

LISACode Reference Manual

1.2

Generated by Doxygen 1.3.4

Thu May 14 16:17:21 2009

Contents

1	Developers' Manual of LISACode	1
2	LISACode Module Index	3
2.1	LISACode Modules	3
3	LISACode Namespace Index	5
3.1	LISACode Namespace List	5
4	LISACode Hierarchical Index	7
4.1	LISACode Class Hierarchy	7
5	LISACode Class Index	9
5.1	LISACode Class List	9
6	LISACode File Index	11
6.1	LISACode File List	11
7	LISACode Page Index	15
7.1	LISACode Related Pages	15
8	LISACode Module Documentation	17
8.1	Noise (directory Bruits)	17
8.2	Detector (directory Dectecteur)	18
8.3	Input data (directory Input_data)	19
8.4	Couple	20
8.5	Elliptic Filter	21
8.6	Filter	31
8.7	Matrix	32
8.8	Mathematical Tools (directory Outils_Math)	33
8.9	Angles handling	34
8.10	Serie	35

8.11 Vector	36
8.12 Gravitational waves (directory Ondes_Gravit)	37
8.13 Background (directory Background)	38
8.14 USO clock (directory USO_Temps)	39
8.15 TDI handling (directory TDI)	40
8.16 TDI	41
8.17 TDI_InterData	42
8.18 TDITools	43
8.19 Geometry (directory Orbitographie)	44
8.20 Main (directory Main)	45
8.21 Memory (directory Memoire)	47
9 LISACode Namespace Documentation	49
9.1 std Namespace Reference	49
10 LISACode Class Documentation	51
10.1 Background Class Reference	51
10.2 BackgroundGalactic Class Reference	54
10.3 ConfigSim Class Reference	60
10.4 Couple Class Reference	96
10.5 ezxml Struct Reference	99
10.6 ezxml_root Struct Reference	102
10.7 Filter Class Reference	104
10.8 Geometry Class Reference	110
10.9 GeometryAnalytic Class Reference	123
10.10 GeometryFile Class Reference	139
10.11 GeometryMLDC Class Reference	154
10.12 GW Class Reference	170
10.13 GWBinary Class Reference	178
10.14 GWCusp Class Reference	189
10.15 GWFastSpinBBH Class Reference	199
10.16 GWFfile Class Reference	229
10.17 GWMono Class Reference	240
10.18 GWNew Class Reference	250
10.19 GWNewton2 Class Reference	260
10.20 GWPeriGate Class Reference	288
10.21 GWSto Class Reference	297

10.22LISA Class Reference	307
10.23Mat Class Reference	315
10.24MathUtils Class Reference	318
10.25Memory Class Reference	319
10.26MemoryReadDisk Class Reference	325
10.27MemoryWriteDisk Class Reference	333
10.28Noise Class Reference	342
10.29NoiseFile Class Reference	351
10.30NoiseFilter Class Reference	362
10.31NoiseFShape Class Reference	374
10.32NoiseOof Class Reference	386
10.33NoiseSpec Struct Reference	397
10.34NoiseTwoFilter Class Reference	399
10.35NoiseWhite Class Reference	410
10.36PhoDetPhaMet Class Reference	421
10.37QuadCell Struct Reference	435
10.38Serie Class Reference	437
10.39SerieC Class Reference	446
10.40TDI Class Reference	452
10.41TDI_InterData Class Reference	462
10.42TDITools Class Reference	469
10.43TrFctGW Class Reference	472
10.44USOClock Class Reference	476
10.45Vect Class Reference	481
11 LISACode File Documentation	485
11.1 com.c File Reference	485
11.2 Doxygen.bibliography File Reference	490
11.3 ezxml.c File Reference	491
11.4 ezxml.h File Reference	500
11.5 linpack.c File Reference	508
11.6 LISACODE-Background.cpp File Reference	509
11.7 LISACODE-Background.h File Reference	510
11.8 LISACODE-BackgroundGalactic.cpp File Reference	511
11.9 LISACODE-BackgroundGalactic.h File Reference	512
11.10LISACODE-ConfigSim.cpp File Reference	513
11.11LISACODE-ConfigSim.h File Reference	514

11.12LISACODE-Couple.cpp File Reference	516
11.13LISACODE-Couple.h File Reference	518
11.14LISACODE-DnonGW.cpp File Reference	519
11.15LISACODE-EllipticFilter.cpp File Reference	520
11.16LISACODE-EllipticFilter.h File Reference	522
11.17LISACODE-Filter.cpp File Reference	524
11.18LISACODE-Filter.h File Reference	525
11.19LISACODE-Geometry.cpp File Reference	526
11.20LISACODE-Geometry.h File Reference	527
11.21LISACODE-GeometryAnalytic.cpp File Reference	528
11.22LISACODE-GeometryAnalytic.h File Reference	529
11.23LISACODE-GeometryFile.cpp File Reference	530
11.24LISACODE-GeometryFile.h File Reference	531
11.25LISACODE-GeometryMLDC.cpp File Reference	532
11.26LISACODE-GeometryMLDC.h File Reference	533
11.27LISACODE-GW.cpp File Reference	534
11.28LISACODE-GW.h File Reference	535
11.29LISACODE-GWBinary.cpp File Reference	536
11.30LISACODE-GWBinary.h File Reference	537
11.31LISACODE-GWCusp.cpp File Reference	538
11.32LISACODE-GWCusp.h File Reference	539
11.33LISACODE-GWFastSpinBBH.cpp File Reference	540
11.34LISACODE-GWFastSpinBBH.h File Reference	541
11.35LISACODE-GWFile.cpp File Reference	542
11.36LISACODE-GWFile.h File Reference	543
11.37LISACODE-GWMono.cpp File Reference	544
11.38LISACODE-GWMono.h File Reference	545
11.39LISACODE-GWNew.cpp File Reference	546
11.40LISACODE-GWNew.h File Reference	547
11.41LISACODE-GWNewton2.cpp File Reference	548
11.42LISACODE-GWNewton2.h File Reference	549
11.43LISACODE-GWPeriGate.cpp File Reference	550
11.44LISACODE-GWPeriGate.h File Reference	551
11.45LISACODE-GWSto.cpp File Reference	552
11.46LISACODE-GWSto.h File Reference	553
11.47LISACODE-LISA.cpp File Reference	554

11.48LISACODE-LISA.h File Reference	555
11.49LISACODE-LISACode.cpp File Reference	556
11.50LISACODE-LISACode.ep.cpp File Reference	558
11.51LISACODE-LISAConstants.h File Reference	559
11.52LISACODE-Mat.cpp File Reference	562
11.53LISACODE-Mat.h File Reference	563
11.54LISACODE-MathUtils.h File Reference	564
11.55LISACODE-Memory.cpp File Reference	565
11.56LISACODE-Memory.h File Reference	566
11.57LISACODE-MemoryReadDisk.cpp File Reference	567
11.58LISACODE-MemoryReadDisk.h File Reference	568
11.59LISACODE-MemoryWriteDisk.cpp File Reference	569
11.60LISACODE-MemoryWriteDisk.h File Reference	570
11.61LISACODE-Noise.cpp File Reference	571
11.62LISACODE-Noise.h File Reference	572
11.63LISACODE-NoiseFile.cpp File Reference	573
11.64LISACODE-NoiseFile.h File Reference	574
11.65LISACODE-NoiseFilter.cpp File Reference	575
11.66LISACODE-NoiseFilter.h File Reference	576
11.67LISACODE-NoiseFShape.cpp File Reference	577
11.68LISACODE-NoiseFShape.h File Reference	578
11.69LISACODE-NoiseOof.cpp File Reference	579
11.70LISACODE-NoiseOof.h File Reference	580
11.71LISACODE-NoiseTwoFilter.cpp File Reference	581
11.72LISACODE-NoiseTwoFilter.h File Reference	582
11.73LISACODE-NoiseWhite.cpp File Reference	583
11.74LISACODE-NoiseWhite.h File Reference	584
11.75LISACODE-PhoDetPhaMet.cpp File Reference	585
11.76LISACODE-PhoDetPhaMet.h File Reference	586
11.77LISACODE-PhysicConstants.h File Reference	587
11.78LISACODE-Serie.cpp File Reference	594
11.79LISACODE-Serie.h File Reference	595
11.80LISACODE-TDI.cpp File Reference	596
11.81LISACODE-TDI.h File Reference	597
11.82LISACODE-TDI_InterData.cpp File Reference	598
11.83LISACODE-TDI_InterData.h File Reference	599

11.84LISACODE-TDIApply.cpp File Reference	600
11.85LISACODE-TDITools.cpp File Reference	601
11.86LISACODE-TDITools.h File Reference	602
11.87LISACODE-TrFctGW.cpp File Reference	603
11.88LISACODE-TrFctGW.h File Reference	604
11.89LISACODE-USOClock.cpp File Reference	605
11.90LISACODE-USOClock.h File Reference	606
11.91LISACODE-Vect.cpp File Reference	607
11.92LISACODE-Vect.h File Reference	609
11.93randlib.c File Reference	610
11.94randlib.h File Reference	616
12 LISACode Page Documentation	621
12.1 Introduction	621
12.2 A description of the Code	623
12.3 LISACode parameters	626
12.4 Bibliography	627
12.5 Todo List	628

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:

Noise (directory Bruits)	17
Detector (directory Dectecteur)	18
Input data (directory Input_data)	19
Mathematical Tools (directory Outils_Math)	33
Couple	20
Elliptic Filter	21
Filter	31
Matrix	32
Angles handling	34
Serie	35
Vector	36
Gravitational waves (directory Ondes_Gravit)	37
Background (directory Background)	38
USO clock (directory USO_Temps)	39
TDI handling (directory TDI)	40
TDI	41
TDI_InterData	42
TDITools	43
Geometry (directory Orbitographie)	44
Main (directory Main)	45
Memory (directory Memoire)	47

Chapter 3

LISACode Namespace Index

3.1 LISACode Namespace List

Here is a list of all namespaces with brief descriptions:

std	49
---------------	----

Chapter 4

LISACode Hierarchical Index

4.1 LISACode Class Hierarchy

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

Background	51
BackgroundGalactic	54
ConfigSim	60
Couple	96
ezxml	99
ezxml_root	102
Filter	104
Geometry	110
GeometryAnalytic	123
GeometryFile	139
GeometryMLDC	154
GW	170
GWBinary	178
GWCusp	189
GWFastSpinBBH	199
GWFile	229
GWMono	240
GWNew	250
GWNewton2	260
GWPeriGate	288
GWSto	297
LISA	307
Mat	315
MathUtils	318
Memory	319
MemoryReadDisk	325
MemoryWriteDisk	333
Noise	342
NoiseFile	351
NoiseFilter	362
NoiseFShape	374
NoiseOof	386

NoiseTwoFilter	399
NoiseWhite	410
NoiseSpec	397
PhoDetPhaMet	421
QuadCell	435
Serie	437
SerieC	446
TDI	452
TDI_InterData	462
TDITools	469
TrFctGW	472
USOClock	476
Vect	481

Chapter 5

LISACode Class Index

5.1 LISACode Class List

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

Background (Background signal received by phasemeters is described in this class)	51
BackgroundGalactic (Background Galactic signal received by phasemeters is described in this class)	54
ConfigSim (Class to configure LISA simulation, that is, LISACode execution)	60
Couple (Couple management class)	96
ezxml	99
ezxml_root	102
Filter (Filter management class)	104
Geometry (Orbit geometry class)	110
GeometryAnalytic (Compute spacecraft position from analytical formulations of the article : S)	123
GeometryFile (Compute spacecraft position from :)	139
GeometryMLDC (Compute spacecraft position from :)	154
GW (Gravitational Waves parameters are described in this class)	170
GWBinary (Gravitational Waves parameters for a monochromatic binary system are defined in this class)	178
GWCusp (Gravitational Waves instantaneous parameters h_+ and h_\times are described in this class)	189
GWFastSpinBBH (Gravitational Waves parameters for a spinning black holes binary system are defined in this class)	199
GWFile (Gravitational Waves file management)	229
GWMono (Gravitational Waves instantaneous parameters h_{plus} and h_{cross} are described in this class)	240
GWNew (Gravitational Waves instantaneous parameters h_{plus} and h_{cross} are described in this class)	250
GNewton2 (Gravitational Waves binary system parameters computation)	260
GWPeriGate (Gravitational Waves periodic gate signal)	288
GWSto (Gravitational Waves instantaneous parameters h_{plus} and h_{cross} are described in this class)	297
LISA (This class contains and manages all the elements necessary to LISA satellites simulation)	307
Mat ((3x3) matrix management class)	315
MathUtils (Angle conversion class)	318
Memory (Memory management class)	319
MemoryReadDisk (Class to manage disk reading. This class (module) reads data in file and stores them)	325

MemoryWriteDisk (Class to manage disk writting. This class (module) manages and stores data supplied to its in a temporary memory and in a file. Index of first series is 0)	333
Noise (Noise base class)	342
NoiseFile (Noise derived class to treat files with noise data)	351
NoiseFilter (Noise derived class to treat noise filters)	362
NoiseFShape (Noise derived class to treat noise filters)	374
NoiseOof (Noise derived class to treat noise filters)	386
NoiseSpec (Noise specification structure)	397
NoiseTwoFilter (Noise derived class to treat noise filters)	399
NoiseWhite (Noise derived class to treat white noise)	410
PhoDetPhaMet (Phasemeter photodiode class)	421
QuadCell (Elliptic cell structure)	435
Serie (Serie interpolation class)	437
SerieC (Complex serie interpolation class)	446
TDI (Time Delay Interferometry combinaison class)	452
TDI_InterData (Time Delay Interferometry interpolated signal class)	462
TDITools (Time Delay Interferometry tools class)	469
TrFctGW (Gravitational Waves Transfer Function class)	472
USOClock (Ultra Stable Oscillator based satellite time is defined in this class)	476
Vect (3 components vector management class)	481

Chapter 6

LISACode File Index

6.1 LISACode File List

Here is a list of all files with brief descriptions:

com.c	485
Doxygen.bibliography	490
ezxml.c	491
ezxml.h	500
linpack.c	508
LISACODE-Background.cpp	509
LISACODE-Background.h	510
LISACODE-BackgroundGalactic.cpp	511
LISACODE-BackgroundGalactic.h	512
LISACODE-ConfigSim.cpp	513
LISACODE-ConfigSim.h	514
LISACODE-Couple.cpp	516
LISACODE-Couple.h	518
LISACODE-DnonGW.cpp	519
LISACODE-EllipticFilter.cpp	520
LISACODE-EllipticFilter.h	522
LISACODE-Filter.cpp	524
LISACODE-Filter.h	525
LISACODE-Geometry.cpp	526
LISACODE-Geometry.h	527
LISACODE-GeometryAnalytic.cpp	528
LISACODE-GeometryAnalytic.h	529
LISACODE-GeometryFile.cpp	530
LISACODE-GeometryFile.h	531
LISACODE-GeometryMLDC.cpp	532
LISACODE-GeometryMLDC.h	533
LISACODE-GW.cpp	534
LISACODE-GW.h	535
LISACODE-GWBinary.cpp	536
LISACODE-GWBinary.h	537
LISACODE-GWCusp.cpp	538
LISACODE-GWCusp.h	539
LISACODE-GWFastSpinBBH.cpp	540

LISACODE-GWFastSpinBBH.h	541
LISACODE-GWFile.cpp	542
LISACODE-GWFile.h	543
LISACODE-GWMono.cpp	544
LISACODE-GWMono.h	545
LISACODE-GWNew.cpp	546
LISACODE-GWNew.h	547
LISACODE-GWNewton2.cpp	548
LISACODE-GWNewton2.h	549
LISACODE-GWPeriGate.cpp	550
LISACODE-GWPeriGate.h	551
LISACODE-GWSto.cpp	552
LISACODE-GWSto.h	553
LISACODE-LISA.cpp	554
LISACODE-LISA.h	555
LISACODE-LISACode.cpp	556
LISACODE-LISACode.ep.cpp	558
LISACODE-LISAConstants.h (Physical constants of LISA instrument)	559
LISACODE-Mat.cpp	562
LISACODE-Mat.h	563
LISACODE-MathUtils.h	564
LISACODE-Memory.cpp	565
LISACODE-Memory.h	566
LISACODE-MemoryReadDisk.cpp	567
LISACODE-MemoryReadDisk.h	568
LISACODE-MemoryWriteDisk.cpp	569
LISACODE-MemoryWriteDisk.h	570
LISACODE-Noise.cpp	571
LISACODE-Noise.h	572
LISACODE-NoiseFile.cpp	573
LISACODE-NoiseFile.h	574
LISACODE-NoiseFilter.cpp	575
LISACODE-NoiseFilter.h	576
LISACODE-NoiseFShape.cpp	577
LISACODE-NoiseFShape.h	578
LISACODE-NoiseOof.cpp	579
LISACODE-NoiseOof.h	580
LISACODE-NoiseTwoFilter.cpp	581
LISACODE-NoiseTwoFilter.h	582
LISACODE-NoiseWhite.cpp	583
LISACODE-NoiseWhite.h	584
LISACODE-PhoDetPhaMet.cpp	585
LISACODE-PhoDetPhaMet.h	586
LISACODE-PhysicConstants.h (Physical constants, reference values and unit conversions)	587
LISACODE-Serie.cpp	594
LISACODE-Serie.h	595
LISACODE-TDI.cpp	596
LISACODE-TDI.h	597
LISACODE-TDI_InterData.cpp	598
LISACODE-TDI_InterData.h	599
LISACODE-TDIApply.cpp	600
LISACODE-TDITools.cpp	601
LISACODE-TDITools.h	602
LISACODE-TrFctGW.cpp	603

LISACODE-TrFctGW.h	604
LISACODE-USOClock.cpp	605
LISACODE-USOClock.h	606
LISACODE-Vect.cpp	607
LISACODE-Vect.h	609
randlib.c	610
randlib.h	616

Chapter 7

LISACode Page Index

7.1 LISACode Related Pages

Here is a list of all related documentation pages:

Introduction	621
A description of the Code	623
LISACode parameters	626
Bibliography	627
Todo List	628

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 [NoiseFShape](#)
Noise derived class to treat noise filters.
- class [NoiseOof](#)
Noise derived class to treat noise filters.
- class [NoiseTwoFilter](#)
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 > **Im** (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 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 |I/D(w)|, where D(w) is the denominator of an elliptic cell (type [QuadCell](#)).
- double [KmQuadCell](#) ([QuadCell](#) Cell)
Returns max|Ell(w)|, where Ell(w) is the frequency response of an elliptic 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.

$$\begin{aligned} \text{returned value} = & 1.38629436112 + 0.09666344259 \cdot x + 0.03590092383 \cdot x^2 + 0.0374256371 \cdot x^3 + 0.01451196212 \cdot x^4 \\ & -alog(x) \cdot (0.5 + 0.12498593597 \cdot x + 0.068802485763 \cdot x^2 + 0.03328355346 \cdot x^3 + 0.004417870122 \cdot x^4) \end{aligned}$$

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) :

$$\begin{aligned} u_{FilterCellsOut[i]} &= (0, 0) \\ CellsCoef[i][0] &= b1_{FilterCells[i]} \\ CellsCoef[i][1] &= b2_{FilterCells[i]} \\ CellsCoef[i][2] &= a0_{FilterCells[i]} \\ CellsCoef[i][3] &= a0_{FilterCells[i]} \cdot a1_{FilterCells[i]} \\ CellsCoef[i][4] &= a0_{FilterCells[i]} \end{aligned}$$

First cell is rescaled :

$$\begin{aligned} CellsCoef[0][2] &= ag \cdot a0_{FilterCells[i]} \\ CellsCoef[0][3] &= ag \cdot a0_{FilterCells[i]} \cdot a1_{FilterCells[i]} \\ CellsCoef[0][4] &= ag \cdot a0_{FilterCells[i]} \end{aligned}$$

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) :

$$\begin{aligned} ac_i &= \frac{1}{a1_{Cell[i]}} - (b1_{Cell[i]} \cdot \frac{1 + b2_{Cell[i]}}{2 \cdot b2_{Cell[i}}}) \cdot Hm_{i+1} \\ n_i &= ceil(\frac{\log(ac_i)}{\log(2) + \frac{1}{2}}) \\ a0_{Cell[i]} &= 2^{n_i} \end{aligned}$$

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 ell (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

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

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

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

- for $i = 0, \dots, N - 1$

$$xmag = 2 \cdot i \cdot \frac{ak1}{2 \cdot N}$$

$$zeros[i] = -ak3 + I \cdot xmag$$

$$poles[i] = U_0 + I \cdot xmag$$

- for $i = 0, \dots, 2 \cdot N - 1$

$$Q = real(zeros[mod(i, N)])$$

$$R = imag(zeros[mod(i, N)])$$

$a1 = sn(Q, dkp, ak3, ak1)$ using sn function

$b1 = sn(R, dk, ak1, ak3)$ using sn function

$$\sigma = \begin{cases} 0 & \text{if } i \leq N \\ a1 \cdot \sqrt{(1 - a1^2) * (1 - b1^2)} \cdot \frac{dn}{de} & \text{else} \end{cases}$$

$$dn = \sqrt{1 - (dk \cdot b1)^2}$$

$$de = 1 - (a1 \cdot dn)^2$$

$$\omega = b1 \cdot \frac{\sqrt{(1 - (dkp \cdot a1)^2)}}{de}$$

$$C[i] = -2 \cdot \sigma \cdot \omega_c$$

$$D[i] = (\sigma^2 + \omega^2) \cdot \omega_c^2$$

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

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

$$\begin{cases} \text{if } i \leq N & zeros[i] = \sigma + I \cdot \omega \\ \text{else} & poles[i] = \sigma + I \cdot \omega \end{cases}$$

- for $i = 2 \cdot N - 1, \dots, 0$

$$\begin{cases} \text{if } i \leq N - 1 & (X, Y) = (real(zeros[i]), imag(zeros[i])) \\ \text{else} & (X, Y) = (real(poles[i]), imag(poles[i])) \end{cases}$$

$$Re = \frac{1 - X^2 - Y^2}{(1 - X)^2 + Y^2}$$

$$V = \frac{2 \cdot Y}{(1 - X)^2 + Y^2}$$

$$c1 = -2 \cdot Re$$

$$d1 = Re^2 + V^2$$

$$\begin{cases} \text{if } i \leq N - 1 & \begin{cases} zeros[i] = Re + I \cdot V \\ CoefB[i] = c1 \\ CoefA[i] = d1 \\ poles[i - N] = Re + I \cdot V \\ CoefD[i - N] = c1 \\ CoefC[i - N] = d1 \end{cases} \\ \text{else} & \end{cases}$$

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 elliptic 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> Im (0, 1) [static]

Pure imaginary=(0,1).

8.5.2.13 double KmQuadCell ([QuadCell Cell](#))

Returns $\max|\text{Ell}(w)|$, where $\text{Ell}(w)$ is the frequency response of an elliptic cell (type [QuadCell](#)).

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

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 :

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

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

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

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

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](#)
- [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 30 of file LISACODE-MathUtils.h.

Referenced by Filter::getDepth(), ConfigSim::gettTDIShift(), PhoDetPhaMet::init(), and NoiseTwo-Filter::loadNoise().

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

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

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

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

8.10 Serie

Classes

- class [Serie](#)

Serie interpolation class.

- class [SerieC](#)

complex serie interpolation class.

Enumerations

- enum [INTERP](#) {
 [TRU](#), [LIN](#), [CUB](#), [LAG](#),
 [SIN](#) }

Interpolation type.

8.10.1 Enumeration Type Documentation

8.10.1.1 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().

8.11 Vector

Classes

- class [Vect](#)

3 components vector management class.

8.12 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 [GWCusp](#)

Gravitational Waves instantaneous parameters h_+ and h_\times are described in this class.

- class [GWFastSpinBBH](#)

Gravitational Waves parameters for a spinning black holes 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 [GWNew](#)

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.

- class [GWSto](#)

Gravitational Waves instantaneous parameters h_{plus} and h_{cross} are described in this class.

8.13 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.

8.14 USO clock (directory USO_Temps)

Classes

- class [USOClock](#)

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

8.15 TDI handling (directory TDI)

Modules

- [TDI](#)
- [TDI_InterData](#)
- [TDITools](#)

8.16 TDI

Classes

- class [TDI](#)

Time Delay Interferometry combinaison class.

8.17 TDI_InterData

Classes

- class [TDI_InterData](#)

Time Delay Interferometry interpolated signal class.

8.18 TDITools

Classes

- class [TDITools](#)

Time Delay Interferometry tools class.

8.19 Geometry (directory Orbitographie)

Classes

- class [Geometry](#)

Orbit geometry class.

- class [GeometryAnalytic](#)

Compute spacecraft position from analytical formulations of the article : S.

- class [GeometryFile](#)

Compute spacecraft position from :.

- class [GeometryMLDC](#)

Compute spacecraft position from :.

8.20 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 $t = 0, \dots, tMemTDI + tTDIShift$ 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.

.

8.20.1 Function Documentation

8.20.1.1 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 $t = 0, \dots, tMemTDI + tTDIShift$ 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 36 of file LISACODE-DnonGW.cpp.

References TrFctGW::deltanu(), ConfigSim::getGeometry(), ConfigSim::getGWs(), ConfigSim::gettDisplay(), ConfigSim::gettMax(), ConfigSim::gettStepMes(), Geometry::gposition(), LCVersion, Vect::p, Geometry::tdelay(), and Geometry::VectNormal().

8.21 Memory (directory Memoire)

Classes

- class [Memory](#)

Memory management class.

- class [MemoryReadDisk](#)

Class to manage disk reading. This class (module) reads data in file and stores them.

- class [MemoryWriteDisk](#)

Class to manage disk writting. This class (module) manages and stores data supplied to its in a temporary memory and in a file. Index of first series is 0.

Chapter 9

LISACode Namespace Documentation

9.1 std Namespace Reference

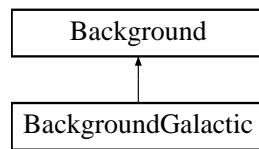
Chapter 10

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 δt_{anu} .

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 δt_{anu} .

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\(\)](#).

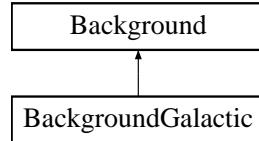
The documentation for this class was generated from the following files:

- [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:

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

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]

$$\begin{aligned} \text{returned value} &= \frac{\text{TimeList}[iRead + 1] - tGeo}{\text{TimeList}[iRead + 1] - \text{TimeList}[iRead]} \cdot \text{tmp_Sig_i}[iSC + 3 \cdot \text{IndirSens} - 1] \\ &+ \frac{tGeo - \text{TimeList}[iRead]}{\text{TimeList}[iRead + 1] - \text{TimeList}[iRead]} \cdot \text{tmp_Sig_ip1}[iSC + 3 \cdot \text{IndirSens} - 1] \end{aligned}$$

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 94 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.
- **int getGlobalRandomSeed ()**
It returns the value of GlobalRandomSeed attribute.
- **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 getDisplay ()**
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 getOrbType ()`

It returns the value of `OrbType` attribute.

- `int getOrbOrder ()`

It returns the value of `OrbOrder` attribute.

- `int getOrbApprox ()`

It returns the value of `OrbApprox` attribute.

- `void getGeometry (Geometry *&SCPos)`

Returns `FileNameSig` corresponding to `iSC` input.

- `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.

- `bool getInternalPhasemeters ()`

It returns `InternalPhasemeters` 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 * getSystemEncoding ()`
It returns `SystemEncoding`, `BigEndian` or `LittleEndian`.
- `char * getFileNameDelays ()`
It returns `FileNameDelays` attribute.
- `char * getFileNamePositions ()`
It returns `FileNamePositions` attribute.
- `int getFileEncodingSig (int iSC)`
Returns `FileEncodingSig` corresponding to `iSC` input.
- `int getFileEncodingTDI ()`
It returns `FileEncodingTDI` attribute.
- `int getFileEncodingDelays ()`
It returns `FileEncodingDelays` attribute.
- `int getFileEncodingPositions ()`
It returns `#FileEncodingPositions` 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.
- `vector< double > getGenTDIPacksFact (int iGen)`
Returns `TDIsPackFact` corresponding to `iGen` input.
- `int getNbMaxDelays ()`
It returns `NbMaxDelays` attribute plus 1.

- `char * getXmlOutputFile ()`
- `string getAuthor ()`
- `string getGenerationDate ()`
- `string getGenerationType ()`
- `string getTDIPParamName ()`
- `string getTDIPParamNameType ()`
- `string getSC1ParamName ()`
- `string getSC2ParamName ()`
- `string getSC3ParamName ()`
- `string getSimulator ()`
- `double getTimeOffset ()`
- `char * getFileNameSigSC1 ()`
- `char * getFileNameSigSC2 ()`
- `char * getFileNameSigSC3 ()`
- `void ReadFile ()`

Read the configuration file.

- `void ReadASCIIFile ()`

Read the ASCII configuration file.

- `void ReadXMLFile ()`

Read the ASCII configuration file.

- `void CreateXmlOutputFile ()`

Creates Xml Output File as header of binary file.

- `void AddTimeSeriesInXMLOutput (ofstream *FichXML, string ind1, string ObsDescr, double t-Offset, int NRec, int DataFileEncoding, char *DataFileName)`

Write in XML output file the bloc corresponding to an output data file.

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

Return unit.

- `double gXMLTime (ezxml_t param)`

Return a time value read in XML file. Unit is Second.

- `double gXMLAngle (ezxml_t param)`

Return an angle value read in XML file. Unit is Radian.

- `double gXMLFrequency (ezxml_t param)`

Return an frequency value read in XML file. Unit is Hertz.

- `double gXMLAstroMass (ezxml_t param)`

Return an astronomic mass value read in XML file. Unit is SolarMass.

- `double gXMLAstroDistance (ezxml_t param)`

Return an astronomic distance value read in XML file. Unit is KiloParsec.

- `char * gXMLWord (ezxml_t param)`

Return the first word read in bloc XML file.

- double `gXMLdouble (ezxml_t param)`
Return the real value read in bloc XML file.
- string `gXMLstring (ezxml_t param)`
Return the string read in bloc XML file.
- int `gXMLint (ezxml_t param)`
Return the integer in read in bloc XML file.
- char * `gXMLTimeSeries (ezxml_t series, int &type, int &encoding, int &length, int &record)`
Return timeseries parameters for an input file.
- void `NoisePlace (NoiseSpec tmp_noise, int iSC, int IndDir, int InstrumentIndex)`
Place a noise in the noises' list.
- void `NoisesCreation ()`
Creation of noise's object.
- double `tMaxDelay ()`
Maximal time travel for one delay.
- double `tMinDelay ()`
Minimal time travel for one delay.
- double `tMemNecInterpTDI ()`
*Return the time the memory time during which data must be saved for apply **TDI** interpolation.*
- double `gettTDIShift ()`
*Offset between the start of data reading and the start of **TDI** application.*
- double `gettMemPhasemeters ()`
Required time storage in phasemeters.
- double `gettMemTDI ()`
*Reauired time storage in **TDI**.*
- double `gettStartPhasemeters ()`
Starting time for phasemeters.
- bool `getNoNoise ()`
Return true if there are no noises.
- bool `UseInternalPhasemeter ()`
Return true if internal phasemeter must be used.
- int `testbyteorder ()`
- char * `uppercase (const char *)`
Convert word in uppercase word.

- void **majuscule** (char &)
Convert letter in lower case in uppercase letter.
- string **upS** (string Str)
- bool **scmp** (const string str1, const string str2)
- char * **stripcopy** (const char *orig)
Makes a copy of the string "orig", stripping all whitespace-type characters.
- bool **FindTDIName** (const char *generatorname, vector< int > &tmp_TDIPacks, vector< double > &tmp_TDIPacksFact)
Find if generatorname is known.

Private Attributes

- char * **ConfigFileName**
File name of simulation parameters. It is called configuration file.
- char * **SystemEncoding**
- int **SystemEncoding_int**
- string **Author**
- string **GenerationDate**
- string **GenerationType**
- string **TDIParamName**
- string **TDIParamNameType**
- string **SC1ParamName**
- string **SC2ParamName**
- string **SC3ParamName**
- string **Simulator**
- double **TimeOffset**
- char **XmlOutputFile** [256]
- ofstream **FichXML**
- int **GlobalRandomSeed**
- int **Endian**
- 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.*
- bool **BigEndian**
- **INTERP TDIIInterp**
TDI interpolator type.
- double **TDIIInterpUtilVal**
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 **OrbType**
Satellites motion.
- int **OrbOrder**
Order for time propagation computing.
- int **OrbApprox**
Orbits approximation.
- 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.*
- bool **InternalPhasemeters**
- 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 spacecraft 1 Signal.
- `char FileNameSigSC2 [256]`
FileName for spacecraft 2 Signal.
- `char FileNameSigSC3 [256]`
FileName for spacecraft 3 Signal.
- `char FileNameTDI [256]`
FileName for TDI generators.
- `char FileNameDelays [256]`
FileName for Delays.
- `char FileNamePositions [256]`
FileName for Positions.
- `int FileEncodingSC1`
- `int FileEncodingSC2`
- `int FileEncodingSC3`
- `int FileEncodingTDI`
- `int FileEncodingDelays`
- `int FileEncodingPos`
- `vector< string > TDIsName`
Name of TDI.
- `vector< vector< int > > TDIsPacks`
Vector of TDI data.
- `vector< vector< double > > TDIsPacksFact`
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 21 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 37 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 54 of file LISACODE-ConfigSim.cpp.

10.3.3 Member Function Documentation

10.3.3.1 void ConfigSim::AddTimeSeriesInXMLOutput (ofstream * *FichXML*, string *ind1*, string *ObsDescr*, double *tOffset*, int *NRec*, int *DataFileEncoding*, char * *DataFileName*)

Write in XML output file the bloc corresponding to an output data file.

Definition at line 2902 of file LISACODE-ConfigSim.cpp.

References [getSystemEncoding\(\)](#), [gettMax\(\)](#), and [gettStepMes\(\)](#).

Referenced by [CreateXmlOutputFile\(\)](#).

10.3.3.2 void ConfigSim::CreateXmlOutputFile ()

Creates Xml Output File as header of binary file.

Definition at line 2829 of file LISACODE-ConfigSim.cpp.

References [AddTimeSeriesInXMLOutput\(\)](#), *FichXML*, [getAuthor\(\)](#), [getFileEncodingSig\(\)](#), [getFileEncodingTDI\(\)](#), [getFileNameSig\(\)](#), [getFileNameTDI\(\)](#), [getSimulator\(\)](#), [gettStartPhasemeters\(\)](#), [getXmlOutputFile\(\)](#), *TDIsName*, [MathUtils::TimeISO8601\(\)](#), and [UseInternalPhasemeter\(\)](#).

Referenced by [ReadXMLFile\(\)](#).

10.3.3.3 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
- TDIIinterp = LAG
- TDIIinterpUtilVal = 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 measurement : PhaMetFilterParam[2]= 0.1
 - high transition frequency in factor of frequency of measurement : 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 101 of file LISACODE-ConfigSim.cpp.

References Armlength, ConfigFileName, Endian, FileEncodingDelays, FileEncodingPos, FileEncodingSC1, FileEncodingSC2, FileEncodingSC3, FileEncodingTDI, FileNameDelays, FileNamePositions, FileNameSigSC1, FileNameSigSC2, FileNameSigSC3, FileNameTDI, GlobalRandomSeed, GWB, InternalPhasemeters, L0_m_default, LAG, LaserPower, LaserPower_W_default, NbMaxDelays, Noises, NoisesData, OrbApprox, OrbInitRot, OrbOrder, OrbStartTime, OrbType, PhaMetFilterON, PhaMetFilterParam, Simulator, tDeltaTDIDelay, TDIDelayApprox, TDIIterp, TDIIterpUtilVal, tDisplay, tMax, tMemNoiseFirst, tMemNoiseLast, tMemSig, tStepMes, tStepPhy, USOs, and XmlOutputFile.

Referenced by ConfigSim().

10.3.3.4 bool ConfigSim::FindTDIName (const char * generatorname, vector< int > & tmp_TDIPacks, vector< double > & tmp_TDIPacksFact)

Find if generatorname is known.

Definition at line 3622 of file LISACODE-ConfigSim.cpp.

References NbMaxDelays.

Referenced by ReadASCIIFile(), and ReadXMLFile().

10.3.3.5 double ConfigSim::getArmlength () [inline]

It returns the value of [Armlength](#) attribute.

Definition at line 347 of file LISACODE-ConfigSim.h.

References Armlength.

10.3.3.6 string ConfigSim::getAuthor () [inline]

Definition at line 410 of file LISACODE-ConfigSim.h.

References Author.

Referenced by CreateXmlOutputFile().

10.3.3.7 int ConfigSim::getFileEncodingDelays () [inline]

It returns [FileEncodingDelays](#) attribute.

Definition at line 390 of file LISACODE-ConfigSim.h.

References FileEncodingDelays.

Referenced by main().

10.3.3.8 int ConfigSim::getFileEncodingPositions () [inline]

It returns #FileEncodingPositions attribute.

Definition at line 392 of file LISACODE-ConfigSim.h.

References FileEncodingPos.

Referenced by main().

10.3.3.9 int ConfigSim::getFileEncodingSig (int iSC)

Returns FileEncodingSig corresponding to iSC input.

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

Definition at line 213 of file LISACODE-ConfigSim.cpp.

References FileEncodingSC1, FileEncodingSC2, and FileEncodingSC3.

Referenced by CreateXmlOutputFile(), and main().

10.3.3.10 int ConfigSim::getFileEncodingTDI () [inline]

It returns FileEncodingTDI attribute.

Definition at line 388 of file LISACODE-ConfigSim.h.

References FileEncodingTDI.

Referenced by CreateXmlOutputFile(), and main().

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

It returns FileNameDelays attribute.

Definition at line 382 of file LISACODE-ConfigSim.h.

References FileNameDelays.

Referenced by main().

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

It returns FileNamePositions attribute.

Definition at line 384 of file LISACODE-ConfigSim.h.

References FileNamePositions.

Referenced by main().

10.3.3.13 char * ConfigSim::getFileNameSig (int iSC)

Returns FileNameSig corresponding to iSC input.

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

Definition at line 188 of file LISACODE-ConfigSim.cpp.

References FileNameSigSC1, FileNameSigSC2, and FileNameSigSC3.

Referenced by CreateXmlOutputFile(), and main().

10.3.3.14 char* ConfigSim::getFileNameSigSC1 () [inline]

Definition at line 420 of file LISACODE-ConfigSim.h.

References FileNameSigSC1.

10.3.3.15 char* ConfigSim::getFileNameSigSC2 () [inline]

Definition at line 421 of file LISACODE-ConfigSim.h.

References FileNameSigSC2.

10.3.3.16 char* ConfigSim::getFileNameSigSC3 () [inline]

Definition at line 422 of file LISACODE-ConfigSim.h.

References FileNameSigSC3.

10.3.3.17 char* ConfigSim::getFileNameTDI () [inline]

It returns [FileNameTDI](#) attribute.

Definition at line 378 of file LISACODE-ConfigSim.h.

References FileNameTDI.

Referenced by CreateXmlOutputFile(), and main().

10.3.3.18 string ConfigSim::getGenerationDate () [inline]

Definition at line 411 of file LISACODE-ConfigSim.h.

References GenerationDate.

10.3.3.19 string ConfigSim::getGenerationType () [inline]

Definition at line 412 of file LISACODE-ConfigSim.h.

References GenerationType.

10.3.3.20 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 261 of file LISACODE-ConfigSim.cpp.

References TDIsPacks.

Referenced by main().

10.3.3.21 vector< double > ConfigSim::getGenTDIPacksFact (int iGen)

Returns TDIsPackFact corresponding to iGen input.

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

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

References TDIsPacksFact.

Referenced by main().

10.3.3.22 void ConfigSim::getGeometry (Geometry *& SPos)

Returns FileNameSig corresponding to iSC input.

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

Definition at line 162 of file LISACODE-ConfigSim.cpp.

References Armlength, OrbApprox, OrbInitRot, OrbOrder, OrbStartTime, OrbType, and tStepPhy.

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

10.3.3.23 int ConfigSim::getGlobalRandomSeed () [inline]

It returns the value of GlobalRandomSeed attribute.

Definition at line 325 of file LISACODE-ConfigSim.h.

References GlobalRandomSeed.

Referenced by main().

10.3.3.24 GW* ConfigSim::getGW () [inline]

It returns the first element of the vector GWS.

Definition at line 372 of file LISACODE-ConfigSim.h.

References GWS.

10.3.3.25 Background* ConfigSim::getGWB () [inline]

It returns GWB attribute.

Definition at line 396 of file LISACODE-ConfigSim.h.

References GWB.

Referenced by LISA::LISA().

10.3.3.26 vector<GW *>* ConfigSim::getGWS () [inline]

It returns a pointer to the GWS attribute.

Definition at line 374 of file LISACODE-ConfigSim.h.

References GWS.

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

10.3.3.27 bool ConfigSim::getInternalPhasemeters () [inline]

It returns [InternalPhasemeters](#) attribute.

Definition at line 370 of file LISACODE-ConfigSim.h.

References InternalPhasemeters.

10.3.3.28 double ConfigSim::getLaserPower () [inline]

It returns the laser power, that is the value of [LaserPower](#) attribute.

Definition at line 363 of file LISACODE-ConfigSim.h.

References LaserPower.

10.3.3.29 char * ConfigSim::getNameGenTDI (int iGen)

Returns NameGenTDI corresponding to iGen input.

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

Definition at line 240 of file LISACODE-ConfigSim.cpp.

References TDIsName, and TDIsPacks.

Referenced by main().

10.3.3.30 int ConfigSim::getNbMaxDelays () [inline]

It returns [NbMaxDelays](#) attribute plus 1.

Definition at line 408 of file LISACODE-ConfigSim.h.

References NbMaxDelays.

Referenced by gettMemTDI().

10.3.3.31 vector<[Noise](#) *> ConfigSim::getNoises () [inline]

It returns [Noises](#) attribute.

Definition at line 394 of file LISACODE-ConfigSim.h.

References Noises.

Referenced by LISA::LISA().

10.3.3.32 bool ConfigSim::getNoNoise ()

Return true if there are no noises.

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

Definition at line 3513 of file LISACODE-ConfigSim.cpp.

References Noises.

Referenced by main(), and UseInternalPhasemeter().

10.3.3.33 int ConfigSim::getOrbApprox () [inline]

It returns the value of [OrbApprox](#) attribute.

Definition at line 359 of file LISACODE-ConfigSim.h.

References OrbApprox.

10.3.3.34 double ConfigSim::getOrbInitRot () [inline]

It returns [OrbInitRot](#) attribute.

Definition at line 351 of file LISACODE-ConfigSim.h.

References OrbInitRot.

10.3.3.35 int ConfigSim::getOrbOrder () [inline]

It returns the value of [OrbOrder](#) attribute.

Definition at line 357 of file LISACODE-ConfigSim.h.

References OrbOrder.

10.3.3.36 double ConfigSim::getOrbStartTime () [inline]

It returns [OrbStartTime](#) attribute.

Definition at line 349 of file LISACODE-ConfigSim.h.

References OrbStartTime.

10.3.3.37 int ConfigSim::getOrbType () [inline]

It returns the value of [OrbType](#) attribute.

Definition at line 355 of file LISACODE-ConfigSim.h.

References OrbType.

10.3.3.38 bool ConfigSim::getPhaMetFilter () [inline]

It returns 1 if a filter is applied in phasemeters, that is the value of [PhaMetFilterON](#) attribute.

Definition at line 366 of file LISACODE-ConfigSim.h.

References PhaMetFilterON.

Referenced by LISA::LISA().

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

It returns [PhaMetFilterParam](#) attribute.

Definition at line 368 of file LISACODE-ConfigSim.h.

References [PhaMetFilterParam](#).

Referenced by [LISA::LISA\(\)](#).

10.3.3.40 string ConfigSim::getSC1ParamName () [inline]

Definition at line 415 of file LISACODE-ConfigSim.h.

References [SC1ParamName](#).

10.3.3.41 string ConfigSim::getSC2ParamName () [inline]

Definition at line 416 of file LISACODE-ConfigSim.h.

References [SC2ParamName](#).

10.3.3.42 string ConfigSim::getSC3ParamName () [inline]

Definition at line 417 of file LISACODE-ConfigSim.h.

References [SC3ParamName](#).

10.3.3.43 string ConfigSim::getSimulator () [inline]

Definition at line 418 of file LISACODE-ConfigSim.h.

References [Simulator](#).

Referenced by [CreateXmlOutputFile\(\)](#).

10.3.3.44 char* ConfigSim::getSystemEncoding () [inline]

It returns [SystemEncoding](#) , BigEndian or LittleEndian.

Definition at line 380 of file LISACODE-ConfigSim.h.

References [SystemEncoding](#).

Referenced by [AddTimeSeriesInXMLOutput\(\)](#), and [ReadFile\(\)](#).

10.3.3.45 double ConfigSim::gettDeltaTDIDelay () [inline]

It returns [tDeltaTDIDelay](#) attribute.

Definition at line 341 of file LISACODE-ConfigSim.h.

References [tDeltaTDIDelay](#).

Referenced by [main\(\)](#).

10.3.3.46 bool ConfigSim::getTDIDelayApprox () [inline]

It returns [TDIDelayApprox](#) attribute.

Definition at line 353 of file LISACODE-ConfigSim.h.

References TDIDelayApprox.

Referenced by main().

10.3.3.47 [INTERP](#) ConfigSim::getTDIInterp () [inline]

It returns [TDIInterp](#) attribute.

Definition at line 343 of file LISACODE-ConfigSim.h.

References INTERP, and TDIIInterp.

Referenced by main().

10.3.3.48 double ConfigSim::getTDIInterpUtilVal () [inline]

It returns [TDIInterpUtilVal](#) attribute.

Definition at line 345 of file LISACODE-ConfigSim.h.

References TDIIInterpUtilVal.

Referenced by main().

10.3.3.49 string ConfigSim::getTDIP paramName () [inline]

Definition at line 413 of file LISACODE-ConfigSim.h.

References TDIP paramName.

10.3.3.50 string ConfigSim::getTDIP paramNameType () [inline]

Definition at line 414 of file LISACODE-ConfigSim.h.

References TDIP paramNameType.

10.3.3.51 double ConfigSim::gettDisplay () [inline]

It returns is the value of [tDisplay](#) attribute.

Definition at line 339 of file LISACODE-ConfigSim.h.

References tDisplay.

Referenced by main().

10.3.3.52 double ConfigSim::getTimeOffset () [inline]

Definition at line 419 of file LISACODE-ConfigSim.h.

References TimeOffset.

10.3.3.53 double ConfigSim::gettMax () [inline]

It returns maximal simulation duration, that is the value of [tMax](#) attribute.

Definition at line 329 of file LISACODE-ConfigSim.h.

References [tMax](#).

Referenced by [AddTimeSeriesInXMLOutput\(\)](#), and [main\(\)](#).

10.3.3.54 double ConfigSim::gettMemNoiseFirst () [inline]

It returns [tMemNoiseFirst](#) attribute.

Definition at line 333 of file LISACODE-ConfigSim.h.

References [tMemNoiseFirst](#).

10.3.3.55 double ConfigSim::gettMemNoiseLast () [inline]

It returns [tMemNoiseLast](#) attribute.

Definition at line 335 of file LISACODE-ConfigSim.h.

References [tMemNoiseLast](#).

10.3.3.56 double ConfigSim::gettMemPhasemeters () [inline]

Required time storage in phasemeters.

Definition at line 473 of file LISACODE-ConfigSim.h.

References [gettTDIShift\(\)](#), [tMaxDelay\(\)](#), and [tMemNecInterpTDI\(\)](#).

Referenced by [gettStartPhasemeters\(\)](#), and [main\(\)](#).

10.3.3.57 double ConfigSim::gettMemSig () [inline]

It returns [tMemSig](#) attribute.

Definition at line 337 of file LISACODE-ConfigSim.h.

References [tMemSig](#).

Referenced by [LISA::LISA\(\)](#).

10.3.3.58 double ConfigSim::gettMemTDI () [inline]

Reauired time storage in [TDI](#).

Definition at line 475 of file LISACODE-ConfigSim.h.

References [getNbMaxDelays\(\)](#), [gettTDIShift\(\)](#), [tMaxDelay\(\)](#), and [tMemNecInterpTDI\(\)](#).

Referenced by [gettStartPhasemeters\(\)](#), and [main\(\)](#).

10.3.3.59 double ConfigSim::gettStartPhasemeters () [inline]

Starting time for phasemeters.

Definition at line 477 of file LISACODE-ConfigSim.h.

References `gettMemPhasemeters()`, `gettMemTDI()`, and `gettTDIShift()`.

Referenced by `CreateXmlOutputFile()`, and `main()`.

10.3.3.60 double ConfigSim::gettStepMes () [inline]

It returns `tStepMes` attribute.

Definition at line 331 of file LISACODE-ConfigSim.h.

References `tStepMes`.

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

10.3.3.61 double ConfigSim::gettStepPhy () [inline]

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

Definition at line 327 of file LISACODE-ConfigSim.h.

References `tStepPhy`.

Referenced by `LISA::LISA()`.

10.3.3.62 double ConfigSim::gettTDIShift () [inline]

Offset between the start of data reading and the start of `TDI` application.

Definition at line 471 of file LISACODE-ConfigSim.h.

References `MAX`, `tMemNecInterpTDI()`, and `tMinDelay()`.

Referenced by `gettMemPhasemeters()`, `gettMemTDI()`, `gettStartPhasemeters()`, and `main()`.

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

It returns `USOs` attribute.

Definition at line 398 of file LISACODE-ConfigSim.h.

References `USOs`.

Referenced by `LISA::LISA()`.

10.3.3.64 char* ConfigSim::getXmlOutputFile () [inline]

Definition at line 409 of file LISACODE-ConfigSim.h.

References `XmlOutputFile`.

Referenced by `CreateXmlOutputFile()`.

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

Return an angle value read in XML file. Unit is Radian.

Checks if unit is degree.

Definition at line 2983 of file LISACODE-ConfigSim.cpp.

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

Referenced by ReadXMLFile().

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

Return an astronomic distance value read in XML file. Unit is KiloParsec.

Checks if unit is Parsec.

Definition at line 3037 of file LISACODE-ConfigSim.cpp.

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

Referenced by ReadXMLFile().

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

Return an astronomic mass value read in XML file. Unit is SolarMass.

Checks if unit is SolarMass.

Definition at line 3021 of file LISACODE-ConfigSim.cpp.

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

Referenced by ReadXMLFile().

10.3.3.68 double ConfigSim::gXMLdouble (*ezxml_t param*)

Return the real value read in bloc XML file.

NO units

Definition at line 3087 of file LISACODE-ConfigSim.cpp.

References ezxml_attr(), ezxml_t, and ezxml_txt.

Referenced by ReadXMLFile().

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

Return an frequency value read in XML file. Unit is Hertz.

Checks if unit is MilliHertz.

Definition at line 3002 of file LISACODE-ConfigSim.cpp.

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

Referenced by ReadXMLFile().

10.3.3.70 int ConfigSim::gXMLInt (*ezxml_t param*)

Return the integer in read in bloc XML file.

NO units

Definition at line 3122 of file LISACODE-ConfigSim.cpp.

References ezxml_attr(), ezxml_t, and ezxml_txt.

Referenced by ReadXMLFile().

10.3.3.71 string ConfigSim::gXMLString (*ezxml_t param*)

Return the string read in bloc XML file.

NO units

Definition at line 3101 of file LISACODE-ConfigSim.cpp.

References ezxml_attr(), ezxml_t, and strcpy().

Referenced by ReadXMLFile().

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

Return a time value read in XML file. Unit is Second.

Checks if unit is second.

Definition at line 2966 of file LISACODE-ConfigSim.cpp.

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

Referenced by ReadASCIIFile(), and ReadXMLFile().

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

Return timeseries parameters for an input file.

Sets parameters :

- typedata get the type of record data : 0 for ASCII, 1 for Binary,LittleEndian and 2 for Binary,Big-Endian (extracted from "Stream" child of series input)
- 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 3140 of file LISACODE-ConfigSim.cpp.

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

Referenced by ReadASCIIFile(), and ReadXMLFile().

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

Return unit.

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

Definition at line 2950 of file LISACODE-ConfigSim.cpp.

Referenced by gXMLAngle(), gXMLAstroDistance(), gXMLAstroMass(), gXMLFrequency(), gXMLTime(), and ReadXMLFile().

10.3.3.75 char * ConfigSim::gXMLWord ([ezxml_t param](#))

Return the first word read in bloc XML file.

Definition at line 3054 of file LISACODE-ConfigSim.cpp.

References ezxml_t.

Referenced by ReadXMLFile().

10.3.3.76 void ConfigSim::majuscule (char &)

Convert letter in lower case in uppercase letter.

Definition at line 3560 of file LISACODE-ConfigSim.cpp.

Referenced by uppercase(), and upS().

10.3.3.77 int ConfigSim::NbGenTDI () [inline]

It returns TDIsName attribute.

Definition at line 400 of file LISACODE-ConfigSim.h.

References TDIsName.

Referenced by main().

10.3.3.78 void ConfigSim::NoisePlace ([NoiseSpec](#) *tmp_noise*, int *iSC*, int *IndDir*, int *InstrumentIndex*)

Place a noise in the noises' list.

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 3223 of file LISACODE-ConfigSim.cpp.

References NoisesData.

Referenced by ReadXMLFile().

10.3.3.79 void ConfigSim::NoisesCreation ()

Creation of noise's object.

Number of created noises is given by size of NoisesData.

Definition at line 3251 of file LISACODE-ConfigSim.cpp.

References Armlength, L0_m_default, LaserPower, LaserPower_W_default, Noises, NoisesData, NoiseSpec::NStr, NoiseSpec::NType, NoiseSpec::NVal0, NoiseSpec::NVal01, NoiseSpec::NVal1, NoiseSpec::NVal2, PRECISION, tMemNoiseFirst, tMemNoiseLast, tStepMes, and tStepPhy.

Referenced by ReadASCIIFile(), and ReadXMLFile().

10.3.3.80 void ConfigSim::ReadASCIIFile ()

Read the 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 353 of file LISACODE-ConfigSim.cpp.

References Armlength, ConfigFileName, MathUtils::deg2rad(), ezxml_attr(), ezxml_child(), ezxml_parse_file(), ezxml_t, FileEncodingDelays, FileEncodingPos, FileEncodingSC1, FileEncodingSC2, FileEncodingSC3, FileEncodingTDI, FileNameDelays, FileNamePositions, FileNameSigSC1, FileNameSigSC2, FileNameSigSC3, FileNameTDI, FindTDIName(), GWB, GWs, gXMLTime(), gXMLTimeSeries(), InternalPhasemeters, LAG, LaserPower, NbMaxDelays, ezxml::next, NoisesCreation(), NoisesData, NoiseSpec::NStr, NoiseSpec::NType, NoiseSpec::NVal0, NoiseSpec::NVal01, NoiseSpec::NVal1, NoiseSpec::NVal2, OrbApprox, OrbInitRot, OrbOrder, OrbStartTime, OrbType, PhaMetFilterON, PhaMetFilterParam, PRECISION, MathUtils::rad2deg(), scmp(), tDeltaTDIDelay, TDIDelayApprox, TDIInterp, TDIIInterpUtilVal, TDIsName, TDIsPacks, TDIsPacksFact, tDisplay, tMax, tMaxDelay(), tMemNoiseFirst, tMemNoiseLast, tMemSig, tStepMes, tStepPhy, and USOs.

Referenced by ReadFile().

10.3.3.81 void ConfigSim::ReadFile ()

Read the configuration file.

Calls ReadXMLFile or ReadASCIIFile depending on [ConfigFileName](#) type

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

References BigEndian, ConfigFileName, Endian, getSystemEncoding(), ReadASCIIFile(), ReadXMLFile(), SystemEncoding, and testbyteorder().

Referenced by ConfigSim().

10.3.3.82 void ConfigSim::ReadXMLFile ()

Read the ASCII configuration file.

Data are read :

- Time Data
- Interpolation Data
- Precision [TDI](#) Data
- Orbit Data
- Detector Data
- USO Data
- Record Data

Definition at line 1498 of file LISACODE-ConfigSim.cpp.

References Armlength, Author, c_SI, ConfigFileName, CreateXmlOutputFile(), ezxml_attr(), ezxml_child(), ezxml_free(), ezxml_parse_file(), ezxml_t, ezxml_txt, FileEncodingDelays, FileEncodingPos, FileEncodingSC1, FileEncodingSC2, FileEncodingSC3, FileEncodingTDI, FileNameDelays, FileNamePositions, FileNameSigSC1, FileNameSigSC2, FileNameSigSC3, FileNameTDI, FindTDIName(), GenerationDate, GenerationType, GlobalRandomSeed, GWs, gXMLAngle(), gXMLAstroDistance(), gXMLAstroMass(), gXMLdouble(), gXMLFrequency(), gXMLInt(), gXMLstring(), gXMLTime(), gXMLTimeSeries(), gXMLUnit(), gXMLWord(), InternalPhasemeters, LAG, LaserPower, NbMaxDelays, ezxml::next, NoisePlace(), NoisesCreation(), NoiseSpec::NTType, NoiseSpec::NVal0, NoiseSpec::NVal01, NoiseSpec::NVal1, NoiseSpec::NVal2, OrbApprox, OrbInitRot, OrbOrder, OrbStartTime, OrbType, PhaMetFilterON, PhaMetFilterParam, PRECISION, MathUtils::rad2deg(), Simulator, tDeltaTDIDelay, TDIDelayApprox, TDIIInterp, TDIIInterpUtilVal, TDIParamName, TDIsName, TDIsPacks, TDIsPacksFact, tDisplay, TimeOffset, tMax, tMaxDelay(), tMemNoiseFirst, tMemNoiseLast, tStepMes, tStepPhy, uppercase(), USOs, XmlOutputFile, and Yr_SI.

Referenced by ReadFile().

10.3.3.83 bool ConfigSim::scmp (const string str1, const string str2)

Definition at line 3581 of file LISACODE-ConfigSim.cpp.

References upS().

Referenced by ReadASCIIFile().

10.3.3.84 char * ConfigSim::stripcopy (const char * orig)

Makes a copy of the string "orig", stripping all whitespace-type characters.

Makes a copy of the string "orig", stripping all whitespace-type characters; the returned string is a private copy that must be deallocated with free().

Definition at line 3593 of file LISACODE-ConfigSim.cpp.

Referenced by gXMLstring().

10.3.3.85 int ConfigSim::testbyteorder ()

Definition at line 3538 of file LISACODE-ConfigSim.cpp.

Referenced by ReadFile().

10.3.3.86 double ConfigSim::tMaxDelay ()

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 3477 of file LISACODE-ConfigSim.cpp.

References ArmLength, c_SI, and tStepMes.

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

10.3.3.87 double ConfigSim::tMemNecInterpTDI ()

Return the time the 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 :

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

Definition at line 3499 of file LISACODE-ConfigSim.cpp.

References LAG, TDIIInterp, TDIIInterpUtilVal, and tStepMes.

Referenced by gettMemPhasemeters(), gettMemTDI(), gettTDIShift(), and main().

10.3.3.88 double ConfigSim::tMinDelay ()

Minimal time travel for one delay.

Minimal time travel for one delay is $tMinDelay = tStepMes \cdot ceil\left(\frac{4 \cdot ArmLength}{5 \cdot tStepMes \cdot C}\right)$.

Definition at line 3486 of file LISACODE-ConfigSim.cpp.

References ArmLength, c_SI, and tStepMes.

Referenced by gettTDIShift().

10.3.3.89 char * ConfigSim::uppercase (const char *)

Convert word in uppercase word.

Definition at line 3548 of file LISACODE-ConfigSim.cpp.

References majuscule().

Referenced by gXMLTimeSeries(), and ReadXMLFile().

10.3.3.90 string ConfigSim::upS (string Str)

Definition at line 3565 of file LISACODE-ConfigSim.cpp.

References majuscule().

Referenced by scmp().

10.3.3.91 bool ConfigSim::UseInternalPhasemeter ()

Return true if internal phasemeter must be used.

Definition at line 3525 of file LISACODE-ConfigSim.cpp.

References getNoNoise(), and InternalPhasemeters.

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

10.3.4 Member Data Documentation

10.3.4.1 double ConfigSim::Armlength [private]

Nominal [LISA](#) arm length.

Definition at line 156 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getArmlength(), getGeometry(), NoisesCreation(), ReadASCIIFile(), ReadXMLFile(), tMaxDelay(), and tMinDelay().

10.3.4.2 string ConfigSim::Author [private]

Definition at line 103 of file LISACODE-ConfigSim.h.

Referenced by getAuthor(), and ReadXMLFile().

10.3.4.3 bool ConfigSim::BigEndian [private]

Definition at line 147 of file LISACODE-ConfigSim.h.

Referenced by ReadFile().

10.3.4.4 char* ConfigSim::ConfigFileName [private]

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

Definition at line 100 of file LISACODE-ConfigSim.h.

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

10.3.4.5 int ConfigSim::Endian [private]

Definition at line 145 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), and ReadFile().

