewton2::commun(), TrFctGW::deltanu(), LISA::gArmLength(), GWNewton2::GWNewton2(), GWBinary::init(), Geometry::tdelay(), Geometry::tdelayOrderContribution(), ConfigSim::tMaxDelay(), and ConfigSim::tMinDelay().

11.59.2.3 const double CE_RG = 0.57721566490153286060651209008240243104215933593994

Euler's constant.

Definition at line 55 of file LISACODE-PhysicConstants.h.

Referenced by GWNewton2::GWNewton2().

11.59.2.4 const double Dy_SI = 24.0*3600.0

Day duration in seconds.

Definition at line 73 of file LISACODE-PhysicConstants.h.

11.59.2.5 const double eps0_SI = 1.0/(4.e-7*M_PI*c_SI*c_SI)

Permittivity of free space or permittivity ϵ_0 in $Farad \cdot meter^{-1}$.

Definition at line 49 of file LISACODE-PhysicConstants.h.

11.59.2.6 const double G_SI = 6.67259e-11

Gravitational constant in $m^3 \cdot Kg^{-1} \cdot s^{-2}$.

Definition at line 35 of file LISACODE-PhysicConstants.h.

Referenced by GWNewton2::commun(), GWNewton2::GWNewton2(), and GWBinary::init().

11.59.2.7 const double gamma_u = 1.

Post-Newtonian constant.

Definition at line 96 of file LISACODE-PhysicConstants.h.

Referenced by Geometry::tdelay(), and Geometry::tdelayOrderContribution().

11.59.2.8 const double H0_cgs = 0.7*3.24e-18

Hubble's constant in CGS.

Definition at line 53 of file LISACODE-PhysicConstants.h.

11.59.2.9 const double h_SI = 6.62620e-34

Planck's constant in $joule \cdot second$.

Definition at line 39 of file LISACODE-PhysicConstants.h.

11.59.2.10 const double hb_SI = h_SI/(2*M_PI)

Planck's constant divided by $2 \cdot \pi$ in $joule \cdot second$.

Definition at line 41 of file LISACODE-PhysicConstants.h.

11.59.2.11 const double k_SI = 1.381e-23

Boltzmann's constant in $joule \cdot kelvin^{-1}$.

Definition at line 37 of file LISACODE-PhysicConstants.h.

11.59.2.12 const double kpc_m = 3.0856675807e19

Number of meters in a kiloparsec (kpc).

Definition at line 94 of file LISACODE-PhysicConstants.h.

Referenced by GWBinary::GWBinary(), and GWNewton2::GWNewton2().

11.59.2.13 const double LS_SI = 3.83e26

Sun's energy flux in Watt.

Definition at line 69 of file LISACODE-PhysicConstants.h.

11.59.2.14 const double ly_au = ly_m/au_m

Light year in astronomical units (63240.17695575401).

Definition at line 86 of file LISACODE-PhysicConstants.h.

11.59.2.15 const double ly_m = c_SI*365.25*24.0*3600.0

Light year in meters ($9.460730472580800 \cdot 10^{15}$).

Definition at line 84 of file LISACODE-PhysicConstants.h.

11.59.2.16 const double me_SI = 9.1091e-31

Electron rest mass in Kg.

Definition at line 59 of file LISACODE-PhysicConstants.h.

11.59.2.17 const double mn_SI = 1.6748e-27

Neutron rest mass in Kg.

Definition at line 63 of file LISACODE-PhysicConstants.h.

11.59.2.18 const double mp_SI = 1.6726e-27

Proton rest mass in Kg.

Definition at line 61 of file LISACODE-PhysicConstants.h.

11.59.2.19 const double MS_SI = 1.9889e30

Sun's mass in Kg.

Definition at line 65 of file LISACODE-PhysicConstants.h.

Referenced by GWBinary::GWBinary(), and GWNewton2::GWNewton2().

11.59.2.20 const double mu0_SI = 4.e-7*M_PI

Permeability of free space or magnetic constant μ_0 in $Henry \cdot meter^{-1}$.

Definition at line 46 of file LISACODE-PhysicConstants.h.

11.59.2.21 const double Na_SI = 6.02252e-27

Avogadro's number in mol^{-1}.

Definition at line 51 of file LISACODE-PhysicConstants.h.

11.59.2.22 const double pc_au = M_PI/(3600.0*180.0)

Parsec in astronomical unit (206265).

Definition at line 88 of file LISACODE-PhysicConstants.h.

11.59.2.23 const double pc_ly = pc_au/ly_au

Parsec in light year (3.262).

Definition at line 92 of file LISACODE-PhysicConstants.h.

11.59.2.24 const double pc_m = pc_au*au_m

Parsec in meters $(3.086 \cdot 10^{16})$.

Definition at line 90 of file LISACODE-PhysicConstants.h.

11.59.2.25 const double PRECISION = 100.0*DBL_EPSILON

Acceptable precision error on doubles.

Definition at line 29 of file LISACODE-PhysicConstants.h.