10.3.4.6 ofstream ConfigSim::FichXML [private]

Definition at line 107 of file LISACODE-ConfigSim.h.

Referenced by CreateXmlOutputFile().

10.3.4.7 int ConfigSim::FileEncodingDelays [private]

Definition at line 302 of file LISACODE-ConfigSim.h.

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

10.3.4.8 int ConfigSim::FileEncodingPos [private]

Definition at line 302 of file LISACODE-ConfigSim.h.

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

10.3.4.9 int ConfigSim::FileEncodingSC1 [private]

Definition at line 302 of file LISACODE-ConfigSim.h.

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

10.3.4.10 int ConfigSim::FileEncodingSC2 [private]

Definition at line 302 of file LISACODE-ConfigSim.h.

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

10.3.4.11 int ConfigSim::FileEncodingSC3 [private]

Definition at line 302 of file LISACODE-ConfigSim.h.

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

10.3.4.12 int ConfigSim::FileEncodingTDI [private]

Definition at line 302 of file LISACODE-ConfigSim.h.

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

10.3.4.13 ConfigSim::FileNameDelays [private]

FileName for Delays.

6 time delays between satellites are stored in this file.

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

10.3.4.14 ConfigSim::FileNamePositions [private]

FileName for Positions.

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

10.3.4.15 ConfigSim::FileNameSigSC1 [private]

FileName for spacecraft 1 Signal.

4 phasemeters data are stored in this file.

Referenced by DefaultConfig(), getFileNameSig(), getFileNameSigSC1(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.16 ConfigSim::FileNameSigSC2 [private]

FileName for spacecraft 2 Signal.

4 phasemeters data are stored in this file.

Referenced by DefaultConfig(), getFileNameSig(), getFileNameSigSC2(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.17 ConfigSim::FileNameSigSC3 [private]

FileName for spacecraft 3 Signal.

4 phasemeters data are stored in this file.

Referenced by DefaultConfig(), getFileNameSig(), getFileNameSigSC3(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.18 ConfigSim::FileNameTDI [private]

FileName for **TDI** generators.

All **TDI** generators data

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

10.3.4.19 string ConfigSim::GenerationDate [private]

Definition at line 103 of file LISACODE-ConfigSim.h.

Referenced by getGenerationDate(), and ReadXMLFile().

10.3.4.20 string ConfigSim::GenerationType [private]

Definition at line 103 of file LISACODE-ConfigSim.h.

Referenced by getGenerationType(), and ReadXMLFile().

10.3.4.21 int ConfigSim::GlobalRandomSeed [private]

Definition at line 144 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getGlobalRandomSeed(), and ReadXMLFile().

10.3.4.22 Background* ConfigSim::GWB [private]

Background signals

Definition at line 237 of file LISACODE-ConfigSim.h.

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

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

Gravitational Waves Parameters.

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

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

10.3.4.24 bool ConfigSim::InternalPhasemeters [private]

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

Referenced by DefaultConfig(), getInternalPhasemeters(), ReadASCIIFile(), ReadXMLFile(), and UseInternalPhasemeter().

10.3.4.25 double ConfigSim::LaserPower [private]

Laser Power in Watts.

It specifies the beam power at the laser output.

Definition at line 217 of file LISACODE-ConfigSim.h.

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

10.3.4.26 int ConfigSim::NbMaxDelays [private]

Maximum Delays Number.

Definition at line 310 of file LISACODE-ConfigSim.h.

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

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

satellites noise

Definition at line 241 of file LISACODE-ConfigSim.h.

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

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

noise specifications

Definition at line 239 of file LISACODE-ConfigSim.h.

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

10.3.4.29 int ConfigSim::OrbApprox [private]

Orbits approximation.

The possible order values are :

- 0 : static, no motion of satellites
- 1 : rigid.
- 2 : eccentric.

Definition at line 209 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getGeometry(), getOrbApprox(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.30 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 177 of file LISACODE-ConfigSim.h.

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

10.3.4.31 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 considering Sagnac effect.
- 2 : time propagation is computed using general relativity.

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

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

10.3.4.32 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 165 of file LISACODE-ConfigSim.h.

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

10.3.4.33 int ConfigSim::OrbType [private]

Satellites motion.

It indicates the type of orbits :

- 0 : Analytical LISACode standard orbits (from gr-qc_0410093).
- 1 : MLDC orbits (from PRD_71_022001).
- 2 : ESA orbits read in file 'Orbit_ESA.input.txt'.

Definition at line 189 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getGeometry(), getOrbType(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.34 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 223 of file LISACODE-ConfigSim.h.

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

10.3.4.35 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 measurement
- high transition frequency in factor of frequency of measurement

Definition at line 232 of file LISACODE-ConfigSim.h.

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

10.3.4.36 string ConfigSim::SC1ParamName [private]

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

Referenced by getSC1ParamName().

10.3.4.37 string ConfigSim::SC2ParamName [private]

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

Referenced by getSC2ParamName().

10.3.4.38 string ConfigSim::SC3ParamName [private]

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

Referenced by getSC3ParamName().

10.3.4.39 string ConfigSim::Simulator [private]

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

Referenced by DefaultConfig(), getSimulator(), and ReadXMLFile().

10.3.4.40 char* ConfigSim::SystemEncoding [private]

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

Referenced by getSystemEncoding(), and ReadFile().

10.3.4.41 int ConfigSim::SystemEncoding_int [private]

Definition at line 102 of file LISACODE-ConfigSim.h.

10.3.4.42 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().

10.3.4.43 bool ConfigSim::TDIDelayApprox [private]

Approximated [TDI](#) delays computation or not.

Definition at line 212 of file LISACODE-ConfigSim.h.

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

10.3.4.44 `INTERP ConfigSim::TDIInterp` [private]

`TDI` interpolator type.

Definition at line 150 of file LISACODE-ConfigSim.h.

Referenced by `DefaultConfig()`, `getTDIInterp()`, `ReadASCIIFile()`, `ReadXMLFile()`, and `tMemNecInterp-TDI()`.

10.3.4.45 `double ConfigSim::TDIInterpUtilVal` [private]

Value used for `TDI` interpolation.

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

Referenced by `DefaultConfig()`, `getTDIInterpUtilVal()`, `ReadASCIIFile()`, `ReadXMLFile()`, and `tMemNec-InterpTDI()`.

10.3.4.46 `string ConfigSim::TDIP paramName` [private]

Definition at line 103 of file LISACODE-ConfigSim.h.

Referenced by `getTDIP paramName()`, and `ReadXMLFile()`.

10.3.4.47 `string ConfigSim::TDIP paramNameType` [private]

Definition at line 103 of file LISACODE-ConfigSim.h.

Referenced by `getTDIP paramNameType()`.

10.3.4.48 `vector<string> ConfigSim::TDIsName` [private]

Name of `TDI`.

Definition at line 304 of file LISACODE-ConfigSim.h.

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

10.3.4.49 `vector<vector<int>> ConfigSim::TDIsPacks` [private]

Vector of `TDI` data.

Definition at line 306 of file LISACODE-ConfigSim.h.

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

10.3.4.50 `vector<vector<double>> ConfigSim::TDIsPacksFact` [private]

Vector of `TDI` data.

Definition at line 308 of file LISACODE-ConfigSim.h.

Referenced by `getGenTDIPacksFact()`, `ReadASCIIFile()`, and `ReadXMLFile()`.

10.3.4.51 ConfigSim::tDisplay [private]

Time step for screen display.

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

10.3.4.52 double ConfigSim::TimeOffset [private]

Definition at line 105 of file LISACODE-ConfigSim.h.

Referenced by getTimeOffset(), and ReadXMLFile().

10.3.4.53 ConfigSim::tMax [private]

It is the maximal simulation duration time.

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

10.3.4.54 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().

10.3.4.55 ConfigSim::tMemNoiseLast [private]

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

later noises are eliminated

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

10.3.4.56 ConfigSim::tMemSig [private]

Duration of satellite memory for signal storage.

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

10.3.4.57 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().

10.3.4.58 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(), getGeometry(), gettStepPhy(), NoisesCreation(), ReadASCIIFile(), and ReadXMLFile().

10.3.4.59 `vector<USOClock> ConfigSim::USOs` [private]

satellites USO time

Definition at line 243 of file LISACODE-ConfigSim.h.

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

10.3.4.60 `char ConfigSim::XmlOutputFile[256]` [private]

Definition at line 106 of file LISACODE-ConfigSim.h.

Referenced by DefaultConfig(), getXmlOutputFile(), and ReadXMLFile().

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

- [LISACODE-ConfigSim.h](#)
- [LISACODE-ConfigSim.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(), GWFile::Interpol(), operator *(), operator+(), operator-(), operator/(), GeometryAnalytic::position(), GWFile::ReadASCIIFile(), GWFile::ReadBinaryFile(), and GeometryAnalytic::velocity().

10.4.4.2 **Couple::y**

Second component.

Referenced by Couple(), GWFile::Interpol(), operator *(), operator+(), operator-(), operator/(), GeometryAnalytic::position(), GWFile::ReadBinaryFile(), and GeometryAnalytic::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()`, `ConfigSim::ReadASCIIFile()`, 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()`, and `ezxml_toxml_r()`.

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 \[EZXML_ERR\]](#)

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 \[EZXML_ERR\]](#)

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 NbData-Stabilization_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.
- **Filter operator= (Filter Model)**
- **void Copy (Filter Model)**
- **void init (vector< vector< double > > alpha_n, vector< vector< double > > beta_n, int NbData-Stabilization_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 to 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 32 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 51 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 65 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` = $\frac{100}{f_e}$
- `TmpData` = (alpha_n size + 1 null) vectors

Definition at line 92 of file LISACODE-Filter.cpp.

References `CalcEllipticFilter4LISACode()`, and `init()`.

10.7.2.5 Filter::~Filter ()

Destructor.

Definition at line 129 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 to 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 $i = 0, \dots, StartBin$

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

Then last TmpData array is copied into FilterData.

Definition at line 219 of file LISACODE-Filter.cpp.

References alpha, beta, and [TmpData](#).

Referenced by [NoiseTwoFilter::generNoise\(\)](#), [NoiseOof::generNoise\(\)](#), [NoiseFilter::generNoise\(\)](#), [PhoDetPhaMet::IntegrateSignal\(\)](#), [NoiseTwoFilter::loadNoise\(\)](#), [NoiseOof::loadNoise\(\)](#), and [NoiseFilter::loadNoise\(\)](#).

10.7.3.2 void Filter::Copy ([Filter Model](#))

Definition at line 143 of file LISACODE-Filter.cpp.

References alpha, beta, init(), and NbDataStab.

Referenced by [NoiseFilter::NoiseFilter\(\)](#), and [NoiseTwoFilter::NoiseTwoFilter\(\)](#).

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

Returns [alpha](#) attribute.

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

References alpha.

Referenced by [NoiseTwoFilter::getFilterAlpha\(\)](#), [NoiseOof::getFilterAlpha\(\)](#), and [NoiseFilter::getFilterAlpha\(\)](#).

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

Returns [beta](#) attribute.

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

References beta.

Referenced by [NoiseTwoFilter::getFilterBeta\(\)](#), [NoiseOof::getFilterBeta\(\)](#), and [NoiseFilter::getFilterBeta\(\)](#).

10.7.3.5 int Filter::getDepth ()

Gives maximum of `alpha` or `beta` attribute size.

Definition at line 190 of file LISACODE-Filter.cpp.

References alpha, beta, and MAX.

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

10.7.3.6 int Filter::getNbDataStab ()

Returns `NbDataStab` attribute.

Definition at line 201 of file LISACODE-Filter.cpp.

References NbDataStab.

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

10.7.3.7 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 160 of file LISACODE-Filter.cpp.

References alpha, beta, NbDataStab, and TmpData.

Referenced by `Copy()`, `Filter()`, `NoiseFilter::NoiseFilter()`, `NoiseOof::NoiseOof()`, and `NoiseTwoFilter::NoiseTwoFilter()`.

10.7.3.8 Filter Filter::operator= (*Filter Model*)

Definition at line 137 of file LISACODE-Filter.cpp.

References alpha, beta, and NbDataStab.

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()`, `Copy()`, `Filter()`, `getAlpha()`, `getDepth()`, `init()`, and `operator=()`.

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(), Copy(), Filter(), getBeta(), getDepth(), init(), and operator=().

10.7.4.3 int Filter::NbDataStab [protected]

Number of data for stabilization.

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

Referenced by Copy(), Filter(), getNbDataStab(), init(), and operator=().

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().

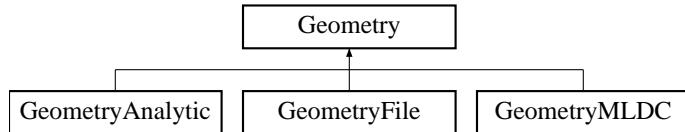
The documentation for this class was generated from the following files:

- [LISACODE-Filter.h](#)
- [LISACODE-Filter.cpp](#)

10.8 Geometry Class Reference

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

Inheritance diagram for Geometry::



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 \(double t0_n, double rot0_n, double L0m_n, int order_default_n, int move_n, double tStep\)](#)
Constructs an instance and initializes it with default values and t0_n and rot0_n inputs. Constructs an instance and initializes it with t0_n, rot0_n, L0m_n, order_default_n, move_n and tStep inputs.
- virtual [~Geometry \(\)](#)
Destructor.
- void [initGlobal \(double t0_n, double rot0_n, double L0m_n, int order_default_n, int move_n, double tStep\)](#)
Globla initialization.
- void [DispGeneralInfo \(\)](#)
Display general inforamtions.
- virtual void [DispInfo \(\)](#)
Display informations.
- double [getL0 \(\)](#)
Returns L0m attribute.
- double [gett0 \(\)](#)
Returns t0 attribute.
- void [sett0 \(double t0_n\)](#)
- double [gettRangeStorePos \(\)](#)
- double [gettRangeStoreDelay \(\)](#)

- void **settRangeStorePos** (double tRangeStorePos_n)
- void **settRangeStoreDelay** (double tRangeStoreDelay_n)
- virtual **Vect position** (int nb, double t)

Returns the position of the spacecraft in the barycentric frame for the time t (s) as argument and spacecraft number (1, 2 or 3).

- virtual **Vect velocity** (int nb, double t)

Returns the position of the spacecraft in the barycentric frame for the time t (s) as argument and spacecraft number (1, 2 or 3).

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

Get delay for specified order (1/2 -> order=1 and 1 -> order=2).

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

Get contribution of specified order (1/2 -> order=1 and 1 -> order=2) in delay.

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

Get 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)

Give direct value for position.

- 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 = $\frac{2\pi}{3}$.
- double **t0**
Initial time (seconds).
- double **rot0**
Initial rotation (radians).
- 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 29 of file LISACODE-Geometry.cpp.

References initGlobal(), and L0_m_default.

10.8.2.2 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 default values and t0_n and rot0_n inputs. 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 56 of file LISACODE-Geometry.cpp.

References initGlobal().

10.8.2.3 Geometry::~Geometry () [virtual]

Destructor.

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

10.8.3 Member Function Documentation

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

Get 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:

$$(\vec{v}_j - \vec{v}_i) \cdot \vec{n},$$

where :

r_i is computed using [position](#) method with rec and trec inputs,

v_i is computed using [velocity](#) method with rec and trec inputs, then divided by [c_SI](#),

r_j is computed using [position](#) method with em and trec inputs,

v_j is computed using [velocity](#) method with em and trec inputs, then divided by [c_SI](#),

$$\vec{r}_{ij} = \vec{r}_j - \vec{r}_i \text{ and } \vec{n} = \frac{\vec{r}_{ij}}{\|\vec{r}_{ij}\|}.$$

[c_SI](#) and [gamma_u](#) constatnts are used

Definition at line 386 of file LISACODE-Geometry.cpp.

References [position\(\)](#), [Vect::unit\(\)](#), and [velocity\(\)](#).

10.8.3.2 void Geometry::DispGeneralInfo ()

Display general inforamtions.

Definition at line 129 of file LISACODE-Geometry.cpp.

References [L0m](#), [move](#), [order_default](#), [rot0](#), [t0](#), [tRangeStoreDelay](#), and [tRangeStorePos](#).

Referenced by [GeometryMLDC::DispInfo\(\)](#), [GeometryFile::DispInfo\(\)](#), [GeometryAnalytic::DispInfo\(\)](#), and [DispInfo\(\)](#).

10.8.3.3 void Geometry::DispInfo () [virtual]

Display informations.

Reimplemented in [GeometryAnalytic](#), [GeometryFile](#), and [GeometryMLDC](#).

Definition at line 143 of file LISACODE-Geometry.cpp.

References [DispGeneralInfo\(\)](#).

Referenced by [LISA::LISA\(\)](#).

10.8.3.4 double Geometry::getL0 () [inline]

Returns [L0m](#) attribute.

Definition at line 111 of file LISACODE-Geometry.h.

References [L0m](#).

10.8.3.5 double Geometry::gett0 () [inline]

Returns [t0](#) attribute.

Definition at line 113 of file LISACODE-Geometry.h.

References [t0](#).

Referenced by [BackgroundGalactic::deltanu\(\)](#).

10.8.3.6 double Geometry::getRangeStoreDelay () [inline]

Definition at line 116 of file LISACODE-Geometry.h.

References tRangeStoreDelay.

10.8.3.7 double Geometry::getRangeStorePos () [inline]

Definition at line 115 of file LISACODE-Geometry.h.

References tRangeStorePos.

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

Give direct value for position.

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 441 of file LISACODE-Geometry.cpp.

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

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

10.8.3.9 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

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

If $(abs(trec-tStoreDelay))>tRangeStorePos$

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

Returns:

$$DelayStore[bs + em - 1]$$

Definition at line 466 of file LISACODE-Geometry.cpp.

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

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

10.8.3.10 void Geometry::initGlobal (double *t0_n*, double *rot0_n*, double *L0m_n*, int *order_default_n*, int *move_n*, double *tStep*)

Globla initialization.

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)
- `SCposStore` : $SCposStore[i] = position(i, 0)$ for $i=1,2,3,4$
- `DelayStore` : for $i=1,2,3,4$ $\begin{cases} DelayStore[i - 1] = tdelay(i, i, mod((i + 1), 3) + 1, 2, 0) \\ DelayStore[i + 2] = tdelay(i, i, mod(i, 3) + 1, 2, 0) \end{cases}$
- `tRangeStorePos` = `tRangeStorePos_default`
- `tRangeStoreDelay` = $\min(tStep, tRangeStoreDelay_default)$

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

References `L0m`, `move`, `order_default`, `rot0`, `t0`, `tRangeStoreDelay`, `tRangeStoreDelay_default`, `tRangeStorePos`, `tRangeStorePos_default`, `tStoreDelay`, and `tStorePos`.

Referenced by `Geometry()`, `GeometryMLDC::init()`, `GeometryFile::init()`, and `GeometryAnalytic::init()`.

10.8.3.11 Vect Geometry::position (int *nb*, double *t*) [virtual]

Returns the position of the spacecraft in the barycentric frame for the time *t* (s) as argument and spacecraft number (1, 2 or 3).

Default function : in correct using this function is not use Returns the started positions

Reimplemented in `GeometryAnalytic`, `GeometryFile`, and `GeometryMLDC`.

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

References `Vect::p`.

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

10.8.3.12 void Geometry::sett0 (double *t0_n*)

Definition at line 148 of file LISACODE-Geometry.cpp.

References `t0`.

10.8.3.13 void Geometry::settRangeStoreDelay (double tRangeStoreDelay_n)

Definition at line 159 of file LISACODE-Geometry.cpp.

References tRangeStoreDelay.

10.8.3.14 void Geometry::settRangeStorePos (double tRangeStorePos_n)

Definition at line 154 of file LISACODE-Geometry.cpp.

References tRangeStorePos.

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

Get delay for specified order (1/2 -> order=1 and 1 -> order=2).

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:

$$\text{delay} = \sum_{i=0}^{\text{order}} c_i,$$

where :

$$\begin{aligned} \mathbf{r}_i &\text{ is computed using } \text{position} \text{ method with rec and trec inputs,} \\ \mathbf{v}_i &\text{ is computed using } \text{velocity} \text{ method with rec and trec inputs, then divided by } \mathbf{c_SI}, \\ \mathbf{r}_j &\text{ is computed using } \text{position} \text{ method with em and trec inputs,} \\ \mathbf{v}_j &\text{ is computed using } \text{velocity} \text{ method with em and trec inputs, then divided by } \mathbf{c_SI}, \\ \vec{r}_{ij} &= \vec{r}_j - \vec{r}_i, \\ c_0 &= \frac{\|\vec{r}_{ij}\|}{C}, \\ \vec{n} &= -\frac{\vec{r}_{ij}}{\|\vec{r}_{ij}\|}, \\ c_1 &= t_{ij} \cdot \vec{n} \cdot \vec{v}_j, \\ c_2 &= c_{21} + c_{22} + c_{23}, \\ c_{21} &= \frac{t_{ij}}{2} \cdot (\vec{v}_j^2 + (\vec{n} \cdot \vec{v}_j)^2), \\ c_{22} &= -\frac{t_{ij}^2}{2} \cdot C \cdot RSchw, \\ c_{23} &= -\frac{RSchw}{C} \cdot (1 + \gamma_u) \cdot \log\left(\frac{\sqrt{(C \cdot t_{ij} \cdot \vec{n} \cdot \vec{r}_{ij})^2 + r_i^2 + C \cdot t_{ij} + \vec{n} \cdot \vec{r}_i}}{\vec{n} \cdot \vec{r}_i + \|\vec{r}_i\|}\right). \end{aligned}$$

$\mathbf{c_SI}$ and $\mathbf{gamma_u}$ constants are used.

Definition at line 247 of file LISACODE-Geometry.cpp.

References $\mathbf{c_SI}$, $\mathbf{gamma_u}$, Vect::norme(), order_default, position(), RSchw, Vect::unit(), and velocity().

Referenced by gtdelay(), GeometryMLDC::init(), GeometryFile::init(), GeometryAnalytic::init(), and main().

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

Get contribution of specified order (1/2 -> order=1 and 1 -> order=2) in delay.

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 :

\mathbf{r}_i is computed using [position](#) method with rec and trec inputs,

\mathbf{v}_i is computed using [velocity](#) method with rec and trec inputs, then divided by [c_SI](#),

\mathbf{r}_j is computed using [position](#) method with em and trec inputs,

\mathbf{v}_j is computed using [velocity](#) method with em and trec inputs, then divided by [c_SI](#),

$$\vec{r}_{ij} = \vec{r}_j - \vec{r}_i,$$

$$\vec{n} = -\frac{\vec{r}_{ij}}{\|\vec{r}_{ij}\|},$$

$$c_1 = t_{ij} \cdot \vec{n} \cdot \vec{v}_j,$$

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

$$c_{21} = \frac{t_{ij}}{2} \cdot (\vec{v}_j^2 + (\vec{n} \cdot \vec{v}_j)^2),$$

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

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

[c_SI](#) and [gamma_u](#) constants are used

Definition at line 324 of file LISACODE-Geometry.cpp.

References [c_SI](#), [gamma_u](#), [Vect::norme\(\)](#), [position\(\)](#), [RSchw](#), [Vect::unit\(\)](#), and [velocity\(\)](#).

10.8.3.17 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:

$$\frac{(\overrightarrow{\text{position}(2,t)} - \overrightarrow{\text{position}(3,t)}) \wedge (\overrightarrow{\text{position}(3,t)} - \overrightarrow{\text{position}(1,t)})}{\|(\overrightarrow{\text{position}(2,t)} - \overrightarrow{\text{position}(3,t)}) \wedge (\overrightarrow{\text{position}(3,t)} - \overrightarrow{\text{position}(1,t)})\|}$$

Definition at line 415 of file LISACODE-Geometry.cpp.

References [Vect::p](#), [position\(\)](#), and [Vect::unit\(\)](#).

Referenced by [main\(\)](#).

10.8.3.18 Vect Geometry::velocity (int nb, double t) [virtual]

Returns the position of the spacecraft in the barycentric frame for the time t (s) as argument and spacecraft number (1, 2 or 3).

Default function : in correct using this function is not use Returns 0 for all the components.

Reimplemented in [GeometryAnalytic](#), [GeometryFile](#), and [GeometryMLDC](#).

Definition at line 210 of file LISACODE-Geometry.cpp.

References Vect::p.

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

10.8.4 Member Data Documentation

10.8.4.1 [Geometry::alpha](#) [protected]

Orbital parameter.

Reimplemented in [GeometryAnalytic](#).

Referenced by GeometryMLDC::position(), and GeometryMLDC::velocity().

10.8.4.2 [Geometry::arot](#) [protected]

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

Reimplemented in [GeometryAnalytic](#).

Referenced by GeometryMLDC::init().

10.8.4.3 [Geometry::cmu](#) [protected]

Orbital constant.

Reimplemented in [GeometryAnalytic](#).

10.8.4.4 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 90 of file LISACODE-Geometry.h.

Referenced by gtdelay(), GeometryMLDC::init(), GeometryFile::init(), and GeometryAnalytic::init().

10.8.4.5 `Geometry::e` [protected]

Eccentricity.

Reimplemented in [GeometryAnalytic](#).

Referenced by `GeometryFile::position()`, and `GeometryFile::velocity()`.

10.8.4.6 `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 `DispGeneralInfo()`, `getL0()`, `GeometryAnalytic::init()`, and `initGlobal()`.

10.8.4.7 `int Geometry::move` [protected]

0 for [LISA](#) fixed, 1 for [LISA](#) moved (default)

Definition at line 71 of file LISACODE-Geometry.h.

Referenced by `DispGeneralInfo()`, `GeometryAnalytic::exanom()`, `gposition()`, `gtdelay()`, `initGlobal()`, `GeometryMLDC::position()`, `GeometryFile::position()`, `GeometryMLDC::velocity()`, and `GeometryFile::velocity()`.

10.8.4.8 `Geometry::nu` [protected]

Orbital angular constant.

Reimplemented in [GeometryAnalytic](#).

10.8.4.9 `int Geometry::order_default` [protected]

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

Definition at line 73 of file LISACODE-Geometry.h.

Referenced by `DispGeneralInfo()`, `initGlobal()`, and `tdelay()`.

10.8.4.10 `Geometry::rot0` [protected]

Initial rotation (radians).

Referenced by `DispGeneralInfo()`, `GeometryMLDC::init()`, `GeometryAnalytic::init()`, and `initGlobal()`.

10.8.4.11 `Vect Geometry::SCposStore[3]` [protected]

Stored positions.

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

Referenced by `gposition()`, `GeometryMLDC::init()`, `GeometryFile::init()`, and `GeometryAnalytic::init()`.

10.8.4.12 `Geometry::smu` [protected]

Orbital constant.

Reimplemented in [GeometryAnalytic](#).

10.8.4.13 `Geometry::sqrtee` [protected]

Eccentricity derived constant.

Reimplemented in [GeometryAnalytic](#).

10.8.4.14 `Geometry::t0` [protected]

Initial time (seconds).

Referenced by `DispGeneralInfo()`, `GeometryAnalytic::exanom()`, `gett0()`, `initGlobal()`, `GeometryMLDC::position()`, `GeometryFile::position()`, `sett0()`, `GeometryMLDC::velocity()`, and `GeometryFile::velocity()`.

10.8.4.15 `Geometry::tmu` [protected]

Orbital constant.

Reimplemented in [GeometryAnalytic](#).

10.8.4.16 `double Geometry::tRangeStoreDelay` [protected]

Last delay storage time.

Definition at line 94 of file LISACODE-Geometry.h.

Referenced by `DispGeneralInfo()`, `gettRangeStoreDelay()`, `gtdelay()`, `initGlobal()`, and `settRangeStoreDelay()`.

10.8.4.17 `double Geometry::tRangeStorePos` [protected]

last position storage time.

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

Referenced by `DispGeneralInfo()`, `gettRangeStorePos()`, `gposition()`, `initGlobal()`, and `settRangeStorePos()`.

10.8.4.18 `double Geometry::tStoreDelay` [protected]

Last delay computation time.

Definition at line 92 of file LISACODE-Geometry.h.

Referenced by `gtdelay()`, and `initGlobal()`.

10.8.4.19 double Geometry::tStorePos [protected]

last position computation time.

Definition at line 77 of file LISACODE-Geometry.h.

Referenced by gposition(), and initGlobal().

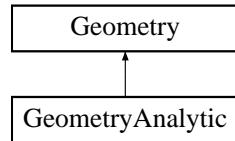
The documentation for this class was generated from the following files:

- [LISACODE-Geometry.h](#)
- [LISACODE-Geometry.cpp](#)

10.9 GeometryAnalytic Class Reference

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

Inheritance diagram for GeometryAnalytic::



10.9.1 Detailed Description

Compute spacecraft position from analytical formulations of the article : S.

V. Dhurandhar, K. Rajesh Nayak, S. Koshti, and Jean-Yves Vinet. Fundamentals of the lisa stable flight formation. gr-qc, 0410093, 2004.

Definition at line 43 of file LISACODE-GeometryAnalytic.h.

Public Member Functions

- [GeometryAnalytic \(\)](#)
Constructs an instance and initializes it with default values.
- [GeometryAnalytic \(double t0_n, double rot0_n\)](#)
Constructs an instance and initializes it with default values and t0_n and rot0_n inputs.
- [GeometryAnalytic \(double t0_n, double rot0_n, double L0m_n\)](#)
Constructs an instance and initializes it with default values and t0_n, rot0_n and L0m_n inputs.
- [GeometryAnalytic \(double t0_n, double rot0_n, double L0m_n, int order_default_n, int move_n\)](#)
Constructs an instance and initializes it with default values and t0_n, rot0_n, L0m_n, order_default_n and move_n inputs.
- [GeometryAnalytic \(double t0_n, double rot0_n, double L0m_n, int order_default_n, int move_n, double tStep\)](#)
Constructs an instance and initializes it with t0_n, rot0_n, L0m_n, order_default_n, move_n and tStep inputs.
- [~GeometryAnalytic \(\)](#)
Destructor.
- [void init \(double t0_n, double rot0_n, double L0m_n, int order_default_n, int move_n, double tStep\)](#)
Initializes attributes using t0_n, rot0_n, L0m_n, order_default_n, move_n and tStep inputs.
- [void DispInfo \(\)](#)
Display informations.

- **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].

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

Globla initialization.

- void **DispGeneralInfo** ()

Display general inforamtions.

- double **getL0** ()

Returns L0m attribute.

- double **gett0** ()

Returns t0 attribute.

- void **sett0** (double t0_n)

- double **gettRangeStorePos** ()

- double **gettRangeStoreDelay** ()

- void **settRangeStorePos** (double tRangeStorePos_n)

- void **settRangeStoreDelay** (double tRangeStoreDelay_n)

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

Get delay for specified order (1/2 -> order=1 and 1 -> order=2).

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

Get contribution of specified order (1/2 -> order=1 and 1 -> order=2) in delay.

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

Get 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)

Give direct value for position.

- 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 **alpha**
0 for staticLISA, 1 for rigid orbits and 2 for eccentric orbits 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 = $\frac{2\pi}{3}$.
- vector< double > **rot**
Satellites phase.
- vector< double > **crot**
Satellites phase cosinus.
- vector< double > **srot**
Satellites phase sinus.
- double **L0m**
Nominal arm length in meters.
- double **t0**
Initial time (seconds).
- double **rot0**
Initial rotation (radians).
- 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.

10.9.2 Constructor & Destructor Documentation

10.9.2.1 GeometryAnalytic::GeometryAnalytic ()

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-GeometryAnalytic.cpp.

References [init\(\)](#), and [L0_m_default](#).

10.9.2.2 GeometryAnalytic::GeometryAnalytic (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-GeometryAnalytic.cpp.

References init(), and L0_m_default.

10.9.2.3 **GeometryAnalytic::GeometryAnalytic (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-GeometryAnalytic.cpp.

References init().

10.9.2.4 **GeometryAnalytic::GeometryAnalytic (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-GeometryAnalytic.cpp.

References init().

10.9.2.5 GeometryAnalytic::GeometryAnalytic (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-GeometryAnalytic.cpp.

References [init\(\)](#).

10.9.2.6 GeometryAnalytic::~GeometryAnalytic ()

Destructor.

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

10.9.3 Member Function Documentation

10.9.3.1 double Geometry::ArmVelocity (int *em*, int *rec*, double *trec*) [inherited]

Get 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:

$$(\vec{v_j} - \vec{v_i}) \cdot \vec{n},$$

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](#),
 $\vec{r_{ij}} = \vec{r_j} - \vec{r_i}$ and $\vec{n} = -\frac{\vec{r_{ij}}}{\|\vec{r_{ij}}\|}$.
[c_SI](#) and [gamma_u](#) constants are used

Definition at line 386 of file LISACODE-Geometry.cpp.

References [Geometry::position\(\)](#), [Vect::unit\(\)](#), and [Geometry::velocity\(\)](#).

10.9.3.2 void Geometry::DispGeneralInfo () [inherited]

Display general inforamtions.

Definition at line 129 of file LISACODE-Geometry.cpp.

References Geometry::L0m, Geometry::move, Geometry::order_default, Geometry::rot0, Geometry::t0, Geometry::tRangeStoreDelay, and Geometry::tRangeStorePos.

Referenced by GeometryMLDC::DispInfo(), GeometryFile::DispInfo(), DispInfo(), and Geometry::DispInfo().

10.9.3.3 void GeometryAnalytic::DispInfo () [virtual]

Display informations.

Reimplemented from [Geometry](#).

Definition at line 180 of file LISACODE-GeometryAnalytic.cpp.

References Geometry::DispGeneralInfo().

10.9.3.4 Couple GeometryAnalytic::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 :
 $t = t0 + move \cdot ts$,
 $\psi = \omega \cdot t - rot_{nb-1}$,
 $ex_0 = \psi + e \cdot \sin(\psi) \cdot \cos\left(\frac{1-3\cdot\cos(\psi)^2}{2}\right)$,
 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 200 of file LISACODE-GeometryAnalytic.cpp.

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

Referenced by position(), and velocity().

10.9.3.5 double Geometry::getL0 () [inline, inherited]

Returns [L0m](#) attribute.

Definition at line 111 of file LISACODE-Geometry.h.

References Geometry::L0m.

10.9.3.6 double Geometry::gett0 () [inline, inherited]

Returns [t0](#) attribute.

Definition at line 113 of file LISACODE-Geometry.h.

References Geometry::t0.

Referenced by BackgroundGalactic::deltanu().

10.9.3.7 double Geometry::getRangeStoreDelay () [inline, inherited]

Definition at line 116 of file LISACODE-Geometry.h.

References Geometry::tRangeStoreDelay.

10.9.3.8 double Geometry::getRangeStorePos () [inline, inherited]

Definition at line 115 of file LISACODE-Geometry.h.

References Geometry::tRangeStorePos.

10.9.3.9 Vect Geometry::gposition (int *nb*, double *t*) [inherited]

Give direct value for position.

`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 441 of file LISACODE-Geometry.cpp.

References Geometry::move, Geometry::position(), Geometry::SCposStore, Geometry::tRangeStorePos, and Geometry::tStorePos.

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

10.9.3.10 double Geometry::gtdelay (int *em*, int *rec*, int *order*, double *trec*) [inherited]

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

`tdelay` method is used to set `DelayStore` attribute

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

If ($abs(trec-tStoreDelay)>tRangeStorePos$)

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

Returns:

$DelayStore[bs + em - 1]$

Definition at line 466 of file LISACODE-Geometry.cpp.

References Geometry::DelayStore, Geometry::move, Geometry::tdelay(), Geometry::tRangeStoreDelay, and Geometry::tStoreDelay.

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

10.9.3.11 void GeometryAnalytic::init (double t0_n, double rot0_n, double L0m_n, int order_default_n, int move_n, double tStep)

Initializes attributes 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 $\begin{cases} DelayStore[i - 1] = tdelay(i, i, mod((i + 1), 3) + 1, 2, 0) \\ DelayStore[i + 2] = tdelay(i, i, mod(i, 3) + 1, 2, 0) \end{cases}$
- **tRangeStorePos** = **tRangeStorePos_default**
- **tRangeStoreDelay** = $\min(tStep, tRangeStoreDelay_default)$

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

References alpha, arot, cmu, crot, Geometry::DelayStore, e, Geometry::initGlobal(), Geometry::L0m, nu, position(), Rgc, rot, Geometry::rot0, Geometry::SCposStore, smu, sqrtee, srot, Geometry::tdelay(), and tmu.

Referenced by GeometryAnalytic().

10.9.3.12 void Geometry::initGlobal (double *t0_n*, double *rot0_n*, double *L0m_n*, int *order_default_n*, int *move_n*, double *tStep*) [inherited]

Globla initialization.

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)
- **SCposStore** : *SCposStore[i] = position(i, 0)* for *i=1,2,3,4*
- **DelayStore** : for *i=1,2,3,4* $\begin{cases} \text{DelayStore}[i - 1] = \text{tdelay}(i, i, \text{mod}((i + 1), 3) + 1, 2, 0) \\ \text{DelayStore}[i + 2] = \text{tdelay}(i, i, \text{mod}(i, 3) + 1, 2, 0) \end{cases}$
- **tRangeStorePos** = *tRangeStorePos_default*
- **tRangeStoreDelay** = *min(tStep, tRangeStoreDelay_default)*

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

References **Geometry::L0m**, **Geometry::move**, **Geometry::order_default**, **Geometry::rot0**, **Geometry::t0**, **Geometry::tRangeStoreDelay**, **tRangeStoreDelay_default**, **Geometry::tRangeStorePos**, **tRangeStorePos_default**, **Geometry::tStoreDelay**, and **Geometry::tStorePos**.

Referenced by **Geometry::Geometry()**, **GeometryMLDC::init()**, **GeometryFile::init()**, and **init()**.

10.9.3.13 Vect GeometryAnalytic::position (int *nb*, double *t*) [virtual]

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

Reimplemented from **Geometry**.

Definition at line 240 of file LISACODE-GeometryAnalytic.cpp.

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

Referenced by **init()**.

10.9.3.14 void Geometry::sett0 (double *t0_n*) [inherited]

Definition at line 148 of file LISACODE-Geometry.cpp.

References Geometry::t0.

10.9.3.15 void Geometry::settRangeStoreDelay (double *tRangeStoreDelay_n*) [inherited]

Definition at line 159 of file LISACODE-Geometry.cpp.

References Geometry::tRangeStoreDelay.

10.9.3.16 void Geometry::settRangeStorePos (double *tRangeStorePos_n*) [inherited]

Definition at line 154 of file LISACODE-Geometry.cpp.

References Geometry::tRangeStorePos.

10.9.3.17 double Geometry::tdelay (int *em*, int *rec*, int *order*, double *trec*) [inherited]

Get delay for specified order (1/2 -> order=1 and 1 -> order=2).

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:

$$\text{delay} = \sum_{i=0}^{\text{order}} c_i,$$

where :

\vec{r}_{ij} is computed using [position](#) method with rec and trec inputs,

\vec{v}_i is computed using [velocity](#) method with rec and trec inputs, then divided by [c_SI](#),

\vec{r}_j is computed using [position](#) method with em and trec inputs,

\vec{v}_j is computed using [velocity](#) method with em and trec inputs, then divided by [c_SI](#),

$$\vec{r}_{ij} = \vec{r}_j - \vec{r}_i,$$

$$c_0 = \frac{\|\vec{r}_{ij}\|}{C},$$

$$\vec{n} = -\frac{\vec{r}_{ij}}{\|\vec{r}_{ij}\|},$$

$$c_1 = t_{ij} \cdot \vec{n} \cdot \vec{v}_j,$$

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

$$c_{21} = \frac{t_{ij}}{2} \cdot (\vec{v}_j^2 + (\vec{n} \cdot \vec{v}_j)^2),$$

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

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

[c_SI](#) and [gamma_u](#) constants are used.

Definition at line 247 of file LISACODE-Geometry.cpp.

References [c_SI](#), [gamma_u](#), [Vect::norme\(\)](#), [Geometry::order_default](#), [Geometry::position\(\)](#), [RSchw](#), [Vect::unit\(\)](#), and [Geometry::velocity\(\)](#).

Referenced by [Geometry::gtdelay\(\)](#), [GeometryMLDC::init\(\)](#), [GeometryFile::init\(\)](#), [init\(\)](#), and [main\(\)](#).

10.9.3.18 double Geometry::tdelayOrderContribution (int em, int rec, int order, double trec) [inherited]

Get contribution of specified order (1/2 -> order=1 and 1 -> order=2) in delay.

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 :

r_i is computed using [position](#) method with rec and trec inputs,

v_i is computed using [velocity](#) method with rec and trec inputs, then divided by [c_SI](#),

r_j is computed using [position](#) method with em and trec inputs,

v_j is computed using [velocity](#) method with em and trec inputs, then divided by [c_SI](#),

$$\vec{r}_{ij} = \vec{r}_j - \vec{r}_i,$$

$$\vec{n} = -\frac{\vec{r}_{ij}}{\|\vec{r}_{ij}\|},$$

$$c_1 = t_{ij} \cdot \vec{n} \cdot \vec{v}_j,$$

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

$$c_{21} = \frac{t_{ij}}{2} \cdot (\vec{v}_j^2 + (\vec{n} \cdot \vec{v}_j)^2),$$

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

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

[c_SI](#) and [gamma_u](#) constants are used

Definition at line 324 of file LISACODE-Geometry.cpp.

References [c_SI](#), [gamma_u](#), [Vect::norme\(\)](#), [Geometry::position\(\)](#), [RSchw](#), [Vect::unit\(\)](#), and [Geometry::velocity\(\)](#).

10.9.3.19 Vect Geometry::VectNormal (double t) [inherited]

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

[position](#) method is used to set [SCposStore](#) attribute

Returns:

$$\frac{(\overrightarrow{\text{position}(2,t)} - \overrightarrow{\text{position}(3,t)}) \wedge (\overrightarrow{\text{position}(3,t)} - \overrightarrow{\text{position}(1,t)})}{\|(\overrightarrow{\text{position}(2,t)} - \overrightarrow{\text{position}(3,t)}) \wedge (\overrightarrow{\text{position}(3,t)} - \overrightarrow{\text{position}(1,t)})\|}$$

Definition at line 415 of file LISACODE-Geometry.cpp.

References [Vect::p](#), [Geometry::position\(\)](#), and [Vect::unit\(\)](#).

Referenced by [main\(\)](#).

10.9.3.20 Vect GeometryAnalytic::velocity (int nb, double t) [virtual]

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: $\vec{v} = \begin{pmatrix} \dot{\psi} \cdot (-spsi \cdot cmu \cdot crot_{nb-1} - sqrtee \cdot cpsi \cdot srot_{nb-1}) \\ \dot{\psi} \cdot (-spsi \cdot cmu \cdot srot_{nb-1} + sqrtee \cdot cpsi \cdot crot_{nb-1}) \\ \dot{\psi} \cdot smu \cdot spsi \end{pmatrix},$
 where $\dot{\psi} = \frac{\omega \cdot Rgc}{1 - e \cdot cpsi}$

Reimplemented from [Geometry](#).

Definition at line 277 of file LISACODE-GeometryAnalytic.cpp.

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

10.9.4 Member Data Documentation

10.9.4.1 GeometryAnalytic::alpha [protected]

0 for staticLISA, 1 for rigid orbits and 2 for eccentric orbits Orbital parameter.

Reimplemented from [Geometry](#).

Referenced by init().

10.9.4.2 GeometryAnalytic::arot [protected]

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

Reimplemented from [Geometry](#).

Referenced by init().

10.9.4.3 GeometryAnalytic::cmu [protected]

Orbital constant.

Reimplemented from [Geometry](#).

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

10.9.4.4 GeometryAnalytic::crot [protected]

Satellites phase cosinus.

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

10.9.4.5 double Geometry::DelayStore[6] [protected, inherited]

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 90 of file LISACODE-Geometry.h.

Referenced by [Geometry::gtdelay\(\)](#), [GeometryMLDC::init\(\)](#), [GeometryFile::init\(\)](#), and [init\(\)](#).

10.9.4.6 double [GeometryAnalytic::e](#) [protected]

Eccentricity.

Reimplemented from [Geometry](#).

Referenced by [exanom\(\)](#), [init\(\)](#), [position\(\)](#), and [velocity\(\)](#).

10.9.4.7 double [Geometry::L0m](#) [protected, inherited]

Nominal arm length in meters.

Nominal distance between two satellites.

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

Referenced by [Geometry::DispGeneralInfo\(\)](#), [Geometry::getL0\(\)](#), [init\(\)](#), and [Geometry::initGlobal\(\)](#).

10.9.4.8 int [Geometry::move](#) [protected, inherited]

0 for [LISA](#) fixed, 1 for [LISA](#) moved (default)

Definition at line 71 of file LISACODE-Geometry.h.

Referenced by [Geometry::DispGeneralInfo\(\)](#), [exanom\(\)](#), [Geometry::gposition\(\)](#), [Geometry::gtdelay\(\)](#), [Geometry::initGlobal\(\)](#), [GeometryMLDC::position\(\)](#), [GeometryFile::position\(\)](#), [GeometryMLDC::velocity\(\)](#), and [GeometryFile::velocity\(\)](#).

10.9.4.9 double [GeometryAnalytic::nu](#) [protected]

Orbital angular constant.

Reimplemented from [Geometry](#).

Referenced by [init\(\)](#).

10.9.4.10 int [Geometry::order_default](#) [protected, inherited]

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

Definition at line 73 of file LISACODE-Geometry.h.

Referenced by [Geometry::DispGeneralInfo\(\)](#), [Geometry::initGlobal\(\)](#), and [Geometry::tdelay\(\)](#).

10.9.4.11 [GeometryAnalytic::rot](#) [protected]

Satellites phase.

Referenced by [exanom\(\)](#), and [init\(\)](#).

10.9.4.12 [Geometry::rot0](#) [protected, inherited]

Initial rotation (radians).

Referenced by [Geometry::DispGeneralInfo\(\)](#), [GeometryMLDC::init\(\)](#), [init\(\)](#), and [Geometry::initGlobal\(\)](#).

10.9.4.13 Vect [Geometry::SCposStore\[3\]](#) [protected, inherited]

Stored positions.

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

Referenced by [Geometry::gposition\(\)](#), [GeometryMLDC::init\(\)](#), [GeometryFile::init\(\)](#), and [init\(\)](#).

10.9.4.14 [GeometryAnalytic::smu](#) [protected]

Orbital constant.

Reimplemented from [Geometry](#).

Referenced by [init\(\)](#), [position\(\)](#), and [velocity\(\)](#).

10.9.4.15 [GeometryAnalytic::sqrtee](#) [protected]

Eccentricity derived constant.

Reimplemented from [Geometry](#).

Referenced by [init\(\)](#), [position\(\)](#), and [velocity\(\)](#).

10.9.4.16 [GeometryAnalytic::srot](#) [protected]

Satellites phase sinus.

Referenced by [init\(\)](#), [position\(\)](#), and [velocity\(\)](#).

10.9.4.17 [Geometry::t0](#) [protected, inherited]

Initial time (seconds).

Referenced by Geometry::DispGeneralInfo(), exanom(), Geometry::gett0(), Geometry::initGlobal(), GeometryMLDC::position(), GeometryFile::position(), Geometry::sett0(), GeometryMLDC::velocity(), and GeometryFile::velocity().

10.9.4.18 **GeometryAnalytic::tmu** [protected]

Orbital constant.

Reimplemented from [Geometry](#).

Referenced by init().

10.9.4.19 **double Geometry::tRangeStoreDelay** [protected, inherited]

Last delay storage time.

Definition at line 94 of file LISACODE-Geometry.h.

Referenced by Geometry::DispGeneralInfo(), Geometry::gettRangeStoreDelay(), Geometry::gtdelay(), Geometry::initGlobal(), and Geometry::settRangeStoreDelay().

10.9.4.20 **double Geometry::tRangeStorePos** [protected, inherited]

last position storage time.

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

Referenced by Geometry::DispGeneralInfo(), Geometry::gettRangeStorePos(), Geometry::gposition(), Geometry::initGlobal(), and Geometry::settRangeStorePos().

10.9.4.21 **double Geometry::tStoreDelay** [protected, inherited]

Last delay computation time.

Definition at line 92 of file LISACODE-Geometry.h.

Referenced by Geometry::gtdelay(), and Geometry::initGlobal().

10.9.4.22 **double Geometry::tStorePos** [protected, inherited]

last position computation time.

Definition at line 77 of file LISACODE-Geometry.h.

Referenced by Geometry::gposition(), and Geometry::initGlobal().

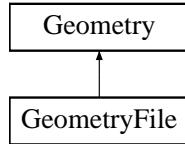
The documentation for this class was generated from the following files:

- [LISACODE-GeometryAnalytic.h](#)
- [LISACODE-GeometryAnalytic.cpp](#)

10.10 GeometryFile Class Reference

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

Inheritance diagram for GeometryFile::



10.10.1 Detailed Description

Compute spacecraft position from ::

This are the orbits used for MLDC

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

Public Member Functions

- [GeometryFile \(\)](#)
Constructs an instance and initializes it with default values.
- [GeometryFile \(double t0_n, double rot0_n\)](#)
Constructs an instance and initializes it with default values and t0_n and rot0_n inputs.
- [GeometryFile \(double t0_n, double rot0_n, double L0m_n\)](#)
Constructs an instance and initializes it with default values and t0_n, rot0_n and L0m_n inputs.
- [GeometryFile \(double t0_n, double rot0_n, double L0m_n, int order_default_n, int move_n, double tStep\)](#)
Constructs an instance and initializes it with t0_n, rot0_n, L0m_n, order_default_n, move_n and tStep inputs. The orbits are readed in file `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.
- [GeometryFile \(double t0_n, double rot0_n, double L0m_n, int order_default_n, int move_n, double tStep, char *InputFileName_n\)](#)
Constructs an instance and initializes it with t0_n, rot0_n, L0m_n, order_default_n, move_n and tStep inputs.
- [~GeometryFile \(\)](#)
Destructor

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

Initialization.
- void **DispInfo** ()

Display informations.
- **Couple exanom** (int nb, double ts)

Returns the position of the spacecraft in the barycentric frame, 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].
- void **initGlobal** (double t0_n, double rot0_n, double L0m_n, int order_default_n, int move_n, double tStep)

Globla initialization.
- void **DispGeneralInfo** ()

Display general inforamtions.
- double **getL0** ()

Returns L0m attribute.
- double **gett0** ()

Returns t0 attribute.
- void **sett0** (double t0_n)
- double **gettRangeStorePos** ()
- double **gettRangeStoreDelay** ()
- void **settRangeStorePos** (double tRangeStorePos_n)
- void **settRangeStoreDelay** (double tRangeStoreDelay_n)
- double **tdelay** (int em, int rec, int order, double trec)

Get delay for specified order (1/2 -> order=1 and 1 -> order=2).
- double **tdelayOrderContribution** (int em, int rec, int order, double trec)

Get contribution of specified order (1/2 -> order=1 and 1 -> order=2) in delay.
- double **ArmVelocity** (int em, int rec, double trec)

Get 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)

Give direct value for position.
- 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 *** **XYZ**
Data of spacecraft's positions : [spacecrat(0->1,1->2,2->3)][component(0->x,1->y,2->z)][itime].
- double * **Time_sec**
Time data.
- int **NbData**
Number of data.
- char * **InputFileName**
Name of file which contains spacecraft's positions.
- 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 = $\frac{2\pi}{3}$.
- double **t0**
Initial time (seconds).
- double **rot0**
Initial rotation (radians).
- 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.

10.10.2 Constructor & Destructor Documentation

10.10.2.1 GeometryFile::GeometryFile ()

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-GeometryFile.cpp.

References `init()`, and `L0_m_default`.

10.10.2.2 GeometryFile::GeometryFile (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-GeometryFile.cpp.

References init(), and [L0_m_default](#).

10.10.2.3 GeometryFile::GeometryFile (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-GeometryFile.cpp.

References init().

10.10.2.4 GeometryFile::GeometryFile (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. The orbits are readed in file [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 75 of file LISACODE-GeometryFile.cpp.

References init().

10.10.2.5 GeometryFile::GeometryFile (double *t0_n*, double *rot0_n*, double *L0m_n*, int *order_default_n*, int *move_n*, double *tStep*, char * *InputFileName_n*)

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-GeometryFile.cpp.

References [init\(\)](#).

10.10.2.6 GeometryFile::~GeometryFile ()

Destructor.

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

10.10.3 Member Function Documentation

10.10.3.1 double Geometry::ArmVelocity (int *em*, int *rec*, double *trec*) [inherited]

Get 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:

$$(\vec{v_j} - \vec{v_i}) \cdot \vec{n},$$

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](#),
 $\vec{r_{ij}} = \vec{r_j} - \vec{r_i}$ and $\vec{n} = -\frac{\vec{r_{ij}}}{\|\vec{r_{ij}}\|}$.
[c_SI](#) and [gamma_u](#) constants are used

Definition at line 386 of file LISACODE-Geometry.cpp.

References [Geometry::position\(\)](#), [Vect::unit\(\)](#), and [Geometry::velocity\(\)](#).

10.10.3.2 void Geometry::DispGeneralInfo () [inherited]

Display general inforamtions.

Definition at line 129 of file LISACODE-Geometry.cpp.

References Geometry::L0m, Geometry::move, Geometry::order_default, Geometry::rot0, Geometry::t0, Geometry::tRangeStoreDelay, and Geometry::tRangeStorePos.

Referenced by GeometryMLDC::DispInfo(), DispInfo(), GeometryAnalytic::DispInfo(), and Geometry::DispInfo().

10.10.3.3 void GeometryFile::DispInfo () [virtual]

Display informations.

Reimplemented from [Geometry](#).

Definition at line 193 of file LISACODE-GeometryFile.cpp.

References Geometry::DispGeneralInfo().

10.10.3.4 Couple GeometryFile::exanom (int nb, double ts)**10.10.3.5 double Geometry::getL0 () [inline, inherited]**

Returns [L0m](#) attribute.

Definition at line 111 of file LISACODE-Geometry.h.

References Geometry::L0m.

10.10.3.6 double Geometry::gett0 () [inline, inherited]

Returns [t0](#) attribute.

Definition at line 113 of file LISACODE-Geometry.h.

References Geometry::t0.

Referenced by BackgroundGalactic::deltanu().

10.10.3.7 double Geometry::gettRangeStoreDelay () [inline, inherited]

Definition at line 116 of file LISACODE-Geometry.h.

References Geometry::tRangeStoreDelay.

10.10.3.8 double Geometry::gettRangeStorePos () [inline, inherited]

Definition at line 115 of file LISACODE-Geometry.h.

References Geometry::tRangeStorePos.

10.10.3.9 Vect Geometry::gposition (int nb, double t) [inherited]

Give direct value for position.

`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 441 of file LISACODE-Geometry.cpp.

References `Geometry::move`, `Geometry::position()`, `Geometry::SCposStore`, `Geometry::tRangeStorePos`, and `Geometry::tStorePos`.

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

10.10.3.10 double Geometry::gtdelay (int em, int rec, int order, double trec) [inherited]

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 ($abs(trec - tStoreDelay) > tRangeStorePos$)

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

Returns:

$DelayStore[bs + em - 1]$

Definition at line 466 of file LISACODE-Geometry.cpp.

References `Geometry::DelayStore`, `Geometry::move`, `Geometry::tdelay()`, `Geometry::tRangeStoreDelay`, and `Geometry::tStoreDelay`.

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

10.10.3.11 void GeometryFile::init (double t0_n, double rot0_n, double L0m_n, int order_default_n, int move_n, double tStep, char * InputFileName_n)

Initialization.

Definition at line 109 of file LISACODE-GeometryFile.cpp.

References `Geometry::DelayStore`, `Geometry::initGlobal()`, `InputFileName`, `NbData`, `position()`, `Geometry::SCposStore`, `Geometry::tdelay()`, `Time_sec`, and `XYZ`.

Referenced by `GeometryFile()`.

10.10.3.12 void Geometry::initGlobal (double *t0_n*, double *rot0_n*, double *L0m_n*, int *order_default_n*, int *move_n*, double *tStep*) [inherited]

Globla initialization.

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)
- `SCposStore` : $SCposStore[i] = position(i, 0)$ for $i=1,2,3,4$
- `DelayStore` : for $i=1,2,3,4$ $\begin{cases} DelayStore[i - 1] = tdelay(i, i, mod((i + 1), 3) + 1, 2, 0) \\ DelayStore[i + 2] = tdelay(i, i, mod(i, 3) + 1, 2, 0) \end{cases}$
- `tRangeStorePos` = `tRangeStorePos_default`
- `tRangeStoreDelay` = $\min(tStep, tRangeStoreDelay_default)$

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

References `Geometry::L0m`, `Geometry::move`, `Geometry::order_default`, `Geometry::rot0`, `Geometry::t0`, `Geometry::tRangeStoreDelay`, `tRangeStoreDelay_default`, `Geometry::tRangeStorePos`, `tRangeStorePos_default`, `Geometry::tStoreDelay`, and `Geometry::tStorePos`.

Referenced by `Geometry::Geometry()`, `GeometryMLDC::init()`, `init()`, and `GeometryAnalytic::init()`.

10.10.3.13 Vect GeometryFile::position (int *nb*, double *t*) [virtual]

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

Reimplemented from `Geometry`.

Definition at line 208 of file LISACODE-GeometryFile.cpp.

References `Geometry::e`, `Geometry::move`, `Vect::p`, `Geometry::t0`, `Time_sec`, and `XYZ`.

Referenced by `init()`.

10.10.3.14 void Geometry::sett0 (double *t0_n*) [inherited]

Definition at line 148 of file LISACODE-Geometry.cpp.

References `Geometry::t0`.

10.10.3.15 void Geometry::settRangeStoreDelay (double tRangeStoreDelay_n) [inherited]

Definition at line 159 of file LISACODE-Geometry.cpp.

References Geometry::tRangeStoreDelay.

10.10.3.16 void Geometry::settRangeStorePos (double tRangeStorePos_n) [inherited]

Definition at line 154 of file LISACODE-Geometry.cpp.

References Geometry::tRangeStorePos.

10.10.3.17 double Geometry::tdelay (int em, int rec, int order, double trec) [inherited]

Get delay for specified order (1/2 -> order=1 and 1 -> order=2).

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:

$$\text{delay} = \sum_{i=0}^{\text{order}} c_i,$$

where :

\vec{r}_{ij} is computed using [position](#) method with rec and trec inputs,

\vec{v}_i is computed using [velocity](#) method with rec and trec inputs, then divided by [c_SI](#),

\vec{r}_j is computed using [position](#) method with em and trec inputs,

\vec{v}_j is computed using [velocity](#) method with em and trec inputs, then divided by [c_SI](#),

$$\vec{r}_{ij} = \vec{r}_j - \vec{r}_i,$$

$$c_0 = \frac{\|\vec{r}_{ij}\|}{C},$$

$$\vec{n} = -\frac{\vec{r}_{ij}}{\|\vec{r}_{ij}\|},$$

$$c_1 = t_{ij} \cdot \vec{n} \cdot \vec{v}_j,$$

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

$$c_{21} = \frac{t_{ij}}{2} \cdot (\vec{v}_j^2 + (\vec{n} \cdot \vec{v}_j)^2),$$

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

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

[c_SI](#) and [gamma_u](#) constants are used.

Definition at line 247 of file LISACODE-Geometry.cpp.

References [c_SI](#), [gamma_u](#), [Vect::norme\(\)](#), [Geometry::order_default](#), [Geometry::position\(\)](#), [RSchw](#), [Vect::unit\(\)](#), and [Geometry::velocity\(\)](#).

Referenced by [Geometry::gtdelay\(\)](#), [GeometryMLDC::init\(\)](#), [init\(\)](#), [GeometryAnalytic::init\(\)](#), and [main\(\)](#).

10.10.3.18 double Geometry::tdelayOrderContribution (int em, int rec, int order, double trec) [inherited]

Get contribution of specified order (1/2 -> order=1 and 1 -> order=2) in delay.

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 :

\vec{r}_i is computed using [position](#) method with rec and trec inputs,

\vec{v}_i is computed using [velocity](#) method with rec and trec inputs, then divided by [c_SI](#),

\vec{r}_j is computed using [position](#) method with em and trec inputs,

\vec{v}_j is computed using [velocity](#) method with em and trec inputs, then divided by [c_SI](#),

$$\vec{r}_{ij} = \vec{r}_j - \vec{r}_i,$$

$$\vec{n} = -\frac{\vec{r}_{ij}}{\|\vec{r}_{ij}\|},$$

$$c_1 = t_{ij} \cdot \vec{n} \cdot \vec{v}_j,$$

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

$$c_{21} = \frac{t_{ij}}{2} \cdot (\vec{v}_j^2 + (\vec{n} \cdot \vec{v}_j)^2),$$

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

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

[c_SI](#) and [gamma_u](#) constants are used

Definition at line 324 of file LISACODE-Geometry.cpp.

References [c_SI](#), [gamma_u](#), [Vect::norme\(\)](#), [Geometry::position\(\)](#), [RSchw](#), [Vect::unit\(\)](#), and [Geometry::velocity\(\)](#).

10.10.3.19 Vect [Geometry::VectNormal \(double t\)](#) [inherited]

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

[position](#) method is used to set [SCposStore](#) attribute

Returns:

$$\frac{(\vec{position}(2, t) - \vec{position}(3, t)) \wedge (\vec{position}(3, t) - \vec{position}(1, t))}{\|(\vec{position}(2, t) - \vec{position}(3, t)) \wedge (\vec{position}(3, t) - \vec{position}(1, t))\|}$$

Definition at line 415 of file LISACODE-Geometry.cpp.

References [Vect::p](#), [Geometry::position\(\)](#), and [Vect::unit\(\)](#).

Referenced by [main\(\)](#).

10.10.3.20 Vect [GeometryFile::velocity \(int nb, double t\)](#) [virtual]

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

Reimplemented from [Geometry](#).

Definition at line 261 of file LISACODE-GeometryFile.cpp.

References [Geometry::e](#), [Geometry::move](#), [Vect::p](#), [Geometry::t0](#), [Time_sec](#), and [XYZ](#).

10.10.4 Member Data Documentation

10.10.4.1 [Geometry::alpha](#) [protected, inherited]

Orbital parameter.

Reimplemented in [GeometryAnalytic](#).

Referenced by [GeometryMLDC::position\(\)](#), and [GeometryMLDC::velocity\(\)](#).

10.10.4.2 [Geometry::arot](#) [protected, inherited]

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

Reimplemented in [GeometryAnalytic](#).

Referenced by [GeometryMLDC::init\(\)](#).

10.10.4.3 [Geometry::cmu](#) [protected, inherited]

Orbital constant.

Reimplemented in [GeometryAnalytic](#).

10.10.4.4 double [Geometry::DelayStore\[6\]](#) [protected, inherited]

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 90 of file LISACODE-Geometry.h.

Referenced by [Geometry::gtdelay\(\)](#), [GeometryMLDC::init\(\)](#), [init\(\)](#), and [GeometryAnalytic::init\(\)](#).

10.10.4.5 [Geometry::e](#) [protected, inherited]

Eccentricity.

Reimplemented in [GeometryAnalytic](#).

Referenced by [position\(\)](#), and [velocity\(\)](#).

10.10.4.6 [GeometryFile::InputFileName](#) [protected]

Name of file which contains spacecraft's positions.

Referenced by init().

10.10.4.7 [double Geometry::L0m](#) [protected, inherited]

Nominal arm length in meters.

Nominal distance between two satellites.

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

Referenced by Geometry::DispGeneralInfo(), Geometry::getL0(), GeometryAnalytic::init(), and Geometry::initGlobal().

10.10.4.8 [int Geometry::move](#) [protected, inherited]

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

Definition at line 71 of file LISACODE-Geometry.h.

Referenced by Geometry::DispGeneralInfo(), GeometryAnalytic::exanom(), Geometry::gposition(), Geometry::gtdelay(), Geometry::initGlobal(), GeometryMLDC::position(), position(), Geometry-MLDC::velocity(), and velocity().

10.10.4.9 [GeometryFile::NbData](#) [protected]

Number of data.

Referenced by init().

10.10.4.10 [Geometry::nu](#) [protected, inherited]

Orbital angular constant.

Reimplemented in [GeometryAnalytic](#).

10.10.4.11 [int Geometry::order_default](#) [protected, inherited]

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

Definition at line 73 of file LISACODE-Geometry.h.

Referenced by Geometry::DispGeneralInfo(), Geometry::initGlobal(), and Geometry::tdelay().

10.10.4.12 [Geometry::rot0](#) [protected, inherited]

Initial rotation (radians).

Referenced by Geometry::DispGeneralInfo(), GeometryMLDC::init(), GeometryAnalytic::init(), and Geometry::initGlobal().

10.10.4.13 Vect Geometry::SCposStore[3] [protected, inherited]

Stored positions.

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

Referenced by Geometry::gposition(), GeometryMLDC::init(), init(), and GeometryAnalytic::init().

10.10.4.14 Geometry::smu [protected, inherited]

Orbital constant.

Reimplemented in [GeometryAnalytic](#).

10.10.4.15 Geometry::sqrtee [protected, inherited]

Eccentricity derived constant.

Reimplemented in [GeometryAnalytic](#).

10.10.4.16 Geometry::t0 [protected, inherited]

Initial time (seconds).

Referenced by Geometry::DispGeneralInfo(), GeometryAnalytic::exanom(), Geometry::gett0(), Geometry::initGlobal(), GeometryMLDC::position(), position(), Geometry::sett0(), GeometryMLDC::velocity(), and velocity().

10.10.4.17 GeometryFile::Time_sec [protected]

Time data.

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

10.10.4.18 Geometry::tmu [protected, inherited]

Orbital constant.

Reimplemented in [GeometryAnalytic](#).

10.10.4.19 double Geometry::tRangeStoreDelay [protected, inherited]

Last delay storage time.

Definition at line 94 of file LISACODE-Geometry.h.

Referenced by Geometry::DispGeneralInfo(), Geometry::gettRangeStoreDelay(), Geometry::gtdelay(), Geometry::initGlobal(), and Geometry::settRangeStoreDelay().

10.10.4.20 double Geometry::tRangeStorePos [protected, inherited]

last position storage time.

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

Referenced by Geometry::DispGeneralInfo(), Geometry::gettRangeStorePos(), Geometry::gposition(), Geometry::initGlobal(), and Geometry::settRangeStorePos().

10.10.4.21 double **Geometry::tStoreDelay** [protected, inherited]

Last delay computation time.

Definition at line 92 of file LISACODE-Geometry.h.

Referenced by Geometry::gtdelay(), and Geometry::initGlobal().

10.10.4.22 double **Geometry::tStorePos** [protected, inherited]

last position computation time.

Definition at line 77 of file LISACODE-Geometry.h.

Referenced by Geometry::gposition(), and Geometry::initGlobal().

10.10.4.23 **GeometryFile::XYZ** [protected]

Data of spacecraft's positions : [spacecrat(0->1,1->2,2->3)][component(0->x,1->y,2->z)][itime].

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

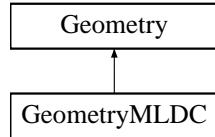
The documentation for this class was generated from the following files:

- [LISACODE-GeometryFile.h](#)
- [LISACODE-GeometryFile.cpp](#)

10.11 GeometryMLDC Class Reference

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

Inheritance diagram for GeometryMLDC::



10.11.1 Detailed Description

Compute spacecraft position from ::

This are the orbits used for MLDC

Definition at line 43 of file LISACODE-GeometryMLDC.h.

Public Member Functions

- [GeometryMLDC \(\)](#)
Constructs an instance and initializes it with default values.
- [GeometryMLDC \(double t0_n, double rot0_n\)](#)
Constructs an instance and initializes it with default values and t0_n and rot0_n inputs.
- [GeometryMLDC \(double t0_n, double rot0_n, double L0m_n\)](#)
Constructs an instance and initializes it with default values and t0_n, rot0_n and L0m_n inputs.
- [GeometryMLDC \(double t0_n, double rot0_n, double L0m_n, int order_default_n, int move_n\)](#)
Constructs an instance and initializes it with default values and t0_n, rot0_n, L0m_n, order_default_n and move_n inputs.
- [GeometryMLDC \(double t0_n, double rot0_n, double L0m_n, int order_default_n, int move_n, double tStep\)](#)
Constructs an instance and initializes it with t0_n, rot0_n, L0m_n, order_default_n, move_n and tStep inputs.
- [~GeometryMLDC \(\)](#)
Destructor
- void [init \(double t0_n, double rot0_n, double L0m_n, int order_default_n, int move_n, double tStep\)](#)
Initializes attributes using t0_n, rot0_n, L0m_n, order_default_n, move_n and tStep inputs.
- void [DispInfo \(\)](#)
Display informations.
- [Couple exanom \(int nb, double ts\)](#)

- **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].
- void **initGlobal** (double t0_n, double rot0_n, double L0m_n, int order_default_n, int move_n, double tStep)

Globla initialization.
- void **DispGeneralInfo** ()

Display general inforamtions.
- double **getL0** ()

Returns L0m attribute.
- double **gett0** ()

Returns t0 attribute.
- void **sett0** (double t0_n)
- double **gettRangeStorePos** ()
- double **gettRangeStoreDelay** ()
- void **settRangeStorePos** (double tRangeStorePos_n)
- void **settRangeStoreDelay** (double tRangeStoreDelay_n)
- double **tdelay** (int em, int rec, int order, double trec)

Get delay for specified order (1/2 -> order=1 and 1 -> order=2).
- double **tdelayOrderContribution** (int em, int rec, int order, double trec)

Get contribution of specified order (1/2 -> order=1 and 1 -> order=2) in delay.
- double **ArmVelocity** (int em, int rec, double trec)

Get 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)

Give direct value for position.
- 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 **Approx**
- vector< double > **rot**

Satellites phase.

- vector< double > **crot**
Satellites phase cosinus.
- vector< double > **srot**
Satellites phase sinus.
- double **e_mldc**
excentricity for MLDC orbits
- double **sqrt_3**
 $\sqrt{3}$ for MLDC orbits
- 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**
 $\text{Phase between satellites : constant, } arot = \frac{2\pi}{3}.$
- double **t0**
Initial time (seconds).
- double **rot0**
Initial rotation (radians).
- 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.

10.11.2 Constructor & Destructor Documentation

10.11.2.1 GeometryMLDC::GeometryMLDC ()

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-GeometryMLDC.cpp.

References init(), and L0_m_default.

10.11.2.2 GeometryMLDC::GeometryMLDC (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-GeometryMLDC.cpp.

References init(), and [L0_m_default](#).

10.11.2.3 GeometryMLDC::GeometryMLDC (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-GeometryMLDC.cpp.

References init().

10.11.2.4 GeometryMLDC::GeometryMLDC (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-GeometryMLDC.cpp.

References init().

10.11.2.5 GeometryMLDC::GeometryMLDC (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-GeometryMLDC.cpp.

References [init\(\)](#).

10.11.2.6 GeometryMLDC::~GeometryMLDC ()

Destructor.

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

10.11.3 Member Function Documentation

10.11.3.1 double Geometry::ArmVelocity (int *em*, int *rec*, double *trec*) [inherited]

Get 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:

$$(\vec{v}_j - \vec{v}_i) \cdot \vec{n},$$

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](#),
- $\vec{r}_{ij} = \vec{r}_j - \vec{r}_i$ and $\vec{n} = -\frac{\vec{r}_{ij}}{\|\vec{r}_{ij}\|}$.

[c_SI](#) and [gamma_u](#) constants are used

Definition at line 386 of file LISACODE-Geometry.cpp.

References [Geometry::position\(\)](#), [Vect::unit\(\)](#), and [Geometry::velocity\(\)](#).

10.11.3.2 void Geometry::DispGeneralInfo () [inherited]

Display general inforamtions.

Definition at line 129 of file LISACODE-Geometry.cpp.

References Geometry::L0m, Geometry::move, Geometry::order_default, Geometry::rot0, Geometry::t0, Geometry::tRangeStoreDelay, and Geometry::tRangeStorePos.

Referenced by DispInfo(), GeometryFile::DispInfo(), GeometryAnalytic::DispInfo(), and Geometry::DispInfo().

10.11.3.3 void GeometryMLDC::DispInfo () [virtual]

Display informations.

Reimplemented from [Geometry](#).

Definition at line 166 of file LISACODE-GeometryMLDC.cpp.

References Approx, and Geometry::DispGeneralInfo().

10.11.3.4 Couple GeometryMLDC::exanom (int nb, double ts)**10.11.3.5 double Geometry::getL0 () [inline, inherited]**

Returns [L0m](#) attribute.

Definition at line 111 of file LISACODE-Geometry.h.

References Geometry::L0m.

10.11.3.6 double Geometry::gett0 () [inline, inherited]

Returns [t0](#) attribute.

Definition at line 113 of file LISACODE-Geometry.h.

References Geometry::t0.

Referenced by BackgroundGalactic::deltanu().

10.11.3.7 double Geometry::gettRangeStoreDelay () [inline, inherited]

Definition at line 116 of file LISACODE-Geometry.h.

References Geometry::tRangeStoreDelay.

10.11.3.8 double Geometry::gettRangeStorePos () [inline, inherited]

Definition at line 115 of file LISACODE-Geometry.h.

References Geometry::tRangeStorePos.

10.11.3.9 Vect Geometry::gposition (int nb, double t) [inherited]

Give direct value for position.

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 441 of file LISACODE-Geometry.cpp.

References **Geometry::move**, **Geometry::position()**, **Geometry::SCposStore**, **Geometry::tRangeStorePos**, and **Geometry::tStorePos**.

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

10.11.3.10 double Geometry::gtdelay (int em, int rec, int order, double trec) [inherited]

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 $(abs(trec-tStoreDelay))>tRangeStorePos)$

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

Returns:

$DelayStore[bs + em - 1]$

Definition at line 466 of file LISACODE-Geometry.cpp.

References **Geometry::DelayStore**, **Geometry::move**, **Geometry::tdelay()**, **Geometry::tRangeStoreDelay**, and **Geometry::tStoreDelay**.

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

10.11.3.11 void GeometryMLDC::init (double t0_n, double rot0_n, double L0m_n, int order_default_n, int move_n, double tStep)

Initializes attributes 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 :

- **arot** = $\frac{2\pi}{3}$
- **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 $i=1,2,3,4$ $\begin{cases} DelayStore[i - 1] = tdelay(i, i, mod((i + 1), 3) + 1, 2, 0) \\ DelayStore[i + 2] = tdelay(i, i, mod(i, 3) + 1, 2, 0) \end{cases}$
- **tRangeStorePos** = **tRangeStorePos_default**
- **tRangeStoreDelay** = $\min(tStep, tRangeStoreDelay_default)$

Definition at line 131 of file LISACODE-GeometryMLDC.cpp.

References Approx, Geometry::arot, crot, Geometry::DelayStore, e_mldc, Geometry::initGlobal(), position(), rot, Geometry::rot0, Geometry::SCposStore, sqrt_3, srot, and Geometry::tdelay().

Referenced by GeometryMLDC().

10.11.3.12 void Geometry::initGlobal (double **t0_n**, double **rot0_n**, double **L0m_n**, int **order_default_n**, int **move_n**, double **tStep**) [inherited]

Globla initialization.

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)
- **SCposStore** : $SCposStore[i] = position(i, 0)$ for $i=1,2,3,4$
- **DelayStore** : for $i=1,2,3,4$ $\begin{cases} DelayStore[i - 1] = tdelay(i, i, mod((i + 1), 3) + 1, 2, 0) \\ DelayStore[i + 2] = tdelay(i, i, mod(i, 3) + 1, 2, 0) \end{cases}$
- **tRangeStorePos** = **tRangeStorePos_default**
- **tRangeStoreDelay** = $\min(tStep, tRangeStoreDelay_default)$

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

References Geometry::L0m, Geometry::move, Geometry::order_default, Geometry::rot0, Geometry::t0, Geometry::tRangeStoreDelay, tRangeStoreDelay_default, Geometry::tRangeStorePos, tRangeStorePos_default, Geometry::tStoreDelay, and Geometry::tStorePos.

Referenced by Geometry::Geometry(), init(), GeometryFile::init(), and GeometryAnalytic::init().

10.11.3.13 Vect GeometryMLDC::position (int nb, double t) [virtual]

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

Reimplemented from [Geometry](#).

Definition at line 182 of file LISACODE-GeometryMLDC.cpp.

References [Geometry::alpha](#), [Approx](#), [au_m](#), [crot](#), [e_mldc](#), [Geometry::move](#), [omega](#), [Vect::p](#), [rot](#), [sqrt_3](#), [srot](#), and [Geometry::t0](#).

Referenced by [init\(\)](#).

10.11.3.14 void Geometry::sett0 (double t0_n) [inherited]

Definition at line 148 of file LISACODE-Geometry.cpp.

References [Geometry::t0](#).

10.11.3.15 void Geometry::settRangeStoreDelay (double tRangeStoreDelay_n) [inherited]

Definition at line 159 of file LISACODE-Geometry.cpp.

References [Geometry::tRangeStoreDelay](#).

10.11.3.16 void Geometry::settRangeStorePos (double tRangeStorePos_n) [inherited]

Definition at line 154 of file LISACODE-Geometry.cpp.

References [Geometry::tRangeStorePos](#).

10.11.3.17 double Geometry::tdelay (int em, int rec, int order, double trec) [inherited]

Get delay for specified order (1/2 -> order=1 and 1 -> order=2).

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:

$$\text{delay} = \sum_{i=0}^{\text{order}} c_i,$$

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](#),

$$\vec{r}_{ij} = \vec{r}_j - \vec{r}_i,$$

$$c_0 = \frac{\|\vec{r}_{ij}\|}{C},$$

$$\vec{n} = -\frac{\vec{r}_{ij}}{\|\vec{r}_{ij}\|},$$

$$\begin{aligned}
c_1 &= t_{ij} \cdot \vec{n} \cdot \vec{v}_j, \\
c_2 &= c_{21} + c_{22} + c_{23}, \\
c_{21} &= \frac{t_{ij}}{2} \cdot (\vec{v}_j^2 + (\vec{n} \cdot \vec{v}_j)^2), \\
c_{22} &= -\frac{t_{ij}^2}{2} \cdot C \cdot RSchw, \\
c_{23} &= -\frac{RSchw}{C} \cdot (1 + \gamma_u) \cdot \log\left(\frac{\sqrt{(C \cdot t_{ij} \cdot \vec{n} \cdot \vec{r}_{ij})^2 + r_i^2 + C \cdot t_{ij} + \vec{n} \cdot \vec{r}_i}}{\vec{n} \cdot \vec{r}_i + \|\vec{r}_i\|}\right).
\end{aligned}$$

[c_SI](#) and [gamma_u](#) constants are used.

Definition at line 247 of file LISACODE-Geometry.cpp.

References [c_SI](#), [gamma_u](#), [Vect::norme\(\)](#), [Geometry::order_default](#), [Geometry::position\(\)](#), [RSchw](#), [Vect::unit\(\)](#), and [Geometry::velocity\(\)](#).

Referenced by [Geometry::gtdelay\(\)](#), [init\(\)](#), [GeometryFile::init\(\)](#), [GeometryAnalytic::init\(\)](#), and [main\(\)](#).

10.11.3.18 double Geometry::tdelayOrderContribution (int em, int rec, int order, double trec) [inherited]

Get contribution of specified order (1/2 -> order=1 and 1 -> order=2) in delay.

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:

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

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](#),

rij 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](#),

$$\vec{r}_{ij} = \vec{r}_j - \vec{r}_i,$$

$$\vec{n} = -\frac{\vec{r}_{ij}}{\|\vec{r}_{ij}\|},$$

$$c_1 = t_{ij} \cdot \vec{n} \cdot \vec{v}_j,$$

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

$$c_{21} = \frac{t_{ij}}{2} \cdot (\vec{v}_j^2 + (\vec{n} \cdot \vec{v}_j)^2),$$

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

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

[c_SI](#) and [gamma_u](#) constants are used

Definition at line 324 of file LISACODE-Geometry.cpp.

References [c_SI](#), [gamma_u](#), [Vect::norme\(\)](#), [Geometry::position\(\)](#), [RSchw](#), [Vect::unit\(\)](#), and [Geometry::velocity\(\)](#).

10.11.3.19 Vect Geometry::VectNormal (double t) [inherited]

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

[position](#) method is used to set [SCposStore](#) attribute

Returns:
$$\frac{(\overrightarrow{\text{position}(2, t)} - \overrightarrow{\text{position}(3, t)}) \wedge (\overrightarrow{\text{position}(3, t)} - \overrightarrow{\text{position}(1, t)})}{\|(\overrightarrow{\text{position}(2, t)} - \overrightarrow{\text{position}(3, t)}) \wedge (\overrightarrow{\text{position}(3, t)} - \overrightarrow{\text{position}(1, t)})\|}$$

Definition at line 415 of file LISACODE-Geometry.cpp.

References [Vect::p](#), [Geometry::position\(\)](#), and [Vect::unit\(\)](#).

Referenced by [main\(\)](#).

10.11.3.20 Vect GeometryMLDC::velocity (int nb, double t) [virtual]

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

Reimplemented from [Geometry](#).

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

References [Geometry::alpha](#), [Approx](#), [au_m](#), [crot](#), [e_mldc](#), [Geometry::move](#), [omega](#), [Vect::p](#), [rot](#), [sqrt_3](#), [srot](#), and [Geometry::t0](#).

10.11.4 Member Data Documentation

10.11.4.1 Geometry::alpha [protected, inherited]

Orbital parameter.

Reimplemented in [GeometryAnalytic](#).

Referenced by [position\(\)](#), and [velocity\(\)](#).

10.11.4.2 double GeometryMLDC::Approx [protected]

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

Referenced by [DispInfo\(\)](#), [init\(\)](#), [position\(\)](#), and [velocity\(\)](#).

10.11.4.3 Geometry::arot [protected, inherited]

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

Reimplemented in [GeometryAnalytic](#).

Referenced by [init\(\)](#).

10.11.4.4 Geometry::cmu [protected, inherited]

Orbital constant.

Reimplemented in [GeometryAnalytic](#).

10.11.4.5 GeometryMLDC::crot [protected]

Satellites phase cosinus.

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

10.11.4.6 double Geometry::DelayStore[6] [protected, inherited]

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 90 of file LISACODE-Geometry.h.

Referenced by Geometry::gtdelay(), init(), GeometryFile::init(), and GeometryAnalytic::init().

10.11.4.7 Geometry::e [protected, inherited]

Eccentricity.

Reimplemented in [GeometryAnalytic](#).

Referenced by GeometryFile::position(), and GeometryFile::velocity().

10.11.4.8 double GeometryMLDC::e_mldc [protected]

excentricity for MLDC orbits

Definition at line 55 of file LISACODE-GeometryMLDC.h.

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

10.11.4.9 double Geometry::L0m [protected, inherited]

Nominal arm length in meters.

Nominal distance between two satellites.

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

Referenced by Geometry::DispGeneralInfo(), Geometry::getL0(), GeometryAnalytic::init(), and Geometry::initGlobal().

10.11.4.10 int [Geometry::move](#) [protected, inherited]

0 for [LISA](#) fixed, 1 for [LISA](#) moved (default)

Definition at line 71 of file LISACODE-Geometry.h.

Referenced by [Geometry::DispGeneralInfo\(\)](#), [GeometryAnalytic::exanom\(\)](#), [Geometry::gposition\(\)](#), [Geometry::gtdelay\(\)](#), [Geometry::initGlobal\(\)](#), [position\(\)](#), [GeometryFile::position\(\)](#), [velocity\(\)](#), and [GeometryFile::velocity\(\)](#).

10.11.4.11 [Geometry::nu](#) [protected, inherited]

Orbital angular constant.

Reimplemented in [GeometryAnalytic](#).

10.11.4.12 int [Geometry::order_default](#) [protected, inherited]

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

Definition at line 73 of file LISACODE-Geometry.h.

Referenced by [Geometry::DispGeneralInfo\(\)](#), [Geometry::initGlobal\(\)](#), and [Geometry::tdelay\(\)](#).

10.11.4.13 [GeometryMLDC::rot](#) [protected]

Satellites phase.

Referenced by [init\(\)](#), [position\(\)](#), and [velocity\(\)](#).

10.11.4.14 [Geometry::rot0](#) [protected, inherited]

Initial rotation (radians).

Referenced by [Geometry::DispGeneralInfo\(\)](#), [init\(\)](#), [GeometryAnalytic::init\(\)](#), and [Geometry::initGlobal\(\)](#).

10.11.4.15 Vect [Geometry::SCposStore\[3\]](#) [protected, inherited]

Stored positions.

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

Referenced by [Geometry::gposition\(\)](#), [init\(\)](#), [GeometryFile::init\(\)](#), and [GeometryAnalytic::init\(\)](#).

10.11.4.16 [Geometry::smu](#) [protected, inherited]

Orbital constant.

Reimplemented in [GeometryAnalytic](#).

10.11.4.17 double [GeometryMLDC::sqrt_3](#) [protected]

$\sqrt{3}$ for MLDC orbits

Definition at line 57 of file LISACODE-GeometryMLDC.h.

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

10.11.4.18 `Geometry::sqrtee` [protected, inherited]

Eccentricity derived constant.

Reimplemented in [GeometryAnalytic](#).

10.11.4.19 `GeometryMLDC::srot` [protected]

Satellites phase sinus.

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

10.11.4.20 `Geometry::t0` [protected, inherited]

Initial time (seconds).

Referenced by `Geometry::DispGeneralInfo()`, `GeometryAnalytic::exanom()`, `Geometry::gett0()`, `Geometry::initGlobal()`, `position()`, `GeometryFile::position()`, `Geometry::sett0()`, `velocity()`, and `GeometryFile::velocity()`.

10.11.4.21 `Geometry::tmu` [protected, inherited]

Orbital constant.

Reimplemented in [GeometryAnalytic](#).

10.11.4.22 `double Geometry::tRangeStoreDelay` [protected, inherited]

Last delay storage time.

Definition at line 94 of file LISACODE-Geometry.h.

Referenced by `Geometry::DispGeneralInfo()`, `Geometry::gettRangeStoreDelay()`, `Geometry::gtdelay()`, `Geometry::initGlobal()`, and `Geometry::settRangeStoreDelay()`.

10.11.4.23 `double Geometry::tRangeStorePos` [protected, inherited]

last position storage time.

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

Referenced by `Geometry::DispGeneralInfo()`, `Geometry::gettRangeStorePos()`, `Geometry::gposition()`, `Geometry::initGlobal()`, and `Geometry::settRangeStorePos()`.

10.11.4.24 `double Geometry::tStoreDelay` [protected, inherited]

Last delay computation time.

Definition at line 92 of file LISACODE-Geometry.h.

Referenced by `Geometry::gtdelay()`, and `Geometry::initGlobal()`.

10.11.4.25 double [Geometry::tStorePos](#) [protected, inherited]

last position computation time.

Definition at line 77 of file LISACODE-Geometry.h.

Referenced by [Geometry::gposition\(\)](#), and [Geometry::initGlobal\(\)](#).

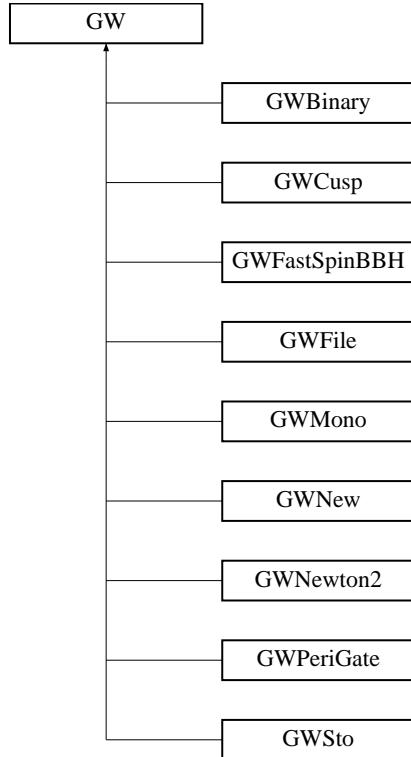
The documentation for this class was generated from the following files:

- [LISACODE-GeometryMLDC.h](#)
- [LISACODE-GeometryMLDC.cpp](#)

10.12 GW Class Reference

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

Inheritance diagram for GW::



10.12.1 Detailed Description

Gravitational Waves parameters are described in this class.

Definition at line 39 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 inputs for sky location.
- **GW (double Beta_n, double Lambda_n, double AnglPol_n)**
Constructs an instance and initializes it with inputs for sky location and polarization angle.
- **virtual ~GW ()**
Destructor

- virtual void `setParam` (int iP, double Param_n)

Sets parameters specified by iP :

 - 0 -> `Beta` : Ecliptic latitude (in radians)
 - 1 -> `Lambda` : Ecliptic longitude (in radians) ... [Redefine in derived class].

- virtual double `getParam` (int iP)

Return parameters specified by iP (see function for `setParam` the iP code).

- int `getNParam` ()
 - double `getBeta` () const
- void `setBeta` (double Beta_n)
- double `getLambda` () const
- void `setLambda` (double Lambda_n)
- vector< double > `getDirProp` () const
- void `setDirProp` (vector< double > DirProp_n)

Sets `DirProp`, `Lambda` and `Beta` attributes.

- double `getAnglPol` ()
 - void `setAnglPol` (double AnglPol_n)
- virtual void `init` ()

Initialization : ...

- virtual double `hp` (double t)

Return $h_+(t)$.

- virtual double `hc` (double t)

Return $h_\times(t)$.

- virtual void `DispTempVal` (double t, ostream *OutDisp)
 - void `CalculDirProp` ()

Computes components of the unit vectors `DirProp` from the angles `Lambda` and `Beta`.

Protected Attributes

- double `Beta`

Ecliptic latitude Source direction angle (in the heliocentric reference frame) in radians.
- double `Lambda`

Ecliptic longitude 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.
- int `NParam`

Number of parameters.

10.12.2 Constructor & Destructor Documentation

10.12.2.1 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 26 of file LISACODE-GW.cpp.

References [AnglPol](#), [Beta](#), [CalculDirProp\(\)](#), [DirProp](#), and [Lambda](#).

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

Constructs an instance and initializes it with inputs for sky location.

Inputs are checked:

$$\begin{aligned}\beta_n &\in [-\frac{\pi}{2}, -\frac{\pi}{2}] \\ \lambda_n &\in [0, 2 \cdot \pi]\end{aligned}$$

- [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](#).

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

Constructs an instance and initializes it with inputs for sky location and polarization angle.

Inputs are checked:

$$\begin{aligned}\beta_n &\in [-\frac{\pi}{2}, -\frac{\pi}{2}] \\ \lambda_n &\in [0, 2 \cdot \pi] \\ AnglPol_n &\in [0, 2 \cdot \pi]\end{aligned}$$

- [Beta](#) = [Beta_n](#)
- [Lambda](#) = [Lambda_n](#)
- [AnglPol](#) = [AnglPol_n](#)
- [CalculDirProp](#) method is called to fill [DirProp](#) attribute

Definition at line 67 of file LISACODE-GW.cpp.

References [AnglPol](#), [Beta](#), [CalculDirProp\(\)](#), [DirProp](#), and [Lambda](#).

10.12.2.4 GW::~GW () [virtual]

Destructor.

Definition at line 83 of file LISACODE-GW.cpp.

10.12.3 Member Function Documentation**10.12.3.1 void GW::CalculDirProp ()**

Computes components of the unit vectors [DirProp](#) from the angles [Lambda](#) and [Beta](#).

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

Definition at line 224 of file LISACODE-GW.cpp.

References [Beta](#), [DirProp](#), and [Lambda](#).

Referenced by [GW\(\)](#), [GWBinary::GWBinary\(\)](#), [GWFastSpinBBH::GWFastSpinBBH\(\)](#), [setBeta\(\)](#), and [setLambda\(\)](#).

10.12.3.2 void GW::DispTempVal (double *t*, ostream * *OutDisp*) [virtual]

Reimplemented in [GWBinary](#), [GWCusp](#), [GWFastSpinBBH](#), [GWFile](#), [GMono](#), [GNew](#), [GNewton2](#), [GPeriGate](#), and [GSto](#).

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

10.12.3.3 double GW::getAnglPol () [inline]

Definition at line 83 of file LISACODE-GW.h.

References [AnglPol](#).

10.12.3.4 double GW::getBeta () const [inline]

Definition at line 76 of file LISACODE-GW.h.

References [Beta](#).

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

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

References [DirProp](#).

10.12.3.6 double GW::getLambda () const [inline]

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

References [Lambda](#).

10.12.3.7 int GW::getNParam () [inline]

Definition at line 75 of file LISACODE-GW.h.

References NParam.

10.12.3.8 double GW::getParam (int iP) [virtual]

Return parameters specified by iP (see function for setParam the iP code).

Reimplemented in [GWBinary](#), [GWCusp](#), [GWFastSpinBBH](#), [GWFile](#), [GMono](#), [GNew](#), [GNewton2](#), [GPeriGate](#), and [GSto](#).

Definition at line 105 of file LISACODE-GW.cpp.

References Beta, and Lambda.

10.12.3.9 double GW::hc (double t) [virtual]

Return $h_x(t)$.

Virtual unused method. Returned value is constant : 0.

Reimplemented in [GWBinary](#), [GWCusp](#), [GWFastSpinBBH](#), [GWFile](#), [GMono](#), [GNew](#), [GNewton2](#), [GPeriGate](#), and [GSto](#).

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

10.12.3.10 double GW::hp (double t) [virtual]

Return $h_+(t)$.

Virtual unused method. Returned value is constant : 1.

Reimplemented in [GWBinary](#), [GWCusp](#), [GWFastSpinBBH](#), [GWFile](#), [GMono](#), [GNew](#), [GNewton2](#), [GPeriGate](#), and [GSto](#).

Definition at line 207 of file LISACODE-GW.cpp.

10.12.3.11 void GW::init () [virtual]

Initialization : ...

Nothing

Reimplemented in [GWBinary](#), [GWCusp](#), [GWFastSpinBBH](#), [GWFile](#), [GMono](#), [GNew](#), [GNewton2](#), [GPeriGate](#), and [GSto](#).

Definition at line 200 of file LISACODE-GW.cpp.

10.12.3.12 void GW::setAnglPol (double AnglPol_n)

* Input is checked:

$$\text{AnglPol}_n \in [0, 2 \cdot \pi]$$

Reimplemented in [GMono](#), and [GNew](#).

Definition at line 187 of file LISACODE-GW.cpp.

References AnglPol.

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

Input is checked:

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

[CalculDirProp](#) method is called to fill [DirProp](#) attribute

Definition at line 125 of file LISACODE-GW.cpp.

References Beta, and [CalculDirProp\(\)](#).

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

Sets [DirProp](#), [Lambda](#) and [Beta](#) attributes.

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

Computations: *DirProp_norm* is normalized *DirProp_n*

•

$$\beta = \arcsin(\text{DirProp}_{\text{norm}}[2])$$

•

$$\lambda = \text{mod}(\text{atan}(\frac{-\text{DirProp}_{\text{norm}}[1]}{-\text{DirProp}_{\text{norm}}[0]}), 2 \cdot \pi)$$

Definition at line 154 of file LISACODE-GW.cpp.

References Beta, DirProp, Lambda, and PRECISION.

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

Input is checked:

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

[CalculDirProp](#) method is called to set [DirProp](#) attribute

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

References [CalculDirProp\(\)](#), and Lambda.

10.12.3.16 void GW::setParam (int *iP*, double *Param_n*) [virtual]

Sets parameters specified by iP :

- 0 -> [Beta](#) : Ecliptic latitude (in radians)
- 1 -> [Lambda](#) : Ecliptic longitude (in radians) ... [Redefine in derived class].

Reimplemented in [GWBinary](#), [GWCusp](#), [GWFastSpinBBH](#), [GWFile](#), [GMono](#), [GNew](#), [GNewton2](#), [GPeriGate](#), and [GSto](#).

Definition at line 92 of file LISACODE-GW.cpp.

References Beta, and Lambda.

10.12.4 Member Data Documentation

10.12.4.1 double [GW::AnglPol](#) [protected]

Polarisation angle.

Reimplemented in [GMono](#), [GNew](#), and [GPeriGate](#).

Definition at line 51 of file LISACODE-GW.h.

Referenced by [getAnglPol\(\)](#), [GNewton2::getParam\(\)](#), [GWFile::getParam\(\)](#), [GWCusp::getParam\(\)](#), [GWBinary::getParam\(\)](#), [GW\(\)](#), [GWBinary::GWBinary\(\)](#), [GWBinary::setParam\(\)](#), [GNewton2::GNewton2\(\)](#), [GWFastSpinBBH::init\(\)](#), [setAnglPol\(\)](#), [GNewton2::setParam\(\)](#), [GWFile::setParam\(\)](#), [GWCusp::setParam\(\)](#), and [GWBinary::setParam\(\)](#).

10.12.4.2 [GW::Beta](#) [protected]

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

Referenced by [CalculDirProp\(\)](#), [getBeta\(\)](#), [GSto::getParam\(\)](#), [GPeriGate::getParam\(\)](#), [GNewton2::getParam\(\)](#), [GNew::getParam\(\)](#), [GMono::getParam\(\)](#), [GWFile::getParam\(\)](#), [GWFastSpinBBH::getParam\(\)](#), [GWCusp::getParam\(\)](#), [GWBinary::getParam\(\)](#), [getParam\(\)](#), [GW\(\)](#), [GWBinary::GWBinary\(\)](#), [GWBinary::setParam\(\)](#), [GNewton2::GNewton2\(\)](#), [GWFastSpinBBH::init\(\)](#), [setBeta\(\)](#), [setDirProp\(\)](#), [GSto::setParam\(\)](#), [GPeriGate::setParam\(\)](#), [GNewton2::setParam\(\)](#), [GNew::setParam\(\)](#), [GMono::setParam\(\)](#), [GWFile::setParam\(\)](#), [GWFastSpinBBH::setParam\(\)](#), [GWCusp::setParam\(\)](#), [GWBinary::setParam\(\)](#), and [setParam\(\)](#).

10.12.4.3 vector<double> [GW::DirProp](#) [protected]

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

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

Referenced by [CalculDirProp\(\)](#), [getDirProp\(\)](#), [GW\(\)](#), and [setDirProp\(\)](#).

10.12.4.4 [GW::Lambda](#) [protected]

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

Referenced by [CalculDirProp\(\)](#), [getLambda\(\)](#), [GSto::getParam\(\)](#), [GPeriGate::getParam\(\)](#), [GNewton2::getParam\(\)](#), [GNew::getParam\(\)](#), [GMono::getParam\(\)](#), [GWFile::getParam\(\)](#), [GWFastSpinBBH::getParam\(\)](#), [GWCusp::getParam\(\)](#), [GWBinary::getParam\(\)](#), [getParam\(\)](#), [GW\(\)](#), [GWBinary::GWBinary\(\)](#), [GWBinary::setParam\(\)](#), [GNewton2::GNewton2\(\)](#), [GWFastSpinBBH::init\(\)](#), [setDirProp\(\)](#), [setLambda\(\)](#), [GSto::setParam\(\)](#), [GPeriGate::setParam\(\)](#), [GNewton2::setParam\(\)](#), [GNew::setParam\(\)](#), [GMono::setParam\(\)](#), [GWFile::setParam\(\)](#), [GWFastSpinBBH::setParam\(\)](#), [GWCusp::setParam\(\)](#), [GWBinary::setParam\(\)](#), and [setParam\(\)](#).

10.12.4.5 int [GW::NParam](#) [protected]

Number of parameters.

Definition at line 53 of file LISACODE-GW.h.

Referenced by `getNParam()`, `GWBinary::GWBinary()`, `GWCusp::GWCusp()`, and `GWFastSpinBBH::GWFastSpinBBH()`.

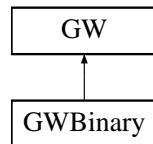
The documentation for this class was generated from the following files:

- [LISACODE-GW.h](#)
- [LISACODE-GW.cpp](#)

10.13 GWBinary Class Reference

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

Inheritance diagram for GWBinary::



10.13.1 Detailed Description

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

Definition at line 32 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
- void [setParam \(int iP, double Param_n\)](#)
Sets parameters specified by iP :
 - 0 -> **Beta** : β : Ecliptic latitude (in radians)
 - 1 -> **Lambda** : λ : Ecliptic longitude (in radians)
 - 2 -> **AnglPol** : ψ : Polarization angle (in radians)
 - 3 -> **M1** : m_1 Mass of object 1 (in solar masses)
 - 4 -> **M2** : m_2 Mass of object 2 (in solar masses)
 - 5 -> **r** : r Distance (in kpc)
 - 6 -> **phi0** : ϕ_0 Initial phase (in radians)
 - 7 -> **forb** : f_{orb} Orbital frequency (in Hertz)
 - 8 -> **inc** : i Inclination (in radians).
- double [getParam \(int iP\)](#)
Return parameters specified by iP (see function for [setParam](#) the iP code).
- double [getM1 \(\)](#)
- double [getM2 \(\)](#)
- double [getForb \(\)](#)
- double [getInc \(\)](#)
- double [getPhi0 \(\)](#)

- double `getDistance ()`
- void `setM1` (double `M1_n`)
- void `setM2` (double `M2_n`)
- void `setForb` (double `forb_n`)
- void `setInc` (double `inc_n`)
- void `setPhi0` (double `phi0_n`)
- void `setDistance` (double `r_n`)
- void `init ()`

Initialization : Sets polarized amplitudes attributes `Ap` and `Ac`.

- double `hp` (double `t`)

Return $h_+(t)$.

- double `hc` (double `t`)

Return $h_\times(t)$.

- void `DispTempVal` (double `t`, ostream *`OutDisp`)

Display ... [for specific use or debug].

- int `getNParam ()`

- double `getBeta () const`

- void `setBeta` (double `Beta_n`)

- double `getLambda () const`

- void `setLambda` (double `Lambda_n`)

- vector< double > `getDirProp () const`

- void `setDirProp` (vector< double > `DirProp_n`)

Sets `DirProp`, `Lambda` and `Beta` attributes.

- double `getAnglPol ()`

- void `setAnglPol` (double `AnglPol_n`)

- void `CalculDirProp ()`

Computes components of the unit vectors `DirProp` from the angles `Lambda` and `Beta`.

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`

- 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](#)
Ecliptic latitude Source direction angle (in the heliocentric reference frame) in radians.
- double [Lambda](#)
Ecliptic longitude 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.
- int [NParam](#)
Number of parameters.

10.13.2 Constructor & Destructor Documentation

10.13.2.1 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 34 of file LISACODE-GWBinary.cpp.

References [GW::AnglPol](#), [GW::Beta](#), [GW::CalculDirProp\(\)](#), [forb](#), [inc](#), [init\(\)](#), [kpc_m](#), [GW::Lambda](#), [M1](#), [M2](#), [MS_SI](#), [GW::NParam](#), [phi0](#), and [r](#).

10.13.2.2 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

$$\beta = \beta_n \text{ rad}$$

$$\lambda = \lambda_n \text{ rad}$$

$$AnglPol = AnglPol_n \text{ rad}$$

$$M_1 = bM_1 \cdot M_{Sun} \text{ kg}$$

$$M_2 = bM_2 \cdot M_{Sun} \text{ kg, where } M_{Sun} = 1.9889 \cdot 10^{30} \text{ kg}$$

$$f_{orb} = b f_{orb} \text{ Hz}$$

$$inc = binc \text{ rad}$$

$$\phi_0 = b\phi_0 \text{ rad}$$

$$r = br \cdot kpc_m \text{ m, where } kpc_m = 3.0856675807 \cdot 10^{19} \text{ m}$$

- [init](#) method is called to fill polarized amplitudes attributes [Ap](#) and [Ac](#).
- [CalculDirProp](#) method is called to fill [DirProp](#) attribute

Definition at line 76 of file LISACODE-GWBinary.cpp.

References [forb](#), [inc](#), [init\(\)](#), [kpc_m](#), [M1](#), [M2](#), [MS_SI](#), [GW::NParam](#), [phi0](#), and [r](#).

10.13.2.3 GWBinary::~GWBinary ()

Destructor.

Definition at line 109 of file LISACODE-GWBinary.cpp.

10.13.3 Member Function Documentation

10.13.3.1 void GW::CalculDirProp () [inherited]

Computes components of the unit vectors [DirProp](#) from the angles [Lambda](#) and [Beta](#).

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

Definition at line 224 of file LISACODE-GW.cpp.

References [GW::Beta](#), [GW::DirProp](#), and [GW::Lambda](#).

Referenced by [GW::GW\(\)](#), [GWBinary\(\)](#), [GWFastSpinBBH::GWFastSpinBBH\(\)](#), [GW::setBeta\(\)](#), and [GW::setLambda\(\)](#).

10.13.3.2 void GWBinary::DispTempVal (double *t*, ostream * *OutDisp*) [virtual]

Display ... [for specific use or debug].

Reimplemented from [GW](#).

Definition at line 287 of file LISACODE-GWBinary.cpp.

10.13.3.3 double GW::getAnglPol () [inline, inherited]

Definition at line 83 of file LISACODE-GW.h.

References [GW::AnglPol](#).

10.13.3.4 double GW::getBeta () const [inline, inherited]

Definition at line 76 of file LISACODE-GW.h.

References [GW::Beta](#).

10.13.3.5 vector<double> GW::getDirProp () const [inline, inherited]

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

References [GW::DirProp](#).

10.13.3.6 double GWBinary::getDistance () [inline]

Definition at line 93 of file LISACODE-GWBinary.h.

References [r](#).

10.13.3.7 double GWBinary::getForb () [inline]

Definition at line 90 of file LISACODE-GWBinary.h.

References [forb](#).

10.13.3.8 double GWBinary::getInc () [inline]

Definition at line 91 of file LISACODE-GWBinary.h.

References [inc](#).

10.13.3.9 double GW::getLambda () const [inline, inherited]

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

References [GW::Lambda](#).

10.13.3.10 double GWBinary::getM1 () [inline]

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

References M1.

10.13.3.11 double GWBinary::getM2 () [inline]

Definition at line 89 of file LISACODE-GWBinary.h.

References M2.

10.13.3.12 int GW::getNParam () [inline, inherited]

Definition at line 75 of file LISACODE-GW.h.

References GW::NParam.

10.13.3.13 double GWBinary::getParam (int iP) [virtual]

Return parameters specified by iP (see function for [setParam](#) the iP code).

Reimplemented from [GW](#).

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

References GW::AnglPol, GW::Beta, forb, inc, GW::Lambda, M1, M2, phi0, and r.

10.13.3.14 double GWBinary::getPhi0 () [inline]

Definition at line 92 of file LISACODE-GWBinary.h.

References phi0.

10.13.3.15 double GWBinary::hc (double t) [virtual]

Return $h_{\times}(t)$.

$$\text{Returned value} = A_c \cdot \sin(4 \cdot \pi \cdot f_{orb} \cdot t + \phi_0)$$

Reimplemented from [GW](#).

Definition at line 279 of file LISACODE-GWBinary.cpp.

References Ac, forb, and phi0.

10.13.3.16 double GWBinary::hp (double t) [virtual]

Return $h_{+}(t)$.

$$\text{Returned value} = A_p \cdot \cos(4 \cdot \pi \cdot f_{orb} \cdot t + \phi_0)$$

Reimplemented from [GW](#).

Definition at line 269 of file LISACODE-GWBinary.cpp.

References Ap, forb, and phi0.

10.13.3.17 void GWBinary::init () [virtual]

Initialization : Sets polarized amplitudes attributes [Ap](#) and [Ac](#).

Returns:

Gravitational wave polarized amplitudes :

$$A_p = A \cdot (1 + \cos^2(\text{inc}))$$

$$A_c = 2 \cdot A \cdot \cos(\text{inc})$$

where binary components orbital separation is :

$$R = \left(\frac{G \cdot (M_1 + M_2)}{(2 \cdot \pi \cdot f_{orb})^2} \right)^{\frac{1}{3}} \text{ m}$$

and gravitational wave intrinsic amplitude is:

$$A = \frac{2 \cdot G^2 \cdot M_1 \cdot M_2}{C^4 \cdot r \cdot R}$$

Reimplemented from [GW](#).

Definition at line 245 of file LISACODE-GWBinary.cpp.

References Ac, Ap, c_SI, forb, G_SI, inc, M1, M2, and r.

Referenced by GWBinary().

10.13.3.18 void GW::setAnglPol (double *AnglPol_n*) [inherited]

* Input is checked:

$$\text{AnglPol}_n \in [0, 2 \cdot \pi]$$

Reimplemented in [GWMono](#), and [GWNew](#).

Definition at line 187 of file LISACODE-GW.cpp.

References GW::AnglPol.

10.13.3.19 void GW::setBeta (double *Beta_n*) [inherited]

Input is checked:

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

[CalculDirProp](#) method is called to fill [DirProp](#) attribute

Definition at line 125 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](#), [Lambda](#) and [Beta](#) attributes.

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

Computations: *DirProp_{norm}* is normalized *DirProp_n*

-

$$\beta = \arcsin(\text{DirProp}_{\text{norm}}[2])$$

-

$$\lambda = \text{mod}(\text{atan}\left(\frac{-\text{DirProp}_{\text{norm}}[1]}{-\text{DirProp}_{\text{norm}}[0]}\right), 2 \cdot \pi)$$

Definition at line 154 of file LISACODE-GW.cpp.

References [GW::Beta](#), [GW::DirProp](#), [GW::Lambda](#), and [PRECISION](#).

10.13.3.21 void GWBinary::setDistance (double *r_n*)

Definition at line 223 of file LISACODE-GWBinary.cpp.

References *r*.

10.13.3.22 void GWBinary::setForb (double *forb_n*)

Definition at line 202 of file LISACODE-GWBinary.cpp.

References *forb*.

10.13.3.23 void GWBinary::setInc (double *inc_n*)

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

References *inc*.

10.13.3.24 void GW::setLambda (double *Lambda_n*) [inherited]

Input is checked:

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

[CalculDirProp](#) method is called to set [DirProp](#) attribute

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

References [GW::CalculDirProp\(\)](#), and [GW::Lambda](#).

10.13.3.25 void GWBinary::setM1 (double *M1_n*)

Definition at line 188 of file LISACODE-GWBinary.cpp.

References *M1*.

10.13.3.26 void GWBinary::setM2 (double M2_n)

Definition at line 195 of file LISACODE-GWBinary.cpp.

References M2.

10.13.3.27 void GWBinary::setParam (int iP, double Param_n) [virtual]

Sets parameters specified by iP :

- 0 -> [Beta](#) : β : Ecliptic latitude (in radians)
- 1 -> [Lambda](#) : λ : Ecliptic longitude (in radians)
- 2 -> [AnglPol](#) : ψ : Polarization angle (in radians)
- 3 -> [M1](#) : m_1 Mass of object 1 (in solar masses)
- 4 -> [M2](#) : m_2 Mass of object 2 (in solar masses)
- 5 -> [r](#) : r Distance (in kpc)
- 6 -> [phi0](#) : ϕ_0 Initial phase (in radians)
- 7 -> [forb](#) : f_{orb} :Orbital frequency (in Hertz)
- 8 -> [inc](#) : i Inclination (in radians).

Reimplemented from [GW](#).

Definition at line 118 of file LISACODE-GWBinary.cpp.

References [GW::AnglPol](#), [GW::Beta](#), [forb](#), [inc](#), [GW::Lambda](#), [M1](#), [M2](#), [phi0](#), and [r](#).

10.13.3.28 void GWBinary::setPhi0 (double phi0_n)

Definition at line 216 of file LISACODE-GWBinary.cpp.

References [phi0](#).

10.13.4 Member Data Documentation

10.13.4.1 double GWBinary::Ac [protected]

Polarized amplitude.

Definition at line 53 of file LISACODE-GWBinary.h.

Referenced by [hc\(\)](#), and [init\(\)](#).

10.13.4.2 double GW::AnglPol [protected, inherited]

Polarisation angle.

Reimplemented in [GWMono](#), [GWNew](#), and [GWPeriGate](#).

Definition at line 51 of file LISACODE-GW.h.

Referenced by GW::getAnglPol(), GWNewton2::getParam(), GWFile::getParam(), GWCusp::getParam(), getParam(), GW::GW(), GWBinary(), GWNewton2::GWNewton2(), GW::setAnglPol(), GWNewton2::setParam(), GWFile::setParam(), GWCusp::setParam(), and setParam().

10.13.4.3 double **GWBinary::Ap** [protected]

Polarized amplitude.

Definition at line 51 of file LISACODE-GWBinary.h.

Referenced by hp(), and init().

10.13.4.4 **GW::Beta** [protected, inherited]

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

Referenced by GW::CalculDirProp(), GW::getBeta(), GWSto::getParam(), GWPeriGate::getParam(), GWNewton2::getParam(), GWNew::getParam(), GWMono::getParam(), GWFile::getParam(), GWFastSpinBBH::getParam(), GWCusp::getParam(), getParam(), GW::getParam(), GW::GW(), GWBinary(), GWNewton2::GWNewton2(), GWFastSpinBBH::init(), GW::setBeta(), GW::setDirProp(), GWSto::setParam(), GWPeriGate::setParam(), GWNewton2::setParam(), GWNew::setParam(), GWMono::setParam(), GWFile::setParam(), GWFastSpinBBH::setParam(), GWCusp::setParam(), setParam(), and GW::setParam().

10.13.4.5 vector<double> **GW::DirProp** [protected, inherited]

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

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

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

10.13.4.6 double **GWBinary::forb** [protected]

orbital frequency (Hertz)

Definition at line 41 of file LISACODE-GWBinary.h.

Referenced by getForb(), getParam(), GWBinary(), hc(), hp(), init(), setForb(), and setParam().

10.13.4.7 double **GWBinary::inc** [protected]

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

Definition at line 43 of file LISACODE-GWBinary.h.

Referenced by getInc(), getParam(), GWBinary(), init(), setInc(), and setParam().

10.13.4.8 **GW::Lambda** [protected, inherited]

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

Referenced by GW::CalculDirProp(), GW::getLambda(), GWSto::getParam(), GWPeriGate::getParam(), GWNewton2::getParam(), GWNew::getParam(), GWMono::getParam(), GWFile::getParam(),

GWFastSpinBBH::getParam(), GWCusp::getParam(), getParam(), GW::getParam(), GW::GW(), GWBinary(), GWNewton2::GWNewton2(), GWFastSpinBBH::init(), GW::setDirProp(), GW::setLambda(), GWSto::setParam(), GWPeriGate::setParam(), GWNewton2::setParam(), GWNew::setParam(), GWMono::setParam(), GWFile::setParam(), GWFastSpinBBH::setParam(), GWCusp::setParam(), setParam(), and GW::setParam().

10.13.4.9 double [GWBinary::M1](#) [protected]

First star mass (in solar masses).

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

Referenced by getM1(), getParam(), GWBinary(), init(), setM1(), and setParam().

10.13.4.10 double [GWBinary::M2](#) [protected]

Second star mass (in solar masses).

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

Referenced by getM2(), getParam(), GWBinary(), init(), setM2(), and setParam().

10.13.4.11 int [GW::NParam](#) [protected, inherited]

Number of parameters.

Definition at line 53 of file LISACODE-GW.h.

Referenced by GW::getNParam(), GWBinary(), GWCusp::GWCusp(), and GWFastSpinBBH::GWFastSpinBBH().

10.13.4.12 double [GWBinary::phi0](#) [protected]

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

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

Referenced by getParam(), getPhi0(), GWBinary(), hc(), hp(), setParam(), and setPhi0().

10.13.4.13 double [GWBinary::r](#) [protected]

Distance to the source in kpc (kiloparsec).

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

Referenced by getDistance(), getParam(), GWBinary(), init(), setDistance(), and setParam().

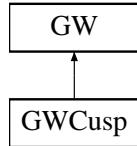
The documentation for this class was generated from the following files:

- [LISACODE-GWBinary.h](#)
- [LISACODE-GWBinary.cpp](#)

10.14 GWCusp Class Reference

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

Inheritance diagram for GWCusp::



10.14.1 Detailed Description

Gravitational Waves instantaneous parameters h_+ and h_\times are described in this class.

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

Public Member Functions

- [GWCusp \(\)](#)
- [GWCusp \(double TStep_n, double Tobs_n, double Tpad_n\)](#)
- [GWCusp \(double Beta_n, double Lambda_n, double AnglPol_n, double Amplitude_n, double CentralTime_n, double MaximumFrequency_n, double TStep_n, double Tobs_n, double Tpad_n\)](#)
- [~GWCusp \(\)](#)
- void [setParam \(int iP, double Param_n\)](#)

Sets parameters specified by iP :

- 0 -> [Beta](#) : β : Ecliptic latitude (in radians)
- 1 -> [Lambda](#) : λ : Ecliptic longitude (in radians)
- 2 -> [AnglPol](#) : ψ : Polarization angle (in radians)
- 3 -> [Amplitude](#) : A : Amplitude (in $\text{Hz}^{1/3}$)
- 4 -> [CentralTime](#) : t_{cen} : Central time (in seconds)
- 5 -> [MaximumFrequency](#) : f_{max} : Maximum frequency (in Hertz).

- double [getParam \(int iP\)](#)

Return parameters specified by iP (see function for [setParam](#) the iP code).

- void [initConst \(\)](#)

Initialization of constants.

- void [AllocMem \(\)](#)

Allocation of memories.

- void [init \(\)](#)

Initialization : Compute [GW](#) strain in Fourier domain and compute the inverse Fourier transform.

- double [hp \(double t\)](#)

Return $h_+(t)$.

- double [hc \(double t\)](#)

Return $h_{\times}(t)$.

- void **DispTempVal** (double t, ostream *OutDisp)
- int **getNParam** ()
- double **getBeta** () const
- void **setBeta** (double Beta_n)
- double **getLambda** () const
- void **setLambda** (double Lambda_n)
- vector< double > **getDirProp** () const
- void **setDirProp** (vector< double > DirProp_n)

*Sets **DirProp**, **Lambda** and **Beta** attributes.*

- double **getAnglPol** ()
- void **setAnglPol** (double AnglPol_n)
- void **CalculDirProp** ()

*Computes components of the unit vectors **DirProp** from the angles **Lambda** and **Beta**.*

Protected Attributes

- double **Amplitude**
Amplitude in ().
- double **CentralTime**
Central time.
- double **MaximumFrequency**
Maximum frequency.
- double **Tobs**
Time offset Duration of observation.
- double **Tstep**
Time step.
- double **T0**
Time for ...
- double **Tpad**
Time for ...
- long **NtDat**
Number of samples.
- double * **th**
Time data of $h(t)$.
- long **NfDat**
Number of samples.

- fftw_complex * **fh**

Fourier data of $h(f)$.

- fftw_plan **FTRevPlan**
- double **Tburst**
- double **Tback**
- double **Flow**
- double **mTPI**
- double **m43**
- double **Beta**

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

- double **Lambda**

Ecliptic longitude 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.

- int **NParam**

Number of parameters.

10.14.2 Constructor & Destructor Documentation

10.14.2.1 GWCusp::GWCusp ()

Definition at line 21 of file LISACODE-GWCusp.cpp.

References AllocMem(), Amplitude, CentralTime, init(), initConst(), MaximumFrequency, GW::NParam, Tobs, Tpad, and Tstep.

10.14.2.2 GWCusp::GWCusp (double *TStep_n*, double *Tobs_n*, double *Tpad_n*)

Definition at line 39 of file LISACODE-GWCusp.cpp.

References AllocMem(), Amplitude, CentralTime, init(), initConst(), MaximumFrequency, GW::NParam, Tobs, Tpad, and Tstep.

10.14.2.3 GWCusp::GWCusp (double *Beta_n*, double *Lambda_n*, double *AnglPol_n*, double *Amplitude_n*, double *CentralTime_n*, double *MaximumFrequency_n*, double *TStep_n*, double *Tobs_n*, double *Tpad_n*)

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

References AllocMem(), Amplitude, CentralTime, init(), initConst(), MaximumFrequency, GW::NParam, Tobs, Tpad, and Tstep.

10.14.2.4 **GWCusp::~GWCusp ()**

Definition at line 83 of file LISACODE-GWCusp.cpp.

10.14.3 Member Function Documentation

10.14.3.1 **void GWCusp::AllocMem ()**

Allocation of memories.

Definition at line 156 of file LISACODE-GWCusp.cpp.

References fh, FTRevPlan, NfDat, NtDat, th, Tobs, and Tstep.

Referenced by GWCusp().

10.14.3.2 **void GW::CalculDirProp () [inherited]**

Computes components of the unit vectors [DirProp](#) from the angles [Lambda](#) and [Beta](#).

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

Definition at line 224 of file LISACODE-GW.cpp.

References [GW::Beta](#), [GW::DirProp](#), and [GW::Lambda](#).

Referenced by [GW::GW\(\)](#), [GWBinary::GWBinary\(\)](#), [GWFastSpinBBH::GWFastSpinBBH\(\)](#), [GW::setBeta\(\)](#), and [GW::setLambda\(\)](#).

10.14.3.3 **void GWCusp::DispTempVal (double t, ostream * OutDisp) [virtual]**

Reimplemented from [GW](#).

Definition at line 252 of file LISACODE-GWCusp.cpp.

10.14.3.4 **double GW::getAnglPol () [inline, inherited]**

Definition at line 83 of file LISACODE-GW.h.

References [GW::AnglPol](#).

10.14.3.5 **double GW::getBeta () const [inline, inherited]**

Definition at line 76 of file LISACODE-GW.h.

References [GW::Beta](#).

10.14.3.6 **vector<double> GW::getDirProp () const [inline, inherited]**

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

References [GW::DirProp](#).

10.14.3.7 double GW::getLambda () const [inline, inherited]

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

References GW::Lambda.

10.14.3.8 int GW::getNParam () [inline, inherited]

Definition at line 75 of file LISACODE-GW.h.

References GW::NParam.

10.14.3.9 double GWCusp::getParam (int iP) [virtual]

Return parameters specified by iP (see function for [setParam](#) the iP code).

Reimplemented from [GW](#).

Definition at line 117 of file LISACODE-GWCusp.cpp.

References Amplitude, GW::AnglPol, GW::Beta, CentralTime, GW::Lambda, and MaximumFrequency.