Referenced by GWNewton2::commun(), BackgroundGalactic::deltanu(), Serie::gData(), Noise::getNoise(), NoiseFile::loadNoise(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), ConfigSim::NoisesCreation(), NoiseWhite::NoiseWhite(), GW::setDirProp(), Noise::settDurAdd(), Noise::settFirst(), and Noise::settLast().

11.59.2.26 const double RS_SI = 6.95e8

Sun's radius in meter.

Definition at line 67 of file LISACODE-PhysicConstants.h.

11.59.2.27 const double RSchw = 1.47664e3

Half of the Schwarzhild radius in $\frac{GM}{c^2}$.

In Schwarzhild radius G is the gravitational constant, m is the mass of the black hole, and c is the speed of light.

Definition at line 78 of file LISACODE-PhysicConstants.h.

Referenced by Geometry::tdelay(), and Geometry::tdelayOrderContribution().

11.59.2.28 const double S_SI = 5.671e-8

Stefan's constant in $Watt \cdot meter^{-2} \cdot kelvin^{-1}$.

Definition at line 43 of file LISACODE-PhysicConstants.h.

11.59.2.29 const double Yr_SI = 3.15581498e7

Sidereal year in seconds.

Definition at line 71 of file LISACODE-PhysicConstants.h.

11.60 LISACODE-Random.cpp File Reference

```
#include "LISACODE-Random.h"
```

11.61 LISACODE-Random.h File Reference

```
#include <stdexcept>
#include <iostream.h>
#include <math.h>
#include <time.h>
#include "randomc.h"
#include "stocc.h"
#include "LISACODE-Serie.h"
```

Classes

- class RandomMT

  Mersenne twister random generator class.

Defines

- #define RANDOM_GENERATOR TRandomMersenne

  Mersenne twister random generator class.

11.62 LISACODE-Serie.cpp File Reference

```
#include "LISACODE-Serie.h"
```

11.63 LISACODE-Serie.h File Reference

```
#include <stdexcept>
#include <fstream.h>
#include <math.h>
#include <complex>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-MathUtils.h"
```

Classes

- class Serie

  Serie interpolation class.

- class SerieC

  complex serie interpolation class.

Enumerations

- enum INTERP {
  TRU, LIN, CUB, LAG,
  SIN }

  Interpolation type.

11.64 LISACODE-TDI.cpp File Reference

```
#include "LISACODE-TDI.h"
```

11.65 LISACODE-TDI.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <fstream.h>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "LISACODE-Memory.h"
#include "LISACODE-TDI_InterData.h"
#include "LISACODE-TDITools.h"
```

Classes

- class TDI

  Time Delay Interferometry combinaison class.

11.66 LISACODE-TDI_InterData.cpp File Reference

```
#include "LISACODE-TDI_InterData.h"
```

11.67 LISACODE-TDI_InterData.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "LISACODE-LISAConstants.h"
#include "LISACODE-Serie.h"
#include "LISACODE-Memory.h"
```

Classes

- class TDI_InterData

  Time Delay Interferometry interpolated signal class.

11.68 LISACODE-TDIApply.cpp File Reference

```
#include <stdexcept>
#include <iostream>
#include <fstream.h>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-MathUtils.h"
#include "LISACODE-LISAConstants.h"
#include "LISACODE-Memory.h"
#include "LISACODE-MemoryWriteDisk.h"
#include "LISACODE-MemoryReadDisk.h"
#include "LISACODE-TDI_InterData.h"
#include "LISACODE-TDITools.h"
#include "LISACODE-TDI.h"
#include "LISACODE-ConfigSim.h"
```

Functions

- int main (int argc, char ∗const argv[ ])

11.68.1 Function Documentation

11.68.1.1 int main (int *argc*, char ∗const *argv*[ ])

Definition at line 36 of file LISACODE-TDIApply.cpp.

References Memory::AddSerieData(), TDI_InterData::ComputeEta(), ConfigSim::getFileNameDelays(), ConfigSim::getFileNameSig(), ConfigSim::getFileNameTDI(), ConfigSim::getGenTDIPacks(), ConfigSim::getNameGenTDI(), ConfigSim::getNbMaxDelays(), ConfigSim::getNoNoise(), TDITools::getRapidOption(), ConfigSim::gettDeltaTDIDelay(), ConfigSim::getTDIDelayApprox(), ConfigSim::getTDIInterp(), ConfigSim::getTDIInterpUtilVal(), ConfigSim::gettDisplay(), ConfigSim::gettMax(), ConfigSim::gettStepMes(), LISACodeVersion, MAX, ConfigSim::NbGenTDI(), Memory::RecordAccData(), TDITools::RefreshDelay(), ConfigSim::tMaxDelay(), ConfigSim::tMemNecInterpTDI(), and ConfigSim::tMinDelay().