10.14.3.10 double GWCusp::hc (double t) [virtual]

Return $h_x(t)$.

returned value = $Amplhc \cdot \sin(2 \cdot \pi \cdot Freq \cdot t + Phi0hc)$

Reimplemented from [GW](#).

Definition at line 246 of file LISACODE-GWCusp.cpp.

10.14.3.11 double GWCusp::hp (double t) [virtual]

Return $h_+(t)$.

returned value = $Amplhp \cdot \sin(2 \cdot \pi \cdot Freq \cdot t + Phi0hp)$

Reimplemented from [GW](#).

Definition at line 229 of file LISACODE-GWCusp.cpp.

References NtDat, T0, Tburst, th, and Tstep.

10.14.3.12 void GWCusp::init () [virtual]

Initialization : Compute [GW](#) strain in Fourier domain and compute the inverse Fourier transform.

Reimplemented from [GW](#).

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

References Amplitude, CentralTime, fh, Flow, FTRevPlan, m43, MaximumFrequency, mTPI, NfDat, NtDat, T0, Tback, Tburst, Tpad, and Tstep.

Referenced by [GWCusp\(\)](#).

10.14.3.13 void GWCusp::initConst ()

Initialization of constants.

Definition at line 148 of file LISACODE-GWCusp.cpp.

References Flow, m43, and mTPI.

Referenced by GWCusp().

10.14.3.14 void GW::setAnglPol (double *AnglPol_n*) [inherited]

* Input is checked:

$$\text{AnglPol}_n \in [0, 2 \cdot \pi]$$

Reimplemented in [GWMono](#), and [GWNew](#).

Definition at line 187 of file LISACODE-GW.cpp.

References GW::AnglPol.

10.14.3.15 void GW::setBeta (double *Beta_n*) [inherited]

Input is checked:

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

[CalculDirProp](#) method is called to fill [DirProp](#) attribute

Definition at line 125 of file LISACODE-GW.cpp.

References GW::Beta, and GW::CalculDirProp().

10.14.3.16 void GW::setDirProp (vector< double > *DirProp_n*) [inherited]

Sets [DirProp](#), [Lambda](#) and [Beta](#) attributes.

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

Computations: *DirProp_{norm}* is normalized *DirProp_n*

•

$$\beta = \arcsin(\text{DirProp}_{\text{norm}}[2])$$

•

$$\lambda = \text{mod}(\text{atan}(\frac{-\text{DirProp}_{\text{norm}}[1]}{-\text{DirProp}_{\text{norm}}[0]}), 2 \cdot \pi)$$

Definition at line 154 of file LISACODE-GW.cpp.

References GW::Beta, GW::DirProp, GW::Lambda, and PRECISION.

10.14.3.17 void GW::setLambda (double *Lambda_n*) [inherited]

Input is checked:

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

[CalculDirProp](#) method is called to set [DirProp](#) attribute

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

References [GW::CalculDirProp\(\)](#), and [GW::Lambda](#).

10.14.3.18 void GWCusp::setParam (int iP, double Param_n) [virtual]

Sets parameters specified by iP :

- 0 -> [Beta](#) : β : Ecliptic latitude (in radians)
- 1 -> [Lambda](#) : λ : Ecliptic longitude (in radians)
- 2 -> [AnglPol](#) : ψ : Polarization angle (in radians)
- 3 -> [Amplitude](#) : A : Amplitude (in $Hz^{1/3}$)
- 4 -> [CentralTime](#) : t_{cen} : Central time (in seconds)
- 5 -> [MaximumFrequency](#) : f_{max} : Maximum frequency (in Hertz).

Reimplemented from [GW](#).

Definition at line 92 of file LISACODE-GWCusp.cpp.

References [Amplitude](#), [GW::AnglPol](#), [GW::Beta](#), [CentralTime](#), [GW::Lambda](#), and [MaximumFrequency](#).

10.14.4 Member Data Documentation

10.14.4.1 double GWCusp::Amplitude [protected]

Amplitude in () .

$$A = \frac{G\mu L^{2/3}}{D_L} \text{ with}$$

- G : Newton constant
- μ : String mass per unit length
- L : Size of the feature that produces cusp
- D_L : Luminosity distance

Definition at line 51 of file LISACODE-GWCusp.h.

Referenced by [getParam\(\)](#), [GWCusp\(\)](#), [init\(\)](#), and [setParam\(\)](#).

10.14.4.2 double GW::AnglPol [protected, inherited]

Polarisation angle.

Reimplemented in [GMono](#), [GNew](#), and [GPeriGate](#).

Definition at line 51 of file LISACODE-GW.h.

Referenced by [GW::getAnglPol\(\)](#), [GNewton2::getParam\(\)](#), [GWFile::getParam\(\)](#), [getParam\(\)](#), [GWBinary::getParam\(\)](#), [GW::GW\(\)](#), [GWBinary::GWBinary\(\)](#), [GNewton2::GNewton2\(\)](#), [GW::setAnglPol\(\)](#), [GNewton2::setParam\(\)](#), [GWFile::setParam\(\)](#), [setParam\(\)](#), and [GWBinary::setParam\(\)](#).

10.14.4.3 GW::Beta [protected, inherited]

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

Referenced by GW::CalculDirProp(), GW::getBeta(), GWSto::getParam(), GWPeriGate::getParam(), GWNewton2::getParam(), GWNew::getParam(), GWMono::getParam(), GWFfile::getParam(), GWFastSpinBBH::getParam(), getParam(), GWBinary::getParam(), GW::getParam(), GW::GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), GWFastSpinBBH::init(), GW::setBeta(), GW::setDirProp(), GWSto::setParam(), GWPeriGate::setParam(), GWNewton2::setParam(), GWNew::setParam(), GWMono::setParam(), GWFfile::setParam(), GWFastSpinBBH::setParam(), setParam(), GWBinary::setParam(), and GW::setParam().

10.14.4.4 double GWCusp::CentralTime [protected]

Central time.

Definition at line 53 of file LISACODE-GWCusp.h.

Referenced by getParam(), GWCusp(), init(), and setParam().

10.14.4.5 vector<double> GW::DirProp [protected, inherited]

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

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

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

10.14.4.6 fftw_complex* GWCusp::fh [protected]

Fourier data of $h(f)$.

Definition at line 76 of file LISACODE-GWCusp.h.

Referenced by AllocMem(), and init().

10.14.4.7 double GWCusp::Flow [protected]

Definition at line 83 of file LISACODE-GWCusp.h.

Referenced by init(), and initConst().

10.14.4.8 fftw_plan GWCusp::FTRevPlan [protected]

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

Referenced by AllocMem(), and init().

10.14.4.9 GW::Lambda [protected, inherited]

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

Referenced by GW::CalculDirProp(), GW::getLambda(), GWSto::getParam(), GWPeriGate::getParam(), GWNewton2::getParam(), GWNew::getParam(), GWMono::getParam(), GWFfile::getParam(),

GWFastSpinBBH::getParam(), getParam(), GWBinary::getParam(), GW::getParam(), GW::GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), GWFastSpinBBH::init(), GW::setDirProp(), GW::setLambda(), GWSto::setParam(), GWPeriGate::setParam(), GWNewton2::setParam(), GWNew::setParam(), GMono::setParam(), GWFile::setParam(), GWFastSpinBBH::setParam(), setParam(), GWBinary::setParam(), and GW::setParam().

10.14.4.10 double **GWCusp::m43** [protected]

Definition at line 83 of file LISACODE-GWCusp.h.

Referenced by init(), and initConst().

10.14.4.11 double **GWCusp::MaximumFrequency** [protected]

Maximum frequency.

Definition at line 55 of file LISACODE-GWCusp.h.

Referenced by getParam(), GWCusp(), init(), and setParam().

10.14.4.12 double **GWCusp::mTPI** [protected]

Definition at line 83 of file LISACODE-GWCusp.h.

Referenced by init(), and initConst().

10.14.4.13 long **GWCusp::NfDat** [protected]

Number of samples.

Definition at line 74 of file LISACODE-GWCusp.h.

Referenced by AllocMem(), and init().

10.14.4.14 int **GW::NParam** [protected, inherited]

Number of parameters.

Definition at line 53 of file LISACODE-GW.h.

Referenced by GW::getNParam(), GWBinary::GWBinary(), GWCusp(), and GWFastSpinBBH::GWFastSpinBBH().

10.14.4.15 long **GWCusp::NtDat** [protected]

Number of samples.

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

Referenced by AllocMem(), hp(), and init().

10.14.4.16 double **GWCusp::T0** [protected]

Time for ...

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

Referenced by hp(), and init().

10.14.4.17 double GWCusp::Tback [protected]

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

Referenced by init().

10.14.4.18 double GWCusp::Tburst [protected]

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

Referenced by hp(), and init().

10.14.4.19 double* GWCusp::th [protected]

Time data of $h(t)$.

Definition at line 72 of file LISACODE-GWCusp.h.

Referenced by AllocMem(), and hp().

10.14.4.20 double GWCusp::Tobs [protected]

Time offset Duration of observation.

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

Referenced by AllocMem(), and GWCusp().

10.14.4.21 double GWCusp::Tpad [protected]

Time for ...

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

Referenced by GWCusp(), and init().

10.14.4.22 double GWCusp::Tstep [protected]

Time step.

Definition at line 64 of file LISACODE-GWCusp.h.

Referenced by AllocMem(), GWCusp(), hp(), and init().

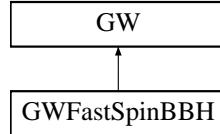
The documentation for this class was generated from the following files:

- [LISACODE-GWCusp.h](#)
- [LISACODE-GWCusp.cpp](#)

10.15 GWFastSpinBBH Class Reference

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

Inheritance diagram for GWFastSpinBBH::



10.15.1 Detailed Description

Gravitational Waves parameters for a spinning black holes binary system are defined in this class.

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

Public Member Functions

- [GWFastSpinBBH \(\)](#)
Constructs an instance and initializes it with default values.
- [GWFastSpinBBH \(double Toffset_n, double Tobs_n\)](#)
Constructs an instance and initializes it with values in arguments.
- [GWFastSpinBBH \(double Beta_n, double Lambda_n, double Mass1_n, double Mass2_n, double CoalescenceTime_n, double Distance_n, double Spin1_n, double Spin2_n, double PolarAngleOfSpin1_n, double PolarAngleOfSpin2_n, double AzimuthalAngleOfSpin1_n, double AzimuthalAngleOfSpin2_n, double PhaseAtCoalescence_n, double InitialPolarAngleL_n, double InitialAzimuthalAngleL_n, double AmplPNorder_n, double Toffset_n, double Tobs_n, double TaperApplied_n, double TaperSteepness_n, double Rmin_n\)](#)
Constructs an instance and initializes it with values in arguments.
- [~GWFastSpinBBH \(\)](#)
Destructor.
- void [setParam \(int iP, double Param_n\)](#)
Sets parameters specified by . Value corresponding to :
 - 0 -> **Beta** : β : Ecliptic latitude (in radians)
 - 1 -> **Lambda** : λ : Ecliptic longitude (in radians)
 - 2 -> **m1** : m_1 : Mass of object 1 (in solar masses)
 - 3 -> **m2** : m_2 : Mass of object 2 (in solar masses)
 - 4 -> **tc** : t_{coal} : (in seconds)
 - 5 -> **DL** : r : Distance (in kpc)
 - 6 -> **chi1** : S_1 : Spin of object 1
 - 7 -> **chi2** : S_2 : Spin of object 2
 - 8 -> **PolarAngleOfSpin1** : θ_{S_1} : Polar angle of spin S1 (in radians)
 - 9 -> **PolarAngleOfSpin2** : θ_{S_2} : Polar angle of spin S2 (in radians)
 - 10 -> **AzimuthalAngleOfSpin1** : φ_{S_1} : Azimuthal angle of spin S1 (in radians)

- *I1 -> AzimuthalAngleOfSpin2 : φ_{S_2} : Azimuthal angle of spin S2 (in radians)*
 - *I2 -> phic : ϕ_c : Phase at coalescence (in radians)*
 - *I2 -> InitialPolarAngleL : $\theta_{L,0}$: Initial polar angle of orbital momentum (in radians)*
 - *I4 -> InitialPolarAngleL : $\varphi_{L,0}$: Initial azimuthal angle of orbital momentum (in radians).*
- double **getParam** (int iP)

*Return parameters specified by iP (see function for **setParam** the iP code).*
 - double **getMass1** ()
 - double **getMass2** ()
 - double **getDistance** ()
 - void **setMass1** (double M1_n)
 - void **setMass2** (double M2_n)
 - void **setDistance** (double r_n)
 - void **setAmpIPNorder** (double AmplPNorder_n)
 - void **initNULL** ()

Initialization of all pointers at NULL.
 - void **AllocMemory** ()

Allocation of memory.
 - void **initRK** ()

Set Runge-Kutta parameters.
 - void **init** ()

Initialisation (integration of precession and spline).
 - void **AmpShared** (double t)

Compute shared data for calculation of amplitude.
 - double **hp** (double t)

Compute wave amplitude component hplus.
 - double **hc** (double t)

Compute wave amplitude component hcross.
 - void **DispTempVal** (double t, ostream *OutDisp)

Display temporary values.
 - double **Freq** (double t, double **Mtot**, double **Mchirp**, double **eta**, double beta, double sigma, double **tc**)

Compute frequency.
 - void **rkckstep** (double outputvals[], double fourthorderoutputvals[], double *outputthomas, double h, double currentvals[], double currentthomas, double t, double **m1**, double **m2**, double **Mtot**, double **Mchirp**, double **mu**, double **eta**, double **chi1**, double **chi2**, double N[], double **tc**)

Compute one step in integration using Runge Kutta method at order 6.
 - void **update** (int j, double A, double h, double intvals[], double t, double **m1**, double **m2**, double **Mtot**, double **Mchirp**, double **mu**, double **eta**, double **chi1**, double **chi2**, double N[], double **tc**, double **k, double kthomas[])

Calculate derivatives for one step of the Runge-Kutta method.

- void **calcderivals** (double derivvals[], double inputs[], double r, double **m1**, double **m2**, double **Mtot**, double **mu**, double **chi1**, double **chi2**)

Calculate derivatives of S1, S2 and L.

- double **S1xdot** (double inputs[], double r, double **m1**, double **m2**, double **Mtot**, double **mu**, double **chi1**, double **chi2**)

Calculate derivative of component x of S1.

- double **S1ydot** (double inputs[], double r, double **m1**, double **m2**, double **Mtot**, double **mu**, double **chi1**, double **chi2**)

Calculate derivative of component y of S1.

- double **S1zdot** (double inputs[], double r, double **m1**, double **m2**, double **Mtot**, double **mu**, double **chi1**, double **chi2**)

Calculate derivative of component z of S1.

- double **S2xdot** (double inputs[], double r, double **m1**, double **m2**, double **Mtot**, double **mu**, double **chi1**, double **chi2**)

Calculate derivative of component x of S2.

- double **S2ydot** (double inputs[], double r, double **m1**, double **m2**, double **Mtot**, double **mu**, double **chi1**, double **chi2**)

Calculate derivative of component y of S2.

- double **S2zdot** (double inputs[], double r, double **m1**, double **m2**, double **Mtot**, double **mu**, double **chi1**, double **chi2**)

Calculate derivative of component z of S2.

- double **calcLdotS1** (double inputs[])

Calculate scalar product L.S1.

- double **calcLdotS2** (double inputs[])

Calculate scalar product L.S1.

- double **calcSdotS** (double inputs[])

Calculate scalar product L.S2.

- double **calcLdotN** (double inputs[], double nvec[])

Calculate scalar product L.N.

- void **calcLcrossN** (double output[], double inputs[], double nvec[])

Calculate scalar product LxN.

- void **spline** (double *x, double *y, int n, double yp1, double ypn, double *y2)

Spline data : compute y2.

- int **getNParam** ()

- double **getBeta** () const

- void `setBeta` (double Beta_n)
- double `getLambda` () const
- void `setLambda` (double Lambda_n)
- vector< double > `getDirProp` () const
- void `setDirProp` (vector< double > DirProp_n)

Sets DirProp, Lambda and Beta attributes.

- double `getAnglPol` ()
- void `setAnglPol` (double AnglPol_n)
- void `CalculDirProp` ()

Computes components of the unit vectors DirProp from the angles Lambda and Beta.

Protected Attributes

- double `m1`

Mass of object 1 (in solar masses).

- double `m2`

Mass of object 2 (in solar masses).

- double `tc`

Time of coalescence (in seconds).

- double `DL`

Distance between source and detector (in kiloparsec).

- double `chi1`

Spin of object 1.

- double `chi2`

Spin of object 2.

- double `PolarAngleOfSpin1`

Polar angle of spin S1.

- double `PolarAngleOfSpin2`

Polar angle of spin S2.

- double `AzimuthalAngleOfSpin1`

Azimuthal angle of spin S1.

- double `AzimuthalAngleOfSpin2`

Azimuthal angle of spin S1.

- double `phic`

Phase at coalescence.

- double `InitialPolarAngleL`

Initial polar angle of orbital momentum .

- double **InitialAzimuthalAngleL**

Initial azimuthal angle of orbital momentum .

- double **TaperApplied**
- double **AmplPNorder**

High Post-Newtonian order in calculation (maximum 2 PN).

- double **Toffset**

Time offset.

- double **Tobs**

Duration of observation.

- double **Rmin**

- double **TaperSteepness**

- double **Mtot**

Total mass.

- double **dm**

Difference of mass.

- double **mu**

Reduced mass.

- double **eta**

Mass ratio and it's inverse.

- double **etain**

Mass ratio and it's inverse.

- double **Mchirp**

Chirp mass.

- double **Amp**

- double **sintheta**

- double **costheta**

- double **gamma0**

- double **tmax**

- double **xmax**

- double **xold**

- double **phi**

- double **Phi0**

- double **Phi10**

- double **p15**

- double **p150**

- double **p20**

- double **p200**

- double **sf**

- double `sx`
- double `ci`
- double `psi`
- double `Ax`
- double `DeltaPhase2L`
- double `timeCur`

Last time of compute amplitude.

- int `idxcur`

Current index for the interpolation and its maximal value.

- int `idxm`

Current index for the interpolation and its maximal value.

- double `c2psi`

sin and cos of twice the polarisaion angle for calculation of amplitude components

- double `s2psi`

sin and cos of twice the polarisaion angle for calculation of amplitude components

- double `shp`

h_{+} and h_x in the source reference frame

- double `shc`

h_{+} and h_x in the source reference frame

- double * `timevec`

- double * `mulvec`

- double * `sphilvec`

- double * `cphilvec`

- double * `betavec`

- double * `sigmavec`

- double * `thomasvec`

- double * `muly2`

- double * `sphily2`

- double * `cphily2`

- double * `betay2`

- double * `sigmay2`

- double * `thomasy2`

- double * `uspline`

- double ** `krk`

Runge-Kutta parameters.

- int `RK_VECLENGTH`

- int `RK_VECLENGTH_Old`

- double `A2`

- double `A3`

- double `A4`

- double `A5`

- double `A6`

- double [B21](#)
- double [B31](#)
- double [B32](#)
- double [B41](#)
- double [B42](#)
- double [B43](#)
- double [B51](#)
- double [B52](#)
- double [B53](#)
- double [B54](#)
- double [B61](#)
- double [B62](#)
- double [B63](#)
- double [B64](#)
- double [B65](#)
- double [C1](#)
- double [C2](#)
- double [C3](#)
- double [C4](#)
- double [C5](#)
- double [C6](#)
- double [D1](#)
- double [D2](#)
- double [D3](#)
- double [D4](#)
- double [D5](#)
- double [D6](#)
- double [RK_H](#)
- double [RK_EPS](#)
- double [Beta](#)

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

- double [Lambda](#)

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

- vector< double > [DirProp](#)

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

- double [AngIPol](#)

Polarisation angle.

- int [NParam](#)

Number of parameters.

10.15.2 Constructor & Destructor Documentation

10.15.2.1 `GWFastSpinBBH::GWFastSpinBBH ()`

Constructs an instance and initializes it with default values.

Definition at line 20 of file LISACODE-GWFastSpinBBH.cpp.

References AllocMemory(), AmplPNorder, GW::CalculDirProp(), init(), initNULL(), initRK(), GW::NParam, Rmin, setParam(), TaperApplied, TaperSteepness, Tobs, and Toffset.

10.15.2.2 `GWFastSpinBBH::GWFastSpinBBH (double Toffset_n, double Tobs_n)`

Constructs an instance and initializes it with values in arguments.

Definition at line 60 of file LISACODE-GWFastSpinBBH.cpp.

References AllocMemory(), AmplPNorder, GW::CalculDirProp(), init(), initNULL(), initRK(), GW::NParam, Rmin, setParam(), TaperApplied, TaperSteepness, Tobs, and Toffset.

10.15.2.3 `GWFastSpinBBH::GWFastSpinBBH (double Beta_n, double Lambda_n, double Mass1_n, double Mass2_n, double CoalescenceTime_n, double Distance_n, double Spin1_n, double Spin2_n, double PolarAngleOfSpin1_n, double PolarAngleOfSpin2_n, double AzimuthalAngleOfSpin1_n, double AzimuthalAngleOfSpin2_n, double PhaseAtCoalescence_n, double InitialPolarAngleL_n, double InitialAzimuthalAngleL_n, double AmplPNorder_n, double Toffset_n, double Tobs_n, double TaperApplied_n, double TaperSteepness_n, double Rmin_n)`

Constructs an instance and initializes it with values in arguments.

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

References AllocMemory(), init(), initNULL(), initRK(), GW::NParam, Rmin, setAmplPNorder(), setParam(), TaperApplied, TaperSteepness, Tobs, and Toffset.

10.15.2.4 `GWFastSpinBBH::~GWFastSpinBBH ()`

Destructor.

Definition at line 156 of file LISACODE-GWFastSpinBBH.cpp.

References betavec, betay2, cphilvec, cphily2, krk, mulvec, muly2, sigmavec, sigmay2, sphilvec, sphily2, thomasvec, thomasy2, timevec, and uspline.

10.15.3 Member Function Documentation

10.15.3.1 `void GWFastSpinBBH::AllocMemory ()`

Allocation of memory.

Definition at line 401 of file LISACODE-GWFastSpinBBH.cpp.

References betavec, betay2, cphilvec, cphily2, krk, mulvec, muly2, RK_VECLENGTH, RK_VECLENGTH_Old, sigmavec, sigmay2, sphilvec, sphily2, thomasvec, thomasy2, timevec, and uspline.

Referenced by `GWFastSpinBBH()`.

10.15.3.2 void GWFastSpinBBH::AmpShared (double *t*)

Compute shared data for calcution of amplitude.

Definition at line 805 of file LISACODE-GWFastSpinBBH.cpp.

References Amp, AmpIPNorder, Ax, betavec, betay2, c2psi, ci, costheta, cphilvec, cphily2, dm, eta, etain, Freq(), gamma0, idxcur, Mchirp, Mtot, mulvec, muly2, p15, p150, p20, p200, phi, Phi0, Phi10, phic, PRECISION, psi, s2psi, sf, shc, shp, sigmavec, sigmay2, syntheta, sphilvec, sphily2, sx, TaperApplied, TaperSteepness, tc, thomasvec, thomasy2, timeCur, timevec, tmax, TSUN, and xold.

Referenced by hc(), and hp().

10.15.3.3 void GWFastSpinBBH::calcederivvals (double *derivvals*[], double *inputs*[], double *r*, double *m1*, double *m2*, double *Mtot*, double *mu*, double *chi1*, double *chi2*)

Calculate derivatives of S1, S2 and L.

Definition at line 1114 of file LISACODE-GWFastSpinBBH.cpp.

References S1xdot(), S1ydot(), S1zdot(), S2xdot(), S2ydot(), and S2zdot().

Referenced by init(), and update().

10.15.3.4 void GWFastSpinBBH::calcLcrossN (double *output*[], double *inputs*[], double *nvec*[])

Calculate scalar product LxN.

Definition at line 1183 of file LISACODE-GWFastSpinBBH.cpp.

Referenced by init(), and update().

10.15.3.5 double GWFastSpinBBH::calcLdotN (double *inputs*[], double *nvec*[])

Calculate scalar product L.N.

Definition at line 1178 of file LISACODE-GWFastSpinBBH.cpp.

Referenced by init(), and update().

10.15.3.6 double GWFastSpinBBH::calcLdotS1 (double *inputs*[])

Calculate scalar product L.S1.

Definition at line 1163 of file LISACODE-GWFastSpinBBH.cpp.

Referenced by init(), S2xdot(), S2ydot(), S2zdot(), and update().

10.15.3.7 double GWFastSpinBBH::calcLdotS2 (double *inputs*[])

Calculate scalar product L.S1.

Definition at line 1168 of file LISACODE-GWFastSpinBBH.cpp.

Referenced by init(), S1ydot(), S1zdot(), and update().

10.15.3.8 double GWFastSpinBBH::calcSdotS (double *inputs*[])

Calculate scalar product L.S2.

Definition at line 1173 of file LISACODE-GWFastSpinBBH.cpp.

Referenced by init(), and update().

10.15.3.9 void GW::CalculDirProp () [inherited]

Computes components of the unit vectors [DirProp](#) from the angles [Lambda](#) and [Beta](#).

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

Definition at line 224 of file LISACODE-GW.cpp.

References [GW::Beta](#), [GW::DirProp](#), and [GW::Lambda](#).

Referenced by [GW::GW\(\)](#), [GWBinary::GWBinary\(\)](#), [GWFastSpinBBH\(\)](#), [GW::setBeta\(\)](#), and [GW::setLambda\(\)](#).

10.15.3.10 void GWFastSpinBBH::DispTempVal (double *t*, ostream * *OutDisp*) [virtual]

Display temporary values.

Reimplemented from [GW](#).

Definition at line 993 of file LISACODE-GWFastSpinBBH.cpp.

References Ax, ci, psi, sf, and sx.

10.15.3.11 double GWFastSpinBBH::Freq (double *t*, double *Mtot*, double *Mchirp*, double *eta*, double *beta*, double *sigma*, double *tc*)

Compute frequency.

Definition at line 1009 of file LISACODE-GWFastSpinBBH.cpp.

References TSUN.

Referenced by AmpShared(), init(), and update().

10.15.3.12 double GW::getAnglPol () [inline, inherited]

Definition at line 83 of file LISACODE-GW.h.

References [GW::AnglPol](#).

10.15.3.13 double GW::getBeta () const [inline, inherited]

Definition at line 76 of file LISACODE-GW.h.

References [GW::Beta](#).

10.15.3.14 `vector<double> GWFastSpinBBH::getDirProp () const [inline, inherited]`

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

References GWFastSpinBBH::DirProp.

10.15.3.15 `double GWFastSpinBBH::getDistance () [inline]`

Definition at line 201 of file LISACODE-GWFastSpinBBH.h.

References GWFastSpinBBH::DL.

10.15.3.16 `double GWFastSpinBBH::getLambda () const [inline, inherited]`

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

References GWFastSpinBBH::Lambda.

10.15.3.17 `double GWFastSpinBBH::getMass1 () [inline]`

Definition at line 199 of file LISACODE-GWFastSpinBBH.h.

References GWFastSpinBBH::m1.

10.15.3.18 `double GWFastSpinBBH::getMass2 () [inline]`

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

References GWFastSpinBBH::m2.

10.15.3.19 `int GWFastSpinBBH::getNParam () [inline, inherited]`

Definition at line 75 of file LISACODE-GW.h.

References GWFastSpinBBH::NPParam.

10.15.3.20 `double GWFastSpinBBH::getParam (int iP) [virtual]`

Return parameters specified by iP (see function for `setParam` the iP code).

Reimplemented from [GW](#).

Definition at line 279 of file LISACODE-GWFastSpinBBH.cpp.

References AzimuthalAngleOfSpin1, AzimuthalAngleOfSpin2, GWFastSpinBBH::Beta, chi1, chi2, DL, eta, InitialAzimuthalAngleL, InitialPolarAngleL, GWFastSpinBBH::Lambda, m1, m2, Mchirp, mu, phic, PolarAngleOfSpin1, PolarAngleOfSpin2, and tc.

10.15.3.21 `double GWFastSpinBBH::hc (double t) [virtual]`

Compute wave amplitude component hcross.

Reimplemented from [GW](#).

Definition at line 986 of file LISACODE-GWFastSpinBBH.cpp.

References AmpShared(), c2psi, s2psi, shc, and shp.

10.15.3.22 double GWFastSpinBBH::hp (double *t*) [virtual]

Compute wave amplitude component hplus.

Reimplemented from [GW](#).

Definition at line 979 of file LISACODE-GWFastSpinBBH.cpp.

References AmpShared(), c2psi, s2psi, shc, and shp.

10.15.3.23 void GWFastSpinBBH::init () [virtual]

Initialisation (integration of precession and spline).

Reimplemented from [GW](#).

Definition at line 504 of file LISACODE-GWFastSpinBBH.cpp.

References Amp, AzimuthalAngleOfSpin1, AzimuthalAngleOfSpin2, GW::Beta, betavec, betay2, c_SI, calcedrivvals(), calcLcrossN(), calcLdotN(), calcLdotS1(), calcLdotS2(), calcSdotS(), chi1, chi2, costheta, cphilvec, cphily2, DL, eta, Freq(), gamma0, idxcur, idxm, InitialAzimuthalAngleL, InitialPolarAngleL, kpc_m, GW::Lambda, m1, m2, Mchirp, Mtot, mu, mulvec, muly2, p15, p150, p20, p200, phi, Phi0, Phi10, phic, PolarAngleOfSpin1, PolarAngleOfSpin2, RK_EPS, RK_H, RK_VECLENGTH, rkckstep(), Rmin, sigmavec, sigmay2, sintheta, sphilvec, sphily2, spline(), tc, thomasvec, thomasy2, timeCur, timevec, tmax, Tobs, Toffset, TSUN, xmax, and xold.

Referenced by GWFastSpinBBH().

10.15.3.24 void GWFastSpinBBH::initNULL ()

Initialization of all pointers at NULL.

Definition at line 381 of file LISACODE-GWFastSpinBBH.cpp.

References betavec, betay2, cphilvec, cphily2, krk, mulvec, muly2, sigmavec, sigmay2, sphilvec, sphily2, thomasvec, thomasy2, timevec, and uspline.

Referenced by GWFastSpinBBH().

10.15.3.25 void GWFastSpinBBH::initRK ()

Set Runge-Kutta parameters.

Definition at line 457 of file LISACODE-GWFastSpinBBH.cpp.

References A2, A3, A4, A5, A6, B21, B31, B32, B41, B42, B43, B51, B52, B53, B54, B61, B62, B63, B64, B65, C1, C2, C3, C4, C5, C6, D1, D2, D3, D4, D5, D6, RK_EPS, RK_H, and RK_VECLENGTH.

Referenced by GWFastSpinBBH().

10.15.3.26 void GWFastSpinBBH::rkckstep (double *outputvals*[], double *fourthorderoutputvals*[], double * *outputthomas*, double *h*, double *currentvals*[], double *currentthomas*, double *t*, double *m1*, double *m2*, double *Mtot*, double *Mchirp*, double *mu*, double *eta*, double *chi1*, double *chi2*, double *N*[], double *tc*)

Compute one step in tegration using Runge Kutta method at order 6.

Definition at line 1026 of file LISACODE-GWFastSpinBBH.cpp.

References A2, A3, A4, A5, A6, B21, B31, B32, B41, B42, B43, B51, B52, B53, B54, B61, B62, B63, B64, B65, C1, C2, C3, C4, C5, C6, D1, D2, D3, D4, D5, D6, krk, and update().

Referenced by init().

10.15.3.27 double GWFastSpinBBH::S1xdot (double *inputs*[], double *r*, double *m1*, double *m2*, double *Mtot*, double *mu*, double *chi1*, double *chi2*)

Calculate derivative of component x of S1.

Referenced by calcdervvals().

10.15.3.28 double GWFastSpinBBH::S1ydot (double *inputs*[], double *r*, double *m1*, double *m2*, double *Mtot*, double *mu*, double *chi1*, double *chi2*)

Calculate derivative of component y of S1.

Definition at line 1134 of file LISACODE-GWFastSpinBBH.cpp.

References calcLdotS2().

Referenced by calcdervvals().

10.15.3.29 double GWFastSpinBBH::S1zdot (double *inputs*[], double *r*, double *m1*, double *m2*, double *Mtot*, double *mu*, double *chi1*, double *chi2*)

Calculate derivative of component z of S1.

Definition at line 1140 of file LISACODE-GWFastSpinBBH.cpp.

References calcLdotS2().

Referenced by calcdervvals().

10.15.3.30 double GWFastSpinBBH::S2xdot (double *inputs*[], double *r*, double *m1*, double *m2*, double *Mtot*, double *mu*, double *chi1*, double *chi2*)

Calculate derivative of component x of S2.

Definition at line 1146 of file LISACODE-GWFastSpinBBH.cpp.

References calcLdotS1().

Referenced by calcdervvals().

10.15.3.31 double GWFastSpinBBH::S2ydot (double *inputs*[], double *r*, double *m1*, double *m2*, double *Mtot*, double *mu*, double *chi1*, double *chi2*)

Calculate derivative of component y of S2.

Definition at line 1152 of file LISACODE-GWFastSpinBBH.cpp.

References calcLdotS1().

Referenced by calcderrivvals().

10.15.3.32 double GWFastSpinBBH::S2zdot (double *inputs*[], double *r*, double *m1*, double *m2*, double *Mtot*, double *mu*, double *chi1*, double *chi2*)

Calculate derivative of component z of S2.

Definition at line 1158 of file LISACODE-GWFastSpinBBH.cpp.

References calcLdotS1().

Referenced by calcderrivvals().

10.15.3.33 void GWFastSpinBBH::setAmplPNorder (double *AmplPNorder_n*)

Definition at line 367 of file LISACODE-GWFastSpinBBH.cpp.

References AmplPNorder.

Referenced by GWFastSpinBBH().

10.15.3.34 void GW::setAnglPol (double *AnglPol_n*) [inherited]

* Input is checked:

$$\text{AnglPol}_n \in [0, 2 \cdot \pi]$$

Reimplemented in [GMono](#), and [GNew](#).

Definition at line 187 of file LISACODE-GW.cpp.

References [GW::AnglPol](#).

10.15.3.35 void GW::setBeta (double *Beta_n*) [inherited]

Input is checked:

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

[CalculDirProp](#) method is called to fill [DirProp](#) attribute

Definition at line 125 of file LISACODE-GW.cpp.

References [GW::Beta](#), and [GW::CalculDirProp\(\)](#).

10.15.3.36 void GW::setDirProp (vector< double > *DirProp_n*) [inherited]

Sets [DirProp](#), [Lambda](#) and [Beta](#) attributes.

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

Computations: $DirProp_{norm}$ is normalized DirProp_n

-

$$\beta = \arcsin(DirProp_{norm}[2])$$

-

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

Definition at line 154 of file LISACODE-GW.cpp.

References GW::Beta, GW::DirProp, GW::Lambda, and PRECISION.

10.15.3.37 void GWFastSpinBBH::setDistance (double *r_n*)

Definition at line 358 of file LISACODE-GWFastSpinBBH.cpp.

References DL.

10.15.3.38 void GW::setLambda (double *Lambda_n*) [inherited]

Input is checked:

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

[CalculDirProp](#) method is called to set [DirProp](#) attribute

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

References GW::CalculDirProp(), and GW::Lambda.

10.15.3.39 void GWFastSpinBBH::setMass1 (double *MI_n*)

Definition at line 342 of file LISACODE-GWFastSpinBBH.cpp.

References m1.

10.15.3.40 void GWFastSpinBBH::setMass2 (double *M2_n*)

Definition at line 350 of file LISACODE-GWFastSpinBBH.cpp.

References m2.

10.15.3.41 void GWFastSpinBBH::setParam (int *iP*, double *Param_n*) [virtual]

Sets parameters specified by . Value corresponding to :

- 0 -> [Beta](#) : β : Ecliptic latitude (in radians)
- 1 -> [Lambda](#) : λ : Ecliptic longitude (in radians)
- 2 -> [m1](#) : m_1 : Mass of object 1 (in solar masses)
- 3 -> [m2](#) : m_2 : Mass of object 2 (in solar masses)

- 4 -> `tc` : t_{coal} : (in seconds)
- 5 -> `DL` : r : Distance (in kpc)
- 6 -> `chi1` : S_1 : Spin of object 1
- 7 -> `chi2` : S_2 : Spin of object 2
- 8 -> `PolarAngleOfSpin1` : θ_{S_1} : Polar angle of spin S1 (in radians)
- 9 -> `PolarAngleOfSpin2` : θ_{S_2} : Polar angle of spin S2 (in radians)
- 10 -> `AzimuthalAngleOfSpin1` : φ_{S_1} : Azimuthal angle of spin S1 (in radians)
- 11 -> `AzimuthalAngleOfSpin2` : φ_{S_2} : Azimuthal angle of spin S2 (in radians)
- 12 -> `phic` : ϕ_c : Phase at coalescence (in radians)
- 12 -> `InitialPolarAngleL` : $\theta_{L,0}$: Initial polar angle of orbital momentum (in radians)
- 14 -> `InitialAzimuthalAngleL` : $\varphi_{L,0}$: Initial azimuthal angle of orbital momentum (in radians).

Reimplemented from [GW](#).

Definition at line 188 of file LISACODE-GWFastSpinBBH.cpp.

References `AzimuthalAngleOfSpin1`, `AzimuthalAngleOfSpin2`, `GW::Beta`, `chi1`, `chi2`, `DL`, `dm`, `eta`, `etain`, `InitialAzimuthalAngleL`, `InitialPolarAngleL`, `GW::Lambda`, `m1`, `m2`, `Mchirp`, `Mtot`, `mu`, `phic`, `PolarAngleOfSpin1`, `PolarAngleOfSpin2`, and `tc`.

Referenced by `GWFastSpinBBH()`.

10.15.3.42 void GWFastSpinBBH::spline (double *x, double *y, int n, double ypI, double ypn, double *y2)

Spline data : compute y2.

Definition at line 1190 of file LISACODE-GWFastSpinBBH.cpp.

References `uspline`.

Referenced by `init()`.

10.15.3.43 void GWFastSpinBBH::update (int j, double A, double h, double intervals[], double t, double m1, double m2, double Mtot, double Mchirp, double mu, double eta, double chi1, double chi2, double N[], double tc, double **k, double *kthomas[])

Calculate derivatives for one step of the Runge-Kutta method.

Definition at line 1066 of file LISACODE-GWFastSpinBBH.cpp.

References `calcedrivvalls()`, `calcLcrossN()`, `calcLdotN()`, `calcLdotS1()`, `calcLdotS2()`, `calcSdotS()`, `Freq()`, and `TSUN`.

Referenced by `rkckstep()`.

10.15.4 Member Data Documentation

10.15.4.1 double GWFastSpinBBH::A2 [protected]

Definition at line 136 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.2 double GWFastSpinBBH::A3 [protected]

Definition at line 136 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.3 double GWFastSpinBBH::A4 [protected]

Definition at line 136 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.4 double GWFastSpinBBH::A5 [protected]

Definition at line 136 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.5 double GWFastSpinBBH::A6 [protected]

Definition at line 136 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.6 double GWFastSpinBBH::Amp [protected]

Definition at line 91 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.7 double GWFastSpinBBH::AmplPNorder [protected]

High Post-Newtonian order in calculation (maximum 2 PN).

Definition at line 68 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), GWFastSpinBBH(), and setAmplPNorder().

10.15.4.8 double GW::AnglPol [protected, inherited]

Polarisation angle.

Reimplemented in [GWMono](#), [GWNew](#), and [GWPeriGate](#).

Definition at line 51 of file LISACODE-GW.h.

Referenced by GW::getAnglPol(), GWNewton2::getParam(), GWFile::getParam(), GWCusp::getParam(), GWBinary::getParam(), GW::GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), GW::setAnglPol(), GWNewton2::setParam(), GWFile::setParam(), GWCusp::setParam(), and GWBinary::setParam().

10.15.4.9 double **GWFastSpinBBH::Ax** [protected]

Definition at line 100 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and DispTempVal().

10.15.4.10 double **GWFastSpinBBH::AzimuthalAngleOfSpin1** [protected]

Azimuthal angle of spin S1.

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

Referenced by getParam(), init(), and setParam().

10.15.4.11 double **GWFastSpinBBH::AzimuthalAngleOfSpin2** [protected]

Azimuthal angle of spin S1.

Definition at line 58 of file LISACODE-GWFastSpinBBH.h.

Referenced by getParam(), init(), and setParam().

10.15.4.12 double **GWFastSpinBBH::B21** [protected]

Definition at line 137 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.13 double **GWFastSpinBBH::B31** [protected]

Definition at line 137 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.14 double **GWFastSpinBBH::B32** [protected]

Definition at line 137 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.15 double **GWFastSpinBBH::B41** [protected]

Definition at line 137 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.16 double GWFastSpinBBH::B42 [protected]

Definition at line 137 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.17 double GWFastSpinBBH::B43 [protected]

Definition at line 137 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.18 double GWFastSpinBBH::B51 [protected]

Definition at line 137 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.19 double GWFastSpinBBH::B52 [protected]

Definition at line 137 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.20 double GWFastSpinBBH::B53 [protected]

Definition at line 137 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.21 double GWFastSpinBBH::B54 [protected]

Definition at line 137 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.22 double GWFastSpinBBH::B61 [protected]

Definition at line 137 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.23 double GWFastSpinBBH::B62 [protected]

Definition at line 137 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.24 double GWFastSpinBBH::B63 [protected]

Definition at line 137 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.25 double `GWFastSpinBBH::B64` [protected]

Definition at line 137 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.26 double `GWFastSpinBBH::B65` [protected]

Definition at line 137 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.27 `GW::Beta` [protected, inherited]

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

Referenced by GW::CalculDirProp(), GW::getBeta(), GWSto::getParam(), GWPeriGate::getParam(), GWNewton2::getParam(), GWNew::getParam(), GWMono::getParam(), GWFile::getParam(), getParam(), GWCusp::getParam(), GWBinary::getParam(), GW::getParam(), GW::GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), init(), GW::setBeta(), GW::setDirProp(), GWSto::setParam(), GWPeriGate::setParam(), GWNewton2::setParam(), GWNew::setParam(), GWMono::setParam(), GWFile::setParam(), setParam(), GWCusp::setParam(), GWBinary::setParam(), and GW::setParam().

10.15.4.28 double* `GWFastSpinBBH::betavec` [protected]

Definition at line 120 of file LISACODE-GWFastSpinBBH.h.

Referenced by AllocMemory(), AmpShared(), init(), initNULL(), and ~GWFastSpinBBH().

10.15.4.29 double* `GWFastSpinBBH::betay2` [protected]

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

Referenced by AllocMemory(), AmpShared(), init(), initNULL(), and ~GWFastSpinBBH().

10.15.4.30 double `GWFastSpinBBH::C1` [protected]

Definition at line 138 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.31 double `GWFastSpinBBH::C2` [protected]

Definition at line 138 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.32 double `GWFastSpinBBH::c2psi` [protected]

sin and cos of twice the polarisaion angle for calculation of amplitude components

Definition at line 110 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), hc(), and hp().

10.15.4.33 double **GWFastSpinBBH::C3** [protected]

Definition at line 138 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.34 double **GWFastSpinBBH::C4** [protected]

Definition at line 138 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.35 double **GWFastSpinBBH::C5** [protected]

Definition at line 138 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.36 double **GWFastSpinBBH::C6** [protected]

Definition at line 138 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.37 double **GWFastSpinBBH::chi1** [protected]

Spin of object 1.

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

Referenced by getParam(), init(), and setParam().

10.15.4.38 double **GWFastSpinBBH::chi2** [protected]

Spin of object 2.

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

Referenced by getParam(), init(), and setParam().

10.15.4.39 double **GWFastSpinBBH::ci** [protected]

Definition at line 98 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and DispTempVal().

10.15.4.40 double **GWFastSpinBBH::costheta** [protected]

Definition at line 92 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.41 double* GWFastSpinBBH::cphilvec [protected]

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

Referenced by AllocMemory(), AmpShared(), init(), initNULL(), and ~GWFastSpinBBH().

10.15.4.42 double* GWFastSpinBBH::cphily2 [protected]

Definition at line 126 of file LISACODE-GWFastSpinBBH.h.

Referenced by AllocMemory(), AmpShared(), init(), initNULL(), and ~GWFastSpinBBH().

10.15.4.43 double GWFastSpinBBH::D1 [protected]

Definition at line 139 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.44 double GWFastSpinBBH::D2 [protected]

Definition at line 139 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.45 double GWFastSpinBBH::D3 [protected]

Definition at line 139 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.46 double GWFastSpinBBH::D4 [protected]

Definition at line 139 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.47 double GWFastSpinBBH::D5 [protected]

Definition at line 139 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.48 double GWFastSpinBBH::D6 [protected]

Definition at line 139 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.49 double GWFastSpinBBH::DeltaPhase2L [protected]

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

10.15.4.50 vector<double> [GW::DirProp](#) [protected, inherited]

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

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

Referenced by [GW::CalculDirProp\(\)](#), [GW::getDirProp\(\)](#), [GW::GW\(\)](#), and [GW::setDirProp\(\)](#).

10.15.4.51 double [GWFastSpinBBH::DL](#) [protected]

Distance between source and detector (in kiloparsec).

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

Referenced by [getDistance\(\)](#), [getParam\(\)](#), [init\(\)](#), [setDistance\(\)](#), and [setParam\(\)](#).

10.15.4.52 double [GWFastSpinBBH::dm](#) [protected]

Difference of mass.

Definition at line 83 of file LISACODE-GWFastSpinBBH.h.

Referenced by [AmpShared\(\)](#), and [setParam\(\)](#).

10.15.4.53 double [GWFastSpinBBH::eta](#) [protected]

Mass ratio and it's inverse.

Definition at line 87 of file LISACODE-GWFastSpinBBH.h.

Referenced by [AmpShared\(\)](#), [getParam\(\)](#), [init\(\)](#), and [setParam\(\)](#).

10.15.4.54 double [GWFastSpinBBH::etain](#) [protected]

Mass ratio and it's inverse.

Definition at line 87 of file LISACODE-GWFastSpinBBH.h.

Referenced by [AmpShared\(\)](#), and [setParam\(\)](#).

10.15.4.55 double [GWFastSpinBBH::gamma0](#) [protected]

Definition at line 93 of file LISACODE-GWFastSpinBBH.h.

Referenced by [AmpShared\(\)](#), and [init\(\)](#).

10.15.4.56 int [GWFastSpinBBH::idxcur](#) [protected]

Current index for the interpolation and its maximal value.

Definition at line 107 of file LISACODE-GWFastSpinBBH.h.

Referenced by [AmpShared\(\)](#), and [init\(\)](#).

10.15.4.57 int [GWFastSpinBBH::idxm](#) [protected]

Current index for the interpolation and its maximal value.

Definition at line 107 of file LISACODE-GWFastSpinBBH.h.

Referenced by init().

10.15.4.58 double [GWFastSpinBBH::InitialAzimuthalAngleL](#) [protected]

Initial azimuthal angle of orbital momentum .

Definition at line 64 of file LISACODE-GWFastSpinBBH.h.

Referenced by getParam(), init(), and setParam().

10.15.4.59 double [GWFastSpinBBH::InitialPolarAngleL](#) [protected]

Initial polar angle of orbital momentum .

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

Referenced by getParam(), init(), and setParam().

10.15.4.60 double [GWFastSpinBBH::krk](#) [protected]**

Runge-Kutta parameters.

Definition at line 134 of file LISACODE-GWFastSpinBBH.h.

Referenced by AllocMemory(), initNULL(), rkckstep(), and ~GWFastSpinBBH().

10.15.4.61 [GW::Lambda](#) [protected, inherited]

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

Referenced by GW::CalculDirProp(), GW::getLambda(), GWSto::getParam(), GWPeriGate::getParam(), GWNewton2::getParam(), GWNew::getParam(), GWMono::getParam(), GWFile::getParam(), getParam(), GWCusp::getParam(), GWBinary::getParam(), GW::getParam(), GW::GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), init(), GW::setDirProp(), GW::setLambda(), GWSto::setParam(), GWPeriGate::setParam(), GWNewton2::setParam(), GWNew::setParam(), GWMono::setParam(), GWFile::setParam(), setParam(), GWCusp::setParam(), GWBinary::setParam(), and GW::setParam().

10.15.4.62 double [GWFastSpinBBH::m1](#) [protected]

Mass of object 1 (in solar masses).

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

Referenced by getMass1(), getParam(), init(), setMass1(), and setParam().

10.15.4.63 double [GWFastSpinBBH::m2](#) [protected]

Mass of object 2 (in solar masses).

Definition at line 42 of file LISACODE-GWFastSpinBBH.h.

Referenced by getMass2(), getParam(), init(), setMass2(), and setParam().

10.15.4.64 double **GWFastSpinBBH::Mchirp** [protected]

Chirp mass.

Definition at line 89 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), getParam(), init(), and setParam().

10.15.4.65 double **GWFastSpinBBH::Mtot** [protected]

Total mass.

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

Referenced by AmpShared(), init(), and setParam().

10.15.4.66 double **GWFastSpinBBH::mu** [protected]

Reduced mass.

Definition at line 85 of file LISACODE-GWFastSpinBBH.h.

Referenced by getParam(), init(), and setParam().

10.15.4.67 double* **GWFastSpinBBH::mulvec** [protected]

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

Referenced by AllocMemory(), AmpShared(), init(), initNULL(), and ~GWFastSpinBBH().

10.15.4.68 double* **GWFastSpinBBH::muly2** [protected]

Definition at line 124 of file LISACODE-GWFastSpinBBH.h.

Referenced by AllocMemory(), AmpShared(), init(), initNULL(), and ~GWFastSpinBBH().

10.15.4.69 int **GW::NParam** [protected, inherited]

Number of parameters.

Definition at line 53 of file LISACODE-GW.h.

Referenced by GW::getNParam(), GWBinary::GWBinary(), GWCusp::GWCusp(), and GWFastSpinBBH().

10.15.4.70 double **GWFastSpinBBH::p15** [protected]

Definition at line 96 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.71 double `GWFastSpinBBH::p150` [protected]

Definition at line 96 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.72 double `GWFastSpinBBH::p20` [protected]

Definition at line 96 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.73 double `GWFastSpinBBH::p200` [protected]

Definition at line 96 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.74 double `GWFastSpinBBH::phi` [protected]

Definition at line 95 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.75 double `GWFastSpinBBH::Phi0` [protected]

Definition at line 96 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.76 double `GWFastSpinBBH::Phi10` [protected]

Definition at line 96 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.77 double `GWFastSpinBBH::phic` [protected]

Phase at coalescence.

Definition at line 60 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), getParam(), init(), and setParam().

10.15.4.78 double `GWFastSpinBBH::PolarAngleOfSpin1` [protected]

Polar angle of spin S1.

Definition at line 52 of file LISACODE-GWFastSpinBBH.h.

Referenced by getParam(), init(), and setParam().

10.15.4.79 double GWFastSpinBBH::PolarAngleOfSpin2 [protected]

Polar angle of spin S2.

Definition at line 54 of file LISACODE-GWFastSpinBBH.h.

Referenced by getParam(), init(), and setParam().

10.15.4.80 double GWFastSpinBBH::psi [protected]

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

Referenced by AmpShared(), and DispTempVal().

10.15.4.81 double GWFastSpinBBH::RK_EPS [protected]

Definition at line 140 of file LISACODE-GWFastSpinBBH.h.

Referenced by init(), and initRK().

10.15.4.82 double GWFastSpinBBH::RK_H [protected]

Definition at line 140 of file LISACODE-GWFastSpinBBH.h.

Referenced by init(), and initRK().

10.15.4.83 int GWFastSpinBBH::RK_VECLENGTH [protected]

Definition at line 135 of file LISACODE-GWFastSpinBBH.h.

Referenced by AllocMemory(), init(), and initRK().

10.15.4.84 int GWFastSpinBBH::RK_VECLENGTH_Old [protected]

Definition at line 135 of file LISACODE-GWFastSpinBBH.h.

Referenced by AllocMemory().

10.15.4.85 double GWFastSpinBBH::Rmin [protected]

Definition at line 74 of file LISACODE-GWFastSpinBBH.h.

Referenced by GWFastSpinBBH(), and init().

10.15.4.86 double GWFastSpinBBH::s2psi [protected]

sin and cos of twice the polarisaion angle for calculation of amplitude components

Definition at line 110 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), hc(), and hp().

10.15.4.87 double `GWFastSpinBBH::sf` [protected]

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

Referenced by AmpShared(), and DispTempVal().

10.15.4.88 double `GWFastSpinBBH::she` [protected]

h_+ and h_x in the source reference frame

Definition at line 113 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), hc(), and hp().

10.15.4.89 double `GWFastSpinBBH::shp` [protected]

h_+ and h_x in the source reference frame

Definition at line 113 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), hc(), and hp().

10.15.4.90 double* `GWFastSpinBBH::sigmavec` [protected]

Definition at line 121 of file LISACODE-GWFastSpinBBH.h.

Referenced by AllocMemory(), AmpShared(), init(), initNULL(), and ~GWFastSpinBBH().

10.15.4.91 double* `GWFastSpinBBH::sigmay2` [protected]

Definition at line 128 of file LISACODE-GWFastSpinBBH.h.

Referenced by AllocMemory(), AmpShared(), init(), initNULL(), and ~GWFastSpinBBH().

10.15.4.92 double `GWFastSpinBBH::sintheta` [protected]

Definition at line 92 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.93 double* `GWFastSpinBBH::sphilvec` [protected]

Definition at line 118 of file LISACODE-GWFastSpinBBH.h.

Referenced by AllocMemory(), AmpShared(), init(), initNULL(), and ~GWFastSpinBBH().

10.15.4.94 double* `GWFastSpinBBH::sphily2` [protected]

Definition at line 125 of file LISACODE-GWFastSpinBBH.h.

Referenced by AllocMemory(), AmpShared(), init(), initNULL(), and ~GWFastSpinBBH().

10.15.4.95 double GWFastSpinBBH::sx [protected]

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

Referenced by AmpShared(), and DispTempVal().

10.15.4.96 double GWFastSpinBBH::TaperApplied [protected]

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

Referenced by AmpShared(), and GWFastSpinBBH().

10.15.4.97 double GWFastSpinBBH::TaperSteepness [protected]

Definition at line 76 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and GWFastSpinBBH().

10.15.4.98 double GWFastSpinBBH::tc [protected]

Time of coalescence (in seconds).

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

Referenced by AmpShared(), getParam(), init(), and setParam().

10.15.4.99 double* GWFastSpinBBH::thomasvec [protected]

Definition at line 122 of file LISACODE-GWFastSpinBBH.h.

Referenced by AllocMemory(), AmpShared(), init(), initNULL(), and ~GWFastSpinBBH().

10.15.4.100 double* GWFastSpinBBH::thomasy2 [protected]

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

Referenced by AllocMemory(), AmpShared(), init(), initNULL(), and ~GWFastSpinBBH().

10.15.4.101 double GWFastSpinBBH::timeCur [protected]

Last time of compute amplitude.

Definition at line 104 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.102 double* GWFastSpinBBH::timevec [protected]

Definition at line 116 of file LISACODE-GWFastSpinBBH.h.

Referenced by AllocMemory(), AmpShared(), init(), initNULL(), and ~GWFastSpinBBH().

10.15.4.103 double [GWFastSpinBBH::tmax](#) [protected]

Definition at line 94 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.104 double [GWFastSpinBBH::Tobs](#) [protected]

Duration of observation.

Definition at line 72 of file LISACODE-GWFastSpinBBH.h.

Referenced by GWFastSpinBBH(), and init().

10.15.4.105 double [GWFastSpinBBH::Toffset](#) [protected]

Time offset.

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

Referenced by GWFastSpinBBH(), and init().

10.15.4.106 double* [GWFastSpinBBH::uspline](#) [protected]

Definition at line 131 of file LISACODE-GWFastSpinBBH.h.

Referenced by AllocMemory(), initNULL(), spline(), and ~GWFastSpinBBH().

10.15.4.107 double [GWFastSpinBBH::xmax](#) [protected]

Definition at line 94 of file LISACODE-GWFastSpinBBH.h.

Referenced by init().

10.15.4.108 double [GWFastSpinBBH::xold](#) [protected]

Definition at line 94 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

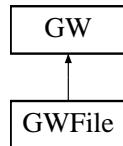
The documentation for this class was generated from the following files:

- [LISACODE-GWFastSpinBBH.h](#)
- [LISACODE-GWFastSpinBBH.cpp](#)

10.16 GWFile Class Reference

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

Inheritance diagram for GWFile::



10.16.1 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, char *FileName)**
Constructs an instance and initializes it with default values and inputs.
- **GWFile (double Beta_n, double Lambda_n, double AnglPol_n, char *FileName, int FileEncoding, double TimeOffset, double TimeStep, int Length_n, int Records_n)**
Constructs an instance and initializes it with default values and inputs.
- **~GWFile ()**
Destructor.
- **void setParam (int iP, double Param_n)**
Sets parameters specified by iP :
 - 0 -> **Beta** : β : Ecliptic latitude (in radians)
 - 1 -> **Lambda** : λ : Ecliptic longitude (in radians)
 - 2 -> **AnglPol** : ψ : Polarization angle (in radians).
- **double getParam (int iP)**
*Return parameters specified by iP (see function for **setParam** the iP code).*
- **void setFileName (char *FileName_n)**
- **char * getFileNames ()**
- **unsigned int getNbStored ()**
*Returns size of **TimeList** attribute.*
- **void init ()**

Initialization : ...

- void **ReadASCIIFile** (char ***FileName**, double **TimeOffset**, double **TimeStep**)
Load ASCII file specified in argument.
- void **ReadBinaryFile** (char ***FileName**, double **TimeOffset**, double **TimeStep**, int **Length**, int **Record**)
Load binary file specified in argument.
- double **Interpol** (double t, int type)
Interpol with Lagrange in specified type (1 -> h+ or 2 -> hx).
- double **hp** (double t)
Return $h_+(t)$.
- double **hc** (double t)
Return $h_\times(t)$.
- virtual void **DispTempVal** (double t, ostream ***OutDisp**)
- int **getNParam** ()
- double **getBeta** () const
- void **setBeta** (double **Beta_n**)
- double **getLambda** () const
- void **setLambda** (double **Lambda_n**)
- vector< double > **getDirProp** () const
- void **setDirProp** (vector< double > **DirProp_n**)
*Sets **DirProp**, **Lambda** and **Beta** attributes.*
- double **getAnglPol** ()
- void **setAnglPol** (double **AnglPol_n**)
- void **CalculDirProp** ()
*Computes components of the unit vectors **DirProp** from the angles **Lambda** and **Beta**.*

Protected Attributes

- double * **TimeList**
Vector of time values associated to components samples.
- **Couple** * **hList**
List of (h_{+} , h_{\times})=(hp , hc) polarisation components.
- int **LastUsedBin**
Last bin read.
- int **NbDat**
Number of data.
- char **FileName** [256]
Name of file which contains data.

- int **FileEncoding**

Type of file encoding : 0->ASCII, 1->Binary.

- double **TimeOffset**

Time offset and time step.

- double **TimeStep**

Time offset and time step.

- int **Length**

Size of data in file.

- int **Records**

Size of data in file.

- double **Beta**

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

- double **Lambda**

Ecliptic longitude 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.

- int **NParam**

Number of parameters.

10.16.2 Constructor & Destructor Documentation

10.16.2.1 GWFile::GWFile ()

Constructs an instance and initializes it with default values.

Inherited attributes from **GW** are set to default values.

- **LastUsedBin = 0**

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

References LastUsedBin.

10.16.2.2 GWFile::GWFile (double *Beta_n*, double *Lambda_n*, double *AnglPol_n*, char * *FileName_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 as arguments

- `LastUsedBin = 0` `TimeList` and `hList` attributes are filled using `ReadASCIIFile` method with `FileName` input as argument

Definition at line 36 of file LISACODE-GWFile.cpp.

References `FileEncoding`, `FileName`, `init()`, `Length`, `Records`, `TimeOffset`, and `TimeStep`.

10.16.2.3 GWFile::GWFile (double Beta_n, double Lambda_n, double AnglPol_n, char * FileName_n, int FileEncoding_n, double TimeOffset_n, double TimeStep_n, int Length_n, int Records_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 as arguments

- `LastUsedBin = 0` `TimeList` and `hList` attributes are filled using `ReadASCIIFile` or `ReadBinaryFile` method with `FileName` input as argument `FileEncoding` get type of file (0: ASCII , 1: Binary)

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

References `FileEncoding`, `FileName`, `init()`, `Length`, `Records`, `TimeOffset`, and `TimeStep`.

10.16.2.4 GWFile::~GWFile ()

Destructor.

Definition at line 79 of file LISACODE-GWFile.cpp.

References `hList`, and `TimeList`.

10.16.3 Member Function Documentation

10.16.3.1 void GW::CalculDirProp () [inherited]

Computes components of the unit vectors `DirProp` from the angles `Lambda` and `Beta`.

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

Definition at line 224 of file LISACODE-GW.cpp.

References `GW::Beta`, `GW::DirProp`, and `GW::Lambda`.