11.69 LISACODE-TDITools.cpp File Reference

```
#include "LISACODE-TDITools.h"
```

11.70 LISACODE-TDITools.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <fstream.h>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "LISACODE-Memory.h"
```

Classes

- class TDITools

  Time Delay Interferometry tools class.

11.71 LISACODE-TrFctGW.cpp File Reference

```
#include "LISACODE-TrFctGW.h"
```

11.72 LISACODE-TrFctGW.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <string.h>
#include <math.h>
#include "LISACODE-MathUtils.h"
#include "LISACODE-Geometry.h"
#include "LISACODE-GW.h"
```

Classes

- class TrFctGW

  Gravitational Waves Transfer Function class.

11.73 LISACODE-USOClock.cpp File Reference

```
#include "LISACODE-USOClock.h"
```

11.74 LISACODE-USOClock.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "randlib.h"
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-MathUtils.h"
```

Classes

- class USOClock
  Ultra Stable Oscillator based satellite time is defined in this class.

11.75 LISACODE-Vect.cpp File Reference

```
#include "LISACODE-Vect.h"
```

Functions

- Vect operator+ (Vect u, Vect v)

  Vectors addition : returns vector u+v.

- Vect operator- (Vect u, Vect v)

  Vectors subtraction : returns vector u-v.

- double operator $*$ (Vect u, Vect v)

  Vectors scalar product, returns a scalar.

- Vect operator $*$ (double a, Vect u)

  Vector and scalar product, returns a vector.

- Vect operator $*$ (Vect u, double a)

  Vector and scalar product, returns a vector.

- Vect operator/ (Vect u, double a)

  Vector and scalar division, returns a vector.

11.75.1 Function Documentation

11.75.1.1 Vect operator $*$ (Vect *u*, double *a*)

Vector and scalar product, returns a vector.

$$\text{returned value} = a \cdot \overrightarrow{u}$$

Definition at line 157 of file LISACODE-Vect.cpp.

References Vect::p.

11.75.1.2 Vect operator $*$ (double *a*, Vect *u*)

Vector and scalar product, returns a vector.

$$\text{returned value} = a \cdot \overrightarrow{u}$$

Definition at line 143 of file LISACODE-Vect.cpp.

References Vect::p.

11.75.1.3 double operator $*$ (Vect *u*, Vect *v*)

Vectors scalar product, returns a scalar.

$$\text{returned value} = \overrightarrow{u} \cdot \overrightarrow{v}$$

Definition at line 128 of file LISACODE-Vect.cpp.

References Vect::p.

11.75.1.4 Vect operator+ (Vect *u*, Vect *v*)

Vectors addition : returns vector u+v.

Definition at line 101 of file LISACODE-Vect.cpp.

References Vect::p.

11.75.1.5 Vect operator- (Vect *u*, Vect *v*)

Vectors subtraction : returns vector u-v.

Definition at line 112 of file LISACODE-Vect.cpp.

References Vect::p.

11.75.1.6 Vect operator/ (Vect *u*, double *a*)

Vector and scalar division, returns a vector.

$$\text{returned value} = \frac{\overrightarrow{u}}{a}$$

Definition at line 171 of file LISACODE-Vect.cpp.

References Vect::p.

11.76 LISACODE-Vect.h File Reference

```
#include <stdexcept>
#include <iostream.h>
#include <math.h>
```

Classes

- class Vect

    3 components vector management class.

11.77 randlib.c File Reference

11.77.1 Detailed Description

Randlib functions.

Definition in file randlib.c.

```
#include "randlib.h"
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
```

Defines

- #define ABS(x) ((x) >= 0 ? (x) : -(x))
- #define min(a, b) ((a) <= (b) ? (a) : (b))
- #define max(a, b) ((a) >= (b) ? (a) : (b))
- #define expmax 87.49823
- #define infnty 1.0E38
- #define minlog 1.0E-37
- #define numg 32L
- #define maxnum 2147483561L
- #define h 32768L

Functions

- void ftnstop (char *)
- float genbet (float aa, float bb)

  Generates beta random deviate.

- float genchi (float df)
- float genexp (float av)
- float genf (float dfn, float dfd)
- float gengam (float a, float r)
- void genmn (float *parm, float *x, float *work)
- void genmul (long n, float *p, long ncat, long *ix)
- float gennch (float df, float xnonc)
- float gennf (float dfn, float dfd, float xnonc)
- float gennor (float av, float sd)
- void genprm (long *iarray, int larray)
- float genunf (float low, float high)

  Generates uniform real between LOW and HIGH.

- void gscgn (long getset, long *g)
- void gsrgs (long getset, long *qvalue)
- void gssst (long getset, long *qset)
- long ignbin (long n, float pp)
- long ignnbn (long n, float p)

- long ignpoi (float mu)
- long ignuin (long low, long high)
- long lennob (char ∗str)
- long mltmod (long a, long s, long m)
- void phrtsd (char ∗phrase, long ∗seed1, long ∗seed2)
- float ranf (void)
- void setgmn (float ∗meanv, float ∗covm, long p, float ∗parm)
- float sexpo (void)
- float sgamma (float a)
- float snorm (void)
- float fsign (float num, float sign)

11.77.2 Define Documentation

11.77.2.1 #define ABS(x) ((x) >= 0 ? (x) : -(x))

Definition at line 8 of file randlib.c.

Referenced by ignbin().

11.77.2.2 #define expmax 87.49823

11.77.2.3 #define h 32768L

11.77.2.4 #define infnty 1.0E38

11.77.2.5 #define max(a, b) ((a) >= (b) ? (a) : (b))

Definition at line 10 of file randlib.c.

Referenced by ezxml_ampencode(), ezxml_str2utf8(), ezxml_toxml(), ezxml_toxml_r(), genbet(), and ignpoi().

11.77.2.6 #define maxnum 2147483561L

11.77.2.7 #define min(a, b) ((a) <= (b) ? (a) : (b))

Definition at line 9 of file randlib.c.

Referenced by genbet(), ignbin(), and ignpoi().

11.77.2.8 #define minlog 1.0E-37

11.77.2.9 #define numg 32L

11.77.3 Function Documentation

11.77.3.1 float fsign (float *num*, float *sign*)

Definition at line 2113 of file randlib.c.

Referenced by ignpoi(), and sgamma().

11.77.3.2 void ftnstop (char $*$)

Definition at line 2124 of file randlib.c.

Referenced by genmul(), ignbin(), and ignnbn().

11.77.3.3 float genbet (float *aa*, float *bb*)

Generates beta random deviate.

Returns a single random deviate from the beta distribution with parameters A and B. The density of the beta is

$$\frac{x^{a-1} \cdot (1-x)^{b-1}}{B(a,b)}$$

for $0 < x < 1$

- aa First parameter of the beta distribution
- bb Second parameter of the beta distribution

Definition at line 24 of file randlib.c.

References genbet(), max, min, and ranf().

Referenced by genbet().

11.77.3.4 float genchi (float *df*)

Definition at line 260 of file randlib.c.

References genchi(), and sgamma().

Referenced by genchi().

11.77.3.5 float genexp (float *av*)

Definition at line 291 of file randlib.c.

References genexp(), and sexpo().

Referenced by genexp().

11.77.3.6 float genf (float *dfn*, float *dfd*)

Definition at line 325 of file randlib.c.

References genf(), and sgamma().

Referenced by genf().

11.77.3.7 float gengam (float *a*, float *r*)

Definition at line 381 of file randlib.c.

References gengam(), and sgamma().

Referenced by gengam().

11.77.3.8 void genmn (float * *parm*, float * *x*, float * *work*)

Definition at line 426 of file randlib.c.

References snorm().

11.77.3.9 void genmul (long *n*, float * *p*, long *ncat*, long * *ix*)

Definition at line 477 of file randlib.c.

References ftnstop(), and ignbin().

11.77.3.10 float gennch (float *df*, float *xnonc*)

Definition at line 538 of file randlib.c.

References gennch(), sgamma(), and snorm().

Referenced by gennch().

11.77.3.11 float gennf (float *dfn*, float *dfd*, float *xnonc*)

Definition at line 583 of file randlib.c.

References gennf(), sgamma(), and snorm().

Referenced by gennf().

11.77.3.12 float gennor (float *av*, float *sd*)

Definition at line 656 of file randlib.c.

References gennor(), and snorm().

Referenced by gennor().

11.77.3.13 void genprm (long * *iarray*, int *larray*)

Definition at line 690 of file randlib.c.

References ignuin().

11.77.3.14 float genunf (float *low*, float *high*)

Generates uniform real between LOW and HIGH.

Generates a real uniformly distributed between LOW and HIGH.

Parameters:

low Low bound (exclusive) on real value to be generated

high High bound (exclusive) on real value to be generated

Definition at line 719 of file randlib.c.

References genunf(), and ranf().

Referenced by NoiseFilter::generNoise(), NoiseWhite::generNoise(), genunf(), USOClock::gGap(), NoiseFilter::loadNoise(), NoiseWhite::loadNoise(), and main().

11.77.3.15 void gscgn (long *getset*, long ∗ *g*)

Definition at line 742 of file randlib.c.