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

10.16.3.2 void GWFile::DispTempVal (double t, ostream * OutDisp) [virtual]

Reimplemented from `GW`.

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

10.16.3.3 double GW::getAnglPol () [inline, inherited]

Definition at line 83 of file LISACODE-GW.h.

References GW::AnglPol.

10.16.3.4 double GW::getBeta () const [inline, inherited]

Definition at line 76 of file LISACODE-GW.h.

References GW::Beta.

10.16.3.5 vector<double> GW::getDirProp () const [inline, inherited]

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

References GW::DirProp.

10.16.3.6 char* GWFile::getFileName ()**10.16.3.7 double GW::getLambda () const [inline, inherited]**

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

References GW::Lambda.

10.16.3.8 unsigned int GWFile::getNbStored () [inline]

Returns size of [TimeList](#) attribute.

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

References NbDat.

10.16.3.9 int GW::getNParam () [inline, inherited]

Definition at line 75 of file LISACODE-GW.h.

References GW::NParam.

10.16.3.10 double GWFile::getParam (int iP) [virtual]

Return parameters specified by iP (see function for [setParam](#) the iP code).

Reimplemented from [GW](#).

Definition at line 105 of file LISACODE-GWFile.cpp.

References GW::AnglPol, GW::Beta, and GW::Lambda.

10.16.3.11 double GWFile::hc (double t) [virtual]

Return $h_x(t)$.

Returns:

Result of [Interpol](#) method with *t* input time and 2 as arguments.

Reimplemented from [GW](#).

Definition at line 364 of file LISACODE-GWFile.cpp.

References [Interpol\(\)](#).

10.16.3.12 double GWFile::hp (double *t*) [virtual]

Return $h_+(t)$.

returned value = result of [Interpol](#) method with *t* input time and 1 as arguments

Reimplemented from [GW](#).

Definition at line 354 of file LISACODE-GWFile.cpp.

References [Interpol\(\)](#).

10.16.3.13 void GWFile::init () [virtual]

Initialization : ...

Reimplemented from [GW](#).

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

References [FileEncoding](#), [FileName](#), [LastUsedBin](#), [Length](#), [ReadASCIIFile\(\)](#), [ReadBinaryFile\(\)](#), [Records](#), [TimeOffset](#), and [TimeStep](#).

Referenced by [GWFile\(\)](#).

10.16.3.14 double GWFile::Interpol (double *t*, int *type*)

Interpol with Lagrange in specified type (1 -> h+ or 2 -> hx).

Returns:

$$\begin{aligned} \text{returnedvalue} &= \begin{cases} \sum_{k=kmin}^{kmax} hList[k].x \cdot P_k & \text{if type=1} \\ \sum_{k=kmin}^{kmax} hList[k].y \cdot P_k & \text{if type=2} \end{cases} \\ \text{where } P_k &= \prod_{j=kmin, j \neq k}^{kmax} \frac{t - TimeList[j]}{TimeList[k] - TimeList[j]} \end{aligned}$$

[LastUsedBin](#) is updated, so that

$$TimeList[LastUsedBin] \leq t \leq TimeList[LastUsedBin + 1]$$

$$ordermin = floor\left(\frac{order + 1}{2}\right) \text{ where } order = 3$$

$$kmin = min(0, LastUsedBin - ordermin + 1)$$

$$kmax = max(LastUsedBin + order + 1 - ordermin, sizeof(TimeList) - 1)$$

Definition at line 296 of file LISACODE-GWFile.cpp.

References hList, LastUsedBin, NbDat, TimeList, Couple::x, and Couple::y.

Referenced by hc(), and hp().

10.16.3.15 void GWFile::ReadASCIIFile (char * *FileName*, double *TimeOffset*, double *TimeStep*)

Load ASCII file specified in argument.

The number of columns of the file must be 2 (h+,hx) or 3 (t,h+,hx). If it is 2, time informations are specific in arguments *TimeOffset* and *TimeStep*. If *TimeStep* is -1.0, it means that the time informations are not specified. Header (lines begining with "#" characted) are ignored. For each line, time and 2 components are read and pushed back into [TimeList](#) and [hList](#) attributes.

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

References hList, NbDat, Records, TimeList, and Couple::x.

Referenced by init().

10.16.3.16 void GWFile::ReadBinaryFile (char * *FileName*, double *TimeOffset*, double *TimeStep*, int *Length*, int *Records*)

Load binary file specified in argument.

The number of columns of the file must be 2 (h+,hx) or 3 (t,h+,hx). If it is 2, time informations are specific in arguments *TimeOffset* and *TimeStep*. If *TimeStep* is -1.0, it means that the time informations are not specified. Header (lines begining with "#" characted) are ignored. For each line, time and 2 components are read and pushed back into [TimeList](#) and [hList](#) attributes.

Definition at line 238 of file LISACODE-GWFile.cpp.

References hList, NbDat, TimeList, Couple::x, and Couple::y.

Referenced by init().

10.16.3.17 void GW::setAnglPol (double *AnglPol_n*) [inherited]

* Input is checked:

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

Reimplemented in [GWMono](#), and [GWNew](#).

Definition at line 187 of file LISACODE-GW.cpp.

References GW::AnglPol.

10.16.3.18 void GW::setBeta (double *Beta_n*) [inherited]

Input is checked:

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

[CalculDirProp](#) method is called to fill [DirProp](#) attribute

Definition at line 125 of file LISACODE-GW.cpp.

References GW::Beta, and GW::CalculDirProp().

10.16.3.19 void GW::setDirProp (vector< double > *DirProp_n*) [inherited]

Sets **DirProp**, **Lambda** and **Beta** attributes.

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

Computations: *DirProp_{norm}* is normalized DirProp_n

•

$$\beta = \arcsin(\text{DirProp}_{\text{norm}}[2])$$

•

$$\lambda = \text{mod}(\text{atan}(\frac{-\text{DirProp}_{\text{norm}}[1]}{-\text{DirProp}_{\text{norm}}[0]}), 2 \cdot \pi)$$

Definition at line 154 of file LISACODE-GW.cpp.

References **GW::Beta**, **GW::DirProp**, **GW::Lambda**, and **PRECISION**.

10.16.3.20 void GWFile::setFileName (char * *FileName_n*)

Definition at line 123 of file LISACODE-GWFile.cpp.

References **FileName**.

10.16.3.21 void GW::setLambda (double *Lambda_n*) [inherited]

Input is checked:

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

CalculDirProp method is called to set **DirProp** attribute

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

References **GW::CalculDirProp()**, and **GW::Lambda**.

10.16.3.22 void GWFile::setParam (int *iP*, double *Param_n*) [virtual]

Sets parameters specified by iP :

- 0 -> **Beta** : β : Ecliptic latitude (in radians)
- 1 -> **Lambda** : λ : Ecliptic longitude (in radians)
- 2 -> **AnglPol** : ψ : Polarization angle (in radians).

Reimplemented from **GW**.

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

References **GW::AnglPol**, **GW::Beta**, and **GW::Lambda**.

10.16.4 Member Data Documentation

10.16.4.1 double **GW::AnglPol** [protected, inherited]

Polarisation angle.

Reimplemented in [GMono](#), [GNew](#), and [GPeriGate](#).

Definition at line 51 of file LISACODE-GW.h.

Referenced by [GW::getAnglPol\(\)](#), [GNewton2::getParam\(\)](#), [getParam\(\)](#), [GWCusp::getParam\(\)](#), [GWBinary::getParam\(\)](#), [GW::GW\(\)](#), [GWBinary::GWBinary\(\)](#), [GNewton2::GNewton2\(\)](#), [GW::setAnglPol\(\)](#), [GNewton2::setParam\(\)](#), [setParam\(\)](#), [GWCusp::setParam\(\)](#), and [GWBinary::setParam\(\)](#).

10.16.4.2 **GW::Beta** [protected, inherited]

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

Referenced by [GW::CalculDirProp\(\)](#), [GW::getBeta\(\)](#), [GWSto::getParam\(\)](#), [GPeriGate::getParam\(\)](#), [GNewton2::getParam\(\)](#), [GNew::getParam\(\)](#), [GMono::getParam\(\)](#), [getParam\(\)](#), [GWFastSpinBBH::getParam\(\)](#), [GWCusp::getParam\(\)](#), [GWBinary::getParam\(\)](#), [GW::getParam\(\)](#), [GW::GW\(\)](#), [GWBinary::GWBinary\(\)](#), [GNewton2::GNewton2\(\)](#), [GWFastSpinBBH::init\(\)](#), [GW::setBeta\(\)](#), [GW::setDirProp\(\)](#), [GWSto::setParam\(\)](#), [GPeriGate::setParam\(\)](#), [GNewton2::setParam\(\)](#), [GNew::setParam\(\)](#), [GMono::setParam\(\)](#), [setParam\(\)](#), [GWFastSpinBBH::setParam\(\)](#), [GWCusp::setParam\(\)](#), [GWBinary::setParam\(\)](#), and [GW::setParam\(\)](#).

10.16.4.3 vector<double> **GW::DirProp** [protected, inherited]

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

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

Referenced by [GW::CalculDirProp\(\)](#), [GW::getDirProp\(\)](#), [GW::GW\(\)](#), and [GW::setDirProp\(\)](#).

10.16.4.4 int **GWFile::FileEncoding** [protected]

Type of file encoding : 0->ASCII, 1->Binary.

Definition at line 52 of file LISACODE-GWFile.h.

Referenced by [GWFile\(\)](#), and [init\(\)](#).

10.16.4.5 char **GWFile::FileName[256]** [protected]

Name of file which contains data.

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

Referenced by [GWFile\(\)](#), [init\(\)](#), and [setFileName\(\)](#).

10.16.4.6 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 Interpol(), ReadASCIIFile(), ReadBinaryFile(), and ~GWFile().

10.16.4.7 `GW::Lambda` [protected, inherited]

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

Referenced by GW::CalculDirProp(), GW::getLambda(), GWSto::getParam(), GWPeriGate::getParam(), GWNewton2::getParam(), GWNew::getParam(), GWMono::getParam(), getParam(), GWFastSpinBBH::getParam(), GWCusp::getParam(), GWBinary::getParam(), GW::getParam(), GW::GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), GWFastSpinBBH::init(), GW::setDirProp(), GW::setLambda(), GWSto::setParam(), GWPeriGate::setParam(), GWNewton2::setParam(), GWNew::setParam(), GWMono::setParam(), setParam(), GWFastSpinBBH::setParam(), GWCusp::setParam(), GWBinary::setParam(), and GW::setParam().

10.16.4.8 int `GWFile::LastUsedBin` [protected]

Last bin read.

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

Referenced by GWFile(), init(), and Interpol().

10.16.4.9 int `GWFile::Length` [protected]

Size of data in file.

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

Referenced by GWFile(), and init().

10.16.4.10 int `GWFile::NbDat` [protected]

Number of data.

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

Referenced by getNbStored(), Interpol(), ReadASCIIFile(), and ReadBinaryFile().

10.16.4.11 int `GW::NParam` [protected, inherited]

Number of parameters.

Definition at line 53 of file LISACODE-GW.h.

Referenced by GW::getNParam(), GWBinary::GWBinary(), GWCusp::GWCusp(), and GWFastSpinBBH::GWFastSpinBBH().

10.16.4.12 int `GWFile::Records` [protected]

Size of data in file.

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

Referenced by GWFile(), init(), and ReadASCIIFile().

10.16.4.13 double* GWFile::TimeList [protected]

Vector of time values associated to components samples.

Definition at line 42 of file LISACODE-GWFile.h.

Referenced by Interpol(), ReadASCIIFile(), ReadBinaryFile(), and ~GWFile().

10.16.4.14 double GWFile::TimeOffset [protected]

Time offset and time step.

Definition at line 54 of file LISACODE-GWFile.h.

Referenced by GWFile(), and init().

10.16.4.15 double GWFile::TimeStep [protected]

Time offset and time step.

Definition at line 54 of file LISACODE-GWFile.h.

Referenced by GWFile(), and init().

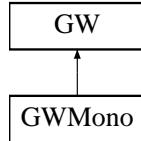
The documentation for this class was generated from the following files:

- [LISACODE-GWFile.h](#)
- [LISACODE-GWFile.cpp](#)

10.17 GWMono Class Reference

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

Inheritance diagram for GWMono::



10.17.1 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 \(\)](#)
- [GWMono \(double Beta_n, double Lambda_n, double AnglPol_n, double Freq_n, double Amplhp_n, double Amplhc_n\)](#)
- [GWMono \(double Beta_n, double Lambda_n, double AnglPol_n, double Freq_n, double Amplhp_n, double Amplhc_n, double Phi0hp_n, double Phi0hc_n\)](#)
- [~GWMono \(\)](#)
Destructor
- void [setParam \(int iP, double Param_n\)](#)
Sets parameters specified by iP :
 - 0 -> [Beta](#) : β : Ecliptic latitude (in radians)
 - 1 -> [Lambda](#) : λ : Ecliptic longitude (in radians)
 - 2 -> [AnglPol](#) : ψ : Polarization angle (in radians)
 - 3 -> [Freq](#) : f_{GW} : Frequency (in Hertz)
 - 4 -> [Amplhp](#) : h_{0+} : Amplitude of component +
 - 5 -> [Amplhc](#) : $h_{0\times}$: Amplitude of component \times
 - 6 -> [Phi0hp](#) : ϕ_{0+} : Initial phase of component + (in radians)
 - 7 -> [Phi0hc](#) : $\phi_{0\times}$: Initial phase of component \times (in radians).
- double [getParam \(int iP\)](#)
Return parameters specified by iP (see function for [setParam](#) the iP code).
- double [getFreq \(\) const](#)
- double [getAmplhp \(\) const](#)
- double [getAmplhc \(\) const](#)
- double [getAnglPol \(\) const](#)
- double [getPhi0hp \(\) const](#)
- double [getPhi0hc \(\) const](#)
- void [setFreq \(double Freq_n\)](#)
- void [setAmplhp \(double Amplhp_n\)](#)

- void **setAmplhc** (double Amplhc_n)
 - void **setAnglPol** (double AnglPol_n)
 - void **setPhi0hp** (double Phi0hp_n)
 - void **setPhi0hc** (double Phi0hc_n)
 - void **init** ()
- Initialization : ...*
- double **hp** (double t)
Return $h_+(t)$.
 - double **hc** (double t)
Return $h_\times(t)$.
 - virtual void **DispTempVal** (double t, ostream *OutDisp)
 - int **getNParam** ()
 - double **getBeta** () const
 - void **setBeta** (double Beta_n)
 - double **getLambda** () const
 - void **setLambda** (double Lambda_n)
 - vector< double > **getDirProp** () const
 - void **setDirProp** (vector< double > DirProp_n)
- Sets **DirProp**, **Lambda** and **Beta** attributes.*
- double **getAnglPol** ()
 - void **CalculDirProp** ()
- Computes components of the unit vectors **DirProp** from the angles **Lambda** and **Beta**.*

Protected Attributes

- double **Freq**
Frequency.
- double **Amplhp**
 h_{+} polarisation component amplitude
- double **Amplhc**
 h_{\times} polarisation component amplitude
- double **AnglPol**
Polarisation angle (rad).
- double **Phi0hp**
Initial phase of h_{+} polarisation component.
- double **Phi0hc**
Initial phase of h_{\times} polarisation component.
- double **Beta**

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

- double **Lambda**

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

- vector< double > **DirProp**

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

- int **NParam**

Number of parameters.

10.17.2 Constructor & Destructor Documentation

10.17.2.1 **GWMono::GWMono ()**

- **Freq** = 0
- **Amplhp** = 0
- **Amplhc** = 0
- **AnglPol** = 0
- **Phi0hp** = 0
- **Phi0hc** = 0

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

References **Amplhc**, **Amplhp**, **AnglPol**, **Freq**, **Phi0hc**, and **Phi0hp**.

10.17.2.2 **GWMono::GWMono (double *Beta_n*, double *Lambda_n*, double *AnglPol_n*, double *Freq_n*, double *Amplhp_n*, double *Amplhc_n*)**

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**.

10.17.2.3 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*)

Inputs are checked :*Freq_n* must be positive or null.

[GW](#) constructor is called using *Beta_n*, *Lambda_n* and *AnglPol_n* inputs.

Definition at line 68 of file LISACODE-GWMono.cpp.

References *Amplhc*, *Amplhp*, *Freq*, *Phi0hc*, and *Phi0hp*.

10.17.2.4 GWMono::~GWMono ()

Destructor.

Definition at line 88 of file LISACODE-GWMono.cpp.

10.17.3 Member Function Documentation

10.17.3.1 void GW::CalculDirProp () [inherited]

Computes components of the unit vectors [DirProp](#) from the angles [Lambda](#) and [Beta](#).

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

Definition at line 224 of file LISACODE-GW.cpp.

References [GW::Beta](#), [GW::DirProp](#), and [GW::Lambda](#).

Referenced by [GW::GW\(\)](#), [GWBinary::GWBinary\(\)](#), [GWFastSpinBBH::GWFastSpinBBH\(\)](#), [GW::setBeta\(\)](#), and [GW::setLambda\(\)](#).

10.17.3.2 void GWMono::DispTempVal (double *t*, ostream * *OutDisp*) [virtual]

Reimplemented from [GW](#).

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

10.17.3.3 double GWMono::getAmplhc () const [inline]

Definition at line 92 of file LISACODE-GWMono.h.

References *Amplhc*.

10.17.3.4 double GWMono::getAmplhp () const [inline]

Definition at line 91 of file LISACODE-GWMono.h.

References *Amplhp*.

10.17.3.5 double GW::getAnglPol () [inline, inherited]

Definition at line 83 of file LISACODE-GW.h.

References GW::AnglPol.

10.17.3.6 double GWMono::getAnglPol () const [inline]

Definition at line 93 of file LISACODE-GWMono.h.

References AnglPol.

10.17.3.7 double GW::getBeta () const [inline, inherited]

Definition at line 76 of file LISACODE-GW.h.

References GW::Beta.

10.17.3.8 vector<double> GW::getDirProp () const [inline, inherited]

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

References GW::DirProp.

10.17.3.9 double GWMono::getFreq () const [inline]

Definition at line 90 of file LISACODE-GWMono.h.

References Freq.

10.17.3.10 double GW::getLambda () const [inline, inherited]

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

References GW::Lambda.

10.17.3.11 int GW::getNParam () [inline, inherited]

Definition at line 75 of file LISACODE-GW.h.

References GW::NParam.

10.17.3.12 double GWMono::getParam (int iP) [virtual]

Return parameters specified by iP (see function for [setParam](#) the iP code).

Reimplemented from [GW](#).

Definition at line 129 of file LISACODE-GWMono.cpp.

References Amplhc, Amplhp, AnglPol, GW::Beta, Freq, GW::Lambda, Phi0hc, and Phi0hp.

10.17.3.13 double GWMono::getPhi0hc () const [inline]

Definition at line 95 of file LISACODE-GWMono.h.

References Phi0hc.

10.17.3.14 double GWMono::getPhi0hp () const [inline]

Definition at line 94 of file LISACODE-GWMono.h.

References Phi0hp.

10.17.3.15 double GWMono::hc (double t) [virtual]

Return $h_x(t)$.

$$\text{returned value} = \text{Amplhc} \cdot \sin(2 \cdot \pi \cdot \text{Freq} \cdot t + \text{Phi0hc})$$

Reimplemented from [GW](#).

Definition at line 202 of file LISACODE-GWMono.cpp.

References Amplhc, Freq, and Phi0hc.

10.17.3.16 double GWMono::hp (double t) [virtual]

Return $h_+(t)$.

$$\text{returned value} = \text{Amplhp} \cdot \sin(2 \cdot \pi \cdot \text{Freq} \cdot t + \text{Phi0hp})$$

Reimplemented from [GW](#).

Definition at line 195 of file LISACODE-GWMono.cpp.

References Amplhp, Freq, and Phi0hp.

10.17.3.17 void GWMono::init () [virtual]

Initialization : ...

Reimplemented from [GW](#).

Definition at line 188 of file LISACODE-GWMono.cpp.

10.17.3.18 void GWMono::setAmplhc (double *Amplhc_n*)

Sets [Amplhc](#) attribute.

Definition at line 177 of file LISACODE-GWMono.cpp.

References Amplhc.

10.17.3.19 void GWMono::setAmplhp (double *Amplhp_n*)

Sets *Amplhp* attribute.

Definition at line 171 of file LISACODE-GWMono.cpp.

References *Amplhp*.

10.17.3.20 void GWMono::setAnglPol (double *AnglPol_n*)

* Input is checked:

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

Reimplemented from [GW](#).

10.17.3.21 void GW::setBeta (double *Beta_n*) [inherited]

Input is checked:

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

[CalculDirProp](#) method is called to fill *DirProp* attribute

Definition at line 125 of file LISACODE-GW.cpp.

References [GW::Beta](#), and [GW::CalculDirProp\(\)](#).

10.17.3.22 void GW::setDirProp (vector< double > *DirProp_n*) [inherited]

Sets *DirProp*, *Lambda* and *Beta* attributes.

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

Computations: *DirProp_{norm}* is normalized *DirProp_n*

-

$$\beta = \arcsin(DirProp_{norm}[2])$$

-

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

Definition at line 154 of file LISACODE-GW.cpp.

References [GW::Beta](#), [GW::DirProp](#), [GW::Lambda](#), and [PRECISION](#).

10.17.3.23 void GWMono::setFreq (double *Freq_n*)

Input is checked: *Freq_n* must be positive or null

Definition at line 163 of file LISACODE-GWMono.cpp.

References *Freq*.

10.17.3.24 void GWMono::setLambda (double *Lambda_n*) [inherited]

Input is checked:

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

[CalculDirProp](#) method is called to set [DirProp](#) attribute

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

References [GWMono::CalculDirProp\(\)](#), and [GWMono::Lambda](#).

10.17.3.25 void GWMono::setParam (int *iP*, double *Param_n*) [virtual]

Sets parameters specified by *iP* :

- 0 -> [Beta](#) : β : Ecliptic latitude (in radians)
- 1 -> [Lambda](#) : λ : Ecliptic longitude (in radians)
- 2 -> [AnglPol](#) : ψ : Polarization angle (in radians)
- 3 -> [Freq](#) : f_{GW} : Frequency (in Hertz)
- 4 -> [Amplhp](#) : h_{0+} : Amplitude of component +
- 5 -> [Amplhc](#) : $h_{0\times}$: Amplitude of component \times
- 6 -> [Phi0hp](#) : ϕ_{0+} : Initial phase of component + (in radians)
- 7 -> [Phi0hc](#) : $\phi_{0\times}$: Initial phase of component \times (in radians).

Reimplemented from [GW](#).

Definition at line 98 of file LISACODE-GWMono.cpp.

References [Amplhc](#), [Amplhp](#), [AnglPol](#), [GW::Beta](#), [Freq](#), [GWMono::Lambda](#), [Phi0hc](#), and [Phi0hp](#).

10.17.3.26 void GWMono::setPhi0hc (double *Phi0hc_n*) [inline]

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

References [Phi0hc](#).

10.17.3.27 void GWMono::setPhi0hp (double *Phi0hp_n*) [inline]

Definition at line 100 of file LISACODE-GWMono.h.

References [Phi0hp](#).

10.17.4 Member Data Documentation**10.17.4.1 double GWMono::Amplhc [protected]**

`h_{x}` polarisation component amplitude

Definition at line 43 of file LISACODE-GWMono.h.

Referenced by [getAmplhc\(\)](#), [getParam\(\)](#), [GWMono\(\)](#), [hc\(\)](#), [setAmplhc\(\)](#), and [setParam\(\)](#).

10.17.4.2 double [GWMono::Amplhp](#) [protected]

`h_{+}` polarisation component amplitude

Definition at line 41 of file LISACODE-GWMono.h.

Referenced by `getAmplhp()`, `getParam()`, `GWMono()`, `hp()`, `setAmplhp()`, and `setParam()`.

10.17.4.3 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()`, `getParam()`, `GWMono()`, and `setParam()`.

10.17.4.4 [GW::Beta](#) [protected, inherited]

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

Referenced by `GW::CalculDirProp()`, `GW::getBeta()`, `GWSto::getParam()`, `GWPeriGate::getParam()`, `GWNewton2::getParam()`, `GWNew::getParam()`, `getParam()`, `GWFile::getParam()`, `GWFastSpinBBH::getParam()`, `GWCSusp::getParam()`, `GWBinary::getParam()`, `GW::getParam()`, `GW::GW()`, `GWBinary::GWBinary()`, `GWNewton2::GWNewton2()`, `GWFastSpinBBH::init()`, `GW::setBeta()`, `GW::setDirProp()`, `GWSto::setParam()`, `GWPeriGate::setParam()`, `GWNewton2::setParam()`, `GWNew::setParam()`, `setParam()`, `GWFile::setParam()`, `GWFastSpinBBH::setParam()`, `GWCSusp::setParam()`, `GWBinary::setParam()`, and `GW::setParam()`.

10.17.4.5 vector<double> [GW::DirProp](#) [protected, inherited]

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

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

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

10.17.4.6 double [GWMono::Freq](#) [protected]

Frequency.

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

Referenced by `getFreq()`, `getParam()`, `GWMono()`, `hc()`, `hp()`, `setFreq()`, and `setParam()`.

10.17.4.7 [GW::Lambda](#) [protected, inherited]

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

Referenced by `GW::CalculDirProp()`, `GW::getLambda()`, `GWSto::getParam()`, `GWPeriGate::getParam()`, `GWNewton2::getParam()`, `GWNew::getParam()`, `getParam()`, `GWFile::getParam()`, `GWFastSpinBBH::getParam()`, `GWCSusp::getParam()`, `GWBinary::getParam()`, `GW::getParam()`, `GW::GW()`,

GWBinary::GWBinary(), GWNewton2::GWNewton2(), GWFastSpinBBH::init(), GW::setDirProp(), GW::setLambda(), GWSto::setParam(), GWPeriGate::setParam(), GWNewton2::setParam(), GWNew::setParam(), setParam(), GWFile::setParam(), GWFastSpinBBH::setParam(), GWCusp::setParam(), GWBinary::setParam(), and GW::setParam().

10.17.4.8 int **GW::NParam** [protected, inherited]

Number of parameters.

Definition at line 53 of file LISACODE-GW.h.

Referenced by GW::getNParam(), GWBinary::GWBinary(), GWCusp::GWCusp(), and GWFastSpinBBH::GWFastSpinBBH().

10.17.4.9 double **GWMono::Phi0hc** [protected]

Initial phase of h_cross polarisation component.

Definition at line 54 of file LISACODE-GWMono.h.

Referenced by getParam(), getPhi0hc(), GWMono(), hc(), setParam(), and setPhi0hc().

10.17.4.10 double **GWMono::Phi0hp** [protected]

Initial phase of h_plus polarisation component.

Definition at line 52 of file LISACODE-GWMono.h.

Referenced by getParam(), getPhi0hp(), GWMono(), hp(), setParam(), and setPhi0hp().

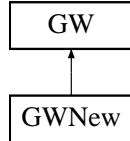
The documentation for this class was generated from the following files:

- [LISACODE-GWMono.h](#)
- [LISACODE-GWMono.cpp](#)

10.18 GWNew Class Reference

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

Inheritance diagram for GWNew::



10.18.1 Detailed Description

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

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

Public Member Functions

- [GWNew \(\)](#)
- [GWNew \(double Beta_n, double Lambda_n, double AnglPol_n, double Freq_n, double Amplhp_n, double Amplhc_n\)](#)
- [GWNew \(double Beta_n, double Lambda_n, double AnglPol_n, double Freq_n, double Amplhp_n, double Amplhc_n, double Phi0hp_n, double Phi0hc_n\)](#)
- [~GWNew \(\)](#)
- void [setParam \(int iP, double Param_n\)](#)

Sets parameters specified by iP :

 - 0 -> [Beta](#) : β : Ecliptic latitude (in radians)
 - 1 -> [Lambda](#) : λ : Ecliptic longitude (in radians)
 - 2 -> [AnglPol](#) : ψ : Polarization angle (in radians)
 - 3 -> [Freq](#) : f_{GW} : Frequency (in Hertz)
 - 4 -> [Amplhp](#) : h_{0+} : Amplitude of component +
 - 5 -> [Amplhc](#) : $h_{0\times}$: Amplitude of component \times
 - 6 -> [Phi0hp](#) : ϕ_{0+} : Initial phase of component + (in radians)
 - 7 -> [Phi0hc](#) : $\phi_{0\times}$: Initial phase of component \times (in radians).
- double [getParam \(int iP\)](#)

Return parameters specified by iP (see function for [setParam](#) the iP code).
- double [getFreq \(\) const](#)
- double [getAmplhp \(\) const](#)
- double [getAmplhc \(\) const](#)
- double [getAnglPol \(\) const](#)
- double [getPhi0hp \(\) const](#)
- double [getPhi0hc \(\) const](#)
- void [setFreq \(double Freq_n\)](#)
- void [setAmplhp \(double Amplhp_n\)](#)
- void [setAmplhc \(double Amplhc_n\)](#)
- void [setAnglPol \(double AnglPol_n\)](#)

- void `setPhi0hp` (double `Phi0hp_n`)
- void `setPhi0hc` (double `Phi0hc_n`)
- void `init` ()
Initialization : ...
- double `hp` (double `t`)
Return $h_+(t)$.
- double `hc` (double `t`)
Return $h_\times(t)$.
- virtual void `DispTempVal` (double `t`, ostream *`OutDisp`)
- int `getNParam` ()
- double `getBeta` () const
- void `setBeta` (double `Beta_n`)
- double `getLambda` () const
- void `setLambda` (double `Lambda_n`)
- vector< double > `getDirProp` () const
- void `setDirProp` (vector< double > `DirProp_n`)
Sets `DirProp`, `Lambda` and `Beta` attributes.
- double `getAnglPol` ()
- void `CalculDirProp` ()
Computes components of the unit vectors `DirProp` from the angles `Lambda` and `Beta`.

Protected Attributes

- double `Freq`
Frequency.
- double `Amplhp`
 h_+ polarisation component amplitude
- double `Amphc`
 h_\times polarisation component amplitude
- double `AnglPol`
Polarisation angle (rad).
- double `Phi0hp`
Initial phase of h_{+} polarisation component.
- double `Phi0hc`
Initial phase of h_{\times} polarisation component.
- double `Beta`
Ecliptic latitude Source direction angle (in the heliocentric reference frame) in radians.

- double **Lambda**

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

- vector< double > **DirProp**

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

- int **NParam**

Number of parameters.

10.18.2 Constructor & Destructor Documentation

10.18.2.1 GWNew::GWNew ()

- **Freq** = 0
- **Amplhp** = 0
- **Amplhc** = 0
- **AnglPol** = 0
- **Phi0hp** = 0
- **Phi0hc** = 0

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

References Amplhc, Amplhp, AnglPol, Freq, Phi0hc, and Phi0hp.

10.18.2.2 GWNew::GWNew (double *Beta_n*, double *Lambda_n*, double *AnglPol_n*, double *Freq_n*, double *Amplhp_n*, double *Amplhc_n*)

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-GWNew.cpp.

References Amplhc, Amplhp, Freq, Phi0hc, and Phi0hp.

10.18.2.3 GWNew::GWNew (double *Beta_n*, double *Lambda_n*, double *AnglPol_n*, double *Freq_n*, double *Amplhp_n*, double *Amplhc_n*, double *Phi0hp_n*, double *Phi0hc_n*)

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 73 of file LISACODE-GWNew.cpp.

References [Amplhc](#), [Amplhp](#), [Freq](#), [Phi0hc](#), and [Phi0hp](#).

10.18.2.4 GWNew::~GWNew ()

Definition at line 92 of file LISACODE-GWNew.cpp.

10.18.3 Member Function Documentation

10.18.3.1 void GW::CalculDirProp () [inherited]

Computes components of the unit vectors [DirProp](#) from the angles [Lambda](#) and [Beta](#).

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

Definition at line 224 of file LISACODE-GW.cpp.

References [GW::Beta](#), [GW::DirProp](#), and [GW::Lambda](#).

Referenced by [GW::GW\(\)](#), [GWBinary::GWBinary\(\)](#), [GWFastSpinBBH::GWFastSpinBBH\(\)](#), [GW::setBeta\(\)](#), and [GW::setLambda\(\)](#).

10.18.3.2 void GWNew::DispTempVal (double *t*, ostream * *OutDisp*) [virtual]

Reimplemented from [GW](#).

Definition at line 211 of file LISACODE-GWNew.cpp.

10.18.3.3 double GWNew::getAmplhc () const [inline]

Definition at line 93 of file LISACODE-GWNew.h.

References [Amplhc](#).

10.18.3.4 double GWNew::getAmplhp () const [inline]

Definition at line 92 of file LISACODE-GWNew.h.

References Amplhp.

10.18.3.5 double GW::getAnglPol () [inline, inherited]

Definition at line 83 of file LISACODE-GW.h.

References GW::AnglPol.

10.18.3.6 double GWNew::getAnglPol () const [inline]

Definition at line 94 of file LISACODE-GWNew.h.

References AnglPol.

10.18.3.7 double GW::getBeta () const [inline, inherited]

Definition at line 76 of file LISACODE-GW.h.

References GW::Beta.

10.18.3.8 vector<double> GW::getDirProp () const [inline, inherited]

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

References GW::DirProp.

10.18.3.9 double GWNew::getFreq () const [inline]

Definition at line 91 of file LISACODE-GWNew.h.

References Freq.

10.18.3.10 double GW::getLambda () const [inline, inherited]

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

References GW::Lambda.

10.18.3.11 int GW::getNParam () [inline, inherited]

Definition at line 75 of file LISACODE-GW.h.

References GW::NParam.

10.18.3.12 double GWNew::getParam (int iP) [virtual]

Return parameters specified by iP (see function for [setParam](#) the iP code).

Reimplemented from [GW](#).

Definition at line 132 of file LISACODE-GWNew.cpp.

References Amplhc, Amplhp, AnglPol, GW::Beta, Freq, GW::Lambda, Phi0hc, and Phi0hp.

10.18.3.13 double GWNew::getPhi0hc () const [inline]

Definition at line 96 of file LISACODE-GWNew.h.

References Phi0hc.

10.18.3.14 double GWNew::getPhi0hp () const [inline]

Definition at line 95 of file LISACODE-GWNew.h.

References Phi0hp.

10.18.3.15 double GWNew::hc (double *t*) [virtual]

Return $h_x(t)$.

$$\text{returned value} = \text{Amplhc} \cdot \sin(2 \cdot \pi \cdot \text{Freq} \cdot t + \text{Phi0hc})$$

Reimplemented from [GW](#).

Definition at line 205 of file LISACODE-GWNew.cpp.

References Amplhc, Freq, and Phi0hc.

10.18.3.16 double GWNew::hp (double *t*) [virtual]

Return $h_+(t)$.

$$\text{returned value} = \text{Amplhp} \cdot \sin(2 \cdot \pi \cdot \text{Freq} \cdot t + \text{Phi0hp})$$

Reimplemented from [GW](#).

Definition at line 196 of file LISACODE-GWNew.cpp.

References Amplhp, Freq, and Phi0hp.

10.18.3.17 void GWNew::init () [virtual]

Initialization : ...

Reimplemented from [GW](#).

Definition at line 188 of file LISACODE-GWNew.cpp.

10.18.3.18 void GWNew::setAmplhc (double *Amplhc_n*)

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

References Amplhc.

10.18.3.19 void GWNew::setAmplhp (double *Amplhp_n*)

Definition at line 173 of file LISACODE-GWNew.cpp.

References Amplhp.

10.18.3.20 void GWNew::setAnglPol (double *AnglPol_n*)

* Input is checked:

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

Reimplemented from [GW](#).

10.18.3.21 void GW::setBeta (double *Beta_n*) [inherited]

Input is checked:

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

[CalculDirProp](#) method is called to fill [DirProp](#) attribute

Definition at line 125 of file LISACODE-GW.cpp.

References [GW::Beta](#), and [GW::CalculDirProp\(\)](#).

10.18.3.22 void GW::setDirProp (vector< double > *DirProp_n*) [inherited]

Sets [DirProp](#), [Lambda](#) and [Beta](#) attributes.

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

Computations: *DirProp_{norm}* is normalized *DirProp_n*

•

$$\beta = \arcsin(DirProp_{norm}[2])$$

•

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

Definition at line 154 of file LISACODE-GW.cpp.

References [GW::Beta](#), [GW::DirProp](#), [GW::Lambda](#), and [PRECISION](#).

10.18.3.23 void GWNew::setFreq (double *Freq_n*)

Input is checked: *Freq_n* must be positive or null

Definition at line 166 of file LISACODE-GWNew.cpp.

References [Freq](#).

10.18.3.24 void GW::setLambda (double *Lambda_n*) [inherited]

Input is checked:

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

[CalculDirProp](#) method is called to set [DirProp](#) attribute

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

References [GW::CalculDirProp\(\)](#), and [GW::Lambda](#).

10.18.3.25 void GWNew::setParam (int *iP*, double *Param_n*) [virtual]

Sets parameters specified by *iP* :

- 0 -> [Beta](#) : β : Ecliptic latitude (in radians)
- 1 -> [Lambda](#) : λ : Ecliptic longitude (in radians)
- 2 -> [AnglPol](#) : ψ : Polarization angle (in radians)
- 3 -> [Freq](#) : f_{GW} : Frequency (in Hertz)
- 4 -> [Amplhp](#) : h_{0+} : Amplitude of component +
- 5 -> [Amplhc](#) : $h_{0\times}$: Amplitude of component \times
- 6 -> [Phi0hp](#) : ϕ_{0+} : Initial phase of component + (in radians)
- 7 -> [Phi0hc](#) : $\phi_{0\times}$: Initial phase of component \times (in radians).

Reimplemented from [GW](#).

Definition at line 101 of file LISACODE-GWNew.cpp.

References [Amplhc](#), [Amplhp](#), [AnglPol](#), [GW::Beta](#), [Freq](#), [GW::Lambda](#), [Phi0hc](#), and [Phi0hp](#).

10.18.3.26 void GWNew::setPhi0hc (double *Phi0hc_n*) [inline]

Definition at line 102 of file LISACODE-GWNew.h.

References [Phi0hc](#).

10.18.3.27 void GWNew::setPhi0hp (double *Phi0hp_n*) [inline]

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

References [Phi0hp](#).

10.18.4 Member Data Documentation**10.18.4.1 double GWNew::Amplhc [protected]**

`h_{x}` polarisation component amplitude

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

Referenced by [getAmplhc\(\)](#), [getParam\(\)](#), [GWNew\(\)](#), [hc\(\)](#), [setAmplhc\(\)](#), and [setParam\(\)](#).

10.18.4.2 double GWNew::Amplhp [protected]

`h_{+}` polarisation component amplitude

Definition at line 42 of file LISACODE-GWNew.h.

Referenced by `getAmplhp()`, `getParam()`, `GWNew()`, `hp()`, `setAmplhp()`, and `setParam()`.

10.18.4.3 double GWNew::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-GWNew.h.

Referenced by `getAnglPol()`, `getParam()`, `GWNew()`, and `setParam()`.

10.18.4.4 GW::Beta [protected, inherited]

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

Referenced by `GW::CalculDirProp()`, `GW::getBeta()`, `GWSto::getParam()`, `GWPeriGate::getParam()`, `GWNewton2::getParam()`, `getParam()`, `GWMono::getParam()`, `GWFile::getParam()`, `GWFastSpinBBH::getParam()`, `GWCusp::getParam()`, `GWBinary::getParam()`, `GW::getParam()`, `GW::GW()`, `GWBinary::GWBinary()`, `GWNewton2::GWNewton2()`, `GWFastSpinBBH::init()`, `GW::setBeta()`, `GW::setDirProp()`, `GWSto::setParam()`, `GWPeriGate::setParam()`, `GWNewton2::setParam()`, `setParam()`, `GWMono::setParam()`, `GWFile::setParam()`, `GWFastSpinBBH::setParam()`, `GWCusp::setParam()`, `GWBinary::setParam()`, and `GW::setParam()`.

10.18.4.5 vector<double> GW::DirProp [protected, inherited]

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

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

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

10.18.4.6 double GWNew::Freq [protected]

Frequency.

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

Referenced by `getFreq()`, `getParam()`, `GWNew()`, `hc()`, `hp()`, `setFreq()`, and `setParam()`.

10.18.4.7 GW::Lambda [protected, inherited]

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

Referenced by `GW::CalculDirProp()`, `GW::getLambda()`, `GWSto::getParam()`, `GWPeriGate::getParam()`, `GWNewton2::getParam()`, `getParam()`, `GWMono::getParam()`, `GWFile::getParam()`, `GWFastSpinBBH::getParam()`, `GWCusp::getParam()`, `GWBinary::getParam()`, `GW::getParam()`, `GW::GW()`, `GWBinary::GWBinary()`, `GWNewton2::GWNewton2()`, `GWFastSpinBBH::init()`, `GW::setDirProp()`,

GW::setLambda(), GWSto::setParam(), GWPeriGate::setParam(), GWNewton2::setParam(), setParam(), GWMono::setParam(), GWFile::setParam(), GWFastSpinBBH::setParam(), GWCusp::setParam(), GWBinary::setParam(), and GW::setParam().

10.18.4.8 int **GW::NParam** [protected, inherited]

Number of parameters.

Definition at line 53 of file LISACODE-GW.h.

Referenced by GW::getNParam(), GWBinary::GWBinary(), GWCusp::GWCusp(), and GWFastSpinBBH::GWFastSpinBBH().

10.18.4.9 double **GWNew::Phi0hc** [protected]

Initial phase of h_cross polarisation component.

Definition at line 55 of file LISACODE-GWNew.h.

Referenced by getParam(), getPhi0hc(), GWNew(), hc(), setParam(), and setPhi0hc().

10.18.4.10 double **GWNew::Phi0hp** [protected]

Initial phase of h_plus polarisation component.

Definition at line 53 of file LISACODE-GWNew.h.

Referenced by getParam(), getPhi0hp(), GWNew(), hp(), setParam(), and setPhi0hp().

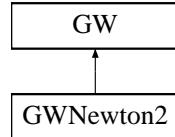
The documentation for this class was generated from the following files:

- [LISACODE-GWNew.h](#)
- [LISACODE-GWNew.cpp](#)

10.19 GWNewton2 Class Reference

```
#include <LISACODE-GWNewton2.h>
```

Inheritance diagram for GWNewton2::



10.19.1 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.
- virtual void [setParam \(int iP, double Param_n\)](#)
Sets parameters specified by iP :
 - 0 -> **Beta** : β : Ecliptic latitude (in radians)
 - 1 -> **Lambda** : λ : Ecliptic longitude (in radians)
 - 2 -> **AnglPol** : ψ : Polarization angle (in radians)
 - 3 -> **m1** : m_1 : Mass of object 1 (in solar masses)
 - 4 -> **m2** : m_2 : Mass of object 2 (in solar masses)
 - 5 -> **tcoal** : t_{coal} : Time of coalescence (in seconds)
 - 6 -> **inc** : i : Inclination angle (in radians)
 - 7 -> **phcoal** : ϕ_c : Phase at coalescence (in radians)
 - 8 -> **rdist** : r : Distance (in kpc).
- virtual double [getParam \(int iP\)](#)
Return parameters specified by iP (see function for `setParam` the iP code).
- double [getM1 \(\)](#)
- double [getM2 \(\)](#)

- double `getTcoal ()`
- double `getInc ()`
- double `getPhCoal ()`
- double `getDistance ()`
- void `setM1` (double `m1_n`)
- void `setM2` (double `m2_n`)
- void `setTcoal` (double `tcoal_n`)

Sets `tcoal` attribute.
- void `setInc` (double `inc_n`)

Gets `inc` attribute.
- void `setPhCoal` (double `phcoal_n`)

Sets `phcoal` attribute.
- void `setDistance` (double `rdist_n`)

Sets `rdist` attribute.
- void `init ()`

Initialization : ...
- double `hp` (double `t`)

Return $h_+(t)$.
- double `hc` (double `t`)

Return $h_\times(t)$.
- virtual void `DispTempVal` (double `t`, ostream *`OutDisp`)
- double `fe` (double `t`)

Return frequency.
- double `phase` (double `t`)

Return phase.
- int `getNParam ()`
- double `getBeta () const`
- void `setBeta` (double `Beta_n`)
- double `getLambda () const`
- void `setLambda` (double `Lambda_n`)
- vector< double > `getDirProp () const`
- void `setDirProp` (vector< double > `DirProp_n`)

Sets `DirProp`, `Lambda` and `Beta` attributes.
- double `getAnglPol ()`
- void `setAnglPol` (double `AnglPol_n`)
- void `CalculDirProp ()`

Computes components of the unit vectors `DirProp` from the angles `Lambda` and `Beta`.

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**
 $\cos(\text{inc})$
- double **si**
 $\sin(\text{inc})$
- double **cmass**
Chirp mass.

- double **time_encour**
Current computation time.
- double **hint**
IPN 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**
Ecliptic latitude Source direction angle (in the heliocentric reference frame) in radians.
- double **Lambda**
Ecliptic longitude Source direction angle (in the heliocentric reference frame) in radians.
- vector< double > **DirProp**
Source direction unit vector (in the heliocentric reference frame).
- double **AngIPol**
Polarisation angle.
- int **NParam**
Number of parameters.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 GWNewton2::GWNewton2 ()

Constructs an instance and initializes it with default values.

$$\begin{aligned}\beta &= 0.917311 - \frac{\pi}{2} \text{ rad} \\ \lambda &= 2.97378 - \pi \text{ rad} \\ AnglPol &= 2.46531 \text{ rad} \\ type &= 1 \\ m_1 &= 10^5 \cdot M_{Sun} \text{ kg} \\ m_2 &= 10^5 \cdot M_{Sun} \text{ kg, where } M_{Sun} = 1.9889 \cdot 10^{30} \text{ kg} \\ rdist &= 10^6 \cdot kpc_m \text{ m, where } kpc_m = 3.0856675807 \cdot 10^{19} \text{ m} \\ \phi_{coal} &= 0 \text{ rad} \\ \tau_{d0} &= 0 \\ \omega_0 &= 0 \\ gw &= 1 \\ inc &= \frac{\pi}{2} \text{ rad} \\ t_{coal} &= 60000 \text{ s}\end{aligned}$$

[init\(\)](#)

Definition at line 37 of file LISACODE-GWNewton2.cpp.

References GW::AnglPol, GW::Beta, gw, inc, init(), kpc_m, GW::Lambda, m1, m2, MS_SI, omega0, phcoal, rdist, taud0, tcoal, and type.

10.19.2.2 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
 - $m_1 = mm1 \cdot M_{Sun} \text{ kg, where } M_{Sun} = 1.9889 \cdot 10^{30} \text{ kg}$
 - $m_2 = mm2 \cdot M_{Sun} \text{ kg, where } M_{Sun} = 1.9889 \cdot 10^{30} \text{ kg}$
 - $rdist = kpc_m \text{ m, where } kpc_m = 3.0856675807 \cdot 10^{19} \text{ m}$
- **phcoal** = pphcoal
- **taud0** = ttaud0
- **omega0** = oomega0

- `gw` = ggw
- `inc` = iinc
- `tcoal` = ttcoal init()

Definition at line 81 of file LISACODE-GWNewton2.cpp.

References gw, inc, init(), kpc_m, m1, m2, MS_SI, omega0, phcoal, rdist, taud0, tcoal, and type.

10.19.2.3 GWNewton2::~GWNewton2 ()

Destructor.

Definition at line 112 of file LISACODE-GWNewton2.cpp.

10.19.3 Member Function Documentation

10.19.3.1 void GW::CalculDirProp () [inherited]

Computes components of the unit vectors `DirProp` from the angles `Lambda` and `Beta`.

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

Definition at line 224 of file LISACODE-GW.cpp.

References `GW::Beta`, `GW::DirProp`, and `GW::Lambda`.

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

10.19.3.2 void GWNewton2::commun (double temps) [protected]

sets `phi`, `omega`, `hint` and `time_encour` attributes depending on `temps` input time.

If $temps \neq \text{time_encour}$, time dependant attributes must be updated and `time_encour` is set to `temps` input :

- if `type=1`

$$\begin{aligned} \phi &= \phi_{coal} - \left(C^3 * \frac{t_{coal} - temps}{5 \cdot G \cdot cmass} \right)^{\frac{5}{8}} \\ \omega &= \frac{C^3}{8 \cdot G \cdot cmass} \cdot * \left(\frac{C^3 (t_{coal} - temps)}{5 \cdot G \cdot cmass} \right)^{-\frac{3}{8}} \\ hint &= -\frac{2 \cdot G \cdot mtot \cdot \nu}{C^2 \cdot rdist} \cdot \left(\frac{G \cdot mtot \cdot \omega}{C^3} \right)^{\frac{2}{3}} \end{aligned}$$

- if `type=2`

$$\begin{aligned} \tau_d &= \nu \cdot C^3 \cdot \frac{t_{coal} - temps}{5 \cdot G \cdot mtot} \\ \phi &= \frac{-1}{\nu} \left(\tau_d^{\frac{5}{8}} + a1 \cdot \tau_d^{\frac{3}{8}} + a2 \cdot \tau_d^{\frac{1}{4}} + a3 \cdot \tau_d^{\frac{1}{8}} + a4 \cdot \log\left(\frac{\tau_d}{\tau_{d0}}\right) + \left(a5 \cdot \log\left(\frac{\tau_d}{256}\right) + a6\right) \cdot \tau_d^{-\frac{1}{8}} + a7 \cdot \tau_d^{-\frac{1}{4}} \right) \end{aligned}$$

$$\omega = b1 \cdot (\tau_d^{-\frac{3}{8}} + b2 \cdot \tau_d^{-\frac{5}{8}} + b3 \cdot \tau_d^{-\frac{4}{8}} + b4 \cdot \tau_d^{-\frac{7}{8}})$$

$$\psi = \phi - \frac{2 \cdot G \cdot m_{tot} \cdot \omega}{C^3} \cdot \log\left(\frac{\omega}{\omega_0}\right)$$

Definition at line 439 of file 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(), hc(), hp(), and phase().

10.19.3.3 void GWNewton2::DispTempVal (double *t*, ostream * *OutDisp*) [virtual]

Reimplemented from [GW](#).

Definition at line 592 of file LISACODE-GWNewton2.cpp.

10.19.3.4 double GWNewton2::fe (double *temps*)

Return frequency.

[commun](#) method is called to update time depending attributes

If *temps* > [tcoal](#) -100, fe=0, else :

$$fe = \frac{\omega}{\pi}$$

Definition at line 618 of file LISACODE-GWNewton2.cpp.

References [commun\(\)](#), fe(), omega, and [tcoal](#).

Referenced by fe().

10.19.3.5 double GW::getAnglPol () [inline, inherited]

Definition at line 83 of file LISACODE-GW.h.

References [GW::AnglPol](#).

10.19.3.6 double GW::getBeta () const [inline, inherited]

Definition at line 76 of file LISACODE-GW.h.

References [GW::Beta](#).

10.19.3.7 vector<double> GW::getDirProp () const [inline, inherited]

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

References [GW::DirProp](#).

10.19.3.8 double GWNewton2::getDistance () [inline]

Definition at line 255 of file LISACODE-GWNewton2.h.

References [rdist](#).

10.19.3.9 double GWNewton2::getInc () [inline]

Definition at line 253 of file LISACODE-GWNewton2.h.

References inc.

10.19.3.10 double GW::getLambda () const [inline, inherited]

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

References GW::Lambda.

10.19.3.11 double GWNewton2::getM1 () [inline]

Definition at line 250 of file LISACODE-GWNewton2.h.

References m1.

10.19.3.12 double GWNewton2::getM2 () [inline]

Definition at line 251 of file LISACODE-GWNewton2.h.

References m2.

10.19.3.13 int GW::getNParam () [inline, inherited]

Definition at line 75 of file LISACODE-GW.h.

References GW::NParam.

10.19.3.14 double GWNewton2::getParam (int iP) [virtual]

Return parameters specified by iP (see function for [setParam](#) the iP code).

Reimplemented from [GW](#).

Definition at line 155 of file LISACODE-GWNewton2.cpp.

References GW::AnglPol, GW::Beta, inc, GW::Lambda, m1, m2, phcoal, rdist, and tcoal.

10.19.3.15 double GWNewton2::getPhCoal () [inline]

Definition at line 254 of file LISACODE-GWNewton2.h.

References phcoal.

10.19.3.16 double GWNewton2::getTcoal () [inline]

Definition at line 252 of file LISACODE-GWNewton2.h.

References tcoal.

10.19.3.17 double GWNewton2::hc (double temps) [virtual]

Return $h_x(t)$.

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

$$hc = a11 \cdot (a22 \cdot \omega)^{\frac{2}{3}} \cdot$$

$$(a1x \cdot \sin(2 \cdot \psi))$$

$$+ \left(\frac{g1 \cdot \omega}{gw} \right)^{\frac{1}{3}} (b1x \cdot (\sin(\psi) - 3 \cdot \sin(3 \cdot \psi)))$$

$$+ \left(\frac{g1 \cdot \omega}{gw} \right)^{\frac{2}{3}} \cdot (c1x \cdot \sin(2 \cdot \psi) + c2x \cdot \sin(4 \cdot \psi))$$

$$+ \frac{g1 \cdot \omega}{gw} (d1x \cdot (d2x \cdot \sin(\psi) + d3x \cdot \sin(3 \cdot \psi) + d4x \cdot \sin(5 \cdot \psi)) + d5x \cdot \sin(2 \cdot \psi))$$

$$+ \left(\frac{g1 \cdot \omega}{gw} \right)^{\frac{4}{3}} \cdot (e1x \sin(2 \cdot \psi) + e2x \sin(4 \cdot \psi) + e3x \sin(6 \cdot \psi))$$

$$+ e4x \cdot ((e5x \cdot \cos(\psi) + e6x \cdot \sin(\psi) + e7x \cdot \sin(3 \cdot \psi) - e8x \cdot \sin(2 \cdot \psi)))$$

Reimplemented from [GW](#).

Definition at line 554 of file 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 hc().

10.19.3.18 double GWNewton2::hp (double temps) [virtual]

Return $h_+(t)$.

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

$$hp = a11 \cdot (a22 \cdot \omega)^{\frac{2}{3}} \cdot (b11 \cdot \cos(2 \cdot \psi))$$

$$+ c1 \cdot \left(\frac{g1 \cdot \omega}{gw} \right)^{\frac{1}{3}} \cdot (c2 \cdot \cos(\psi) - c3 \cdot \cos(3 \cdot \psi))$$

$$+ \left(\frac{g1 \cdot \omega}{gw} \right)^{\frac{2}{3}} \cdot (d1 \cdot \cos(2 \cdot \psi) - d2 \cdot \cos(4 \cdot \psi))$$

$$\begin{aligned}
& + \frac{g1 \cdot \omega}{gw} \cdot (e1(e2 \cdot \cos(\psi) - e3 \cdot \cos(3 \cdot \psi) + e4 \cdot \cos(5 \cdot \psi)) - e5 \cdot \cos(2 \cdot \psi)) \\
& + \left(\frac{g1 \cdot \omega}{gw} \right)^{\frac{2}{3}} \cdot (f1 \cdot \cos(2 \cdot \psi) + f3 \cdot \cos(4 \cdot \psi) - f5 \cdot \cos(6 \cdot \psi) \\
& + f6 \cdot (f7 \sin(\psi) + f8 \cdot \cos(\psi) - f9 \cdot \sin(3 \cdot \psi) + f10 \cdot \cos(3 \cdot \psi)))
\end{aligned}$$

Reimplemented from [GW](#).

Definition at line 486 of file 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 hp().

10.19.3.19 void GWNewton2::init () [virtual]

Initialization : ...

$$m_{tot} = m1 + m2$$

$$deltam = m1 - m2$$

$$\mu = \frac{m1 * m2}{m1 + m2}$$

$$\nu = \frac{\mu}{m_{tot}}$$

$$ci = \cos(inc)$$

$$si = \sin(inc)$$

$$c_{mass} = \mu^{\frac{2}{3}} \cdot m_{tot}^{\frac{2}{5}}$$

$$time_{encour} = -1000$$

- If (type=2)

$$\theta = -\frac{11831}{9240}$$

$$\lambda = \frac{3}{7} \cdot (\theta - \frac{1039}{4620})$$

$$a1 = \frac{55 \cdot \nu}{96} + \frac{3715}{8064}$$

$$a2 = -\frac{3 \cdot \pi}{4}$$

$$a3 = \frac{9275495}{14450688} + \frac{284875}{258048} \cdot \nu + \frac{1855}{2048} \cdot \nu^2$$

$$a4 = \left(-\frac{38645}{172032} + \frac{65}{2048} \cdot \nu \right) \cdot \pi$$

$$a5 = \frac{831032450749357}{57682522275840} - \frac{53}{40} \cdot \pi^2 - \frac{107}{56} \cdot CE_{RG} + \frac{107}{448} \text{ where } CE_{RG} \text{ is Euler constant}$$

$$a6 = \left(-\frac{123292747421}{4161798144} + \frac{2255}{2048} \cdot \pi^2 + \frac{385}{48} \cdot \lambda - \frac{55}{16} \cdot \theta \right) \cdot \nu + \frac{154565}{1835008} \cdot \nu^2 - \frac{1179625}{1769472} \cdot \nu^3$$

$$a7 = \left(\frac{188516689}{173408256} + \frac{488825}{516096} \cdot \nu - \frac{141769}{516096} \cdot \nu^2 \right) \cdot \pi$$

$$\begin{aligned}
b1 &= \frac{C^3}{8 \cdot G \cdot m_{tot}} \\
b2 &= \frac{743}{2688} + \frac{11}{32} \cdot \nu \\
b3 &= -\frac{3 \cdot \pi}{10} \\
b4 &= \frac{1855099}{14450688} + \frac{56975}{258048} \cdot \nu + \frac{371}{2048} \cdot \nu^2 \\
a11 &= \frac{2 \cdot G \cdot m_{tot} \cdot \nu}{r_{dist} \cdot C^2} \\
a22 &= \frac{G \cdot m_{tot}}{C^3} \\
b11 &= -(1 + ci^2) \\
c1 &= -\frac{si}{8} \cdot \frac{deltam}{m_{tot}} \\
c2 &= 5 + ci^2 \\
c3 &= 9 \cdot (1 + ci^2) \\
d1 &= \frac{1}{6} \cdot (19 + 9 \cdot ci^2 - 2 \cdot ci^4 - \nu \cdot (19 - 11 \cdot ci^2 - 6 \cdot ci^4)) \\
d2 &= \frac{4}{3} \cdot si^2 \cdot (1 + ci^2) \cdot (1 - 3 \cdot \nu) \\
e1 &= \frac{si}{192} \cdot \frac{deltam}{m_{tot}} \\
e2 &= (57 + 60 \cdot ci^2 - ci^4) - 2 \cdot \nu \cdot (49 - 12 \cdot ci^2 - ci^4) \\
e3 &= \frac{27}{2} \cdot ((73 + 40 \cdot ci^2 - 9 \cdot ci^4) - 2 \cdot \nu \cdot (25 - 8 \cdot ci^2 - 9 \cdot ci^4)) \\
e4 &= \frac{625}{2} \cdot (1 - 2 \cdot \nu) \cdot si^2 \cdot (1 + ci^2) \\
e5 &= 2 \cdot \pi \cdot (1 + ci^2) \\
f1 &= \frac{1}{120} \cdot (22 + 396 \cdot ci^2 + 145 \cdot ci^4 - 5 \cdot ci^6 + \frac{5}{3} \cdot \nu \cdot (706 - 216 \cdot ci^2 - 251 \cdot ci^4 + 15 \cdot ci^6) - 5 \cdot \nu^2 \cdot (98 - 108 \cdot ci^2 + 7 \cdot ci^4 + 5 \cdot ci^6)) \\
f3 &= \frac{2}{15} \cdot si^2 \cdot ((59 + 35 \cdot ci^2 - 8 \cdot ci^4) - \frac{5}{3} \cdot \nu \cdot (131 + 59 \cdot ci^2 - 24 \cdot ci^4) + 5 \cdot \nu^2 \cdot (21 - 3 \cdot ci^2 - 8 \cdot ci^4)) \\
f5 &= \frac{81}{40} \cdot (1 - 5 \cdot \nu + 5 \cdot \nu^2) \cdot si^4 \cdot (1 + ci^2) \\
f6 &= \frac{si}{40} \cdot \frac{deltam}{m_{tot}} \\
f7 &= 11 + 7 \cdot ci^2 + 10 \cdot (5 + ci^2) \cdot \log(2) \\
f8 &= 5 \cdot \pi \cdot (5 + ci^2) \\
f9 &= 27 \cdot (7 - 10 \cdot \log(\frac{3}{2})) \cdot (1 + ci^2) \\
f10 &= 135 \cdot \pi \cdot (1 + ci^2) \\
a1x &= -2 \cdot ci \\
b1x &= -\frac{3}{4} \cdot si \cdot ci \cdot \frac{deltam}{m_{tot}}
\end{aligned}$$

$$\begin{aligned}
c1x &= \frac{ci}{3} \cdot ((17 - 4 \cdot ci^2) - \nu \cdot (13 - 12 \cdot ci^2)) \\
c2x &= -\frac{8}{3} \cdot (1 - 3 \cdot \nu) \cdot ci \cdot si^2 \\
d1x &= \frac{si \cdot ci}{96} \cdot \frac{deltam}{mtot} \\
d2x &= 63 - 5 \cdot ci^2 - 2 \cdot \nu \cdot (23 - 5 \cdot ci^2) \\
d3x &= -\frac{27}{2} \cdot (67 - 15 \cdot ci^2 - 2 \cdot \nu \cdot (19 - 15 \cdot ci^2)) \\
d4x &= \frac{625}{2} \cdot (1 - 2 \cdot \nu) \cdot si^2 \\
d5x &= -4 \cdot \pi \cdot ci \\
e1x &= \frac{ci}{60} \cdot (68 + 226 \cdot ci^2 - 15 \cdot ci^4 + \frac{5}{3} \cdot \nu \cdot (572 - 490 \cdot ci^2 + 45 \cdot ci^4) - 5 \cdot \nu^2 \cdot (56 - 70 \cdot ci^2 + 15 \cdot ci^4)) \\
e2x &= \frac{4}{15} \cdot ci \cdot si^2 \cdot (55 - 12 \cdot ci^2 - \frac{5}{3} \cdot \nu \cdot (119 - 36 \cdot ci^2) + 5 \cdot \nu^2 \cdot (17 - 12 \cdot ci^2)) \\
e3x &= -\frac{81}{20} \cdot (1 - 5 \cdot \nu + 5 \cdot \nu^2) \cdot ci \cdot si^4 \\
e4x &= -\frac{3}{20} \cdot si \cdot ci \cdot \frac{deltam}{mtot} \\
e5x &= 3 + 10 \cdot \log(2) \\
e6x &= 5 \cdot \pi \\
e7x &= -9 \cdot (7 - 10 \cdot \log(\frac{3}{2})) \\
e8x &= -45 \cdot \pi \\
g1 &= \frac{\cdot G \cdot mtot}{C^3}
\end{aligned}$$

Reimplemented from [GW](#).