Referenced by advnst(), getsd(), ignlgi(), initgn(), setall(), setant(), and setsd().

11.77.3.16 void gsrgs (long *getset*, long ∗ *qvalue*)

Definition at line 767 of file randlib.c.

Referenced by advnst(), getsd(), ignlgi(), initgn(), inrgcm(), setall(), setant(), and setsd().

11.77.3.17 void gssst (long *getset*, long ∗ *qset*)

Definition at line 784 of file randlib.c.

Referenced by ignlgi(), and setall().

11.77.3.18 long ignbin (long *n*, float *pp*)

Definition at line 800 of file randlib.c.

References ABS, ftnstop(), ignbin(), min, and ranf().

Referenced by genmul(), and ignbin().

11.77.3.19 long ignnbn (long *n*, float *p*)

Definition at line 1067 of file randlib.c.

References ftnstop(), ignnbn(), ignpoi(), and sgamma().

Referenced by ignnbn().

11.77.3.20 long ignpoi (float *mu*)

Definition at line 1122 of file randlib.c.

References fsign(), ignpoi(), max, min, omega, ranf(), sexpo(), and snorm().

Referenced by ignnbn(), and ignpoi().

11.77.3.21 long ignuin (long *low*, long *high*)

Definition at line 1372 of file randlib.c.

References ignlgi(), and ignuin().

Referenced by genprm(), and ignuin().

11.77.3.22 long lennob (char ∗ *str*)

Definition at line 1425 of file randlib.c.

Referenced by phrtsd().

11.77.3.23 long mltmod (long *a*, long *s*, long *m*)

Definition at line 1437 of file randlib.c.

References mltmod().

Referenced by advnst(), initgn(), mltmod(), and setall().

11.77.3.24 void phrtsd (char ∗ *phrase*, long ∗ *seed1*, long ∗ *seed2*)

Definition at line 1531 of file randlib.c.

References lennob().

11.77.3.25 float ranf (void)

Definition at line 1593 of file randlib.c.

References ignlgi(), and ranf().

Referenced by genbet(), genunf(), ignbin(), ignpoi(), ranf(), sexpo(), sgamma(), and snorm().

11.77.3.26 void setgmn (float ∗ *meanv*, float ∗ *covm*, long *p*, float ∗ *parm*)

Definition at line 1617 of file randlib.c.

References spofa().

11.77.3.27 float sexpo (void)

Definition at line 1682 of file randlib.c.

References ranf(), and sexpo().

Referenced by genexp(), ignpoi(), sexpo(), and sgamma().

11.77.3.28 float sgamma (float *a*)

Definition at line 1747 of file randlib.c.

References fsign(), ranf(), sexpo(), sgamma(), and snorm().

Referenced by genchi(), genf(), gengam(), gennch(), gennf(), ignnbn(), and sgamma().

11.77.3.29 float snorm (void)

Definition at line 1986 of file randlib.c.

References ranf(), and snorm().

Referenced by genmn(), gennch(), gennf(), gennor(), ignpoi(), sgamma(), and snorm().

11.78 randlib.h File Reference

Functions

- void advnst (long k)
- float genbet (float aa, float bb)

  Generates beta random deviate.

- float genchi (float df)
- float genexp (float av)
- float genf (float dfn, float dfd)
- float gengam (float a, float r)
- void genmn (float ∗parm, float ∗x, float ∗work)
- void genmul (long n, float ∗p, long ncat, long ∗ix)
- float gennch (float df, float xnonc)
- float gennf (float dfn, float dfd, float xnonc)
- float gennor (float av, float sd)
- void genprm (long ∗iarray, int larray)
- float genunf (float low, float high)

  Generates uniform real between LOW and HIGH.

- void getsd (long ∗iseed1, long ∗iseed2)
- void gscgn (long getset, long ∗g)
- long ignbin (long n, float pp)
- long ignnbn (long n, float p)
- long ignlgi (void)
- long ignpoi (float mu)
- long ignuin (long low, long high)
- void initgn (long isdtyp)
- long mltmod (long a, long s, long m)
- void phrtsd (char ∗phrase, long ∗seed1, long ∗seed2)
- float ranf (void)
- void setall (long iseed1, long iseed2)
- void setant (long qvalue)
- void setgmn (float ∗meanv, float ∗covm, long p, float ∗parm)
- void setsd (long iseed1, long iseed2)
- float sexpo (void)
- float sgamma (float a)
- float snorm (void)

11.78.1 Function Documentation

11.78.1.1 void advnst (long *k*)

Definition at line 7 of file com.c.

References gscgn(), gsrgs(), mltmod(), setsd(), Xa1, Xa2, Xcg1, Xcg2, Xm1, and Xm2.

11.78.1.2 float genbet (float *aa*, float *bb*)

Generates beta random deviate.

Returns a single random deviate from the beta distribution with parameters A and B. The density of the beta is

$$\frac{x^{a-1} \cdot (1-x)^{b-1}}{B(a,b)}$$

for $0 < x < 1$

- aa First parameter of the beta distribution
- bb Second parameter of the beta distribution

Definition at line 24 of file randlib.c.

References genbet(), max, min, and ranf().

Referenced by genbet().

11.78.1.3 float genchi (float *df*)

Definition at line 260 of file randlib.c.

References genchi(), and sgamma().

Referenced by genchi().

11.78.1.4 float genexp (float *av*)

Definition at line 291 of file randlib.c.

References genexp(), and sexpo().

Referenced by genexp().

11.78.1.5 float genf (float *dfn*, float *dfd*)

Definition at line 325 of file randlib.c.

References genf(), and sgamma().

Referenced by genf().

11.78.1.6 float gengam (float *a*, float *r*)

Definition at line 381 of file randlib.c.

References gengam(), and sgamma().

Referenced by gengam().

11.78.1.7 void genmn (float $*$ *parm*, float $*$ *x*, float $*$ *work*)