Definition at line 331 of file 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, inc, lambda, m1, m2, mtot, mu, nu, rdist, si, teta, time_encour, and type.

Referenced by GWNewton2().

10.19.3.20 double GWNewton2::phase (double temps)

Return phase.

[commun](#) method is called to update time depending attributes

If *temps* > [tcoal](#) -100, fe=0, else :

$$fe = \frac{\omega}{\pi}$$

Definition at line 640 of file LISACODE-GWNewton2.cpp.

References [commun\(\)](#), [phase\(\)](#), [psi](#), and [tcoal](#).

Referenced by [phase\(\)](#).

10.19.3.21 void GW::setAnglPol (double *AnglPol_n*) [inherited]

* Input is checked:

$$\text{AnglPol}_n \in [0, 2 \cdot \pi]$$

Reimplemented in [GWMono](#), and [GWNew](#).

Definition at line 187 of file LISACODE-GW.cpp.

References [GW::AnglPol](#).

10.19.3.22 void GW::setBeta (double *Beta_n*) [inherited]

Input is checked:

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

[CalculDirProp](#) method is called to fill [DirProp](#) attribute

Definition at line 125 of file LISACODE-GW.cpp.

References [GW::Beta](#), and [GW::CalculDirProp\(\)](#).

10.19.3.23 void GW::setDirProp (vector< double > *DirProp_n*) [inherited]

Sets [DirProp](#), [Lambda](#) and [Beta](#) attributes.

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

Computations: *DirProp_{norm}* is normalized *DirProp_n*

•

$$\beta = \arcsin(\text{DirProp}_{\text{norm}}[2])$$

•

$$\lambda = \text{mod}(\text{atan}(\frac{-\text{DirProp}_{\text{norm}}[1]}{-\text{DirProp}_{\text{norm}}[0]}), 2 \cdot \pi)$$

Definition at line 154 of file LISACODE-GW.cpp.

References [GW::Beta](#), [GW::DirProp](#), [GW::Lambda](#), and [PRECISION](#).

10.19.3.24 void GWNewton2::setDistance (double *rdist_n*)

Sets [rdist](#) attribute.

Input is checked:

$$\text{rdist}_n \geq 0$$

Definition at line 244 of file LISACODE-GWNewton2.cpp.

References [rdist](#).

10.19.3.25 void GWNewton2::setInc (double *inc_n*)

Gets `inc` attribute.

Input is checked:

$$0 \leq inc_n \leq 1 \cdot \pi$$

Definition at line 222 of file LISACODE-GWNewton2.cpp.

References `inc`.

10.19.3.26 void GW::setLambda (double *Lambda_n*) [inherited]

Input is checked:

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

`CalculDirProp` method is called to set `DirProp` attribute

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

References `GW::CalculDirProp()`, and `GW::Lambda`.

10.19.3.27 void GWNewton2::setM1 (double *M1_n*)

Input is checked:

$$M1_n \geq 0$$

Definition at line 192 of file LISACODE-GWNewton2.cpp.

References `m1`.

10.19.3.28 void GWNewton2::setM2 (double *M2_n*)

Input is checked:

$$M2_n \geq 0$$

Definition at line 200 of file LISACODE-GWNewton2.cpp.

References `m2`.

10.19.3.29 void GWNewton2::setParam (int *iP*, double *Param_n*) [virtual]

Sets parameters specified by `iP`:

- 0 -> `Beta` : β : Ecliptic latitude (in radians)
- 1 -> `Lambda` : λ : Ecliptic longitude (in radians)
- 2 -> `AnglPol` : ψ : Polarization angle (in radians)
- 3 -> `m1` : m_1 : Mass of object 1 (in solar masses)
- 4 -> `m2` : m_2 : Mass of object 2 (in solar masses)
- 5 -> `tcoal` : t_{coal} : Time of coalescence (in seconds)

- 6 -> `inc` : i : Inclination angle (in radians)
- 7 -> `phcoal` : ϕ_c : Phase at coalescence (in radians)
- 8 -> `rdist` : r : Distance (in kpc).

Reimplemented from [GW](#).

Definition at line 121 of file LISACODE-GWNewton2.cpp.

References `GW::AnglPol`, `GW::Beta`, `inc`, `GW::Lambda`, `m1`, `m2`, `phcoal`, `rdist`, and `tcoal`.

10.19.3.30 void GWNewton2::setPhCoal (double *phcoal_n*)

Sets `phcoal` attribute.

Input is checked:

$$0 \leq phcoal_n \leq 1 \cdot \pi$$

Definition at line 233 of file LISACODE-GWNewton2.cpp.

References `phcoal`.

10.19.3.31 void GWNewton2::setTcoal (double *tcoal_n*)

Sets `tcoal` attribute.

Input is checked:

$$tcoal_n \geq 0$$

Definition at line 211 of file LISACODE-GWNewton2.cpp.

References `tcoal`.

10.19.4 Member Data Documentation

10.19.4.1 `GWNewton2::a1` [protected]

2.5 PN approximation parameter.

Referenced by `commun()`, and `init()`.

10.19.4.2 `GWNewton2::a11` [protected]

2.5 PN approximation parameter.

Referenced by `hc()`, `hp()`, and `init()`.

10.19.4.3 `GWNewton2::a1x` [protected]

2.5 PN approximation parameter.

Referenced by `hc()`, and `init()`.

10.19.4.4 [GWNewton2::a2](#) [protected]

2.5 PN approximation parameter.

Referenced by `commun()`, and `init()`.

10.19.4.5 [GWNewton2::a22](#) [protected]

2.5 PN approximation parameter.

Referenced by `hc()`, `hp()`, and `init()`.

10.19.4.6 [GWNewton2::a3](#) [protected]

2.5 PN approximation parameter.

Referenced by `commun()`, and `init()`.

10.19.4.7 [GWNewton2::a4](#) [protected]

2.5 PN approximation parameter.

Referenced by `commun()`, and `init()`.

10.19.4.8 [GWNewton2::a5](#) [protected]

2.5 PN approximation parameter.

Referenced by `commun()`, and `init()`.

10.19.4.9 [GWNewton2::a6](#) [protected]

2.5 PN approximation parameter.

Referenced by `commun()`, and `init()`.

10.19.4.10 [GWNewton2::a7](#) [protected]

2.5 PN approximation parameter.

Referenced by `commun()`, and `init()`.

10.19.4.11 [double GW::AnglPol](#) [protected, inherited]

Polarisation angle.

Reimplemented in [GWMono](#), [GWNew](#), and [GWPeriGate](#).

Definition at line 51 of file LISACODE-GW.h.

Referenced by `GW::getAnglPol()`, `getParam()`, `GWFile::getParam()`, `GWCusp::getParam()`, `GWBinary::getParam()`, `GW::GW()`, `GWBinary::GWBinary()`, `GWNewton2()`, `GW::setAnglPol()`, `setParam()`, `GWFile::setParam()`, `GWCusp::setParam()`, and `GWBinary::setParam()`.

10.19.4.12 GWNewton2::b1 [protected]

2.5 PN approximation parameter.

Referenced by commun(), and init().

10.19.4.13 GWNewton2::b11 [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.14 GWNewton2::b1x [protected]

2.5 PN approximation parameter.

Referenced by hc(), and init().

10.19.4.15 GWNewton2::b2 [protected]

2.5 PN approximation parameter.

Referenced by commun(), and init().

10.19.4.16 GWNewton2::b3 [protected]

2.5 PN approximation parameter.

Referenced by commun(), and init().

10.19.4.17 GWNewton2::b4 [protected]

2.5 PN approximation parameter.

Referenced by commun(), and init().

10.19.4.18 GW::Beta [protected, inherited]

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

Referenced by GW::CalculDirProp(), GW::getBeta(), GWSto::getParam(), GWPeriGate::getParam(), getParam(), getParam(), GWNew::getParam(), GWMono::getParam(), GWFile::getParam(), GWFastSpinBBH::getParam(), GWCusp::getParam(), GWBinary::getParam(), GW::getParam(), GW::GW(), GWBinary::GWBinary(), GWNewton2(), GWFastSpinBBH::init(), GW::setBeta(), GW::setDirProp(), GWSto::setParam(), GWPeriGate::setParam(), setParam(), GWNew::setParam(), GWMono::setParam(), GWFile::setParam(), GWFastSpinBBH::setParam(), GWCusp::setParam(), GWBinary::setParam(), and GW::setParam().

10.19.4.19 GWNewton2::c1 [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.20 `GWNewton2::c1x` [protected]

2.5 PN approximation parameter.

Referenced by `hc()`, and `init()`.

10.19.4.21 `GWNewton2::c2` [protected]

2.5 PN approximation parameter.

Referenced by `hp()`, and `init()`.

10.19.4.22 `GWNewton2::c2x` [protected]

2.5 PN approximation parameter.

Referenced by `hc()`, and `init()`.

10.19.4.23 `GWNewton2::c3` [protected]

2.5 PN approximation parameter.

Referenced by `hp()`, and `init()`.

10.19.4.24 double `GWNewton2::ci` [protected]

`cos(inc)`

Definition at line 77 of file LISACODE-GWNewton2.h.

Referenced by `hc()`, `hp()`, and `init()`.

10.19.4.25 double `GWNewton2::cmass` [protected]

Chirp mass.

Definition at line 81 of file LISACODE-GWNewton2.h.

Referenced by `commun()`, and `init()`.

10.19.4.26 `GWNewton2::d1` [protected]

2.5 PN approximation parameter.

Referenced by `hp()`, and `init()`.

10.19.4.27 `GWNewton2::d1x` [protected]

2.5 PN approximation parameter.

Referenced by `hc()`, and `init()`.

10.19.4.28 GWNewton2::d2 [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.29 GWNewton2::d2x [protected]

2.5 PN approximation parameter.

Referenced by hc(), and init().

10.19.4.30 GWNewton2::d3x [protected]

2.5 PN approximation parameter.

Referenced by hc(), and init().

10.19.4.31 GWNewton2::d4x [protected]

2.5 PN approximation parameter.

Referenced by hc(), and init().

10.19.4.32 GWNewton2::d5x [protected]

2.5 PN approximation parameter.

Referenced by hc(), and init().

10.19.4.33 double GWNewton2::deltam [protected]

Mass difference.

Definition at line 71 of file LISACODE-GWNewton2.h.

Referenced by init().

10.19.4.34 vector<double> GW::DirProp [protected, inherited]

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

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

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

10.19.4.35 GWNewton2::e1 [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.36 [GWN](#)₂^{ton2::e1x} [protected]

2.5 PN approximation parameter.

Referenced by hc(), and init().

10.19.4.37 [GWN](#)₂^{ton2::e2} [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.38 [GWN](#)₂^{ton2::e2x} [protected]

2.5 PN approximation parameter.

Referenced by hc(), and init().

10.19.4.39 [GWN](#)₂^{ton2::e3} [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.40 [GWN](#)₂^{ton2::e3x} [protected]

2.5 PN approximation parameter.

Referenced by hc(), and init().

10.19.4.41 [GWN](#)₂^{ton2::e4} [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.42 [GWN](#)₂^{ton2::e4x} [protected]

2.5 PN approximation parameter.

Referenced by hc(), and init().

10.19.4.43 [GWN](#)₂^{ton2::e5} [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.44 [GWN](#)₂^{ton2::e5x} [protected]

2.5 PN approximation parameter.

Referenced by hc(), and init().

10.19.4.45 GWNewton2::e6x [protected]

2.5 PN approximation parameter.

Referenced by hc(), and init().

10.19.4.46 GWNewton2::e7x [protected]

2.5 PN approximation parameter.

Referenced by hc(), and init().

10.19.4.47 GWNewton2::e8x [protected]

2.5 PN approximation parameter.

Referenced by hc(), and init().

10.19.4.48 GWNewton2::f1 [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.49 GWNewton2::f10 [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.50 GWNewton2::f3 [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.51 GWNewton2::f5 [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.52 GWNewton2::f6 [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.53 GWNewton2::f7 [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.54 GWNewton2::f8 [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.55 GWNewton2::f9 [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.56 GWNewton2::g1 [protected]

2.5 PN approximation parameter.

Referenced by hc(), hp(), and init().

10.19.4.57 GWNewton2::gw [protected]

System parameter.

Referenced by GWNewton2(), hc(), and hp().

10.19.4.58 double GWNewton2::hint [protected]

1PN amplitude (depending on time)

Definition at line 85 of file LISACODE-GWNewton2.h.

Referenced by commun(), hc(), and hp().

10.19.4.59 double GWNewton2::inc [protected]

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

Definition at line 63 of file LISACODE-GWNewton2.h.

Referenced by getInc(), getParam(), GWNewton2(), init(), setInc(), and setParam().

10.19.4.60 GW::Lambda [protected, inherited]

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

Referenced by GW::CalculDirProp(), GW::getLambda(), GWSto::getParam(), GWPeriGate::getParam(), getParam(), getParam(), GWNew::getParam(), GWMono::getParam(), GWFile::getParam(), GWFastSpinBBH::getParam(), GWCusp::getParam(), GWBinary::getParam(), GW::getParam(), GW::GW(), GWBinary::GWBinary(), GWNewton2(), GWFastSpinBBH::init(), GW::setDirProp(), GW::setLambda(), GWSto::setParam(), GWPeriGate::setParam(), setParam(), GWNew::setParam(), GWMono::setParam(), GWFile::setParam(), GWFastSpinBBH::setParam(), GWCusp::setParam(), GWBinary::setParam(), and GW::setParam().

10.19.4.61 double [GWNewton2::lambda](#) [protected]

System constant.

Referenced by init().

10.19.4.62 double [GWNewton2::m1](#) [protected]

First star mass (in solar masses).

Definition at line 57 of file LISACODE-GWNewton2.h.

Referenced by getM1(), getParam(), GWNewton2(), init(), setM1(), and setParam().

10.19.4.63 double [GWNewton2::m2](#) [protected]

Second star mass (in solar masses).

Definition at line 59 of file LISACODE-GWNewton2.h.

Referenced by getM2(), getParam(), GWNewton2(), init(), setM2(), and setParam().

10.19.4.64 double [GWNewton2::mtot](#) [protected]

Total mass.

Definition at line 69 of file LISACODE-GWNewton2.h.

Referenced by commun(), and init().

10.19.4.65 double [GWNewton2::mu](#) [protected]

Reduced mass.

Definition at line 73 of file LISACODE-GWNewton2.h.

Referenced by init().

10.19.4.66 int [GW::NParam](#) [protected, inherited]

Number of parameters.

Definition at line 53 of file LISACODE-GW.h.

Referenced by GW::getNParam(), GWBinary::GWBinary(), GWCusp::GWCusp(), and GWFastSpin-BBH::GWFastSpinBBH().

10.19.4.67 double [GWNewton2::nu](#) [protected]

Mass ratio.

Definition at line 75 of file LISACODE-GWNewton2.h.

Referenced by commun(), and init().

10.19.4.68 double GWNewton2::omega [protected]

Pulsation (depending on time).

Definition at line 87 of file LISACODE-GWNewton2.h.

Referenced by commun(), fe(), hc(), and hp().

10.19.4.69 double GWNewton2::omega0 [protected]

System parameter.

Referenced by commun(), and GWNewton2().

10.19.4.70 double GWNewton2::phcoal [protected]

Phase before the coalescence.

Definition at line 65 of file LISACODE-GWNewton2.h.

Referenced by commun(), getParam(), getPhCoal(), GWNewton2(), setParam(), and setPhCoal().

10.19.4.71 double GWNewton2::phi [protected]

Phase (depending on time).

Definition at line 89 of file LISACODE-GWNewton2.h.

Referenced by commun(), hc(), and hp().

10.19.4.72 double GWNewton2::psi [protected]

2.5 PN parameter (depending on time).

Referenced by commun(), hc(), hp(), and phase().

10.19.4.73 double GWNewton2::rdist [protected]

Distance to the source in kpc(kiloparsec).

Definition at line 67 of file LISACODE-GWNewton2.h.

Referenced by commun(), getDistance(), getParam(), GWNewton2(), init(), setDistance(), and setParam().

10.19.4.74 double GWNewton2::si [protected]

sin([inc](#))

Definition at line 79 of file LISACODE-GWNewton2.h.

Referenced by init().

10.19.4.75 double GWNewton2::taud [protected]

2.5 PN approximation parameter.

Referenced by `commun()`.

10.19.4.76 **GWNewton2::taud0** [protected]

System parameter.

Referenced by `commun()`, and `GWNewton2()`.

10.19.4.77 **double GWNewton2::tcoal** [protected]

Time before the coalescence.

Definition at line 61 of file LISACODE-GWNewton2.h.

Referenced by `commun()`, `fe()`, `getParam()`, `getTcoal()`, `GWNewton2()`, `hc()`, `hp()`, `phase()`, `setParam()`, and `setTcoal()`.

10.19.4.78 **GWNewton2::teta** [protected]

System constant.

Referenced by `init()`.

10.19.4.79 **double GWNewton2::time_encour** [protected]

Current computation time.

Definition at line 83 of file LISACODE-GWNewton2.h.

Referenced by `commun()`, and `init()`.

10.19.4.80 **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()`, `hp()`, and `init()`.

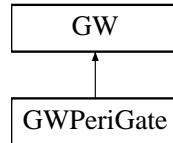
The documentation for this class was generated from the following files:

- [LISACODE-GWNewton2.h](#)
- [LISACODE-GWNewton2.cpp](#)

10.20 GWPeriGate Class Reference

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

Inheritance diagram for GWPeriGate::



10.20.1 Detailed Description

Gravitational Waves periodic gate signal.

`h_plus` and `h_cross` polarisation components 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.
- **~GWPeriGate ()**
Destructor.
- **void setParam (int iP, double Param_n)**
Sets parameters specified by iP :
 - 0 -> **Beta** : β : Ecliptic latitude (in radians)
 - 1 -> **Lambda** : λ : Ecliptic longitude (in radians)
 - 2 -> **AnglPol** : ψ : Polarization angle (in radians)
 - 3 -> **Freq** : f_{GW} : Frequency (in Hertz)
 - 4 -> **Amplhp** : h_{0+} : Amplitude of component +
 - 5 -> **Amplhc** : $h_{0\times}$: Amplitude of component \times .
- **double getParam (int iP)**
Return parameters specified by iP (see function for `setParam` the iP code).
- **double getFreq () const**
Returns `Freq` attribute.
- **void setFreq (double Freq_n)**
- **double getAmplhp () const**
Returns `Amplhp` attribute.

- void `setAmplhp` (double Amplhp_n)
- double `getAmplhc` () const

Returns Amplhc attribute.
- void `setAmplhc` (double Amplhc_n)
- void `init` ()

Initialization : ...
- double `hp` (double t)

Return $h_+(t)$.
- double `hc` (double t)

Return $h_\times(t)$.
- virtual void `DispTempVal` (double t, ostream *OutDisp)
- int `getNParam` ()
- double `getBeta` () const
- void `setBeta` (double Beta_n)
- double `getLambda` () const
- void `setLambda` (double Lambda_n)
- vector< double > `getDirProp` () const
- void `setDirProp` (vector< double > DirProp_n)

Sets DirProp, Lambda and Beta attributes.
- double `getAnglPol` ()
- void `setAnglPol` (double AnglPol_n)
- void `CalculDirProp` ()

Computes components of the unit vectors DirProp from the angles Lambda and Beta.

Protected Attributes

- double `Freq`

Frequency.
- double `Amplhp`

hp component amplitude
- double `Amplhc`

hc component amplitude
- double `AnglPol`

Polarisation angle (rad).
- double `Beta`

Ecliptic latitude Source direction angle (in the heliocentric reference frame) in radians.
- double `Lambda`

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

- `vector< double > DirProp`
Source direction unit vector (in the heliocentric reference frame).
- `int NParam`
Number of parameters.

10.20.2 Constructor & Destructor Documentation

10.20.2.1 GWPeriGate::GWPeriGate ()

Constructs an instance and initializes it with default values.

- `Freq = 0`
- `Amplhp = 0`
- `Amplhc = 0`
- `AnglPol = 0`

Definition at line 26 of file LISACODE-GWPeriGate.cpp.

References Amplhc, Amplhp, AnglPol, and Freq.

10.20.2.2 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 45 of file LISACODE-GWPeriGate.cpp.

References Amplhc, Amplhp, and Freq.

10.20.2.3 GWPeriGate::~GWPeriGate ()

Destructor.

Definition at line 60 of file LISACODE-GWPeriGate.cpp.

10.20.3 Member Function Documentation

10.20.3.1 void GW::CalcuDirProp () [inherited]

Computes components of the unit vectors [DirProp](#) from the angles [Lambda](#) and [Beta](#).

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

Definition at line 224 of file LISACODE-GW.cpp.

References [GW::Beta](#), [GW::DirProp](#), and [GW::Lambda](#).

Referenced by [GW::GW\(\)](#), [GWBinary::GWBinary\(\)](#), [GWFastSpinBBH::GWFastSpinBBH\(\)](#), [GW::setBeta\(\)](#), and [GW::setLambda\(\)](#).

10.20.3.2 void GWPeriGate::DispTempVal (double *t*, ostream * *OutDisp*) [virtual]

Reimplemented from [GW](#).

Definition at line 173 of file LISACODE-GWPeriGate.cpp.

10.20.3.3 double GWPeriGate::getAmplhc () const [inline]

Returns [Amplhc](#) attribute.

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

References [Amplhc](#).

10.20.3.4 double GWPeriGate::getAmplhp () const [inline]

Returns [Amplhp](#) attribute.

Definition at line 77 of file LISACODE-GWPeriGate.h.

References [Amplhp](#).

10.20.3.5 double GW::getAnglPol () [inline, inherited]

Definition at line 83 of file LISACODE-GW.h.

References [GW::AnglPol](#).

10.20.3.6 double GW::getBeta () const [inline, inherited]

Definition at line 76 of file LISACODE-GW.h.

References [GW::Beta](#).

10.20.3.7 vector<double> GW::getDirProp () const [inline, inherited]

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

References `GW::DirProp`.

10.20.3.8 double GWPeriGate::getFreq () const [inline]

Returns `Freq` attribute.

Definition at line 74 of file LISACODE-GWPeriGate.h.

References `Freq`.

10.20.3.9 double GW::getLambda () const [inline, inherited]

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

References `GW::Lambda`.

10.20.3.10 int GW::getNParam () [inline, inherited]

Definition at line 75 of file LISACODE-GW.h.

References `GW::NParam`.

10.20.3.11 double GWPeriGate::getParam (int iP) [virtual]

Return parameters specified by iP (see function for `setParam` the iP code).

Reimplemented from `GW`.

Definition at line 93 of file LISACODE-GWPeriGate.cpp.

References `Amplhc`, `Amplhp`, `AnglPol`, `GW::Beta`, `Freq`, and `GW::Lambda`.

10.20.3.12 double GWPeriGate::hc (double t) [virtual]

Return $h_x(t)$.

$$\text{returned value} = \begin{cases} Amplhc & \text{if } \frac{Freq \cdot t}{1} - ceil(\frac{Freq \cdot t}{1}) < 0.5 \\ 0 & \text{else} \end{cases}$$

Reimplemented from `GW`.

Definition at line 166 of file LISACODE-GWPeriGate.cpp.

References `Amplhc`, and `Freq`.

10.20.3.13 double GWPeriGate::hp (double t) [virtual]

Return $h_+(t)$.

$$\text{returned value} = \begin{cases} Amplhp & \text{if } \frac{Freq \cdot t}{1} - ceil(\frac{Freq \cdot t}{1}) < 0.5 \\ 0 & \text{else} \end{cases}$$

Reimplemented from `GW`.

Definition at line 155 of file LISACODE-GWPeriGate.cpp.

References Amplhp, and Freq.

10.20.3.14 void GWPeriGate::init () [virtual]

Initialization : ...

Nothing !

Reimplemented from [GW](#).

Definition at line 143 of file LISACODE-GWPeriGate.cpp.

10.20.3.15 void GWPeriGate::setAmplhc (double *Amplhc_n*)

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

References Amplhc.

10.20.3.16 void GWPeriGate::setAmplhp (double *Amplhp_n*)

Definition at line 128 of file LISACODE-GWPeriGate.cpp.

References Amplhp.

10.20.3.17 void GW::setAnglPol (double *AnglPol_n*) [inherited]

* Input is checked:

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

Reimplemented in [GWMono](#), and [GWNew](#).

Definition at line 187 of file LISACODE-GW.cpp.

References GW::AnglPol.

10.20.3.18 void GW::setBeta (double *Beta_n*) [inherited]

Input is checked:

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

[CalculDirProp](#) method is called to fill [DirProp](#) attribute

Definition at line 125 of file LISACODE-GW.cpp.

References GW::Beta, and GW::CalculDirProp().

10.20.3.19 void GW::setDirProp (vector< double > *DirProp_n*) [inherited]

Sets [DirProp](#), [Lambda](#) and [Beta](#) attributes.

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

Computations: *DirProp_{norm}* is normalized DirProp_n

-

$$\beta = \arcsin(\text{DirProp}_{norm}[2])$$

-

$$\lambda = \text{mod}(\text{atan}(\frac{-\text{DirProp}_{norm}[1]}{-\text{DirProp}_{norm}[0]}), 2 \cdot \pi)$$

Definition at line 154 of file LISACODE-GW.cpp.

References GW::Beta, GW::DirProp, GW::Lambda, and PRECISION.

10.20.3.20 void GWPeriGate::setFreq (double *Freq_n*)

Input is checked: *Freq_n* must be positive or null

Definition at line 121 of file LISACODE-GWPeriGate.cpp.

References Freq.

10.20.3.21 void GW::setLambda (double *Lambda_n*) [inherited]

Input is checked:

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

[CalculDirProp](#) method is called to set [DirProp](#) attribute

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

References GW::CalculDirProp(), and GW::Lambda.

10.20.3.22 void GWPeriGate::setParam (int *iP*, double *Param_n*) [virtual]

Sets parameters specified by iP :

- 0 -> [Beta](#) : β : Ecliptic latitude (in radians)
- 1 -> [Lambda](#) : λ : Ecliptic longitude (in radians)
- 2 -> [AnglPol](#) : ψ : Polarization angle (in radians)
- 3 -> [Freq](#) : f_{GW} : Frequency (in Hertz)
- 4 -> [Amplhp](#) : h_{0+} : Amplitude of component +
- 5 -> [Amplhc](#) : $h_{0\times}$: Amplitude of component \times .

Reimplemented from [GW](#).

Definition at line 68 of file LISACODE-GWPeriGate.cpp.

References Amplhc, Amplhp, AnglPol, GW::Beta, Freq, and GW::Lambda.

10.20.4 Member Data Documentation

10.20.4.1 double **GWPeriGate::Amplhc** [protected]

hc component amplitude

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

Referenced by getAmplhc(), getParam(), GWPeriGate(), hc(), setAmplhc(), and setParam().

10.20.4.2 double **GWPeriGate::Amplhp** [protected]

hp component amplitude

Definition at line 42 of file LISACODE-GWPeriGate.h.

Referenced by getAmplhp(), getParam(), GWPeriGate(), hp(), setAmplhp(), and setParam().

10.20.4.3 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 getParam(), GWPeriGate(), and setParam().

10.20.4.4 **GW::Beta** [protected, inherited]

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

Referenced by GW::CalculDirProp(), GW::getBeta(), GWSto::getParam(), getParam(), GWNewton2::getParam(), GWNew::getParam(), GWMono::getParam(), GWFfile::getParam(), GWFastSpinBBH::getParam(), GWCusp::getParam(), GBinary::getParam(), GW::getParam(), GW::GW(), GBinary::GBinary(), GWNewton2::GWNewton2(), GWFastSpinBBH::init(), GW::setBeta(), GW::setDirProp(), GWSto::setParam(), setParam(), GWNewton2::setParam(), GWNew::setParam(), GWMono::setParam(), GWFfile::setParam(), GWFastSpinBBH::setParam(), GWCusp::setParam(), GBinary::setParam(), and GW::setParam().

10.20.4.5 vector<double> **GW::DirProp** [protected, inherited]

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

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

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

10.20.4.6 double **GWPeriGate::Freq** [protected]

Frequency.

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

Referenced by getFreq(), getParam(), GWPeriGate(), hc(), hp(), setFreq(), and setParam().

10.20.4.7 [GW::Lambda](#) [protected, inherited]

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

Referenced by `GW::CalculDirProp()`, `GW::getLambda()`, `GWSto::getParam()`, `getParam()`, `GWNewton2::getParam()`, `GWNew::getParam()`, `GWMono::getParam()`, `GWFile::getParam()`, `GWFastSpinBBH::getParam()`, `GWCusp::getParam()`, `GWBinary::getParam()`, `GW::getParam()`, `GW::GW()`, `GWBinary::GWBinary()`, `GWNewton2::GWNewton2()`, `GWFastSpinBBH::init()`, `GW::setDirProp()`, `GW::setLambda()`, `GWSto::setParam()`, `setParam()`, `GWNewton2::setParam()`, `GWNew::setParam()`, `GWMono::setParam()`, `GWFile::setParam()`, `GWFastSpinBBH::setParam()`, `GWCusp::setParam()`, `GWBinary::setParam()`, and `GW::setParam()`.

10.20.4.8 [int GW::NParam](#) [protected, inherited]

Number of parameters.

Definition at line 53 of file LISACODE-GW.h.

Referenced by `GW::getNParam()`, `GWBinary::GWBinary()`, `GWCusp::GWCusp()`, and `GWFastSpinBBH::GWFastSpinBBH()`.

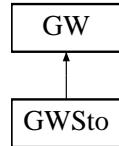
The documentation for this class was generated from the following files:

- [LISACODE-GWPeriGate.h](#)
- [LISACODE-GWPeriGate.cpp](#)

10.21 GWSto Class Reference

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

Inheritance diagram for GWSto::



10.21.1 Detailed Description

Gravitational Waves instantaneous parameters h_{+} and h_{\times} are described in this class.

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

Public Member Functions

- [GWSto \(\)](#)
Constructs an instance and initializes
 - *Beta_n : Ecliptic latitude*
 - *Lambda_n : Ecliptic longitude*
 - *tStep_n : Sample time*
 - *Slope_n : Slope (power of frequency) : α in f^α*
 - *Fknee_n : Minimum value where the shape is f^α .*
 - *Fmin_n : Maximum value where the shape is f^α .*
 - *Fac_Hp_n : Multiplicatif parameter on the amplitude of H_p .*
 - *Fac_Hc_n : Multiplicatif parameter on the amplitude of H_c .*
- [GWSto \(double Beta_n, double Lambda_n, double tStep_n, double Slope_n, double Fknee_n, double Fmin_n, double Fac_Hp_n, double Fac_Hc_n\)](#)
Constructs an instance and initializes
 - *Beta_n : Ecliptic latitude*
 - *Lambda_n : Ecliptic longitude*
 - *tStep_n : Sample time*
 - *Slope_n : Slope (power of frequency) : α in f^α*
 - *Fknee_n : Minimum value where the shape is f^α .*
 - *Fmin_n : Maximum value where the shape is f^α .*
 - *Fac_Hp_n : Multiplicatif parameter on the amplitude of H_p .*
 - *Fac_Hc_n : Multiplicatif parameter on the amplitude of H_c .*
- [GWSto \(double Beta_n, double Lambda_n, double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, double tder_n, double Slope_n, double Tsamp_n, double Fknee_n, double Fmin_n, int Nb_Ageing_n, double Fac_Hp_n, double Fac_Hc_n\)](#)
Sets parameters specified by iP :
 - 0 -> *Beta* : β : Ecliptic latitude (in radians)
 - 1 -> *Lambda* : λ : Ecliptic longitude (in radians)
 - 2 -> *Slope* : α : Slope (power of frequency f^α)
 - 3 -> *Fknee* : f_{knee} : Minimum value where the shape is f^α (in Hertz)
 - 4 -> *Fmin* : f_{min} : Maximum value where the shape is f^α (in Hertz)
 - 5 -> *Fac_Hp* : C_{h+} : Multiplicatif parameter on the amplitude of h_+
 - 6 -> *Fac_Hc* : $C_{h\times}$: Multiplicatif parameter on the amplitude of h_\times .
- [double getParam \(int iP\)](#)
Return parameters specified by iP (see function for [setParam](#) the iP code).
- [void init \(\)](#)

Initialization : ...

- double `hp` (double t)
Return $h_+(t)$.
- double `hc` (double t)
Return $h_\times(t)$.
- virtual void `DispTempVal` (double t, ostream *OutDisp)
- int `getNParam` ()
- double `getBeta` () const
- void `setBeta` (double Beta_n)
- double `getLambda` () const
- void `setLambda` (double Lambda_n)
- vector< double > `getDirProp` () const
- void `setDirProp` (vector< double > DirProp_n)
Sets `DirProp`, `Lambda` and `Beta` attributes.
- double `getAnglPol` ()
- void `setAnglPol` (double AnglPol_n)
- void `CalculDirProp` ()

Computes components of the unit vectors `DirProp` from the angles `Lambda` and `Beta`.

Protected Attributes

- double `tStep`
- double `tDurAdd`
- double `tFirst`
- double `tLast`
- double `Slope`
- double `Tsample`
- double `Fknee`
- double `Fmin`
- int `Nb_Ageing`
- double `Fac_Hp`
- double `Fac_Hc`
- double `Fac_norm`
- double `tder_hp`
- double `tder_hc`
- `Noise * NFhp`
- `Noise * NFhc`
- int `OrderLa`
- double `Beta`

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

- double `Lambda`
Ecliptic longitude Source direction angle (in the heliocentric reference frame) in radians.
- vector< double > `DirProp`

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

- double [AngIPol](#)
Polarisation angle.
- int [NParam](#)
Number of parameters.

10.21.2 Constructor & Destructor Documentation

10.21.2.1 GWSto::GWSto ()

Definition at line 19 of file LISACODE-GWSto.cpp.

10.21.2.2 GWSto::GWSto (double Beta_n, double Lambda_n, double tStep_n, double Slope_n, double Fknee_n, double Fmin_n, double Fac_Hp_n, double Fac_Hc_n)

Constructs an instance and initializes

- Beta_n : Ecliptic latitude
- Lambda_n : Ecliptic longitude
- tStep_n : Sample time
- Slope_n : Slope (power of frequency) : α in f^α
- Fknee_n : Minimum value where the shape is f^α .
- Fmin_n : Maximum value where the shape is f^α .
- Fac_Hp_n : Multiplicatif parameter on the amplitude of Hp.
- Fac_Hc_n : Multiplicatif parameter on the amplitude of Hc.

Definition at line 25 of file LISACODE-GWSto.cpp.

References Fac_Hc, Fac_Hp, Fknee, Fmin, init(), Nb_Ageing, OrderLa, Slope, tDurAdd, tFirst, tLast, and tStep.

10.21.2.3 GWSto::GWSto (double Beta_n, double Lambda_n, double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, double tder_n, double Slope_n, double Tsamp_n, double Fknee_n, double Fmin_n, int Nb_Ageing_n, double Fac_Hp_n, double Fac_Hc_n)

Definition at line 63 of file LISACODE-GWSto.cpp.

References Fac_Hc, Fac_Hp, Fknee, Fmin, init(), Nb_Ageing, Slope, tder_hc, tder_hp, tDurAdd, tFirst, tLast, Tsamp, and tStep.

10.21.3 Member Function Documentation

10.21.3.1 void GW::CalcuDirProp () [inherited]

Computes components of the unit vectors [DirProp](#) from the angles [Lambda](#) and [Beta](#).

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

Definition at line 224 of file LISACODE-GW.cpp.

References [GW::Beta](#), [GW::DirProp](#), and [GW::Lambda](#).

Referenced by [GW::GW\(\)](#), [GWBinary::GWBinary\(\)](#), [GWFastSpinBBH::GWFastSpinBBH\(\)](#), [GW::setBeta\(\)](#), and [GW::setLambda\(\)](#).

10.21.3.2 void GWSto::DispTempVal (double *t*, ostream * *OutDisp*) [virtual]

Reimplemented from [GW](#).

Definition at line 274 of file LISACODE-GWSto.cpp.

10.21.3.3 double GW::getAnglPol () [inline, inherited]

Definition at line 83 of file LISACODE-GW.h.

References [GW::AnglPol](#).

10.21.3.4 double GW::getBeta () const [inline, inherited]

Definition at line 76 of file LISACODE-GW.h.

References [GW::Beta](#).

10.21.3.5 vector<double> GW::getDirProp () const [inline, inherited]

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

References [GW::DirProp](#).

10.21.3.6 double GW::getLambda () const [inline, inherited]

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

References [GW::Lambda](#).

10.21.3.7 int GW::getNParam () [inline, inherited]

Definition at line 75 of file LISACODE-GW.h.

References [GW::NParam](#).

10.21.3.8 double GWSto::getParam (int iP) [virtual]

Return parameters specified by iP (see function for setParam the iP code).

Reimplemented from [GW](#).

Definition at line 134 of file LISACODE-GWSto.cpp.

References [GW::Beta](#), [Fac_Hc](#), [Fac_Hp](#), [Fknee](#), [Fmin](#), [GW::Lambda](#), and [Slope](#).

10.21.3.9 double GWSto::hc (double t) [virtual]

Return $h_x(t)$.

Reimplemented from [GW](#).

Definition at line 248 of file LISACODE-GWSto.cpp.

References [Noise::addNoise\(\)](#), [Fac_Hc](#), [Fac_norm](#), [Noise::getNoise\(\)](#), [NFhc](#), [OrderLa](#), [tder_hc](#), and [tDurAdd](#).

10.21.3.10 double GWSto::hp (double t) [virtual]

Return $h_+(t)$.

Reimplemented from [GW](#).

Definition at line 222 of file LISACODE-GWSto.cpp.

References [Noise::addNoise\(\)](#), [Fac_Hp](#), [Fac_norm](#), [Noise::getNoise\(\)](#), [NFhp](#), [OrderLa](#), [tder_hp](#), and [tDurAdd](#).

10.21.3.11 void GWSto::init () [virtual]

Initialization : ...

Reimplemented from [GW](#).

Definition at line 169 of file LISACODE-GWSto.cpp.

References [Fac_norm](#), [Fknee](#), [Fmin](#), [Nb_Ageing](#), [NFhc](#), [NFhp](#), [Slope](#), [tder_hc](#), [tder_hp](#), [tDurAdd](#), [tFirst](#), [tLast](#), [Tsample](#), and [tStep](#).

Referenced by [GWSto\(\)](#).

10.21.3.12 void GW::setAnglPol (double AnglPol_n) [inherited]

* Input is checked:

$$\text{AnglPol}_n \in [0, 2 \cdot \pi]$$

Reimplemented in [GWMono](#), and [GWNew](#).

Definition at line 187 of file LISACODE-GW.cpp.

References [GW::AnglPol](#).

10.21.3.13 void GW::setBeta (double *Beta_n*) [inherited]

Input is checked:

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

[CalculDirProp](#) method is called to fill [DirProp](#) attribute

Definition at line 125 of file LISACODE-GW.cpp.

References [GW::Beta](#), and [GW::CalculDirProp\(\)](#).

10.21.3.14 void GW::setDirProp (vector< double > *DirProp_n*) [inherited]

Sets [DirProp](#), [Lambda](#) and [Beta](#) attributes.

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

Computations: *DirProp_norm* is normalized *DirProp_n*

-

$$\beta = \arcsin(\text{DirProp}_{\text{norm}}[2])$$

-

$$\lambda = \text{mod}(\text{atan}(\frac{-\text{DirProp}_{\text{norm}}[1]}{-\text{DirProp}_{\text{norm}}[0]}), 2 \cdot \pi)$$

Definition at line 154 of file LISACODE-GW.cpp.

References [GW::Beta](#), [GW::DirProp](#), [GW::Lambda](#), and [PRECISION](#).

10.21.3.15 void GW::setLambda (double *Lambda_n*) [inherited]

Input is checked:

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

[CalculDirProp](#) method is called to set [DirProp](#) attribute

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

References [GW::CalculDirProp\(\)](#), and [GW::Lambda](#).

10.21.3.16 void GWSto::setParam (int *iP*, double *Param_n*) [virtual]

Sets parameters specified by *iP* :

- 0 -> [Beta](#) : β : Ecliptic latitude (in radians)
- 1 -> [Lambda](#) : λ : Ecliptic longitude (in radians)
- 2 -> [Slope](#) : α : Slope (power of frequency f^α)
- 3 -> [Fknee](#) : f_{knee} : Minimum value where the shape is f^α (in Hertz)
- 4 -> [Fmin](#) : f_{min} : Maximum value where the shape is f^α (in Hertz)
- 5 -> [Fac_Hp](#) : C_{h+} : Multiplicatif parameter on the amplitude of h_+

- 6 -> **Fac_Hc** : $C_{h\times}$: Multiplicatif parameter on the amplitude of h_\times .

Reimplemented from [GW](#).

Definition at line 106 of file LISACODE-GWSto.cpp.

References [GW::Beta](#), [Fac_Hc](#), [Fac_Hp](#), [Fknee](#), [Fmin](#), [GW::Lambda](#), and [Slope](#).

10.21.4 Member Data Documentation

10.21.4.1 double **GW::AnglPol** [protected, inherited]

Polarisation angle.

Reimplemented in [GWMono](#), [GWNew](#), and [GWPeriGate](#).

Definition at line 51 of file LISACODE-GW.h.

Referenced by [GW::getAnglPol\(\)](#), [GWNewton2::getParam\(\)](#), [GWFfile::getParam\(\)](#), [GWCusp::getParam\(\)](#), [GWBinary::getParam\(\)](#), [GW::GW\(\)](#), [GWBinary::GWBinary\(\)](#), [GWNewton2::GWNewton2\(\)](#), [GW::setAnglPol\(\)](#), [GWNewton2::setParam\(\)](#), [GWFfile::setParam\(\)](#), [GWCusp::setParam\(\)](#), and [GWBinary::setParam\(\)](#).

10.21.4.2 **GW::Beta** [protected, inherited]

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

Referenced by [GW::CalculDirProp\(\)](#), [GW::getBeta\(\)](#), [getParam\(\)](#), [GWPeriGate::getParam\(\)](#), [GWNewton2::getParam\(\)](#), [GWNew::getParam\(\)](#), [GWMono::getParam\(\)](#), [GWFfile::getParam\(\)](#), [GWFastSpinBBH::getParam\(\)](#), [GWCusp::getParam\(\)](#), [GWBinary::getParam\(\)](#), [GW::getParam\(\)](#), [GW::GW\(\)](#), [GWBinary::GWBinary\(\)](#), [GWNewton2::GWNewton2\(\)](#), [GWFastSpinBBH::init\(\)](#), [GW::setBeta\(\)](#), [GW::setDirProp\(\)](#), [setParam\(\)](#), [GWPeriGate::setParam\(\)](#), [GWNewton2::setParam\(\)](#), [GWNew::setParam\(\)](#), [GWMono::setParam\(\)](#), [GWFfile::setParam\(\)](#), [GWFastSpinBBH::setParam\(\)](#), [GWCusp::setParam\(\)](#), [GWBinary::setParam\(\)](#), and [GW::setParam\(\)](#).

10.21.4.3 vector<double> **GW::DirProp** [protected, inherited]

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

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

Referenced by [GW::CalculDirProp\(\)](#), [GW::getDirProp\(\)](#), [GW::GW\(\)](#), and [GW::setDirProp\(\)](#).

10.21.4.4 double **GWSto::Fac_Hc** [protected]

Definition at line 58 of file LISACODE-GWSto.h.

Referenced by [getParam\(\)](#), [GWSto\(\)](#), [hc\(\)](#), and [setParam\(\)](#).

10.21.4.5 double **GWSto::Fac_Hp** [protected]

Definition at line 57 of file LISACODE-GWSto.h.

Referenced by [getParam\(\)](#), [GWSto\(\)](#), [hp\(\)](#), and [setParam\(\)](#).

10.21.4.6 double GWSto::Fac_norm [protected]

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

Referenced by hc(), hp(), and init().

10.21.4.7 double GWSto::Fknee [protected]

Definition at line 54 of file LISACODE-GWSto.h.

Referenced by getParam(), GWSto(), init(), and setParam().

10.21.4.8 double GWSto::Fmin [protected]

Definition at line 55 of file LISACODE-GWSto.h.

Referenced by getParam(), GWSto(), init(), and setParam().

10.21.4.9 GW::Lambda [protected, inherited]

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

Referenced by GW::CalculDirProp(), GW::getLambda(), getParam(), GWPeriGate::getParam(), GWNewton2::getParam(), GWNew::getParam(), GWMono::getParam(), GWFile::getParam(), GWFastSpinBBH::getParam(), GWCusp::getParam(), GWBinary::getParam(), GW::getParam(), GW::GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), GWFastSpinBBH::init(), GW::setDirProp(), GW::setLambda(), setParam(), GWPeriGate::setParam(), GWNewton2::setParam(), GWNew::setParam(), GWMono::setParam(), GWFile::setParam(), GWFastSpinBBH::setParam(), GWCusp::setParam(), GWBinary::setParam(), and GW::setParam().

10.21.4.10 int GWSto::Nb_Ageing [protected]

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

Referenced by GWSto(), and init().

10.21.4.11 Noise* GWSto::NFhc [protected]

Definition at line 64 of file LISACODE-GWSto.h.

Referenced by hc(), and init().

10.21.4.12 Noise* GWSto::NFhp [protected]

Definition at line 63 of file LISACODE-GWSto.h.

Referenced by hp(), and init().

10.21.4.13 int GW::NParam [protected, inherited]

Number of parameters.

Definition at line 53 of file LISACODE-GW.h.

Referenced by GW::getNParam(), GWBinary::GWBinary(), GWCusp::GWCusp(), and GWFastSpin-BBH::GWFastSpinBBH().

10.21.4.14 int [GWSto::OrderLa](#) [protected]

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

Referenced by GWSto(), hc(), and hp().

10.21.4.15 double [GWSto::Slope](#) [protected]

Definition at line 52 of file LISACODE-GWSto.h.

Referenced by getParam(), GWSto(), init(), and setParam().

10.21.4.16 double [GWSto::tder_hc](#) [protected]

Definition at line 61 of file LISACODE-GWSto.h.

Referenced by GWSto(), hc(), and init().

10.21.4.17 double [GWSto::tder_hp](#) [protected]

Definition at line 61 of file LISACODE-GWSto.h.

Referenced by GWSto(), hp(), and init().

10.21.4.18 double [GWSto::tDurAdd](#) [protected]

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

Referenced by GWSto(), hc(), hp(), and init().

10.21.4.19 double [GWSto::tFirst](#) [protected]

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

Referenced by GWSto(), and init().

10.21.4.20 double [GWSto::tLast](#) [protected]

Definition at line 51 of file LISACODE-GWSto.h.

Referenced by GWSto(), and init().

10.21.4.21 double [GWSto::Tsample](#) [protected]

Definition at line 53 of file LISACODE-GWSto.h.

Referenced by GWSto(), and init().

10.21.4.22 double GWSto::tStep [protected]

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

Referenced by GWSto(), and init().

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

- [LISACODE-GWSto.h](#)
- [LISACODE-GWSto.cpp](#)

10.22 LISA Class Reference

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

10.22.1 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 satellites position ([SCPos](#)), the set of photodiodes-phasemeters ([PhotoDetects](#)) and the gravitational waves transfert function ([sGW](#)). It is related to memory object to which data are sent ([RecordPDPM](#)).

Definition at line 62 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 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.`

10.22.2 Constructor & Destructor Documentation

10.22.2.1 LISACode::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 intialized with `tMemRAM` and `tStepMes` attributes

- **USOs** = 3 **USOClock** elements initialized 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**.

10.22.2.2 **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 initialized with 0.0 value
- **NoisePointers** = 6 white noises (NULL if **PSDLaser** input = 0.0), 6 opticals benches 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**.

10.22.2.3 **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 intialized 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 Geometry::DispInfo(), ConfigSim::getGeometry(), ConfigSim::getGWB(), ConfigSim::getGWs(), ConfigSim::getNoises(), 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, ConfigSim::UseInternalPhasemeter(), and USOs.

10.22.2.4 LISA::~LISA ()

Destructor.

Definition at line 340 of file LISACODE-LISA.cpp.

10.22.3 Member Function Documentation

10.22.3.1 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 406 of file LISACODE-LISA.cpp.

References [c_SI](#), [Geometry::gtdelay\(\)](#), and [SCPos](#).

10.22.3.2 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 366 of file LISACODE-LISA.cpp.

References [Geometry::gtdelay\(\)](#), and [SCPos](#).

Referenced by [main\(\)](#).

10.22.3.3 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 427 of file LISACODE-LISA.cpp.

References [Geometry::gposition\(\)](#), and [SCPos](#).

Referenced by [main\(\)](#).

10.22.3.4 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 494 of file LISACODE-LISA.cpp.

References NoisePointers, PhotoDetects, tStepMes, and USOs.

Referenced by main().

10.22.3.5 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 523 of file LISACODE-LISA.cpp.

References NoisePointers, and tStepPhy.

10.22.3.6 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 455 of file LISACODE-LISA.cpp.

References NoisePointers, PhotoDetects, and tStepMes.

Referenced by LISA().

10.22.4 Member Data Documentation

10.22.4.1 **Background* LISA::GWB** [protected]

Background pointer : gravitational background noise (small mass binaries signal received by phasemeters).

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

Referenced by LISA().

10.22.4.2 **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 72 of file LISACODE-LISA.h.

Referenced by LISA(), MakeOneStepOfTime(), PresentMeanNoise(), and Stabilization().

10.22.4.3 **vector<PhoDetPhaMet> LISA::PhotoDetects** [protected]

Vector of photodetector-phasemeters.

Definition at line 75 of file LISACODE-LISA.h.

Referenced by LISA(), MakeOneStepOfTime(), and Stabilization().

10.22.4.4 **vector<Memory *>* LISA::RecordPDPM** [protected]

Vector of memory objects where the photodetector-phasemeters signals are stored.

Definition at line 77 of file LISACODE-LISA.h.

Referenced by LISA().

10.22.4.5 **Geometry* LISA::SCPos** [protected]

Geometry pointer : spacecraft orbitography structure.

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

Referenced by gArmLength(), gDelayT(), gPosSC(), and LISA().

10.22.4.6 **TrFctGW LISA::sGW** [protected]

Transfer fonction to compute the LISA answer to gravitationnals waves.

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

Referenced by LISA().

10.22.4.7 **double LISA::tMemRAM** [protected]

Duration (time) of photodetector-phasemeters signals stored.

Definition at line 85 of file LISACODE-LISA.h.

Referenced by LISA().

10.22.4.8 double [LISA::tStepMes](#) [protected]

Time step for phasemeters measurement.

Definition at line 87 of file LISACODE-LISA.h.

Referenced by LISA(), MakeOneStepOfTime(), and Stabilization().

10.22.4.9 double [LISA::tStepPhy](#) [protected]

Time step to simulate continuous physical process.

Definition at line 83 of file LISACODE-LISA.h.

Referenced by LISA(), and PresentMeanNoise().

10.22.4.10 vector<[USOClock](#)> [LISA::USOs](#) [protected]

Vector of LISA USO clocks.

There is one USO clock by spacecraft.

Definition at line 92 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](#)

10.23 Mat Class Reference

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

10.23.1 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.

10.23.2 Constructor & Destructor Documentation

10.23.2.1 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.

10.23.2.2 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.

10.23.2.3 Mat::~Mat ()

Destructor.

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

10.23.3 Member Function Documentation

10.23.3.1 void Mat::display ()

Displays matrix components.

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

References p.

10.23.4 Friends And Related Function Documentation

10.23.4.1 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.

10.23.4.2 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.

10.23.4.3 Mat operator+ (Mat A, Mat B) [friend]

Matrices addition. It returns matrix A+B.

Definition at line 70 of file LISACODE-Mat.cpp.

10.23.4.4 Mat operator- (Mat A, Mat B) [friend]

Matrices subtraction. It returns matrix A-B.

Definition at line 82 of file LISACODE-Mat.cpp.

10.23.5 Member Data Documentation**10.23.5.1 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](#)

10.24 MathUtils Class Reference

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

10.24.1 Detailed Description

Angle conversion class.

Definition at line 36 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).
- char * [TimeISO8601](#) ()
Return time in convention ISO-8601.

10.24.2 Member Function Documentation

10.24.2.1 double MathUtils::deg2rad (double *angle_*) [inline, static]

Angle conversion (from degrees to radians).

Definition at line 41 of file LISACODE-MathUtils.h.

Referenced by ConfigSim::gXMLAngle(), and ConfigSim::ReadASCIIFile().

10.24.2.2 double MathUtils::rad2deg (double *angle_*) [inline, static]

Angle conversion (from radians to degrees).

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

Referenced by ConfigSim::ReadASCIIFile(), and ConfigSim::ReadXMLFile().

10.24.2.3 char* MathUtils::TimeISO8601 () [inline, static]

Return time in convention ISO-8601.

Definition at line 54 of file LISACODE-MathUtils.h.

Referenced by ConfigSim::CreateXmlOutputFile(), main(), and MemoryWriteDisk::MakeTitles().

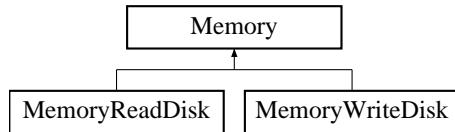
The documentation for this class was generated from the following file:

- [LISACODE-MathUtils.h](#)

10.25 Memory Class Reference

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

Inheritance diagram for Memory::



10.25.1 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.

10.25.2 Constructor & Destructor Documentation

10.25.2.1 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**.

10.25.2.2 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`.

10.25.2.3 Memory::~Memory () [virtual]

Destructor.

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

10.25.3 Member Function Documentation**10.25.3.1 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()`.

10.25.3.2 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`.

10.25.3.3 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(), TDI::ComputeNoEta(), and
TDITools::RefreshDelay().

10.25.3.4 int Memory::getNbSerie () [inline]

Gets the number of the serie.

Definition at line 65 of file LISACODE-Memory.h.

References [ListTmpData](#).

10.25.3.5 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](#).

10.25.3.6 double Memory::gettStepRecord () [inline]

Gets [tStepRecord](#) attribute.

Definition at line 71 of file LISACODE-Memory.h.

References [tStepRecord](#).

10.25.3.7 double Memory::gettStoreData () [inline]

Gets [tStoreData](#) attribute.

Definition at line 67 of file LISACODE-Memory.h.

References [tStoreData](#).

10.25.3.8 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().

10.25.3.9 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().

10.25.3.10 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().

10.25.3.11 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.

10.25.3.12 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.

10.25.3.13 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.

10.25.4 Member Data Documentation

10.25.4.1 `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().

10.25.4.2 `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(), MemoryWriteDisk::MakeTitles(), ReceiveData(), MemoryWriteDisk::RecordAccData(), MemoryReadDisk::RecordAccData(), and RecordAccData().

10.25.4.3 `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(), MemoryWriteDisk::MakeTitles(), Memory(), MemoryReadDisk::ReadASCIIFile(), MemoryReadDisk::ReadBinaryFile(), and settStepRecord().

10.25.4.4 `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().

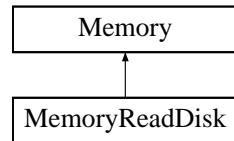
The documentation for this class was generated from the following files:

- [LISACODE-Memory.h](#)
- [LISACODE-Memory.cpp](#)

10.26 MemoryReadDisk Class Reference

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

Inheritance diagram for MemoryReadDisk::



10.26.1 Detailed Description

Class to manage disk reading. This class (module) reads data in file and stores them.

Definition at line 36 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 \(double tStoreData_n, double tStepRecord_n, char *NomFichMem_n, int Encoding_n, int NbColumns, double tTot\)](#)
Adds series in ListTmpData attribute, updates AlreadyRecDat attribute, fills IndexInReadData[SerieNumber] attribute.
- [~MemoryReadDisk \(\)](#)
Destructor.
- [double getMax \(\)](#)
Gets maximum time.
- [void ReadASCIIFile \(\)](#)
Read data in ASCII file.
- [void ReadBinaryFile \(int NbColumns, double tTot\)](#)
- [void AddSerieData \(int SerieNumber, char *TypeName, int IndirectDirName, int iSCName\)](#)
Sets first value of ListTmpData[SerieNumber] to data input value and sets AlreadyRecDat[SerieNumber] to TRUE.
- [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.
- int **FEncoding**
File encoding on which data are recorded (0: ASCII, 1: BINARY).
- 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.

10.26.2 Constructor & Destructor Documentation

10.26.2.1 [MemoryReadDisk::MemoryReadDisk \(\)](#)

Constructs an empty instance.

Definition at line 17 of file LISACODE-MemoryReadDisk.cpp.

10.26.2.2 [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 FEncoding, NomFichMem, and ReadASCIIFile().

10.26.2.3 [MemoryReadDisk::MemoryReadDisk \(double tStoreData_n, double tStepRecord_n, char * NomFichMem_n, int Encoding_n, int NbColumns, double tTot\)](#)

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

References FEncoding, NomFichMem, ReadASCIIFile(), and ReadBinaryFile().

10.26.2.4 [MemoryReadDisk::~MemoryReadDisk \(\)](#)

Destructor.

Definition at line 55 of file LISACODE-MemoryReadDisk.cpp.

10.26.3 Member Function Documentation

10.26.3.1 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 compairing [TitlesReadData](#)[*SerieNumber*] to RequiredTitle build with *IndirectDirName* and *iSCName* inputs.

Reimplemented from [Memory](#).

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

References [Memory::AlreadyRecDat](#), [FEncoding](#), [IndexInReadData](#), [Memory::ListTmpData](#), [TitlesReadData](#), and [Memory::tStepRecord](#).

10.26.3.2 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](#).

10.26.3.3 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\(\)](#), [TDI::ComputeNoEta\(\)](#), and [TDITools::RefreshDelay\(\)](#).

10.26.3.4 int Memory::getNbSerie () [inline, inherited]

Gets the number of the serie.

Definition at line 65 of file LISACODE-Memory.h.

References [Memory::ListTmpData](#).

10.26.3.5 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 66 of file LISACODE-MemoryReadDisk.cpp.

References [ReadData](#), and [Memory::tStepRecord](#).

10.26.3.6 double Memory::gettStepRecord () [inline, inherited]

Gets [tStepRecord](#) attribute.

Definition at line 71 of file LISACODE-Memory.h.

References [Memory::tStepRecord](#).

10.26.3.7 double Memory::gettStoreData () [inline, inherited]

Gets [tStoreData](#) attribute.

Definition at line 67 of file LISACODE-Memory.h.

References [Memory::tStoreData](#).

10.26.3.8 void MemoryReadDisk::MakeTitles (char * *FileNameHead* = " ") [virtual]

Empty method.

Reimplemented from [Memory](#).

Definition at line 244 of file LISACODE-MemoryReadDisk.cpp.

10.26.3.9 void MemoryReadDisk::ReadASCIIFile ()

Read data in ASCII file.

Definition at line 77 of file LISACODE-MemoryReadDisk.cpp.

References [FichMem](#), [NomFichMem](#), [ReadData](#), [TitlesReadData](#), and [Memory::tStepRecord](#).

Referenced by [MemoryReadDisk\(\)](#).

10.26.3.10 void MemoryReadDisk::ReadBinaryFile (int *NbColumns*, double *tTot*)

Definition at line 159 of file LISACODE-MemoryReadDisk.cpp.