Definition at line 426 of file randlib.c.

References snorm().

11.78.1.8 void genmul (long *n*, float $*$ *p*, long *ncat*, long $*$ *ix*)

Definition at line 477 of file randlib.c.

References ftnstop(), and ignbin().

11.78.1.9 float gennch (float *df*, float *xnonc*)

Definition at line 538 of file randlib.c.

References gennch(), sgamma(), and snorm().

Referenced by gennch().

11.78.1.10 float gennf (float *dfn*, float *dfd*, float *xnonc*)

Definition at line 583 of file randlib.c.

References gennf(), sgamma(), and snorm().

Referenced by gennf().

11.78.1.11 float gennor (float *av*, float *sd*)

Definition at line 656 of file randlib.c.

References gennor(), and snorm().

Referenced by gennor().

11.78.1.12 void genprm (long $*$ *iarray*, int *larray*)

Definition at line 690 of file randlib.c.

References ignuin().

11.78.1.13 float genunf (float *low*, float *high*)

Generates uniform real between LOW and HIGH.

Generates a real uniformly distributed between LOW and HIGH.

Parameters:

low Low bound (exclusive) on real value to be generated

high High bound (exclusive) on real value to be generated

Definition at line 719 of file randlib.c.

References genunf(), and ranf().

Referenced by NoiseWhite::generNoise(), NoiseFilter::generNoise(), genunf(), USOClock::gGap(), NoiseWhite::loadNoise(), NoiseFilter::loadNoise(), and main().

11.78.1.14 void getsd (long * *iseed1*, long * *iseed2*)

Definition at line 52 of file com.c.

References gscgn(), gsrgs(), Xcg1, and Xcg2.

Referenced by main().

11.78.1.15 void gscgn (long *getset*, long * *g*)

Definition at line 742 of file randlib.c.

Referenced by advnst(), getsd(), ignlgi(), initgn(), setall(), setant(), and setsd().

11.78.1.16 long ignbin (long *n*, float *pp*)

Definition at line 800 of file randlib.c.

References ABS, ftnstop(), ignbin(), min, and ranf().

Referenced by genmul(), and ignbin().

11.78.1.17 long ignlgi (void)

Definition at line 89 of file com.c.

References gscgn(), gsrgs(), gssst(), ignlgi(), inrgcm(), setall(), Xa1, Xa2, Xcg1, Xcg2, Xm1, Xm2, and Xqanti.

Referenced by ignlgi(), ignuin(), and ranf().

11.78.1.18 long ignnbn (long *n*, float *p*)

Definition at line 1067 of file randlib.c.

References ftnstop(), ignnbn(), ignpoi(), and sgamma().

Referenced by ignnbn().

11.78.1.19 long ignpoi (float *mu*)

Definition at line 1122 of file randlib.c.

References fsign(), ignpoi(), max, min, omega, ranf(), sexpo(), and snorm().

Referenced by ignnbn(), and ignpoi().

11.78.1.20 long ignuin (long *low*, long *high*)

Definition at line 1372 of file randlib.c.

References ignlgi(), and ignuin().

Referenced by genprm(), and ignuin().

11.78.1.21 void initgn (long *isdtyp*)

Definition at line 143 of file com.c.

References gscgn(), gsrgs(), mltmod(), Xa1w, Xa2w, Xcg1, Xcg2, Xig1, Xig2, Xlg1, Xlg2, Xm1, and Xm2.

Referenced by setall(), and setsd().

11.78.1.22 long mltmod (long *a*, long *s*, long *m*)

Definition at line 1437 of file randlib.c.

References mltmod().

Referenced by advnst(), initgn(), mltmod(), and setall().

11.78.1.23 void phrtsd (char ∗ *phrase*, long ∗ *seed1*, long ∗ *seed2*)

Definition at line 1531 of file randlib.c.

References lennob().

11.78.1.24 float ranf (void)

Definition at line 1593 of file randlib.c.

References ignlgi(), and ranf().

Referenced by genbet(), genunf(), ignbin(), ignpoi(), ranf(), sexpo(), sgamma(), and snorm().

11.78.1.25 void setall (long *iseed1*, long *iseed2*)

Definition at line 245 of file com.c.

References gscgn(), gsrgs(), gssst(), initgn(), inrgcm(), mltmod(), Xa1vw, Xa2vw, Xig1, Xig2, Xm1, and Xm2.

Referenced by ignlgi(), and main().

11.78.1.26 void setant (long *qvalue*)

Definition at line 296 of file com.c.

References gscgn(), gsrgs(), and Xqanti.

11.78.1.27 void setgmn (float ∗ *meanv*, float ∗ *covm*, long *p*, float ∗ *parm*)

Definition at line 1617 of file randlib.c.

References spofa().

11.78.1.28 void setsd (long *iseed1*, long *iseed2*)

Definition at line 337 of file com.c.

References gscgn(), gsrgs(), initgn(), Xig1, and Xig2.

Referenced by advnst().

11.78.1.29 float sexpo (void)

Definition at line 1682 of file randlib.c.

References ranf(), and sexpo().

Referenced by genexp(), ignpoi(), sexpo(), and sgamma().

11.78.1.30 float sgamma (float *a*)

Definition at line 1747 of file randlib.c.

References fsign(), ranf(), sexpo(), sgamma(), and snorm().

Referenced by genchi(), genf(), gengam(), gennch(), gennf(), ignnbn(), and sgamma().

11.78.1.31 float snorm (void)

Definition at line 1986 of file randlib.c.

References ranf(), and snorm().

Referenced by genmn(), gennch(), gennf(), gennor(), ignpoi(), sgamma(), and snorm().

Chapter 12

LISACode Page Documentation

12.1 Introduction

LISACode is a LISA mission simulator. It is highly structured and programmed in C++. The simulator has the purpose to bridge the gap between the basic principles of LISA and a sophisticated end-to-end engineering level simulator. This software package, which runs on most computer platforms, can be downloaded from the Lisa-France web site (http://www.apc.univ-paris7.fr/LISA-France/analyse.phtml).

12.1.1 LISACode technical description

LISACode simulates the LISA gravitational wave (GW) detector (see http://www.esa.int/esa-SC/SEMEJRR1VED_index_0.html). It does not aim at simulating the LISA detector in detail but rather it uses the response function of its main components, particularly because they will affect the noise level of the detector response. It also includes an implementation of the TDI (Time Delay Interferometry, [Tinto 2004]) technique which allows to suppress the noise introduced by lasers frequency instability.

The main inputs and outputs of LISACode are time-dependent sequences. Input sequences describe the GW strain and output sequences describe the phasemeters response or their treatment via various TDI combination.

A number of elementary GW signals can be defined, but the main aim of the code is to be used in conjunction with more sophisticated GW simulators via intermediate data files.

12.2 A description of the Code

12.2.1 Code organisation

LISACode is written in C++ and has a very modular structure. The main structure of LISACode is shown in the figure below. This structure maps the main components of the LISA detector as well as its physical inputs (see details in in [LISACode]). Its main components are:

- a variety of GW inputs,
- a detailed description of the orbits [Nayak and all.] of the three satellites (including the breathing and rotation modes of the LISA triangle),
- the different noise sources (lasers, DFS and the interferometric measurements),
- the phasemetre measurements and
- the Ultra Stable Oscillator (USO) clock performances.

latex New_Structure_Anglais.eps

The next figure shows the organisation of the libraries used by LISACode. The green boxes represent objects, the red boxes the libraries or modules (see Modules) and the pink boxes the executables.

latex droppedImage-2_-_petit.eps

12.2.2 Physical constants

Constants used by the LISA simulator are described in next files:

LISACODE-LISAConstants.h : Physical constants of LISA instrument.

LISACODE-PhysicConstants.h : Physical constants, reference values and unit conversions.

12.2.3 Details of LISACode simulator input/outputs

The first input is the GW itself. It can be defined internally through some simplified models which produce signals of constant frequency. The GW may also be input via a time sequence as well as produced by more sophisticated simulator codes. An example of this is given below.

The orbits of LISA are generated internally by the simulator. They correspond to realistic orbits that contain both the breathing and rotation modes of Lisa as a function of its rotation around the sun (see [Dhurandhar and all. 2004]). The parameters of these orbits can be adjusted to modify the average distance between the satellites (nominally $510^9 m$) or be defined in such a way to keep LISA at a fixed given location. The initial position on the orbits can also be defined in inputs.

An important element of the LISA response and hence of the code are the different nature of noise inputs. This include the optical noise due to shot noise and related factors as described in table 4.1 of [Tinto 2004].The inertial mass and the laser noises can also be defined at input. Normally, these noises are defined as bandwidth limited white noise but different shapes of noise can be used.

Using the orbits, the response of LISA to the GW will be calculated and a relative frequency fluctuation (see [Dhurandhar -TDelay 2004]) will be input to the Phasemeter Module. These will be combined with the different noise contribution to produce the primary phasemeter output signal which then will be processed through a Butterworth filtering module. In standard operation, the primary signal will be produced at a 10 Hz signal and outputted, after filtering, at a 1 Hz rate.

These signals can be saved on disk files and/or processed by a TDI module using a variety of TDI combinations that are defined in input.

The above description of the code provides only a brief summary of its capabilities. The LISACode parameters will provide a (non-exhaustive) list of the parameters that can be used to control the input, the processing and the output of the code and will therefore give a more complete idea of its possibilities.

12.3 LISACode parameters

As the use of the different parameters may be complicated, the reader may again refer to section IV of http://www.apc.univ-paris7.fr/LISA-France/analyse.phtml where examples of various input and output are given.

The following figures give the different reference frames that are used in the LISACode. latex figure_-psi-last.eps width=10cm In an input file, you may include your own comments. These must be preceded (column 1 and 2) by the # symbol followed by one space. The file should preferably end with the Keyword "END".

Times are in seconds, lengths in meters, angles in degrees, frequencies in Hz, ...