References [NomFichMem](#), [ReadData](#), [TitlesReadData](#), and [Memory::tStepRecord](#).

Referenced by [MemoryReadDisk\(\)](#).

10.26.3.11 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](#).

10.26.3.12 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 259 of file LISACODE-MemoryReadDisk.cpp.

References IndexInReadData, [Memory::ListTmpData](#), [ReadData](#), and [Memory::tStoreData](#).

10.26.3.13 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](#).

10.26.3.14 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](#).

10.26.3.15 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](#).

10.26.4 Member Data Documentation

10.26.4.1 `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()`.

10.26.4.2 `int MemoryReadDisk::FEncoding` [protected]

File encoding on which data are recorded (0: ASCII, 1: BINARY).

Definition at line 42 of file LISACODE-MemoryReadDisk.h.

Referenced by `AddSerieData()`, and `MemoryReadDisk()`.

10.26.4.3 `ifstream MemoryReadDisk::FichMem` [protected]

File object managing reading file `NomFichMem`.

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

Referenced by `ReadASCIIFile()`.

10.26.4.4 `vector<int> MemoryReadDisk::IndexInReadData` [protected]

Index in read data.

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

Referenced by `AddSerieData()`, and `RecordAccData()`.

10.26.4.5 `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()`, `MemoryWriteDisk::MakeTitles()`, `Memory::ReceiveData()`, `MemoryWriteDisk::RecordAccData()`, `RecordAccData()`, and `Memory::RecordAccData()`.

10.26.4.6 `char* MemoryReadDisk::NomFichMem` [protected]

File name on which data are read.

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

Referenced by `MemoryReadDisk()`, `ReadASCIIFile()`, and `ReadBinaryFile()`.

10.26.4.7 `vector< vector<double> > MemoryReadDisk::ReadData` [protected]

Data read from file.

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

Referenced by getMax(), ReadASCIIFile(), ReadBinaryFile(), and RecordAccData().

10.26.4.8 **vector<string> MemoryReadDisk::TitlesReadData** [protected]

Titles of read series.

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

Referenced by AddSerieData(), ReadASCIIFile(), and ReadBinaryFile().

10.26.4.9 **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(), getMax(), Memory::gettStepRecord(), MemoryWriteDisk::MakeTitles(), Memory::Memory(), ReadASCIIFile(), ReadBinaryFile(), and Memory::settStepRecord().

10.26.4.10 **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::getMax(), Memory::getMax(), Memory::gettStoreData(), Memory::Memory(), MemoryWriteDisk::RecordAccData(), RecordAccData(), Memory::RecordAccData(), Memory::settStoreData(), and Memory::unusable().

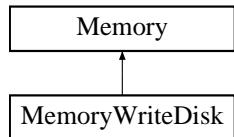
The documentation for this class was generated from the following files:

- [LISACODE-MemoryReadDisk.h](#)
- [LISACODE-MemoryReadDisk.cpp](#)

10.27 MemoryWriteDisk Class Reference

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

Inheritance diagram for MemoryWriteDisk::



10.27.1 Detailed Description

Class to manage disk writing. This class (module) manages and stores data supplied to its in a temporary memory and in a file. Index of first series is 0.

Definition at line 41 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 (double tStoreData_n, double tStepRecord_n, char *NomFichMem_n, int Encoding_n)**
Constructs an instance and initializes it with default values and tStoreData_n, tStepRecord_n and NomFichMem_n inputs.
- **MemoryWriteDisk (double tStoreData_n, double tStepRecord_n, char *NomFichMem_n, int Encoding_n, bool BinHeader_n, double TimeOffset_n, double TimeEnd_n)**
Constructs an instance and initializes it with default values and tStoreData_n, tStepRecord_n and NomFichMem_n inputs.
- **~MemoryWriteDisk ()**
Destructor.
- **double getMax ()**
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 SCSeries[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.
- int **FEncoding**
File encoding on which data are recorded (0: ASCII, 1: BINARY).
- ofstream **FichMem**
File object managing recording file [NomFichMem](#).
- vector< int > **SCSerie**

Spacecraft index for data series.

- `vector< string > TitleSerie`

Vector of series titles.

- `bool BinHeader`

True if header must be written in binary file.

- `double TimeOffset`

For the header.

- `double TimeEnd`

For the header.

- `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.

10.27.2 Constructor & Destructor Documentation

10.27.2.1 MemoryWriteDisk::MemoryWriteDisk ()

Constructs an empty instance.

Definition at line 17 of file LISACODE-MemoryWriteDisk.cpp.

10.27.2.2 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 BinHeader, FEncoding, FichMem, NomFichMem, SCSerie, and TitleSerie.

10.27.2.3 MemoryWriteDisk::MemoryWriteDisk (double tStoreData_n, double tStepRecord_n, char * NomFichMem_n, int Encoding_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 55 of file LISACODE-MemoryWriteDisk.cpp.

References BinHeader, FEncoding, FichMem, NomFichMem, SCSerie, and TitleSerie.

10.27.2.4 MemoryWriteDisk::MemoryWriteDisk (double tStoreData_n, double tStepRecord_n, char * NomFichMem_n, int Encoding_n, bool BinHeader_n, double TimeOffset_n, double TimeEnd_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
- [BinHeader](#) = BinHeader_n

Definition at line 84 of file LISACODE-MemoryWriteDisk.cpp.

References BinHeader, FEncoding, FichMem, NomFichMem, SCSerie, TimeEnd, TimeOffset, and TitleSerie.

10.27.2.5 MemoryWriteDisk::~MemoryWriteDisk ()

Destructor.

Definition at line 105 of file LISACODE-MemoryWriteDisk.cpp.

References CloseFile().

10.27.3 Member Function Documentation

10.27.3.1 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 131 of file LISACODE-MemoryWriteDisk.cpp.

References [Memory::AlreadyRecDat](#), [Memory::ListTmpData](#), [SCSerie](#), [TitleSerie](#), and [Memory::tStepRecord](#).

10.27.3.2 void MemoryWriteDisk::CloseFile ()

Closes [FichMem](#).

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

References [FEncoding](#), and [FichMem](#).

Referenced by [~MemoryWriteDisk\(\)](#).

10.27.3.3 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](#).

10.27.3.4 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\(\)](#), [TDI::ComputeNoEta\(\)](#), and [TDITools::RefreshDelay\(\)](#).

10.27.3.5 int Memory::getNbSerie () [inline, inherited]

Gets the number of the serie.

Definition at line 65 of file LISACODE-Memory.h.

References Memory::ListTmpData.

10.27.3.6 double MemoryWriteDisk::gettMax () [virtual]

Gets maximum time.

returned value = [tStoreData](#) attribute

Reimplemented from [Memory](#).

Definition at line 117 of file LISACODE-MemoryWriteDisk.cpp.

References Memory::tStoreData.

10.27.3.7 double Memory::gettStepRecord () [inline, inherited]

Gets [tStepRecord](#) attribute.

Definition at line 71 of file LISACODE-Memory.h.

References Memory::tStepRecord.

10.27.3.8 double Memory::gettStoreData () [inline, inherited]

Gets [tStoreData](#) attribute.

Definition at line 67 of file LISACODE-Memory.h.

References Memory::tStoreData.

10.27.3.9 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 159 of file LISACODE-MemoryWriteDisk.cpp.

References BinHeader, FEncoding, FichMem, LCVersion, Memory::ListTmpData, SCSerie, TimeEnd, MathUtils::TimeISO8601(), TimeOffset, TitleSerie, and Memory::tStepRecord.

10.27.3.10 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().

10.27.3.11 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 216 of file LISACODE-MemoryWriteDisk.cpp.

References Memory::AlreadyRecDat, FEncoding, [FichMem](#), Memory::ListTmpData, and Memory::tStoreData.

10.27.3.12 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.

10.27.3.13 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.

10.27.3.14 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.

10.27.4 Member Data Documentation

10.27.4.1 `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().

10.27.4.2 `bool MemoryWriteDisk::BinHeader` [protected]

True if header must be written in binary file.

Definition at line 55 of file LISACODE-MemoryWriteDisk.h.

Referenced by MakeTitles(), and MemoryWriteDisk().

10.27.4.3 `int MemoryWriteDisk::FEncoding` [protected]

File encoding on which data are recorded (0: ASCII, 1: BINARY).

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

Referenced by CloseFile(), MakeTitles(), MemoryWriteDisk(), and RecordAccData().

10.27.4.4 `ofstream MemoryWriteDisk::FichMem` [protected]

File object managing recording file `NomFichMem`.

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

Referenced by CloseFile(), MakeTitles(), MemoryWriteDisk(), and RecordAccData().

10.27.4.5 `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(), MakeTitles(), Memory::ReceiveData(), RecordAccData(), MemoryReadDisk::RecordAccData(), and Memory::RecordAccData().

10.27.4.6 `char* MemoryWriteDisk::NomFichMem` [protected]

File name on which data are recorded.

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

Referenced by MemoryWriteDisk().

10.27.4.7 `vector<int> MemoryWriteDisk::SCSerie` [protected]

Spacecraft index for data series.

Definition at line 51 of file LISACODE-MemoryWriteDisk.h.

Referenced by AddSerieData(), MakeTitles(), and MemoryWriteDisk().

10.27.4.8 double MemoryWriteDisk::TimeEnd [protected]

For the header.

Definition at line 57 of file LISACODE-MemoryWriteDisk.h.

Referenced by MakeTitles(), and MemoryWriteDisk().

10.27.4.9 double MemoryWriteDisk::TimeOffset [protected]

For the header.

Definition at line 57 of file LISACODE-MemoryWriteDisk.h.

Referenced by MakeTitles(), and MemoryWriteDisk().

10.27.4.10 vector<string> MemoryWriteDisk::TitleSerie [protected]

Vector of series titles.

Definition at line 53 of file LISACODE-MemoryWriteDisk.h.

Referenced by AddSerieData(), MakeTitles(), and MemoryWriteDisk().

10.27.4.11 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(), MakeTitles(), Memory::Memory(), MemoryReadDisk::ReadASCIIFile(), MemoryReadDisk::ReadBinaryFile(), and Memory::settStepRecord().

10.27.4.12 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 getMax(), Memory::getMax(), Memory::gettStoreData(), Memory::Memory(), RecordAccData(), MemoryReadDisk::RecordAccData(), Memory::RecordAccData(), Memory::settStoreData(), and Memory::unusable().

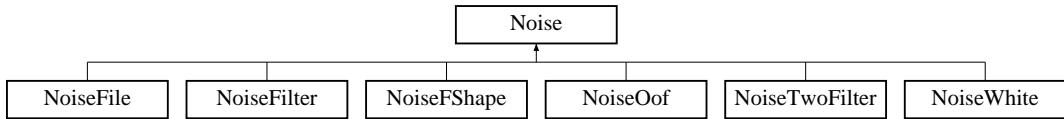
The documentation for this class was generated from the following files:

- [LISACODE-MemoryWriteDisk.h](#)
- [LISACODE-MemoryWriteDisk.cpp](#)

10.28 Noise Class Reference

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

Inheritance diagram for Noise::



10.28.1 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 47 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.
- double [getNoise \(double tDelay, int order\) 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.

10.28.2 Constructor & Destructor Documentation

10.28.2.1 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 34 of file LISACODE-Noise.cpp.

References `loadNoise()`, `NbData`, `NoiseType`, `PRECISION`, `tDurAdd`, `tFirst`, `tLast`, and `tStep`.

10.28.2.2 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 52 of file LISACODE-Noise.cpp.

References `loadNoise()`, `NbData`, `NoiseType`, `PRECISION`, `settDurAdd()`, `settStep()`, `tFirst`, `tLast`, and `tStep`.

10.28.2.3 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 73 of file LISACODE-Noise.cpp.

References [loadNoise\(\)](#), [NbData](#), [NoiseType](#), [PRECISION](#), [settDurAdd\(\)](#), [settFirst\(\)](#), [settLast\(\)](#), [settStep\(\)](#), [tFirst](#), [tLast](#), and [tStep](#).

10.28.2.4 Noise::~Noise () [virtual]

Destructor.

It does not any particular action.

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

10.28.3 Member Function Documentation**10.28.3.1 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 207 of file LISACODE-Noise.cpp.

References [generNoise\(\)](#), [NbBinAdd](#), [NbData](#), and [NoiseData](#).

Referenced by [GWSto::hc\(\)](#), and [GWSto::hp\(\)](#).

10.28.3.2 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](#), [NoiseFShape](#), [NoiseOof](#), [NoiseTwoFilter](#), and [NoiseWhite](#).

Definition at line 194 of file LISACODE-Noise.cpp.

References NoiseData.

Referenced by addNoise().

10.28.3.3 int Noise::getNbBinAdd () const [inline]

It returns the number of bins added, that is the value of [NbBinAdd](#) attribute.

Definition at line 98 of file LISACODE-Noise.h.

References NbBinAdd.

10.28.3.4 double Noise::getNoise (double tDelay, int order) 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 299 of file LISACODE-Noise.cpp.

References NoiseData, PRECISION, [tFirst](#), and [tStep](#).

10.28.3.5 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 226 of file LISACODE-Noise.cpp.

References NoiseData, PRECISION, [tFirst](#), and [tStep](#).

Referenced by [GWSto::hc\(\)](#), and [GWSto::hp\(\)](#).

10.28.3.6 double Noise::gettDurAdd () const [inline]

It returns the duration for a noise addition, that is the value of [tDurAdd](#) attribute.

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

References [tDurAdd](#).

10.28.3.7 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 91 of file LISACODE-Noise.h.

References [tFirst](#).

10.28.3.8 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 95 of file LISACODE-Noise.h.

References [tLast](#).

10.28.3.9 double Noise::gettStep () const [inline]

It returns the time step between two saved data, that is the value of [tStep](#) attribute.

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

References [tStep](#).

10.28.3.10 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](#), [NoiseFShape](#), [NoiseOof](#), [NoiseTwoFilter](#), and [NoiseWhite](#).

Definition at line 184 of file LISACODE-Noise.cpp.

References [NbData](#), and [NoiseData](#).

Referenced by [Noise\(\)](#).

10.28.3.11 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 128 of file LISACODE-Noise.cpp.

References `NbBinAdd`, `PRECISION`, `tDurAdd`, and `tStep`.

Referenced by `Noise()`, `NoiseFile::NoiseFile()`, `NoiseFilter::NoiseFilter()`, `NoiseFShape::NoiseFShape()`, `NoiseOof::NoiseOof()`, `NoiseTwoFilter::NoiseTwoFilter()`, and `NoiseWhite::NoiseWhite()`.

10.28.3.12 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 149 of file LISACODE-Noise.cpp.

References `PRECISION`, `tFirst`, and `tStep`.

Referenced by `Noise()`, `NoiseFile::NoiseFile()`, `NoiseFilter::NoiseFilter()`, `NoiseFShape::NoiseFShape()`, `NoiseOof::NoiseOof()`, `NoiseTwoFilter::NoiseTwoFilter()`, and `NoiseWhite::NoiseWhite()`.

10.28.3.13 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 164 of file LISACODE-Noise.cpp.

References `PRECISION`, `tLast`, and `tStep`.

Referenced by `Noise()`, `NoiseFile::NoiseFile()`, `NoiseFilter::NoiseFilter()`, `NoiseFShape::NoiseFShape()`, `NoiseOof::NoiseOof()`, `NoiseTwoFilter::NoiseTwoFilter()`, and `NoiseWhite::NoiseWhite()`.

10.28.3.14 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 117 of file LISACODE-Noise.cpp.

References tStep.

Referenced by Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), and NoiseWhite::NoiseWhite().

10.28.3.15 bool Noise::TestType (char * *SubmitType*)

It verifies the type of the noise ([NoiseType](#)).

Parameters:

SubmitType String containing the expected noise type.

Returns:

It returns `true` if the Noise object has the expected type. `false` otherwise.

Definition at line 106 of file LISACODE-Noise.cpp.

References NoiseType.

10.28.4 Member Data Documentation

10.28.4.1 int Noise::NbBinAdd [protected]

Number of bins (for each measurement).

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

Referenced by addNoise(), NoiseTwoFilter::generNoise(), NoiseOof::generNoise(), NoiseFShape::generNoise(), NoiseFilter::generNoise(), getNbBinAdd(), and settDurAdd().

10.28.4.2 int Noise::NbData [protected]

Nominal number of data in the noise data vector [NoiseData](#).

Definition at line 61 of file LISACODE-Noise.h.

Referenced by addNoise(), NoiseTwoFilter::generNoise(), NoiseOof::generNoise(), NoiseFShape::generNoise(), NoiseFilter::generNoise(), NoiseWhite::loadNoise(), NoiseTwoFilter::loadNoise(), NoiseOof::loadNoise(), NoiseFShape::loadNoise(), NoiseFilter::loadNoise(), NoiseFile::loadNoise(), loadNoise(), Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), and NoiseWhite::NoiseWhite().

10.28.4.3 vector<double> Noise::NoiseData [protected]

Vector of noise data.

Definition at line 63 of file LISACODE-Noise.h.

Referenced by addNoise(), NoiseWhite::generNoise(), NoiseTwoFilter::generNoise(), NoiseOof::generNoise(), NoiseFShape::generNoise(), NoiseFilter::generNoise(), NoiseFile::generNoise(), generNoise(), getNoise(), NoiseWhite::loadNoise(), NoiseTwoFilter::loadNoise(), NoiseOof::loadNoise(), NoiseFShape::loadNoise(), NoiseFilter::loadNoise(), NoiseFile::loadNoise(), and loadNoise().

10.28.4.4 char Noise::NoiseType[30] [protected]

String to describe the noise type.

Definition at line 65 of file LISACODE-Noise.h.

Referenced by Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and TestType().

10.28.4.5 double Noise::tDurAdd [protected]

Noise computation time step (for each measurement).

Definition at line 53 of file LISACODE-Noise.h.

Referenced by gettDurAdd(), Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and settDurAdd().

10.28.4.6 double Noise::tFirst [protected]

Time of the first data in data vector.

Definition at line 55 of file LISACODE-Noise.h.

Referenced by getNoise(), gettFirst(), Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and settFirst().

10.28.4.7 double Noise::tLast [protected]

Time of the last data in data vector.

Definition at line 57 of file LISACODE-Noise.h.

Referenced by gettLast(), Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and settLast().

10.28.4.8 double Noise::tStep [protected]

Time step in seconds between two saved data. It is used to simulate the continuous signal.

Definition at line 51 of file LISACODE-Noise.h.

Referenced by NoiseTwoFilter::generNoise(), NoiseOof::generNoise(), NoiseFShape::generNoise(), NoiseFilter::generNoise(), getNoise(), NoiseWhite::getPSD(), NoiseWhite::getSqPSD(), gettStep(), NoiseTwoFilter::loadNoise(), NoiseOof::loadNoise(), NoiseFShape::loadNoise(), NoiseFilter::loadNoise(), NoiseFile::loadNoise(), Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), NoiseWhite::setSqPSD(), settDurAdd(), settFirst(), settLast(), and settStep().

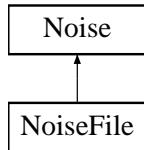
The documentation for this class was generated from the following files:

- [LISACODE-Noise.h](#)
- [LISACODE-Noise.cpp](#)

10.29 NoiseFile Class Reference

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

Inheritance diagram for NoiseFile::



10.29.1 Detailed Description

[Noise](#) derived class to treat files with noise data.

Definition at line 41 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.
- [NoiseFile \(double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, char *FileName_n, double FactMult_n\)](#)
- [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 getStep \(\) 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.
- double `getNoise` (double tDelay, int order) const
It returns the noise value for the specified delay.

Protected Attributes

- char * `FileName`
Name of the file where the noise is read.
- double * `StoredData`
List of noise date read in the file.
- int `NbDataStored`
Number of data stored in the noise data list `StoredData`.
- double `FactMult`
Multiplication factor on data.
- 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.

10.29.2 Constructor & Destructor Documentation

10.29.2.1 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, Noise::tDurAdd, Noise::tFirst, Noise::tLast, and Noise::tStep.

10.29.2.2 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 ([NoiseFile::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 54 of file LISACODE-NoiseFile.cpp.

References [FactMult](#), [FileName](#), [loadNoise\(\)](#), [Noise::NbData](#), [Noise::NoiseType](#), [PRECISION](#), [Noise::settDurAdd\(\)](#), [Noise::settFirst\(\)](#), [Noise::settLast\(\)](#), [Noise::settStep\(\)](#), [Noise::tFirst](#), [Noise::tLast](#), and [Noise::tStep](#).

10.29.2.3 NoiseFile::NoiseFile (double *tStep_n*, double *tDurAdd_n*, double *tFirst_n*, double *tLast_n*, char * *FileName_n*, double *FactMult_n*)

Todo

Make a function to compute a int from a double. Is there a reason to no call `rint` ?

Definition at line 71 of file LISACODE-NoiseFile.cpp.

References [FactMult](#), [FileName](#), [loadNoise\(\)](#), [Noise::NbData](#), [Noise::NoiseType](#), [PRECISION](#), [Noise::settDurAdd\(\)](#), [Noise::settFirst\(\)](#), [Noise::settLast\(\)](#), [Noise::settStep\(\)](#), [Noise::tFirst](#), [Noise::tLast](#), and [Noise::tStep](#).

10.29.3 Member Function Documentation

10.29.3.1 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 207 of file LISACODE-Noise.cpp.

References [Noise::generNoise\(\)](#), [Noise::NbBinAdd](#), [Noise::NbData](#), and [Noise::NoiseData](#).

Referenced by [GWSto::hc\(\)](#), and [GWSto::hp\(\)](#).

10.29.3.2 void NoiseFile::generNoise (int StartBin) [virtual]

Noise generation (for one measurement), using Startbin input as beginning index.

NoiseData StartBin first data are set.

$$\text{for } i=0, \dots, \text{Startbin} \quad \text{NoiseData}[StartBin - i] = \begin{cases} \text{StoredData}[i] & \text{if } i \leq \text{NbDataStored} \\ \text{StoredData}[0] & \text{else} \end{cases}$$

Reimplemented from [Noise](#).

Definition at line 305 of file LISACODE-NoiseFile.cpp.

References NbDataStored, Noise::NoiseData, ReadBin, and StoredData.

10.29.3.3 char * NoiseFile::getFileName ()

It returns the name of the file where the noise is read, [FileName](#).

Definition at line 91 of file LISACODE-NoiseFile.cpp.

References FileName.

10.29.3.4 int Noise::getNbBinAdd () const [inline, inherited]

It returns the number of bins added, that is the value of [NbBinAdd](#) attribute.

Definition at line 98 of file LISACODE-Noise.h.

References Noise::NbBinAdd.

10.29.3.5 int NoiseFile::getNbDataStored ()

It returns [NbDataStored](#), the number of data stored in the noise data list [StoredData](#).

Definition at line 103 of file LISACODE-NoiseFile.cpp.

References NbDataStored.

10.29.3.6 double Noise::getNoise (double tDelay, int order) 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 299 of file LISACODE-Noise.cpp.

References Noise::NoiseData, PRECISION, Noise::tFirst, and Noise::tStep.

10.29.3.7 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 226 of file LISACODE-Noise.cpp.

References Noise::NoiseData, PRECISION, Noise::tFirst, and Noise::tStep.

Referenced by GWSto::hc(), and GWSto::hp().

10.29.3.8 double Noise::gettDurAdd () const [inline, inherited]

It returns the duration for a noise addition, that is the value of tDurAdd attribute.

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

References Noise::tDurAdd.

10.29.3.9 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 91 of file LISACODE-Noise.h.

References Noise::tFirst.

10.29.3.10 double Noise::getLast () const [inline, inherited]

It returns the delay of the last data in data vector, that is the value of [tLast](#) attribute.

Definition at line 95 of file LISACODE-Noise.h.

References Noise::tLast.

10.29.3.11 double Noise::getStep () const [inline, inherited]

It returns the time step between two saved data, that is the value of [tStep](#) attribute.

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

References Noise::tStep.

10.29.3.12 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](#).

Reimplemented from [Noise](#).

Definition at line 116 of file LISACODE-NoiseFile.cpp.

References FactMult, [FileName](#), genunf(), [Noise::NbData](#), [NbDataStored](#), [Noise::NoiseData](#), PRECISION, ReadBin, [StoredData](#), and [Noise::tStep](#).

Referenced by [NoiseFile\(\)](#).

10.29.3.13 void NoiseFile::setFileName (char * *FileName_n*)

It sets the name of the file where the noise is read, [FileName](#).

Definition at line 97 of file LISACODE-NoiseFile.cpp.

References [FileName](#).

10.29.3.14 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 128 of file LISACODE-Noise.cpp.

References [Noise::NbBinAdd](#), PRECISION, [Noise::tDurAdd](#), and [Noise::tStep](#).

Referenced by [Noise::Noise\(\)](#), [NoiseFile\(\)](#), [NoiseFilter::NoiseFilter\(\)](#), [NoiseFShape::NoiseFShape\(\)](#), [NoiseOof::NoiseOof\(\)](#), [NoiseTwoFilter::NoiseTwoFilter\(\)](#), and [NoiseWhite::NoiseWhite\(\)](#).

10.29.3.15 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 149 of file LISACODE-Noise.cpp.

References PRECISION, Noise::tFirst, and Noise::tStep.

Referenced by Noise::Noise(), NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), and NoiseWhite::NoiseWhite().

10.29.3.16 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 164 of file LISACODE-Noise.cpp.

References PRECISION, Noise::tLast, and Noise::tStep.

Referenced by Noise::Noise(), NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), and NoiseWhite::NoiseWhite().

10.29.3.17 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 117 of file LISACODE-Noise.cpp.

References Noise::tStep.

Referenced by Noise::Noise(), NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), and NoiseWhite::NoiseWhite().

10.29.3.18 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 106 of file LISACODE-Noise.cpp.

References Noise::NoiseType.

10.29.4 Member Data Documentation

10.29.4.1 double **NoiseFile::FactMult** [protected]

Multiplication factor on data.

Definition at line 52 of file LISACODE-NoiseFile.h.

Referenced by loadNoise(), and NoiseFile().

10.29.4.2 char* **NoiseFile::FileName** [protected]

Name of the file where the noise is read.

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

Referenced by getFileName(), loadNoise(), NoiseFile(), and setFileName().

10.29.4.3 int **Noise::NbBinAdd** [protected, inherited]

Number of bins (for each measurement).

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

Referenced by Noise::addNoise(), NoiseTwoFilter::generNoise(), NoiseOof::generNoise(), NoiseFShape::generNoise(), NoiseFilter::generNoise(), Noise::getNbBinAdd(), and Noise::settDurAdd().

10.29.4.4 int **Noise::NbData** [protected, inherited]

Nominal number of data in the noise data vector [NoiseData](#).

Definition at line 61 of file LISACODE-Noise.h.

Referenced by Noise::addNoise(), NoiseTwoFilter::generNoise(), NoiseOof::generNoise(), NoiseFShape::generNoise(), NoiseFilter::generNoise(), NoiseWhite::loadNoise(), NoiseTwoFilter::loadNoise(), NoiseOof::loadNoise(), NoiseFShape::loadNoise(), NoiseFilter::loadNoise(), loadNoise(), Noise::loadNoise(), Noise::Noise(), NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), and NoiseWhite::NoiseWhite().

10.29.4.5 int **NoiseFile::NbDataStored** [protected]

Number of data stored in the noise data list [StoredData](#).

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

Referenced by generNoise(), getNbDataStored(), and loadNoise().

10.29.4.6 vector<double> **Noise::NoiseData** [protected, inherited]

Vector of noise data.

Definition at line 63 of file LISACODE-Noise.h.

Referenced by Noise::addNoise(), NoiseWhite::generNoise(), NoiseTwoFilter::generNoise(), NoiseOof::generNoise(), NoiseFShape::generNoise(), NoiseFilter::generNoise(), generNoise(), Noise::gener-

Noise(), Noise::getNoise(), NoiseWhite::loadNoise(), NoiseTwoFilter::loadNoise(), NoiseOof::loadNoise(), NoiseFShape::loadNoise(), NoiseFilter::loadNoise(), loadNoise(), and Noise::loadNoise().

10.29.4.7 **char Noise::NoiseType[30]** [protected, inherited]

String to describe the noise type.

Definition at line 65 of file LISACODE-Noise.h.

Referenced by Noise::Noise(), NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and Noise::TestType().

10.29.4.8 **int NoiseFile::ReadBin** [protected]

Index of last bin read in [StoredData](#).

Definition at line 54 of file LISACODE-NoiseFile.h.

Referenced by generNoise(), and loadNoise().

10.29.4.9 **double* NoiseFile::StoredData** [protected]

List of noise date read in the file.

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

Referenced by generNoise(), and loadNoise().

10.29.4.10 **double Noise::tDurAdd** [protected, inherited]

Noise computation time step (for each measurement).

Definition at line 53 of file LISACODE-Noise.h.

Referenced by Noise::gettDurAdd(), Noise::Noise(), NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and Noise::settDurAdd().

10.29.4.11 **double Noise::tFirst** [protected, inherited]

Time of the first data in data vector.

Definition at line 55 of file LISACODE-Noise.h.

Referenced by Noise::getNoise(), Noise::gettFirst(), Noise::Noise(), NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and Noise::settFirst().

10.29.4.12 **double Noise::tLast** [protected, inherited]

Time of the last data in data vector.

Definition at line 57 of file LISACODE-Noise.h.

Referenced by Noise::gettLast(), Noise::Noise(), NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and Noise::settLast().

10.29.4.13 double **Noise::tStep** [protected, inherited]

Time step in seconds between two saved data. It is used to simulate the continuous signal.

Definition at line 51 of file LISACODE-Noise.h.

Referenced by NoiseTwoFilter::generNoise(), NoiseOof::generNoise(), NoiseFShape::generNoise(), NoiseFilter::generNoise(), Noise::getNoise(), NoiseWhite::getPSD(), NoiseWhite::getSqPSD(), Noise::gettStep(), NoiseTwoFilter::loadNoise(), NoiseOof::loadNoise(), NoiseFShape::loadNoise(), NoiseFilter::loadNoise(), loadNoise(), Noise::Noise(), NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), NoiseWhite::setSqPSD(), Noise::settDurAdd(), Noise::settFirst(), Noise::settLast(), and Noise::settStep().

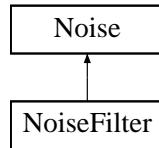
The documentation for this class was generated from the following files:

- [LISACODE-NoiseFile.h](#)
- [LISACODE-NoiseFile.cpp](#)

10.30 NoiseFilter Class Reference

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

Inheritance diagram for NoiseFilter::



10.30.1 Detailed Description

[Noise](#) derived class to treat noise filters.

It creates a noise signal from a filtered 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.
- double `getNoise` (double tDelay, int order) 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.

10.30.2 Constructor & Destructor Documentation

10.30.2.1 [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](#).

10.30.2.2 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](#).

10.30.2.3 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 `Filter::Copy()`, `loadNoise()`, `Noise::NbData`, `NFilter`, `Noise::NoiseType`, `PRECISION`, `Noise::settDurAdd()`, `Noise::settFirst()`, `Noise::settLast()`, `Noise::settStep()`, `Noise::tFirst`, `Noise::tLast`, and `Noise::tStep`.

10.30.2.4 `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`.

10.30.2.5 `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`.

10.30.3 Member Function Documentation

10.30.3.1 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 207 of file LISACODE-Noise.cpp.

References `Noise::generNoise()`, `Noise::NbBinAdd`, `Noise::NbData`, and `Noise::NoiseData`.

Referenced by `GWSto::hc()`, and `GWSto::hp()`.

10.30.3.2 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 :

$$\text{for } i=0, \dots, \text{Startbin } \text{WhiteData}[i] = \sqrt{-\frac{\log(r1)}{tStep} \cdot \cos(2 \cdot \pi \cdot r1)}$$

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`.

10.30.3.3 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`.

10.30.3.4 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](#).

10.30.3.5 int Noise::getNbBinAdd () const [inline, inherited]

It returns the number of bins added, that is the value of [NbBinAdd](#) attribute.

Definition at line 98 of file LISACODE-Noise.h.

References [Noise::NbBinAdd](#).

10.30.3.6 double Noise::getNoise (double tDelay, int order) 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 299 of file LISACODE-Noise.cpp.

References [Noise::NoiseData](#), [PRECISION](#), [Noise::tFirst](#), and [Noise::tStep](#).

10.30.3.7 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 226 of file LISACODE-Noise.cpp.

References `Noise::NoiseData`, `PRECISION`, `Noise::tFirst`, and `Noise::tStep`.

Referenced by `GWSto::hc()`, and `GWSto::hp()`.

10.30.3.8 double Noise::gettDurAdd () const [inline, inherited]

It returns the duration for a noise addition, that is the value of `tDurAdd` attribute.

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

References `Noise::tDurAdd`.

10.30.3.9 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 91 of file LISACODE-Noise.h.

References `Noise::tFirst`.

10.30.3.10 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 95 of file LISACODE-Noise.h.

References `Noise::tLast`.

10.30.3.11 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 82 of file LISACODE-Noise.h.

References `Noise::tStep`.

10.30.3.12 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 :

$$\text{for } i=0, \dots, \text{Startbin } WhiteData[i] = \sqrt{-\frac{\log(r1)}{tStep} \cdot \cos(2 \cdot \pi \cdot r1)}$$

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()`.

10.30.3.13 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 128 of file LISACODE-Noise.cpp.

References `Noise::NbBinAdd`, `PRECISION`, `Noise::tDurAdd`, and `Noise::tStep`.

Referenced by `Noise::Noise()`, `NoiseFile::NoiseFile()`, `NoiseFilter()`, `NoiseFShape::NoiseFShape()`, `NoiseOof::NoiseOof()`, `NoiseTwoFilter::NoiseTwoFilter()`, and `NoiseWhite::NoiseWhite()`.

10.30.3.14 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 149 of file LISACODE-Noise.cpp.

References `PRECISION`, `Noise::tFirst`, and `Noise::tStep`.

Referenced by `Noise::Noise()`, `NoiseFile::NoiseFile()`, `NoiseFilter()`, `NoiseFShape::NoiseFShape()`, `NoiseOof::NoiseOof()`, `NoiseTwoFilter::NoiseTwoFilter()`, and `NoiseWhite::NoiseWhite()`.

10.30.3.15 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 164 of file LISACODE-Noise.cpp.

References PRECISION, Noise::tLast, and Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), and NoiseWhite::NoiseWhite().

10.30.3.16 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 117 of file LISACODE-Noise.cpp.

References Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), and NoiseWhite::NoiseWhite().

10.30.3.17 bool Noise::TestType (char * SubmitType) [inherited]

It verifies the type of the noise ([NoiseType](#)).

Parameters:

SubmitType String containing the expected noise type.

Returns:

It returns `true` if the Noise object has the expected type. `false` otherwise.

Definition at line 106 of file LISACODE-Noise.cpp.

References Noise::NoiseType.

10.30.4 Member Data Documentation**10.30.4.1 int Noise::NbBinAdd [protected, inherited]**

Number of bins (for each measurement).

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

Referenced by Noise::addNoise(), NoiseTwoFilter::generNoise(), NoiseOof::generNoise(), NoiseFShape::generNoise(), generNoise(), Noise::getNbBinAdd(), and Noise::settDurAdd().

10.30.4.2 int Noise::NbData [protected, inherited]

Nominal number of data in the noise data vector [NoiseData](#).

Definition at line 61 of file LISACODE-Noise.h.

Referenced by Noise::addNoise(), NoiseTwoFilter::generNoise(), NoiseOof::generNoise(), NoiseFShape::generNoise(), generNoise(), NoiseWhite::loadNoise(), NoiseTwoFilter::loadNoise(), NoiseOof::loadNoise(), NoiseFShape::loadNoise(), loadNoise(), NoiseFile::loadNoise(), Noise::loadNoise(),

Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), and NoiseWhite::NoiseWhite().

10.30.4.3 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().

10.30.4.4 vector<double> Noise::NoiseData [protected, inherited]

Vector of noise data.

Definition at line 63 of file LISACODE-Noise.h.

Referenced by Noise::addNoise(), NoiseWhite::generNoise(), NoiseTwoFilter::generNoise(), NoiseOof::generNoise(), NoiseFShape::generNoise(), generNoise(), NoiseFile::generNoise(), Noise::generNoise(), Noise::getNoise(), NoiseWhite::loadNoise(), NoiseTwoFilter::loadNoise(), NoiseOof::loadNoise(), NoiseFShape::loadNoise(), loadNoise(), NoiseFile::loadNoise(), and Noise::loadNoise().

10.30.4.5 char Noise::NoiseType[30] [protected, inherited]

String to describe the noise type.

Definition at line 65 of file LISACODE-Noise.h.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and Noise::TestType().

10.30.4.6 double Noise::tDurAdd [protected, inherited]

Noise computation time step (for each measurement).

Definition at line 53 of file LISACODE-Noise.h.

Referenced by Noise::gettDurAdd(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and Noise::settDurAdd().

10.30.4.7 double Noise::tFirst [protected, inherited]

Time of the first data in data vector.

Definition at line 55 of file LISACODE-Noise.h.

Referenced by Noise::getNoise(), Noise::gettFirst(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and Noise::settFirst().

10.30.4.8 double Noise::tLast [protected, inherited]

Time of the last data in data vector.

Definition at line 57 of file LISACODE-Noise.h.

Referenced by Noise::getLast(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and Noise::settLast().

10.30.4.9 double Noise::tStep [protected, inherited]

Time step in seconds between two saved data. It is used to simulate the continuous signal.

Definition at line 51 of file LISACODE-Noise.h.

Referenced by NoiseTwoFilter::generNoise(), NoiseOof::generNoise(), NoiseFShape::generNoise(), generNoise(), Noise::getNoise(), NoiseWhite::getPSD(), NoiseWhite::getSqPSD(), Noise::gettStep(), NoiseTwoFilter::loadNoise(), NoiseOof::loadNoise(), NoiseFShape::loadNoise(), loadNoise(), NoiseFile::loadNoise(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), NoiseWhite::setSqPSD(), Noise::settDurAdd(), Noise::settFirst(), Noise::settLast(), and Noise::settStep().

10.30.4.10 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().

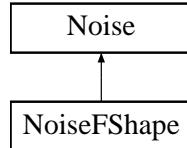
The documentation for this class was generated from the following files:

- [LISACODE-NoiseFilter.h](#)
- [LISACODE-NoiseFilter.cpp](#)

10.31 NoiseFShape Class Reference

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

Inheritance diagram for NoiseFShape::



10.31.1 Detailed Description

[Noise](#) derived class to treat noise filters.

It creates a noise signal from a filtered white noise.

Definition at line 42 of file LISACODE-NoiseFShape.h.

Public Member Functions

- [NoiseFShape \(\)](#)
Base constructor.
- [NoiseFShape \(double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, int NFp_n, int NFm_n, double *FactF_n\)](#)
Constructor.
- [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.
- double `getNoise (double tDelay, int order) const`
It returns the noise value for the specified delay.

Protected Attributes

- vector< double > `WhiteData`
Vector of raw data before filtering.
- vector< `Filter` > `FilterFp`
List of filter in f.
- vector< `Filter` > `FilterFm`
List of filter in 1/f.
- vector< vector< double > > `tmpData_p`
Temporary data for positive power off.
- vector< vector< double > > `tmpData_m`
Temporary data for negative power off.
- int `NFp`
Number of positive power off (f^0, f^1, f^2, \dots).
- int `NFm`
Number of negative power off (f^{-1}, f^{-2}, \dots).
- vector< vector< double > > `FpData`

Data filtered in f, f^2, f^3, \dots

- vector< vector< double > > **FmData**

Data filtered in $f^{-1}, f^{-2}, f^{-3}, \dots$

- double **FactF0**

Factor on f^0 .

- double * **FactFp**

Factor on positive power of f .

- double * **FactFm**

Factor on negative power of f .

- 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.

10.31.2 Constructor & Destructor Documentation

10.31.2.1 NoiseFShape::NoiseFShape ()

Base constructor.

It sets default values for class attributes. ([loadNoise](#)).

- tStep = 0.5
- 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](#))

Definition at line 28 of file LISACODE-NoiseFShape.cpp.

References FactF0, FactFm, FactFp, FilterFm, FilterFp, loadNoise(), Noise::NbData, NFm, Nfp, Noise::NoiseType, PRECISION, Noise::tDurAdd, Noise::tFirst, Noise::tLast, tmpData_m, tmpData_p, and Noise::tStep.