There is a variety of output files. Some of them are related to the phasemeter outputs. One of them is related to the output of the various TDI combinations. These may be defined in the configuration file and hence the output file will reflect this choice. In the examples given in http://www.apc.univ-paris7.fr/LISA-France/analyse.phtml the output file consists of 9 columns. The first column is the time, the second, third, fourth and fifth columns give the alpha, beta, gamma and zeta TDI combination [Tinto 2004]. The sixth column gives the first generation Michelson combination (X1s1) related to satellite s1 (that is to say using the arms between s1 and s2 and s1 and s3). The seventh, eighth and ninth columns give the 2nd generation Michelson combination for satellite 1,2 and 3.

12.4 Bibliography

[Tinto 2004]	Massimo Tinto, Sanjeev V. Dhurandhar **Time-Delay Interferometry**,Living Rev.Rel. 8 (2005) 4, http://xxx.lanl.gov/abs/gr-qc/0409034
[Dhurandhar and all. 2004]	S. V. Dhurandhar, K. Rajesh Nayak, S. Koshti, J-Y. Vinet **Fundamentals of the LISA Stable Flight Formation**,Class.Quant.Grav. 22 (2005) 481-488, http://xxx.lanl.gov/abs/gr-qc/0410093
[Dhurandhar -TDelay 2004]	S. V. Dhurandhar, K. Rajesh Nayak, J-Y. Vinet **Algebraic approach to time-delay data analysis for LISA**,Phys.Rev. D70 (2004) 102003. http://xxx.lanl.gov/abs/gr-qc/0112059
[LISACode]	Antoine Petiteau, Gerard Auger, Hubert Halloin, Olivier Jeannin, Sophie Pireaux, Eric Plagnol, Tania Regimbau and Jean-YvesVinet. **LISACode : Simulating Lisa**
[Nayak and all.]	NK.R. Nayak and S. Koshti and S.V. Dhurandhar and J.-Y.Vinet **On the minimum flexing of LISA** Classical and Quantum Gravity 23, 1763 (2006)

12.5 Todo List

Member ConfigSim::ConfigSim() Create a variable for the default configuration file name ("ConfigBase").

Member Noise::Noise() Define variables for the default Noise values in Noise.h and use them in Noise and its derivated classes.

Create a function to compute NbData from tFirst, tLast and tStep and use it in Noise constructors and its derivated classes.

Member Noise::Noise() Make a function to compute a int from a double. Is there a reason to no call rint ?

Member Noise::Noise(double tStep_n, double tDurAdd_n) Make a function to compute a int from a double. Is there a reason to no call rint ?

Member Noise::Noise(double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n) Make a function to compute a int from a double. Is there a reason to no call rint ?

Member Noise::getNoise(double tDelay) const Make a function to compute a int from a double. Is there a reason to no call rint ?

Make a function to compute a int from a double. Is there a reason to no call rint ?

Replace code by call to InterLagrange or its optimised function

Member Noise::settDurAdd(double tDurAdd_n) Make a function to verify that a value is an integer. and replace code here after.

Make a function to compute a int from a double. Is there a reason to no call rint ?

Member Noise::settFirst(double tFirst_n) Make a function to verify that a value is an integer. and replace code here after.

Member Noise::settLast(double tLast_n) Make a function to verifie that a value is an integer. and replace code here after.

Class NoiseFile Correct class description in french

Member NoiseFile::NoiseFile() Make a function to compute a int from a double. Is there a reason to no call rint ?

Member NoiseFile::NoiseFile(double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, char ∗FileName_n) Make a function to compute a int from a double. Is there a reason to no call rint ?

Member NoiseFile::loadNoise() Make a function to compute a int from a double. Is there a reason to no call rint ?

Member NoiseFile::StoredData Are StoredData and NbDataStored necessary. Are they different to #NbData and #NoiseData?

Member NoiseFilter::NoiseFilter() loadNoise and generateNoise have confuse names in relation to the activities of the methods.

Member NoiseFilter::NoiseFilter() Make a function to compute a int from a double. Is there a reason to no call rint ?

Member NoiseFilter::NoiseFilter(double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n) Make a function to compute a int from a double. Is there a reason to no call rint ?

Member NoiseFilter::NoiseFilter(double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, vector< vector< Make a function to compute a int from a double. Is there a reason to no call rint ?

Member NoiseFilter::NoiseFilter(double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, vector< vector< Make a function to compute a int from a double. Is there a reason to no call rint ?

Member NoiseWhite::NoiseWhite() Make a function to compute a int from a double. Is there a reason to no call rint ?

Member NoiseWhite::NoiseWhite(double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n) Make a function to compute a int from a double. Is there a reason to no call rint ?

Member NoiseWhite::NoiseWhite(double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, double SqPSD) Make a function to compute a int from a double. Is there a reason to no call rint ?

Index