10.31.2.2 NoiseFShape::NoiseFShape (double *tStep_n*, double *tDurAdd_n*, double *tFirst_n*, double *tLast_n*, int *NFp_n*, int *NFm_n*, double * *FactF_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](#).

```
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 63 of file LISACODE-NoiseFShape.cpp.

References FactF0, FactFm, FactFp, FilterFm, FilterFp, loadNoise(), Noise::NbData, NFm, Nfp, Noise::NoiseType, PRECISION, Noise::settDurAdd(), Noise::settFirst(), Noise::settLast(), Noise::settStep(), Noise::tFirst, Noise::tLast, tmpData_m, tmpData_p, and Noise::tStep.

10.31.3 Member Function Documentation

10.31.3.1 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 207 of file LISACODE-Noise.cpp.

References Noise::generNoise(), Noise::NbBinAdd, Noise::NbData, and Noise::NoiseData.

Referenced by GWSto::hc(), and GWSto::hp().

10.31.3.2 void NoiseFShape::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 :

$$\text{for } i=0, \dots, \text{Startbin} \quad \text{WhiteData}[i] = \sqrt{-\frac{\log(r1)}{tStep} \cdot \cos(2 \cdot \pi \cdot r1)}$$

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 252 of file LISACODE-NoiseFShape.cpp.

References FactF0, FactFm, FactFp, FilterFm, FilterFp, genunf(), Noise::NbBinAdd, Noise::NbData, NFm, NFp, Noise::NoiseData, tmpData_m, tmpData_p, Noise::tStep, and WhiteData.

10.31.3.3 int Noise::getNbBinAdd () const [inline, inherited]

It returns the number of bins added, that is the value of [NbBinAdd](#) attribute.

Definition at line 98 of file LISACODE-Noise.h.

References Noise::NbBinAdd.

10.31.3.4 double Noise::getNoise (double tDelay, int order) 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 299 of file LISACODE-Noise.cpp.

References Noise::NoiseData, PRECISION, Noise::tFirst, and Noise::tStep.

10.31.3.5 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 226 of file LISACODE-Noise.cpp.

References Noise::NoiseData, PRECISION, Noise::tFirst, and Noise::tStep.

Referenced by GWSto::hc(), and GWSto::hp().

10.31.3.6 double Noise::gettDurAdd () const [inline, inherited]

It returns the duration for a noise addition, that is the value of tDurAdd attribute.

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

References Noise::tDurAdd.

10.31.3.7 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 91 of file LISACODE-Noise.h.

References Noise::tFirst.

10.31.3.8 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 95 of file LISACODE-Noise.h.

References Noise::tLast.

10.31.3.9 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 82 of file LISACODE-Noise.h.

References Noise::tStep.

10.31.3.10 void NoiseFShape::loadNoise () [virtual]

Initialization.

Number of bins must be adjusted depending on filter stabilization :

$$NbDataStab = 2 \cdot NbData + NbDataStab_{NFilter} + \text{Max}(\text{size}(\alpha_{NFilter}), \text{size}(\beta_{NFilter}))$$

[WhiteData](#) size and [NoiseData](#) size are set to [NbDataStab](#).

[WhiteData](#) is generated as a standard gaussian :

$$\text{for } i=0, \dots, \text{Startbin } WhiteData[i] = \sqrt{-\frac{\log(r1)}{tStep} \cdot \cos(2 \cdot \pi \cdot r1)}$$

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 161 of file LISACODE-NoiseFShape.cpp.

References FactF0, FactFm, FactFp, FilterFm, FilterFp, genunf(), [Noise::NbData](#), [NFm](#), [Nfp](#), [Noise::NoiseData](#), [tmpData_m](#), [tmpData_p](#), [Noise::tStep](#), and [WhiteData](#).

Referenced by [NoiseFShape\(\)](#).

10.31.3.11 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 128 of file LISACODE-Noise.cpp.

References [Noise::NbBinAdd](#), PRECISION, [Noise::tDurAdd](#), and [Noise::tStep](#).

Referenced by [Noise::Noise\(\)](#), [NoiseFile::NoiseFile\(\)](#), [NoiseFilter::NoiseFilter\(\)](#), [NoiseFShape\(\)](#), [NoiseOof::NoiseOof\(\)](#), [NoiseTwoFilter::NoiseTwoFilter\(\)](#), and [NoiseWhite::NoiseWhite\(\)](#).

10.31.3.12 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 149 of file LISACODE-Noise.cpp.

References PRECISION, Noise::tFirst, and Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), and NoiseWhite::NoiseWhite().

10.31.3.13 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 164 of file LISACODE-Noise.cpp.

References PRECISION, Noise::tLast, and Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), and NoiseWhite::NoiseWhite().

10.31.3.14 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 117 of file LISACODE-Noise.cpp.

References Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), and NoiseWhite::NoiseWhite().

10.31.3.15 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 106 of file LISACODE-Noise.cpp.

References Noise::NoiseType.

10.31.4 Member Data Documentation

10.31.4.1 double NoiseFShape::FactF0 [protected]

Factor on f^0 .

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

Referenced by generNoise(), loadNoise(), and NoiseFShape().

10.31.4.2 double* NoiseFShape::FactFm [protected]

Factor on negative power of f.

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

Referenced by generNoise(), loadNoise(), and NoiseFShape().

10.31.4.3 double* NoiseFShape::FactFp [protected]

Factor on positive power of f.

Definition at line 68 of file LISACODE-NoiseFShape.h.

Referenced by generNoise(), loadNoise(), and NoiseFShape().

10.31.4.4 vector<Filter> NoiseFShape::FilterFm [protected]

List of filter in 1/f.

Definition at line 52 of file LISACODE-NoiseFShape.h.

Referenced by generNoise(), loadNoise(), and NoiseFShape().

10.31.4.5 vector<Filter> NoiseFShape::FilterFp [protected]

List of filter in f.

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

Referenced by generNoise(), loadNoise(), and NoiseFShape().

10.31.4.6 vector<vector<double>> NoiseFShape::FmData [protected]

Data filtered in $f^{-1}, f^{-2}, f^{-3}, \dots$

Definition at line 64 of file LISACODE-NoiseFShape.h.

10.31.4.7 vector<vector<double>> NoiseFShape::FpData [protected]

Data filtered in f, f^2, f^3, \dots

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

10.31.4.8 int Noise::NbBinAdd [protected, inherited]

Number of bins (for each measurement).

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

Referenced by Noise::addNoise(), NoiseTwoFilter::generNoise(), NoiseOof::generNoise(), generNoise(), NoiseFilter::generNoise(), Noise::getNbBinAdd(), and Noise::settDurAdd().

10.31.4.9 int Noise::NbData [protected, inherited]

Nominal number of data in the noise data vector [NoiseData](#).

Definition at line 61 of file LISACODE-Noise.h.

Referenced by [Noise::addNoise\(\)](#), [NoiseTwoFilter::generNoise\(\)](#), [NoiseOof::generNoise\(\)](#), [generNoise\(\)](#), [NoiseFilter::generNoise\(\)](#), [NoiseWhite::loadNoise\(\)](#), [NoiseTwoFilter::loadNoise\(\)](#), [NoiseOof::loadNoise\(\)](#), [loadNoise\(\)](#), [NoiseFilter::loadNoise\(\)](#), [NoiseFile::loadNoise\(\)](#), [Noise::loadNoise\(\)](#), [Noise::Noise\(\)](#), [NoiseFile::NoiseFile\(\)](#), [NoiseFilter::NoiseFilter\(\)](#), [NoiseFShape\(\)](#), [NoiseOof::NoiseOof\(\)](#), [NoiseTwoFilter::NoiseTwoFilter\(\)](#), and [NoiseWhite::NoiseWhite\(\)](#).

10.31.4.10 int NoiseFShape::NFm [protected]

Number of negative power of f (f^{-1}, f^{-2}, \dots).

Definition at line 60 of file LISACODE-NoiseFShape.h.

Referenced by [generNoise\(\)](#), [loadNoise\(\)](#), and [NoiseFShape\(\)](#).

10.31.4.11 int NoiseFShape::Nfp [protected]

Number of positive power of f (f^0, f^1, f^2, \dots).

Definition at line 58 of file LISACODE-NoiseFShape.h.

Referenced by [generNoise\(\)](#), [loadNoise\(\)](#), and [NoiseFShape\(\)](#).

10.31.4.12 vector<double> Noise::NoiseData [protected, inherited]

Vector of noise data.

Definition at line 63 of file LISACODE-Noise.h.

Referenced by [Noise::addNoise\(\)](#), [NoiseWhite::generNoise\(\)](#), [NoiseTwoFilter::generNoise\(\)](#), [NoiseOof::generNoise\(\)](#), [generNoise\(\)](#), [NoiseFilter::generNoise\(\)](#), [NoiseFile::generNoise\(\)](#), [Noise::generNoise\(\)](#), [Noise::getNoise\(\)](#), [NoiseWhite::loadNoise\(\)](#), [NoiseTwoFilter::loadNoise\(\)](#), [NoiseOof::loadNoise\(\)](#), [loadNoise\(\)](#), [NoiseFilter::loadNoise\(\)](#), [NoiseFile::loadNoise\(\)](#), and [Noise::loadNoise\(\)](#).

10.31.4.13 char Noise::NoiseType[30] [protected, inherited]

String to describe the noise type.

Definition at line 65 of file LISACODE-Noise.h.

Referenced by [Noise::Noise\(\)](#), [NoiseFile::NoiseFile\(\)](#), [NoiseFilter::NoiseFilter\(\)](#), [NoiseFShape\(\)](#), [NoiseOof::NoiseOof\(\)](#), [NoiseTwoFilter::NoiseTwoFilter\(\)](#), [NoiseWhite::NoiseWhite\(\)](#), and [Noise::TestType\(\)](#).

10.31.4.14 double Noise::tDurAdd [protected, inherited]

Noise computation time step (for each measurement).

Definition at line 53 of file LISACODE-Noise.h.

Referenced by [Noise::gettDurAdd\(\)](#), [Noise::Noise\(\)](#), [NoiseFile::NoiseFile\(\)](#), [NoiseFilter::NoiseFilter\(\)](#), [NoiseFShape\(\)](#), [NoiseOof::NoiseOof\(\)](#), [NoiseTwoFilter::NoiseTwoFilter\(\)](#), [NoiseWhite::NoiseWhite\(\)](#),

and Noise::settDurAdd().

10.31.4.15 double Noise::tFirst [protected, inherited]

Time of the first data in data vector.

Definition at line 55 of file LISACODE-Noise.h.

Referenced by Noise::getNoise(), Noise::gettFirst(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and Noise::settFirst().

10.31.4.16 double Noise::tLast [protected, inherited]

Time of the last data in data vector.

Definition at line 57 of file LISACODE-Noise.h.

Referenced by Noise::gettLast(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and Noise::settLast().

10.31.4.17 vector<vector<double>> NoiseFShape::tmpData_m [protected]

Temporary data for negative power of f.

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

Referenced by generNoise(), loadNoise(), and NoiseFShape().

10.31.4.18 vector<vector<double>> NoiseFShape::tmpData_p [protected]

Temporary data for positive power of f.

Definition at line 54 of file LISACODE-NoiseFShape.h.

Referenced by generNoise(), loadNoise(), and NoiseFShape().

10.31.4.19 double Noise::tStep [protected, inherited]

Time step in seconds between two saved data. It is used to simulate the continuous signal.

Definition at line 51 of file LISACODE-Noise.h.

Referenced by NoiseTwoFilter::generNoise(), NoiseOof::generNoise(), generNoise(), NoiseFilter::generNoise(), Noise::getNoise(), NoiseWhite::getPSD(), NoiseWhite::getSqPSD(), Noise::gettStep(), NoiseTwoFilter::loadNoise(), NoiseOof::loadNoise(), loadNoise(), NoiseFilter::loadNoise(), NoiseFile::loadNoise(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), NoiseWhite::setSqPSD(), Noise::settDurAdd(), Noise::settFirst(), Noise::settLast(), and Noise::settStep().

10.31.4.20 vector<double> NoiseFShape::WhiteData [protected]

Vector of raw data before filtering.

White noise data are generated with a standard gaussian.

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

Referenced by generNoise(), and loadNoise().

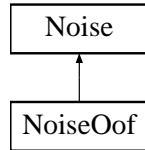
The documentation for this class was generated from the following files:

- [LISACODE-NoiseFShape.h](#)
- [LISACODE-NoiseFShape.cpp](#)

10.32 NoiseOof Class Reference

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

Inheritance diagram for NoiseOof::



10.32.1 Detailed Description

[Noise](#) derived class to treat noise filters.

It creates a noise signal from a filtered 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 45 of file LISACODE-NoiseOof.h.

Public Member Functions

- [**NoiseOof \(\)**](#)
Base constructor.
- [**NoiseOof \(double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, double fmin, double fmax, double alpha, int Nb_Ageing\)**](#)
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.
- double **getNoise** (double tDelay, int order) const
It returns the noise value for the specified delay.

Protected Attributes

- vector< double > **WhiteData**
Vector of raw data before filtering.
- double **fmin**
Minimal and maximal frequency for the range where the filter have the good shape.
- double **fmax**
Minimal and maximal frequency for the range where the filter have the good shape.
- double **alpha**
Power of $1/f \rightarrow$ Shape : $1/f^\alpha$.

- **Filter NFilter**

Filter for the white noise used to generate the final noise.

- int **Nb_Ageing**
- 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.

10.32.2 Constructor & Destructor Documentation

10.32.2.1 NoiseOof::NoiseOof ()

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 29 of file LISACODE-NoiseOof.cpp.

References `Filter::init()`, `loadNoise()`, `Noise::NbData`, `NFilter`, `Noise::NoiseType`, `PRECISION`, `Noise::tDurAdd`, `Noise::tFirst`, `Noise::tLast`, and `Noise::tStep`.

10.32.2.2 `NoiseOof::NoiseOof (double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, double fmin_n, double fmax_n, double alpha_n, int Nb_Ageing_n)`

Constructor.

It sets some class attributes. The other attributes are set to default values (see [NoiseOof::NoiseOof\(\)](#)). It acts like the base constructor ([NoiseOof::NoiseOof\(\)](#)).

Parameters:

tStep_n Value of `tStep`.

tDurAdd_n Value of `tDurAdd`.

tFirst_n Value of `tFirst`.

tLast_n Value of `tLast`.

fmin_n Value of `#fmint`.

fmax_n Value of `fmax`.

alpha_n Value of `alpha`.

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 65 of file LISACODE-NoiseOof.cpp.

References `alpha`, `fmax`, `fmin`, `Filter::init()`, `loadNoise()`, `Nb_Ageing`, `Noise::NbData`, `NFilter`, `Noise::NoiseType`, `PRECISION`, `Noise::settDurAdd()`, `Noise::settFirst()`, `Noise::settLast()`, `Noise::settStep()`, `Noise::tFirst`, `Noise::tLast`, and `Noise::tStep`.

10.32.3 Member Function Documentation

10.32.3.1 `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 207 of file LISACODE-Noise.cpp.

References `Noise::generNoise()`, `Noise::NbBinAdd`, `Noise::NbData`, and `Noise::NoiseData`.

Referenced by `GWSto::hc()`, and `GWSto::hp()`.

10.32.3.2 void NoiseOof::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 :

$$\text{for } i=0, \dots, \text{Startbin } \text{WhiteData}[i] = \sqrt{-\frac{\log(r1)}{tStep} \cdot \cos(2 \cdot \pi \cdot r1)}$$

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 225 of file LISACODE-NoiseOof.cpp.

References [Filter::App\(\)](#), [genunf\(\)](#), [Noise::NbBinAdd](#), [Noise::NbData](#), [NFilter](#), [Noise::NoiseData](#), [Noise::tStep](#), and [WhiteData](#).

10.32.3.3 vector< vector <double> > NoiseOof::getFilterAlpha () [inline]

It returns the list of the alpha parameters (see [Filter::alpha](#)) of the filter [NFilter](#).

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

References [Filter::getAlpha\(\)](#), and [NFilter](#).

10.32.3.4 vector< vector <double> > NoiseOof::getFilterBeta () [inline]

It returns the list of the beta parameters (see [Filter::beta](#)) of the filter [NFilter](#).

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

References [Filter::getBeta\(\)](#), and [NFilter](#).

10.32.3.5 int Noise::getNbBinAdd () const [inline, inherited]

It returns the number of bins added, that is the value of [NbBinAdd](#) attribute.

Definition at line 98 of file LISACODE-Noise.h.

References [Noise::NbBinAdd](#).

10.32.3.6 double Noise::getNoise (double tDelay, int order) 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 299 of file LISACODE-Noise.cpp.

References `Noise::NoiseData`, `PRECISION`, `Noise::tFirst`, and `Noise::tStep`.

10.32.3.7 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 226 of file LISACODE-Noise.cpp.

References `Noise::NoiseData`, `PRECISION`, `Noise::tFirst`, and `Noise::tStep`.

Referenced by `GWSto::hc()`, and `GWSto::hp()`.

10.32.3.8 double Noise::gettDurAdd () const [inline, inherited]

It returns the duration for a noise addition, that is the value of `tDurAdd` attribute.

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

References `Noise::tDurAdd`.

10.32.3.9 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 91 of file LISACODE-Noise.h.

References Noise::tFirst.

10.32.3.10 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 95 of file LISACODE-Noise.h.

References Noise::tLast.

10.32.3.11 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 82 of file LISACODE-Noise.h.

References Noise::tStep.

10.32.3.12 void NoiseOof::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 :

$$\text{for } i=0, \dots, \text{Startbin } WhiteData[i] = \sqrt{-\frac{\log(r1)}{tStep} \cdot \cos(2 \cdot \pi \cdot r1)}$$

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 179 of file LISACODE-NoiseOof.cpp.

References [Filter::App\(\)](#), [genunf\(\)](#), [Filter::getDepth\(\)](#), [Filter::getNbDataStab\(\)](#), [Noise::NbData](#), [NFilter](#), [Noise::NoiseData](#), [Noise::tStep](#), and [WhiteData](#).

Referenced by [NoiseOof\(\)](#).

10.32.3.13 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 128 of file LISACODE-Noise.cpp.

References Noise::NbBinAdd, PRECISION, Noise::tDurAdd, and Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), and NoiseWhite::NoiseWhite().

10.32.3.14 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 149 of file LISACODE-Noise.cpp.

References PRECISION, Noise::tFirst, and Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), and NoiseWhite::NoiseWhite().

10.32.3.15 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 164 of file LISACODE-Noise.cpp.

References PRECISION, Noise::tLast, and Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), and NoiseWhite::NoiseWhite().

10.32.3.16 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 117 of file LISACODE-Noise.cpp.

References Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), and NoiseWhite::NoiseWhite().

10.32.3.17 bool Noise::TestType (char * *SubmitType*) [inherited]

It verifies the type of the noise ([NoiseType](#)).

Parameters:

SubmitType String containing the expected noise type.

Returns:

It returns `true` if the Noise object has the expected type. `false` otherwise.

Definition at line 106 of file LISACODE-Noise.cpp.

References Noise::NoiseType.

10.32.4 Member Data Documentation

10.32.4.1 double NoiseOof::alpha [protected]

Power of $1/f \rightarrow$ Shape : $1/f^\alpha$.

Definition at line 55 of file LISACODE-NoiseOof.h.

Referenced by NoiseOof().

10.32.4.2 double NoiseOof::fmax [protected]

Minimal and maximal frequency for the range where the filter have the good shape.

Definition at line 53 of file LISACODE-NoiseOof.h.

Referenced by NoiseOof().

10.32.4.3 double NoiseOof::fmin [protected]

Minimal and maximal frequency for the range where the filter have the good shape.

Definition at line 53 of file LISACODE-NoiseOof.h.

Referenced by NoiseOof().

10.32.4.4 int NoiseOof::Nb_Ageing [protected]

Definition at line 58 of file LISACODE-NoiseOof.h.

Referenced by NoiseOof().

10.32.4.5 int Noise::NbBinAdd [protected, inherited]

Number of bins (for each measurement).

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

Referenced by Noise::addNoise(), NoiseTwoFilter::generNoise(), generNoise(), NoiseFShape::generNoise(), NoiseFilter::generNoise(), Noise::getNbBinAdd(), and Noise::settDurAdd().

10.32.4.6 int Noise::NbData [protected, inherited]

Nominal number of data in the noise data vector [NoiseData](#).

Definition at line 61 of file LISACODE-Noise.h.

Referenced by [Noise::addNoise\(\)](#), [NoiseTwoFilter::generNoise\(\)](#), [generNoise\(\)](#), [NoiseFShape::generNoise\(\)](#), [NoiseFilter::generNoise\(\)](#), [NoiseWhite::loadNoise\(\)](#), [NoiseTwoFilter::loadNoise\(\)](#), [loadNoise\(\)](#), [NoiseFShape::loadNoise\(\)](#), [NoiseFilter::loadNoise\(\)](#), [NoiseFile::loadNoise\(\)](#), [Noise::loadNoise\(\)](#), [Noise::Noise\(\)](#), [NoiseFile::NoiseFile\(\)](#), [NoiseFilter::NoiseFilter\(\)](#), [NoiseFShape::NoiseFShape\(\)](#), [NoiseOof\(\)](#), [NoiseTwoFilter::NoiseTwoFilter\(\)](#), and [NoiseWhite::NoiseWhite\(\)](#).

10.32.4.7 Filter NoiseOof::NFilter [protected]

[Filter](#) for the white noise used to generate the final noise.

Definition at line 57 of file LISACODE-NoiseOof.h.

Referenced by [generNoise\(\)](#), [getFilterAlpha\(\)](#), [getFilterBeta\(\)](#), [loadNoise\(\)](#), and [NoiseOof\(\)](#).

10.32.4.8 vector<double> Noise::NoiseData [protected, inherited]

Vector of noise data.

Definition at line 63 of file LISACODE-Noise.h.

Referenced by [Noise::addNoise\(\)](#), [NoiseWhite::generNoise\(\)](#), [NoiseTwoFilter::generNoise\(\)](#), [generNoise\(\)](#), [NoiseFShape::generNoise\(\)](#), [NoiseFilter::generNoise\(\)](#), [NoiseFile::generNoise\(\)](#), [Noise::generNoise\(\)](#), [Noise::getNoise\(\)](#), [NoiseWhite::loadNoise\(\)](#), [NoiseTwoFilter::loadNoise\(\)](#), [loadNoise\(\)](#), [NoiseFShape::loadNoise\(\)](#), [NoiseFilter::loadNoise\(\)](#), [NoiseFile::loadNoise\(\)](#), and [Noise::loadNoise\(\)](#).

10.32.4.9 char Noise::NoiseType[30] [protected, inherited]

String to describe the noise type.

Definition at line 65 of file LISACODE-Noise.h.

Referenced by [Noise::Noise\(\)](#), [NoiseFile::NoiseFile\(\)](#), [NoiseFilter::NoiseFilter\(\)](#), [NoiseFShape::NoiseFShape\(\)](#), [NoiseOof\(\)](#), [NoiseTwoFilter::NoiseTwoFilter\(\)](#), [NoiseWhite::NoiseWhite\(\)](#), and [Noise::TestType\(\)](#).

10.32.4.10 double Noise::tDurAdd [protected, inherited]

Noise computation time step (for each measurement).

Definition at line 53 of file LISACODE-Noise.h.

Referenced by [Noise::gettDurAdd\(\)](#), [Noise::Noise\(\)](#), [NoiseFile::NoiseFile\(\)](#), [NoiseFilter::NoiseFilter\(\)](#), [NoiseFShape::NoiseFShape\(\)](#), [NoiseOof\(\)](#), [NoiseTwoFilter::NoiseTwoFilter\(\)](#), [NoiseWhite::NoiseWhite\(\)](#), and [Noise::settDurAdd\(\)](#).

10.32.4.11 double Noise::tFirst [protected, inherited]

Time of the first data in data vector.

Definition at line 55 of file LISACODE-Noise.h.

Referenced by Noise::getNoise(), Noise::gettFirst(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and Noise::settFirst().

10.32.4.12 double **Noise::tLast** [protected, inherited]

Time of the last data in data vector.

Definition at line 57 of file LISACODE-Noise.h.

Referenced by Noise::getLast(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and Noise::settLast().

10.32.4.13 double **Noise::tStep** [protected, inherited]

Time step in seconds between two saved data. It is used to simulate the continuous signal.

Definition at line 51 of file LISACODE-Noise.h.

Referenced by NoiseTwoFilter::generNoise(), generNoise(), NoiseFShape::generNoise(), NoiseFilter::generNoise(), Noise::getNoise(), NoiseWhite::getPSD(), NoiseWhite::getSqPSD(), Noise::gettStep(), NoiseTwoFilter::loadNoise(), loadNoise(), NoiseFShape::loadNoise(), NoiseFilter::loadNoise(), NoiseFile::loadNoise(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), NoiseWhite::setSqPSD(), Noise::settDurAdd(), Noise::settFirst(), Noise::settLast(), and Noise::settStep().

10.32.4.14 vector<double> **NoiseOof::WhiteData** [protected]

Vector of raw data before filtering.

White noise data are generated with a standard gaussian.

Definition at line 51 of file LISACODE-NoiseOof.h.

Referenced by generNoise(), and loadNoise().

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

- [LISACODE-NoiseOof.h](#)
- [LISACODE-NoiseOof.cpp](#)

10.33 NoiseSpec Struct Reference

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

10.33.1 Detailed Description

`Noise` specification structure.

Definition at line 68 of file LISACODE-ConfigSim.h.

Public Attributes

- int `NType`
Type of `Noise`.
- double `NVal0`
Noise level (used if `NType`=4,5 or 6).
- double `NVal01`
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]
Filenname (used if `NType`=2).

10.33.2 Member Data Documentation

10.33.2.1 char `NoiseSpec::NStr`[256]

Filenname (used if `NType`=2).

Definition at line 89 of file LISACODE-ConfigSim.h.

Referenced by ConfigSim::NoisesCreation(), and ConfigSim::ReadASCIIFile().

10.33.2.2 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 80 of file LISACODE-ConfigSim.h.

Referenced by ConfigSim::NoisesCreation(), ConfigSim::ReadASCIIFile(), and ConfigSim::ReadXMLFile().

10.33.2.3 double NoiseSpec::NVal0

Noise level (used if **NType**=4,5 or 6).

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

Referenced by ConfigSim::NoisesCreation(), ConfigSim::ReadASCIIFile(), and ConfigSim::ReadXMLFile().

10.33.2.4 double NoiseSpec::NVal01

Noise level (used if **NType**=4,5 or 6).

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

Referenced by ConfigSim::NoisesCreation(), ConfigSim::ReadASCIIFile(), and ConfigSim::ReadXMLFile().

10.33.2.5 vector<double> NoiseSpec::NVal1

α recursive coefficients (used if **NType**=3)

Definition at line 85 of file LISACODE-ConfigSim.h.

Referenced by ConfigSim::NoisesCreation(), ConfigSim::ReadASCIIFile(), and ConfigSim::ReadXMLFile().

10.33.2.6 vector<double> NoiseSpec::NVal2

β recursive coefficients (used if **NType**=3)

Definition at line 87 of file LISACODE-ConfigSim.h.

Referenced by ConfigSim::NoisesCreation(), ConfigSim::ReadASCIIFile(), and ConfigSim::ReadXMLFile().

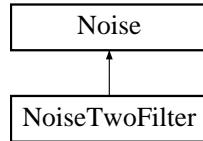
The documentation for this struct was generated from the following file:

- [LISACODE-ConfigSim.h](#)

10.34 NoiseTwoFilter Class Reference

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

Inheritance diagram for NoiseTwoFilter::



10.34.1 Detailed Description

[Noise](#) derived class to treat noise filters.

It creates a noise signal from a filtered 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 46 of file LISACODE-NoiseTwoFilter.h.

Public Member Functions

- [**NoiseTwoFilter \(\)**](#)
Base constructor.
- [**NoiseTwoFilter \(double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, Filter NFilter_n, Filter NFilter_n2\)**](#)
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.
- double `getNoise` (double `tDelay`, int `order`) const

It returns the noise value for the specified delay.

Protected Attributes

- `vector< double > WhiteData`

Vector of raw data before filtering.
- `vector< double > WhiteData2`
- `vector< double > NoiseData_tmp1`
- `vector< double > NoiseData_tmp2`
- `Filter NFilter`

Filter for the white noise used to generate the final noise.
- `Filter NFilter_2`
- 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.

10.34.2 Constructor & Destructor Documentation

10.34.2.1 NoiseTwoFilter::NoiseTwoFilter ()

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-NoiseTwoFilter.cpp.

References [Filter::init\(\)](#), [loadNoise\(\)](#), [Noise::NbData](#), [NFilter](#), [NFilter_2](#), [Noise::NoiseType](#), [PRECISION](#), [Noise::tDurAdd](#), [Noise::tFirst](#), [Noise::tLast](#), and [Noise::tStep](#).

10.34.2.2 NoiseTwoFilter::NoiseTwoFilter (double *tStep_n*, double *tDurAdd_n*, double *tFirst_n*, double *tLast_n*, Filter *NFilter_n*, Filter *NFilter_n2*)

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](#).
- NFilter_n2* [Filter](#) for [NFilter](#).

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

References [Filter::Copy](#)(), [loadNoise](#)(), [Noise::NbData](#), [NFilter](#), [NFilter_2](#), [Noise::NoiseType](#), PRECISION, [Noise::settDurAdd](#)(), [Noise::settFirst](#)(), [Noise::settLast](#)(), [Noise::settStep](#)(), [Noise::tFirst](#), [Noise::tLast](#), and [Noise::tStep](#).

10.34.3 Member Function Documentation

10.34.3.1 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 207 of file LISACODE-Noise.cpp.

References [Noise::generNoise](#)(), [Noise::NbBinAdd](#), [Noise::NbData](#), and [Noise::NoiseData](#).

Referenced by [GWSto::hc](#)(), and [GWSto::hp](#)().

10.34.3.2 void NoiseTwoFilter::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 :

$$\text{for } i=0, \dots, \text{Startbin } \text{WhiteData}[i] = \sqrt{-\frac{\log(r1)}{tStep} \cdot \cos(2 \cdot \pi \cdot r1)}$$

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 176 of file LISACODE-NoiseTwoFilter.cpp.

References [Filter::App](#)(), [genunf](#)(), [Noise::NbBinAdd](#), [Noise::NbData](#), [NFilter](#), [NFilter_2](#), [Noise::NoiseData](#), [NoiseData_tmp1](#), [NoiseData_tmp2](#), [Noise::tStep](#), [WhiteData](#), and [WhiteData2](#).

10.34.3.3 vector< vector <double> > NoiseTwoFilter::getFilterAlpha () [inline]

It returns the list of the alpha parameters (see [Filter::alpha](#)) of the filter [NFilter](#).

Definition at line 74 of file LISACODE-NoiseTwoFilter.h.

References [Filter::getAlpha\(\)](#), and [NFilter](#).

10.34.3.4 vector< vector <double> > NoiseTwoFilter::getFilterBeta () [inline]

It returns the list of the beta parameters (see [Filter::beta](#)) of the filter [NFilter](#).

Definition at line 76 of file LISACODE-NoiseTwoFilter.h.

References [Filter::getBeta\(\)](#), and [NFilter](#).

10.34.3.5 int Noise::getNbBinAdd () const [inline, inherited]

It returns the number of bins added, that is the value of [NbBinAdd](#) attribute.

Definition at line 98 of file LISACODE-Noise.h.

References [Noise::NbBinAdd](#).

10.34.3.6 double Noise::getNoise (double tDelay, int order) 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 299 of file LISACODE-Noise.cpp.

References [Noise::NoiseData](#), [PRECISION](#), [Noise::tFirst](#), and [Noise::tStep](#).

10.34.3.7 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 226 of file LISACODE-Noise.cpp.

References `Noise::NoiseData`, `PRECISION`, `Noise::tFirst`, and `Noise::tStep`.

Referenced by `GWSto::hc()`, and `GWSto::hp()`.

10.34.3.8 double Noise::gettDurAdd () const [inline, inherited]

It returns the duration for a noise addition, that is the value of `tDurAdd` attribute.

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

References `Noise::tDurAdd`.

10.34.3.9 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 91 of file LISACODE-Noise.h.

References `Noise::tFirst`.

10.34.3.10 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 95 of file LISACODE-Noise.h.

References `Noise::tLast`.

10.34.3.11 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 82 of file LISACODE-Noise.h.

References `Noise::tStep`.

10.34.3.12 void NoiseTwoFilter::loadNoise () [virtual]

Initialization.

Number of bins must be adjusted depending on filter stabilization :

$$NbDataStab = 2 \cdot NbData + NbDataStab_{NFilter} + \text{Max}(\text{size}(\alpha_{NFilter}), \text{size}(\beta_{NFilter}))$$

[WhiteData](#) size and [NoiseData](#) size are set to [NbDataStab](#).

[WhiteData](#) is generated as a standard gaussian :

$$\text{for i=0, ..., Startbin } WhiteData[i] = \sqrt{-\frac{\log(r1)}{tStep} \cdot \cos(2 \cdot \pi \cdot r1)}$$

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 104 of file LISACODE-NoiseTwoFilter.cpp.

References [Filter::App\(\)](#), [genunf\(\)](#), [Filter::getDepth\(\)](#), [Filter::getNbDataStab\(\)](#), [MAX](#), [Noise::NbData](#), [NFilter](#), [NFilter_2](#), [Noise::NoiseData](#), [NoiseData_tmp1](#), [NoiseData_tmp2](#), [Noise::tStep](#), [WhiteData](#), and [WhiteData2](#).

Referenced by [NoiseTwoFilter\(\)](#).

10.34.3.13 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 128 of file LISACODE-Noise.cpp.

References [Noise::NbBinAdd](#), [PRECISION](#), [Noise::tDurAdd](#), and [Noise::tStep](#).

Referenced by [Noise::Noise\(\)](#), [NoiseFile::NoiseFile\(\)](#), [NoiseFilter::NoiseFilter\(\)](#), [NoiseFShape::NoiseFShape\(\)](#), [NoiseOof::NoiseOof\(\)](#), [NoiseTwoFilter\(\)](#), and [NoiseWhite::NoiseWhite\(\)](#).

10.34.3.14 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 149 of file LISACODE-Noise.cpp.

References PRECISION, Noise::tFirst, and Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter(), and NoiseWhite::NoiseWhite().

10.34.3.15 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 164 of file LISACODE-Noise.cpp.

References PRECISION, Noise::tLast, and Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter(), and NoiseWhite::NoiseWhite().

10.34.3.16 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 117 of file LISACODE-Noise.cpp.

References Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter(), and NoiseWhite::NoiseWhite().

10.34.3.17 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 106 of file LISACODE-Noise.cpp.

References Noise::NoiseType.

10.34.4 Member Data Documentation

10.34.4.1 int Noise::NbBinAdd [protected, inherited]

Number of bins (for each measurement).

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

Referenced by Noise::addNoise(), generNoise(), NoiseOof::generNoise(), NoiseFShape::generNoise(), NoiseFilter::generNoise(), Noise::getNbBinAdd(), and Noise::settDurAdd().

10.34.4.2 int Noise::NbData [protected, inherited]

Nominal number of data in the noise data vector [NoiseData](#).

Definition at line 61 of file LISACODE-Noise.h.

Referenced by [Noise::addNoise\(\)](#), [generNoise\(\)](#), [NoiseOof::generNoise\(\)](#), [NoiseFShape::generNoise\(\)](#), [NoiseFilter::generNoise\(\)](#), [NoiseWhite::loadNoise\(\)](#), [loadNoise\(\)](#), [NoiseOof::loadNoise\(\)](#), [NoiseFShape::loadNoise\(\)](#), [NoiseFilter::loadNoise\(\)](#), [NoiseFile::loadNoise\(\)](#), [Noise::loadNoise\(\)](#), [Noise::Noise\(\)](#), [NoiseFile::NoiseFile\(\)](#), [NoiseFilter::NoiseFilter\(\)](#), [NoiseFShape::NoiseFShape\(\)](#), [NoiseOof::NoiseOof\(\)](#), [NoiseTwoFilter\(\)](#), and [NoiseWhite::NoiseWhite\(\)](#).

10.34.4.3 Filter NoiseTwoFilter::NFilter [protected]

[Filter](#) for the white noise used to generate the final noise.

Definition at line 57 of file LISACODE-NoiseTwoFilter.h.

Referenced by [generNoise\(\)](#), [getFilterAlpha\(\)](#), [getFilterBeta\(\)](#), [loadNoise\(\)](#), and [NoiseTwoFilter\(\)](#).

10.34.4.4 Filter NoiseTwoFilter::NFilter_2 [protected]

Definition at line 58 of file LISACODE-NoiseTwoFilter.h.

Referenced by [generNoise\(\)](#), [loadNoise\(\)](#), and [NoiseTwoFilter\(\)](#).

10.34.4.5 vector<double> Noise::NoiseData [protected, inherited]

Vector of noise data.

Definition at line 63 of file LISACODE-Noise.h.

Referenced by [Noise::addNoise\(\)](#), [NoiseWhite::generNoise\(\)](#), [generNoise\(\)](#), [NoiseOof::generNoise\(\)](#), [NoiseFShape::generNoise\(\)](#), [NoiseFilter::generNoise\(\)](#), [NoiseFile::generNoise\(\)](#), [Noise::generNoise\(\)](#), [Noise::getNoise\(\)](#), [NoiseWhite::loadNoise\(\)](#), [loadNoise\(\)](#), [NoiseOof::loadNoise\(\)](#), [NoiseFShape::loadNoise\(\)](#), [NoiseFilter::loadNoise\(\)](#), [NoiseFile::loadNoise\(\)](#), and [Noise::loadNoise\(\)](#).

10.34.4.6 vector<double> NoiseTwoFilter::NoiseData_tmp1 [protected]

Definition at line 54 of file LISACODE-NoiseTwoFilter.h.

Referenced by [generNoise\(\)](#), and [loadNoise\(\)](#).

10.34.4.7 vector<double> NoiseTwoFilter::NoiseData_tmp2 [protected]

Definition at line 55 of file LISACODE-NoiseTwoFilter.h.

Referenced by [generNoise\(\)](#), and [loadNoise\(\)](#).

10.34.4.8 char Noise::NoiseType[30] [protected, inherited]

String to describe the noise type.

Definition at line 65 of file LISACODE-Noise.h.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and Noise::TestType().

10.34.4.9 double Noise::tDurAdd [protected, inherited]

Noise computation time step (for each measurement).

Definition at line 53 of file LISACODE-Noise.h.

Referenced by Noise::gettDurAdd(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and Noise::settDurAdd().

10.34.4.10 double Noise::tFirst [protected, inherited]

Time of the first data in data vector.

Definition at line 55 of file LISACODE-Noise.h.

Referenced by Noise::getNoise(), Noise::gettFirst(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and Noise::settFirst().

10.34.4.11 double Noise::tLast [protected, inherited]

Time of the last data in data vector.

Definition at line 57 of file LISACODE-Noise.h.

Referenced by Noise::gettLast(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter(), NoiseWhite::NoiseWhite(), and Noise::settLast().

10.34.4.12 double Noise::tStep [protected, inherited]

Time step in seconds between two saved data. It is used to simulate the continuous signal.

Definition at line 51 of file LISACODE-Noise.h.

Referenced by generNoise(), NoiseOof::generNoise(), NoiseFShape::generNoise(), NoiseFilter::generNoise(), Noise::getNoise(), NoiseWhite::getPSD(), NoiseWhite::getSqPSD(), Noise::gettStep(), loadNoise(), NoiseOof::loadNoise(), NoiseFShape::loadNoise(), NoiseFilter::loadNoise(), NoiseFile::loadNoise(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter(), NoiseWhite::NoiseWhite(), NoiseWhite::setSqPSD(), Noise::settDurAdd(), Noise::settFirst(), Noise::settLast(), and Noise::settStep().

10.34.4.13 vector<double> NoiseTwoFilter::WhiteData [protected]

Vector of raw data before filtering.

White noise data are generated with a standard gaussian.

Definition at line 52 of file LISACODE-NoiseTwoFilter.h.

Referenced by generNoise(), and loadNoise().

10.34.4.14 vector<double> NoiseTwoFilter::WhiteData2 [protected]

Definition at line 53 of file LISACODE-NoiseTwoFilter.h.

Referenced by generNoise(), and loadNoise().

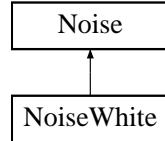
The documentation for this class was generated from the following files:

- [LISACODE-NoiseTwoFilter.h](#)
- [LISACODE-NoiseTwoFilter.cpp](#)

10.35 NoiseWhite Class Reference

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

Inheritance diagram for NoiseWhite::



10.35.1 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.
- double `getNoise` (double `tDelay`, int `order`) 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.

10.35.2 Constructor & Destructor Documentation

10.35.2.1 [NoiseWhite::NoiseWhite \(\)](#)

Base constructor.

It sets default values for class attributes and generate white noise vector (see [loadNoise](#)).

$\Sigma(\sigma)$ 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](#).

10.35.2.2 [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\(\)](#)). $\Sigma(\sigma)$ is computed from a by default PSD value.

[NoiseType](#) attribute is set to `White`.

$\Sigma(\sigma)$ 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`.

10.35.2.3 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`.

10.35.3 Member Function Documentation

10.35.3.1 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 207 of file LISACODE-Noise.cpp.

References `Noise::generNoise()`, `Noise::NbBinAdd`, `Noise::NbData`, and `Noise::NoiseData`.

Referenced by `GWSto::hc()`, and `GWSto::hp()`.

10.35.3.2 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 :

$$\text{for } i=0, \dots, \text{StartBin} \quad \text{WhiteData}[i] = \sigma \cdot \sqrt{-2 \cdot \log(r2)} \cdot \cos(2 \cdot \pi \cdot r1)$$

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.

10.35.3.3 int Noise::getNbBinAdd () const [inline, inherited]

It returns the number of bins added, that is the value of NbBinAdd attribute.

Definition at line 98 of file LISACODE-Noise.h.

References Noise::NbBinAdd.

10.35.3.4 double Noise::getNoise (double tDelay, int order) 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 299 of file LISACODE-Noise.cpp.

References Noise::NoiseData, PRECISION, Noise::tFirst, and Noise::tStep.

10.35.3.5 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 226 of file LISACODE-Noise.cpp.

References `Noise::NoiseData`, `PRECISION`, `Noise::tFirst`, and `Noise::tStep`.

Referenced by `GWSto::hc()`, and `GWSto::hp()`.

10.35.3.6 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`.

10.35.3.7 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`.

10.35.3.8 double Noise::gettDurAdd () const [inline, inherited]

It returns the duration for a noise addition, that is the value of `tDurAdd` attribute.

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

References `Noise::tDurAdd`.

10.35.3.9 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 91 of file LISACODE-Noise.h.

References Noise::tFirst.

10.35.3.10 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 95 of file LISACODE-Noise.h.

References Noise::tLast.

10.35.3.11 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 82 of file LISACODE-Noise.h.

References Noise::tStep.

10.35.3.12 void NoiseWhite::loadNoise () [virtual]

Initialization.

[NoiseData](#) size is set to [NbData](#).

[NoiseData](#) is generated as below :

$$\text{for } i=0, \dots, \text{NbData} \quad \text{WhiteData}[i] = \sigma \cdot \sqrt{-2 \cdot \log(r2)} \cdot \cos(2 \cdot \pi \cdot r1)$$

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\(\)](#).

10.35.3.13 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:

$$\sigma = \frac{\text{SqPSD}}{\sqrt{2.0 \cdot tStep}}$$

where $tStep$ 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\(\)](#).

10.35.3.14 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 128 of file LISACODE-Noise.cpp.

References `Noise::NbBinAdd`, `PRECISION`, `Noise::tDurAdd`, and `Noise::tStep`.

Referenced by `Noise::Noise()`, `NoiseFile::NoiseFile()`, `NoiseFilter::NoiseFilter()`, `NoiseFShape::NoiseFShape()`, `NoiseOof::NoiseOof()`, `NoiseTwoFilter::NoiseTwoFilter()`, and `NoiseWhite()`.

10.35.3.15 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 149 of file LISACODE-Noise.cpp.

References `PRECISION`, `Noise::tFirst`, and `Noise::tStep`.

Referenced by `Noise::Noise()`, `NoiseFile::NoiseFile()`, `NoiseFilter::NoiseFilter()`, `NoiseFShape::NoiseFShape()`, `NoiseOof::NoiseOof()`, `NoiseTwoFilter::NoiseTwoFilter()`, and `NoiseWhite()`.

10.35.3.16 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 164 of file LISACODE-Noise.cpp.

References `PRECISION`, `Noise::tLast`, and `Noise::tStep`.

Referenced by `Noise::Noise()`, `NoiseFile::NoiseFile()`, `NoiseFilter::NoiseFilter()`, `NoiseFShape::NoiseFShape()`, `NoiseOof::NoiseOof()`, `NoiseTwoFilter::NoiseTwoFilter()`, and `NoiseWhite()`.

10.35.3.17 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 117 of file LISACODE-Noise.cpp.

References Noise::tStep.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), and NoiseWhite().

10.35.3.18 bool Noise::TestType (char * *SubmitType*) [inherited]

It verifies the type of the noise ([NoiseType](#)).

Parameters:

SubmitType String containing the expected noise type.

Returns:

It returns `true` if the Noise object has the expected type. `false` otherwise.

Definition at line 106 of file LISACODE-Noise.cpp.

References Noise::NoiseType.

10.35.4 Member Data Documentation

10.35.4.1 int Noise::NbBinAdd [protected, inherited]

Number of bins (for each measurement).

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

Referenced by Noise::addNoise(), NoiseTwoFilter::generNoise(), NoiseOof::generNoise(), NoiseFShape::generNoise(), NoiseFilter::generNoise(), Noise::getNbBinAdd(), and Noise::settDurAdd().

10.35.4.2 int Noise::NbData [protected, inherited]

Nominal number of data in the noise data vector [NoiseData](#).

Definition at line 61 of file LISACODE-Noise.h.

Referenced by Noise::addNoise(), NoiseTwoFilter::generNoise(), NoiseOof::generNoise(), NoiseFShape::generNoise(), NoiseFilter::generNoise(), loadNoise(), NoiseTwoFilter::loadNoise(), NoiseOof::loadNoise(), NoiseFShape::loadNoise(), NoiseFilter::loadNoise(), NoiseFile::loadNoise(), Noise::loadNoise(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), and NoiseWhite().

10.35.4.3 vector<double> Noise::NoiseData [protected, inherited]

Vector of noise data.

Definition at line 63 of file LISACODE-Noise.h.

Referenced by Noise::addNoise(), generNoise(), NoiseTwoFilter::generNoise(), NoiseOof::generNoise(), NoiseFShape::generNoise(), NoiseFilter::generNoise(), NoiseFile::generNoise(), Noise::generNoise(), Noise::getNoise(), loadNoise(), NoiseTwoFilter::loadNoise(), NoiseOof::loadNoise(), NoiseFShape::loadNoise(), NoiseFilter::loadNoise(), NoiseFile::loadNoise(), and Noise::loadNoise().

10.35.4.4 char Noise::NoiseType[30] [protected, inherited]

String to describe the noise type.

Definition at line 65 of file LISACODE-Noise.h.

Referenced by Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite(), and Noise::TestType().

10.35.4.5 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().

10.35.4.6 double Noise::tDurAdd [protected, inherited]

Noise computation time step (for each measurement).

Definition at line 53 of file LISACODE-Noise.h.

Referenced by Noise::gettDurAdd(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite(), and Noise::settDurAdd().

10.35.4.7 double Noise::tFirst [protected, inherited]

Time of the first data in data vector.

Definition at line 55 of file LISACODE-Noise.h.

Referenced by Noise::getNoise(), Noise::gettFirst(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite(), and Noise::settFirst().

10.35.4.8 double Noise::tLast [protected, inherited]

Time of the last data in data vector.

Definition at line 57 of file LISACODE-Noise.h.

Referenced by Noise::gettLast(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite(), and Noise::settLast().

10.35.4.9 double Noise::tStep [protected, inherited]

Time step in seconds between two saved data. It is used to simulate the continuous signal.

Definition at line 51 of file LISACODE-Noise.h.

Referenced by NoiseTwoFilter::generNoise(), NoiseOof::generNoise(), NoiseFShape::generNoise(), NoiseFilter::generNoise(), Noise::getNoise(), getPSD(), getSqPSD(), Noise::gettStep(), NoiseTwo-

Filter::loadNoise(), NoiseOof::loadNoise(), NoiseFShape::loadNoise(), NoiseFilter::loadNoise(), NoiseFile::loadNoise(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), NoiseTwoFilter::NoiseTwoFilter(), 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](#)

10.36 PhoDetPhaMet Class Reference

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

10.36.1 Detailed Description

Phasemeter photodiode class.

All noises are phase differences.

Noises are combined (addition, subtraction, 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 Gravitational 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 * SCPPos`
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.

10.36.2 Constructor & Destructor Documentation

10.36.2.1 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.

10.36.2.2 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().

10.36.2.3 **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().

10.36.2.4 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()`.

10.36.2.5 PhoDetPhaMet::~PhoDetPhaMet ()

Destructor.

Definition at line 183 of file LISACODE-PhoDetPhaMet.cpp.

10.36.3 Member Function Documentation

10.36.3.1 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 309 of file LISACODE-PhoDetPhaMet.cpp.

References [FilterON](#), [FilterPhyData](#), [IndirectDir](#), [InterfPhyData](#), [InterfType](#), and [iSC](#).

10.36.3.2 double PhoDetPhaMet::getIndirectDir () const [inline]

Returns [IndirectDir](#) attribute.

Definition at line 178 of file LISACODE-PhoDetPhaMet.h.

References [IndirectDir](#).

10.36.3.3 double PhoDetPhaMet::getiSC () const [inline]

Returns [iSC](#) attribute.

Definition at line 180 of file LISACODE-PhoDetPhaMet.h.

References [iSC](#).

10.36.3.4 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 636 of file LISACODE-PhoDetPhaMet.cpp.

Referenced by [init\(\)](#).

10.36.3.5 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 323 of file LISACODE-PhoDetPhaMet.cpp.

References [FilterON](#), [Filter::getNbDataStab\(\)](#), [PBFilter](#), and [tStepPhy](#).

10.36.3.6 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 404 of file LISACODE-PhoDetPhaMet.cpp.

References [Background::deltanu\(\)](#), and [GWB](#).

Referenced by [ReceiveSignal\(\)](#).

10.36.3.7 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 :

$$\text{indexNPs} = \begin{cases} 3 \cdot \text{IndirectDir} + \text{iSC} - 1 & \text{if OrigN=LA} \\ 3 \cdot \text{IndirectDir} + \text{iSC} - 1 + 6 & \text{if OrigN=OB} \\ 3 \cdot \text{IndirectDir} + \text{iSC} - 1 + 12 & \text{if OrigN=IM} \\ 3 \cdot \text{IndirectDir} + \text{iSC} - 1 + 18 & \text{if OrigN=OP} \\ 24 & \text{else} \end{cases}$$

Then, noise corresponding to that index and tDelay time is given by [Noise::getNoise](#) method and its value is returned.

Definition at line 354 of file LISACODE-PhoDetPhaMet.cpp.

References IM, LA, OB, and OP.

Referenced by [ReceiveSignal\(\)](#).

10.36.3.8 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

$$\text{InterfPhyData} = \text{NbDataStored} \text{ null elements, with } \text{NbDataAdd} = 2 \cdot (\text{int})\left(\frac{\text{tStepMes}}{\text{tStepPhy}} + \frac{1}{2}\right)$$

```
if (FilterON) { PBFILTER = new Filter(1/tStepPhy, FilterParam[0], FilterParam[1], FilterParam[2]/tStepMes, FilterParam[3]); FilterPhyData = NbDataStored null elements }
```

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\(\)](#).

10.36.3.9 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 581 of file LISACODE-PhoDetPhaMet.cpp.

References Filter::App(), FilterON, FilterPhyData, IndirectDir, InterfPhyData, InterfType, NbDataAdd, NbDataStored, PBFilter, Memory::ReceiveData(), ReceiveSignal(), RecordData, S, and TAU.

10.36.3.10 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.

Phasemeter S : Between distant and local laser beam :

$$\begin{aligned}s_1 &= s_1^{GW} + d_1^{OPN} + D_3 p'_2 - p_1 - 2 * d_1^{IM} \dots (+D_3 d_2^{OB} + d_1^{OB}) \\s'_1 &= s'_1^{GW} + d'_1^{OPN} + D'_2 p_3 - p'_1 + 2 * d'_1^{IM} \dots (-D_2 d_3^{OB} - d'_1^{OB})\end{aligned}$$

Phasemeter S : Between distant and local laser beam :

$$\begin{aligned}\tau_1 &= p'_1 - p_1 - 2 * d_1^{IM} \dots (+2d_1^{OB}) \\ \tau'_1 &= p_1 - p'_1 + 2 * d_1^{IM} \dots (-2d_1^{OB})\end{aligned}$$

Computations :

- if (InterfType=S)


```
GWSignal = sGW - > deltanu(iSC, modulo(iSC + 1 + IndirectDir, 3), 2, t)
      +gGWB(iSC, IndirectDir, t)
      for i = 0, ..., NbDataAdd - 1,
      { if (NoNoise)  InterfPhyData[i] = GWSignal
        else          InterfPhyData[i] = GWSignal + noise_i
      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 · 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 · tStepPhy

• if (InterfType=TAU)
    for i = 0, . . . , NbDataAdd - 1
        InterfPhyData[i] = gN(LA, iSC, 1, tDPhy_i)
        -2 · gN(IM, iSC, 1 - IndirectDir, tDPhy_i)
        +2 · 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 460 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\(\)](#).

10.36.4 Member Data Documentation

10.36.4.1 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\(\)](#).

10.36.4.2 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\(\)](#).

10.36.4.3 vector<double> [PhoDetPhaMet::FilterPhyData](#) [protected]

Filtered physical data vector.

Definition at line 105 of file LISACODE-PhoDetPhaMet.h.

Referenced by [DisplayStoredData\(\)](#), [init\(\)](#), and [IntegrateSignal\(\)](#).

10.36.4.4 Background* PhoDetPhaMet::GWB [protected]

Background pointer : confusion whites dwarfs background.

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

Referenced by gGWB(), and init().

10.36.4.5 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().

10.36.4.6 vector<double> PhoDetPhaMet::InterfPhyData [protected]

Physical data vector.

Definition at line 103 of file LISACODE-PhoDetPhaMet.h.

Referenced by DisplayStoredData(), init(), IntegrateSignal(), and ReceiveSignal().

10.36.4.7 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().

10.36.4.8 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().

10.36.4.9 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().

10.36.4.10 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().

10.36.4.11 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().

10.36.4.12 vector<Noise *>* PhoDetPhaMet::NPs [protected]

Noise pointers vector.

Definition at line 89 of file LISACODE-PhoDetPhaMet.h.

Referenced by init().

10.36.4.13 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().

10.36.4.14 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().

10.36.4.15 Geometry* PhoDetPhaMet::SCPos [protected]

Pointer to **LISA** geometry.

Definition at line 87 of file LISACODE-PhoDetPhaMet.h.

Referenced by init().

10.36.4.16 TrFctGW* PhoDetPhaMet::sGW [protected]

Pointer to the transfer function.

Definition at line 95 of file LISACODE-PhoDetPhaMet.h.

Referenced by init(), and ReceiveSignal().

10.36.4.17 double PhoDetPhaMet::tStepMes [protected]

Measurement time step.

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

Referenced by init().

10.36.4.18 double PhoDetPhaMet::tStepPhy [protected]

Physical simulation time step.

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

Referenced by gettStab(), init(), and ReceiveSignal().

10.36.4.19 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](#)

10.37 QuadCell Struct Reference

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

10.37.1 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.

10.37.2 Member Data Documentation

10.37.2.1 [QuadCell::a0](#)

Scale factor.

Referenced by AbsRespFunctQuadCell(), FilterQuadCell(), KmQuadCell(), and TransfZQuadCell().

10.37.2.2 [QuadCell::a1](#)

First direct coefficient divided by scale factor.

Referenced by AbsRespFunctQuadCell(), FilterQuadCell(), KmQuadCell(), and TransfZQuadCell().

10.37.2.3 [QuadCell::b1](#)

First recursive coefficient.

Referenced by AbsRespFunctQuadCell(), CalcEllipticFilter4LISACode(), FilterQuadCell(), HmQuadCell(), KmQuadCell(), and TransfZQuadCell().

10.37.2.4 [QuadCell::b2](#)

Second recursive coefficient.

Referenced by AbsRespFunctQuadCell(), FilterQuadCell(), HmQuadCell(), KmQuadCell(), and TransfZQuadCell().

10.37.2.5 [QuadCell::pole](#)

Complex value corresponding to pole.

10.37.2.6 [QuadCell::u](#)

Memory.

Referenced by CalcEllipticFilter(), and FilterQuadCell().

10.37.2.7 [QuadCell::zero](#)

Complex value corresponding to zero (0).

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

- [LISACODE-EllipticFilter.h](#)

10.38 Serie Class Reference

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

10.38.1 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.

10.38.2 Constructor & Destructor Documentation

10.38.2.1 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.

10.38.2.2 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.

10.38.2.3 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.

10.38.2.4 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.

10.38.2.5 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 between second and first read elements
- ys = read elements

Definition at line 84 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

10.38.2.6 Serie::~Serie ()

Destructor.

Definition at line 112 of file LISACODE-Serie.cpp.

References ys.

10.38.3 Member Function Documentation

10.38.3.1 void Serie::addData (double *y*)

Adds data at the begining of the serie.

Definition at line 183 of file LISACODE-Serie.cpp.

References ys.

10.38.3.2 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.

10.38.3.3 void Serie::delLastData ()

Deletes the last data of the serie.

Definition at line 190 of file LISACODE-Serie.cpp.

References ys.

10.38.3.4 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:

$$\begin{cases} \text{TruncVal}(x) & \text{if } \text{modulo}\left(\frac{x-x_0}{dx}, 1\right) \approx 0 \\ \text{TruncVal}(x) & \text{if } \text{InterpolType} = \text{TRU} \\ \text{InterLinear}(x) & \text{if } \text{InterpolType} = \text{LIN} \\ \text{InterCubic}(x) & \text{if } \text{InterpolType} = \text{CUB} \\ \text{InterLagrange}(x, \text{int}(\text{InterpUtilValue})) & \text{if } \text{InterpolType} = \text{LAG} \end{cases}$$

Definition at line 259 of file LISACODE-Serie.cpp.

References CUB, dx, InterCubic(), InterLagrange(), InterLinear(), LAG, LIN, PRECISION, TRU, TruncVal(), and x0.

10.38.3.5 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*.

10.38.3.6 int Serie::getNbVal () const [inline]

Returns number of values (size of *ys* attribute).

Definition at line 67 of file LISACODE-Serie.h.

References *ys*.

10.38.3.7 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:

$x_0 + dx \cdot bin$

Definition at line 145 of file LISACODE-Serie.cpp.

References dx, x0, and *ys*.

10.38.3.8 double Serie::getRefStep () const [inline]

Returns *dx* attribute.

Definition at line 71 of file LISACODE-Serie.h.

References dx.

10.38.3.9 double Serie::InterCubic (double *x*) const

Returns cubic interpolation result.

Indices are computed :

$$bin_1 = \text{floor}\left(\frac{x-x_0}{dx}\right)$$

$$bin_2 = bin_1 + 1$$

$$bin_0 = bin_1 - 1$$

$$bin_3 = bin_2 + 1$$

Indices are checked ; bin_0 must be positive or null, and bin_3 must be lower than `ys` attribute size.

Returns:

$$\begin{aligned} & \mu^3 \cdot (ys[bin_3] - ys[bin_2] - ys[bin_0] + ys[bin_1]) \\ & + \mu^2 \cdot (2 \cdot ys[bin_0] - 2 \cdot ys[bin_1] + ys[bin_2] - ys[bin_3]) \\ & + \mu \cdot (ys[bin_2] - ys[bin_0]) \\ & + ys[bin_3], \end{aligned}$$

where :

$$\mu = \frac{x-x_0}{dx} - \text{floor}\left(\frac{x-x_0}{dx}\right)$$

Definition at line 349 of file LISACODE-Serie.cpp.

References `dx`, `x0`, and `ys`.

Referenced by `gData()`.

10.38.3.10 double Serie::InterHermite (double *x*, double *tension*, double *bias*) const

Returns hermite interpolation result, depending on *x*, tension and bias inputs.

Indices are computed :

$$bin_1 = \text{floor}\left(\frac{x-x_0}{dx}\right)$$

$$bin_2 = bin_1 + 1$$

$$bin_0 = bin_1 - 1$$

$$bin_3 = bin_2 + 1$$

Indices are checked ; bin_0 must be positive or null, and bin_3 must be lower than `ys` attribute size.

Returns:

$$\begin{aligned} & (\mu^3 - 2 \cdot \mu^2 + \mu) \cdot ys[bin_1] + (\mu^3 - 2 \cdot \mu^2 + \mu) \cdot ((ys[bin_1] - ys[bin_0]) \cdot (1 + bias) \cdot \frac{1-tension}{2} \\ & + (ys[bin_2] - ys[bin_1]) \cdot (1 - bias) \cdot \frac{1-tension}{2}), \end{aligned}$$

where :

$$\mu = \frac{x-x_0}{dx} - \text{floor}\left(\frac{x-x_0}{dx}\right)$$

Definition at line 394 of file LISACODE-Serie.cpp.

References `dx`, `x0`, and `ys`.

10.38.3.11 double Serie::InterLagrange (double *x*, int *order*) const

Returns Lagrange interpolation result.

Indices are computed :

$$bin = floor(\frac{x-x_0}{dx})$$

$$kmin = bin - ordermin + 1, \text{ where } ordermin = floor(\frac{order+1}{2})$$

$$kmax = bin + order + 1 - ordermin$$

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:

$$\sum_{k=kmin}^{kmax} ys[k] \cdot P_k, \text{ where } P_k = \prod_{j=kmin, j \neq k}^{kmax} \frac{x-x_0-j \cdot dx}{(k-j) \cdot dx}$$

Definition at line 445 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

Referenced by gData().

10.38.3.12 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().

10.38.3.13 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.

10.38.3.14 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.

10.38.3.15 void Serie::setNbVal (int lenght)

Sets `ys` attribute size to lenght input.

Definition at line 121 of file LISACODE-Serie.cpp.

References ys.

10.38.3.16 void Serie::setRefStart (double start)

Sets `x0` attribute size to start input.

Definition at line 127 of file LISACODE-Serie.cpp.

References x0.

10.38.3.17 void Serie::setRefStep (double delta)

Sets `dx` attribute size to delta input.

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

References dx.

10.38.3.18 double Serie::TruncVal (double x) const

Returns truncated value.

First, bin index is computed :

$$\text{bin} = \text{floor}\left(\frac{x - x_0}{dx}\right)$$

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().

10.38.3.19 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.

10.38.4 Member Data Documentation

10.38.4.1 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().

10.38.4.2 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().

10.38.4.3 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](#)

10.39 SerieC Class Reference

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

10.39.1 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.

10.39.2 Constructor & Destructor Documentation

10.39.2.1 SerieC::SerieC ()

Constructs an instance and initializes it with default values.

- `x0 = 0`

- `dx =1`

Definition at line 511 of file LISACODE-Serie.cpp.

References dx, and x0.

10.39.2.2 SerieC::SerieC (double *start*, double *delta*)

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

- x0 = start
- dx = delta

Definition at line 522 of file LISACODE-Serie.cpp.

References dx, and x0.

10.39.2.3 SerieC::SerieC (double *start*, double *delta*, int *length*)

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

- x0 = start
- dx = delta
- ys = lenght elements, with 0 values

Definition at line 535 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

10.39.2.4 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 553 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

10.39.2.5 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 569 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

10.39.2.6 SerieC::~SerieC ()

Destructor.

Definition at line 601 of file LISACODE-Serie.cpp.

References ys.

10.39.3 Member Function Documentation**10.39.3.1 void SerieC::addDataC (complex< double > y)**

Adds data at the begining of the serie.

Definition at line 672 of file LISACODE-Serie.cpp.

References ys.

10.39.3.2 void SerieC::delLastDataC (double xMax)

Deletes the last data of the serie, while x reference is greater than xMax input.

Definition at line 686 of file LISACODE-Serie.cpp.

References dx, and ys.

10.39.3.3 void SerieC::delLastDataC ()

Deletes the last data of the serie.

Definition at line 679 of file LISACODE-Serie.cpp.

References ys.

10.39.3.4 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 647 of file LISACODE-Serie.cpp.

References ys.

10.39.3.5 int SerieC::getNbValC () const [inline]

Returns number of values (size of [ys](#) attribute).

Definition at line 122 of file LISACODE-Serie.h.

References ys.

10.39.3.6 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:

$$x_0 + dx \cdot bin$$

Definition at line 634 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

10.39.3.7 double SerieC::getRefStepC () const [inline]

Returns [dx](#) attribute.

Definition at line 126 of file LISACODE-Serie.h.

References dx.

10.39.3.8 void SerieC::rfileC (char **fname*)

Writes the serie as 3 columns (X,Y.real,Y.imag) from fname input file.

Definition at line 714 of file LISACODE-Serie.cpp.

References ys.

10.39.3.9 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 660 of file LISACODE-Serie.cpp.

References ys.

10.39.3.10 void SerieC::setNbValC (int *length*)

Sets [ys](#) attribute size to lenght input.

Definition at line 610 of file LISACODE-Serie.cpp.

References ys.

10.39.3.11 void SerieC::setRefStartC (double *start*)

Sets [x0](#) attribute size to start input.

Definition at line 616 of file LISACODE-Serie.cpp.

References x0.

10.39.3.12 void SerieC::setRefStepC (double *delta*)

Sets `dx` attribute size to delta input.

Definition at line 622 of file LISACODE-Serie.cpp.

References `dx`.

10.39.3.13 void SerieC::wfileC (char **fname*)

Writes the serie as 3 columns (X,Y.real,Y.imag) into fname input file.

Definition at line 697 of file LISACODE-Serie.cpp.

References `dx`, `x0`, and `ys`.

10.39.4 Member Data Documentation

10.39.4.1 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()`.

10.39.4.2 double SerieC::`x0` [protected]

Reference serie start value.

Definition at line 105 of file LISACODE-Serie.h.

Referenced by `getRefC()`, `SerieC()`, `setRefStartC()`, and `wfileC()`.

10.39.4.3 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](#)

10.40 TDI Class Reference

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

10.40.1 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 OutFileEncoding_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 OutFileEncoding_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 OutFileEncoding_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.
- **TDI (Memory *TDelay_n, TDI_InterData *Eta_n, ofstream *OutFile_n, int OutFileEncoding_n, int iSerie_n, vector< int > SignEtaDelays, vector< double > Fact_n, TDITools *TDIQuickMod_n)**
Constructs an instance and initializes it using TDelay_n, Eta_n, OutFile_n, iSerie_n, SignEtaDelays , Fact_n and TDIQuickMod_n inputs.
- **TDI (Memory *TDelay_n, Memory *SCSig_n, vector< int > SignEtaDelays, vector< double > Fact_n, TDITools *TDIQuickMod_n)**
Constructor for directly use spacecraft signal (no use of Eta) and no records in out file.
- virtual **~TDI ()**
Destructor.
- **int getCountInterDelay ()**
Returns tmpCountInterDelay attribute.
- **int getCountInterEta ()**
Returns tmpCountInterEta attribute.
- **void ReadSignEtaDelays (vector< int > SignEtaDelays)**
Read list of packs that are in the following form : -1231 for - D1 D2 D3 Eta1.

- double **Compute** (double tComputeDelay)
Compute the result of generator in case where Eta is used.
- double **ComputeNoEta** (double tComputeDelay)
Compute the result of generator in case where spacecraft signal are used.
- double **RecordAndReturnResult** (double tComputeDelay)
Record the result of generator and return the result.
- int **NbDelayMax** ()
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.
- Memory * **SCSig**
Memory where the 6 signals of spacecraft are stored .
- ofstream * **OutFile**
File where TDI result are recorded.
- int **OutFileEncoding**
Encoding of 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]).
- vector< double > **Fact**
List of factor of the pack (double).
- TDITools * **TDIQuickMod**
TDI access tools.

- int tmpCountInterDelay
Temporary data (unused).

- int tmpCountInterEta
Temporary data (unused).

10.40.2 Constructor & Destructor Documentation

10.40.2.1 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
- TDQuickMod = empty

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

References Eta, Fact, IndexDelay, IndexEta, iSerie, OutFile, OutFileEncoding, SCSig, Sign, TDelay, and TDQuickMod.

10.40.2.2 TDI::TDI (**Memory** * TDelay_n, **TDI_InterData** * Eta_n, **ofstream** * OutFile_n, **int** OutFileEncoding_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
- TDQuickMod = empty

Definition at line 61 of file LISACODE-TDI.cpp.

References Eta, Fact, IndexDelay, IndexEta, iSerie, OutFile, OutFileEncoding, SCSig, Sign, TDelay, and TDITools.

10.40.2.3 TDI::TDI ([Memory](#) * TDelay_n, [TDI_InterData](#) * Eta_n, [ofstream](#) * OutFile_n, int OutFileEncoding_n, int iSerie_n, [vector](#)< int > Sign_n, [vector](#)< int > IndexEta_n, [vector](#)< [vector](#)< int > > IndexDelay_n, [TDITools](#) * TDITools_n)

Constructs an instance and initializes it using TDelay_n, Eta_n, OutFile_n, iSerie_n, Sign_n, IndexEta_n, IndexDelay_n and TDITools_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
- [TDITools](#) = TDITools_n
- [tmpCountInterDelay](#) = 0
- [tmpCountInterEta](#) = 0

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

References Eta, Fact, IndexDelay, IndexEta, iSerie, OutFile, OutFileEncoding, SCSig, Sign, TDelay, TDITools, tmpCountInterDelay, and tmpCountInterEta.

10.40.2.4 TDI::TDI ([Memory](#) * TDelay_n, [TDI_InterData](#) * Eta_n, [ofstream](#) * OutFile_n, int OutFileEncoding_n, int iSerie_n, [vector](#)< int > SignEtaDelays, [TDITools](#) * TDITools_n)

Constructs an instance and initializes it using TDelay_n, Eta_n, OutFile_n, iSerie_n, SignEtaDelays and TDITools_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 146 of file LISACODE-TDI.cpp.

References Eta, Fact, IndexDelay, IndexEta, iSerie, OutFile, OutFileEncoding, ReadSignEtaDelays(), SCSig, Sign, TDelay, TDIQuickMod, tmpCountInterDelay, and tmpCountInterEta.

10.40.2.5 TDI::TDI ([Memory](#) * `TDelay_n`, [TDI_InterData](#) * `Eta_n`, [ostream](#) * `OutFile_n`, int `OutFileEncoding_n`, int `iSerie_n`, [vector](#)< int > `SignEtaDelays`, [vector](#)< double > `Fact_n`, [TDITools](#) * `TDIQuickMod_n`)

Constructs an instance and initializes it using `TDelay_n`, `Eta_n`, `OutFile_n`, `iSerie_n`, `SignEtaDelays`, `Fact_n` 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 190 of file LISACODE-TDI.cpp.

References Eta, Fact, IndexDelay, IndexEta, iSerie, OutFile, OutFileEncoding, ReadSignEtaDelays(), SCSig, Sign, TDelay, TDIQuickMod, tmpCountInterDelay, and tmpCountInterEta.

10.40.2.6 TDI::TDI ([Memory](#) * `TDelay_n`, [Memory](#) * `SCSig_n`, [vector](#)< int > `SignEtaDelays`, [vector](#)< double > `Fact_n`, [TDITools](#) * `TDIQuickMod_n`)

Constructor for directly use spacecraft signal (no use of Eta) and no records in out file.

Definition at line 223 of file LISACODE-TDI.cpp.

References Eta, Fact, IndexDelay, IndexEta, OutFile, OutFileEncoding, ReadSignEtaDelays(), SCSig, Sign, TDelay, TDIQuickMod, tmpCountInterDelay, and tmpCountInterEta.

10.40.2.7 TDI::~TDI () [virtual]

Destructor.

Definition at line 247 of file LISACODE-TDI.cpp.

10.40.3 Member Function Documentation

10.40.3.1 double TDI::Compute (double *tComputeDelay*)

Compute the result of generator in case where Eta is used.

Computes delay using tComputeDelay input and class attributes.

If approximations are done in order to compute delays more quickly

$$\begin{aligned} TotalDelay = & \sum_{iPack=0}^{size(IndexEta)-1} \sum_{iDelay=0}^{size(IndexDelay[iPack])-1} \\ & TDITools::getDelay((IndexDelay[iPack])[iDelay]) \\ \text{else} \quad & TDITools::getDelay((IndexDelay[iPack])[iDelay] - 1, TotalDelay, LAG, 6) \end{aligned}$$

[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 353 of file LISACODE-TDI.cpp.

References Eta, Fact, TDI_InterData::gData(), Memory::gData(), TDITools::getDelay(), TDITools::getRapidOption(), TDI_InterData::getUsable(), IndexDelay, IndexEta, LAG, Sign, TDelay, and TDITools::getQuickMod.

Referenced by RecordAndReturnResult().

10.40.3.2 double TDI::ComputeNoEta (double *tComputeDelay*)

Compute the result of generator in case where spacecraft signal are used.

Definition at line 401 of file LISACODE-TDI.cpp.

References Fact, Memory::gData(), TDITools::getDelay(), IndexDelay, IndexEta, LIN, SCSig, Sign, and TDITools::getQuickMod.

10.40.3.3 int TDI::getCountInterDelay () [inline]

Returns tmpCountInterDelay attribute.

Definition at line 140 of file LISACODE-TDI.h.

References tmpCountInterDelay.

10.40.3.4 int TDI::getCountInterEta () [inline]

Returns tmpCountInterEta attribute.

Definition at line 142 of file LISACODE-TDI.h.

References tmpCountInterEta.

10.40.3.5 int TDI::NbDelayMax ()

Maximum number of delays.

Maximum number of delays is size of [IndexDelay](#) attribute.

Definition at line 457 of file LISACODE-TDI.cpp.

References IndexDelay.

10.40.3.6 void TDI::ReadSignEtaDelays (vector< int > SignEtaDelays)

Read list 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 $iPack = 0, \dots, \text{size}(\text{SignEtaDelays}) - 1$:

$\text{SignEtaDelays}[iPack]$ is checked : $\text{SignEtaDelays}[iPack] \neq 0$

$$\begin{cases} \text{if } (\text{SignEtaDelays}[iPack] > 0) & +1 \text{ is pushed back in Sign attribute} \\ \text{if } (\text{SignEtaDelays}[iPack] < 0) & -1 \text{ is pushed back in Sign attribute} \end{cases}$$

$$\text{TmpInfo} = \text{abs}(\text{SignEtaDelays}[iPack])$$

$$\text{TmpIndexEta} = \text{ceil}(10 \cdot (\frac{\text{TmpInfo}}{10} - \text{floor}(\frac{\text{TmpInfo}}{10}) + 10^{-6}))$$

TmpIndexEta is checked : $1 \leq \text{TmpIndexEta} \leq 6$

TmpIndexEta is pushed back in [IndexEta](#) attribute

- $\text{TmpInfo} = \text{ceil}(\frac{\text{TmpInfo}}{10})$; while $\text{tmpInfo} \neq 0$:

$$\text{TmpIndexDelay} = \text{ceil}(10 \cdot (\frac{\text{TmpInfo}}{10} - \text{floor}(\frac{\text{TmpInfo}}{10}) + 10^{-6}))$$

TmpIndexDelay is checked : $1 \leq \text{TmpIndexDelay} \leq 6$

TmpIndexDelay is pushed back in [IndexDelay](#) attribute

$$\text{TmpInfo} = \text{ceil}(\frac{\text{TmpInfo}}{10})$$

Definition at line 277 of file LISACODE-TDI.cpp.

References Fact, [IndexDelay](#), [IndexEta](#), and [Sign](#).

Referenced by [TDI\(\)](#).

10.40.3.7 double TDI::RecordAndReturnResult (double *tComputeDelay*)

Record the result of generator and return the result.

Computes delay using *tComputeDelay* input, writes result into [OutFile](#) and returns it.

Definition at line 443 of file LISACODE-TDI.cpp.

References Compute(), OutFile, and OutFileEncoding.

10.40.4 Member Data Documentation

10.40.4.1 [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().

10.40.4.2 [vector<double>](#) TDI::Fact [protected]

List of factor of the pack (double).

Definition at line 71 of file LISACODE-TDI.h.

Referenced by Compute(), ComputeNoEta(), ReadSignEtaDelays(), and TDI().

10.40.4.3 [vector<vector<int>>](#) TDI::IndexDelay [protected]

List of list of delay's index for each pack (value=[1,6]).

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

Referenced by Compute(), ComputeNoEta(), NbDelayMax(), ReadSignEtaDelays(), and TDI().

10.40.4.4 [vector<int>](#) TDI::IndexEta [protected]

List of index of signal Eta where the pack work (value=[1,6]).

Definition at line 64 of file LISACODE-TDI.h.

Referenced by Compute(), ComputeNoEta(), ReadSignEtaDelays(), and TDI().

10.40.4.5 int TDI::iSerie [protected]

Index of the serie in memory RecordMem.

Definition at line 60 of file LISACODE-TDI.h.

Referenced by TDI().

10.40.4.6 [ofstream*](#) TDI::OutFile [protected]

File where TDI result are recorded.

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

Referenced by RecordAndReturnResult(), and TDI().

10.40.4.7 int [TDI::OutFileEncoding](#) [protected]

Encoding of file where TDI result are recorded.

Definition at line 58 of file LISACODE-TDI.h.

Referenced by RecordAndReturnResult(), and TDI().

10.40.4.8 [Memory*](#) [TDI::SCSig](#) [protected]

[Memory](#) where the 6 signals of spacecraft are stored .

Definition at line 54 of file LISACODE-TDI.h.

Referenced by ComputeNoEta(), and TDI().

10.40.4.9 [vector<int>](#) [TDI::Sign](#) [protected]

List of sign of the pack (1 for + and -1 for -).

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

Referenced by Compute(), ComputeNoEta(), ReadSignEtaDelays(), and TDI().

10.40.4.10 [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().

10.40.4.11 [TDITools*](#) [TDI::TDIQuickMod](#) [protected]

TDI access tools.

Definition at line 73 of file LISACODE-TDI.h.

Referenced by Compute(), ComputeNoEta(), and TDI().

10.40.4.12 int [TDI::tmpCountInterDelay](#) [protected]

Temporary data (unused).

Definition at line 75 of file LISACODE-TDI.h.

Referenced by getCountInterDelay(), and TDI().

10.40.4.13 int [TDI::tmpCountInterEta](#) [protected]

Temporary data (unused).

Definition at line 77 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](#)

10.41 TDI_InterData Class Reference

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

10.41.1 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 ()**
Desctructor.
- **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 getDelayCompute ()**
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 * > * PDPMem**
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.

10.41.2 Constructor & Destructor Documentation

10.41.2.1 TDI_InterData::TDI_InterData ()

Constructs an instance and initializes it with default values.

Attributes are :

- **TDelay** = NULL
- **PDPMem** = NULL
- **TimeStore** = 30.0
- **tShift** = 0.0
- **Eta** = 6 elements initialized 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, PDPMem, TDelay, TimeStore, tShift, and Usable.

10.41.2.2 TDI_InterData::TDI_InterData (Memory** * **TDelay_n**, **vector<Memory*>** * **PDPMem_n**)**

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

Attributes are :

- **TDelay** = TDelay_n
- **PDPMem** = NULL
- **TimeStore** = 30.0
- **tShift** = 0.0
- **Eta** = 6 elements initialized 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, PDPMem, TDelay, TimeStore, tShift, and Usable.

10.41.2.3 TDI_InterData::TDI_InterData (Memory** * **TDelay_n**, **vector<Memory*>** * **PDPMem_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 PDPMem_n inputs and default values.

Attributes are :

- **TDelay** = TDelay_n
- **PDPMem** = PDPMem_n
- **TimeStore** = TimeStore_n (checked; expected positive or null value)
- **tShift** = tShift_n
- **Eta** = 6 elements initialized with **tShift** and step (from **PDPMem**) 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, PDPMem, TDelay, TimeStore, tShift, and Usable.

10.41.2.4 TDI_InterData::~TDI_InterData ()

Desctrutor.

Definition at line 119 of file LISACODE-TDI_InterData.cpp.

10.41.3 Member Function Documentation

10.41.3.1 void TDI_InterData::ComputeEta ()

Computes Eta using `TimeStore` and `NoNoise` attributes.

for all spacecrafts (index iSC=1,2,3) :

- if there is no noise

$Eta[iSC-1].addData((*PDPMem)[iSC-1] -> gData(0, tShift, InterpType, InterpUtilValue))$

$Eta[iSC+2].addData((*PDPMem)[iSC-1] -> gData(1, tShift, InterpType, InterpUtilValue))$

using `Memory::gData` method

- else the following value is added at the begining of Eta[iSC-1] serie, using `Serie::addData` method

$(*PDPMem)[iSC - 1] -> gData(0, tShift, InterpType, InterpUtilValue)$

$-\frac{1}{2} \cdot (*PDPMem)[mod(iSC+1, 3)] -> gData(2, tShift + TDelay -> gData(mod(iSC+2, 3), tShift),$

$InterpType, InterpUtilValue)$

$-\frac{1}{2} \cdot (*PDPMem)[mod(iSC+1, 3)] -> gData(3, tShift + TDelay -> gData(mod(iSC+2, 3), tShift),$

$InterpType, InterpUtilValue)$

and the following value is added at the begining of Eta[iSC+2] serie, using `Serie::addData` method

$(*PDPMem)[iSC - 1] -> gData(1, tShift, InterpType, InterpUtilValue)$

$+\frac{1}{2} \cdot (*PDPMem)[iSC - 1] -> gData(2, tShift, InterpType, InterpUtilValue)$

$-\frac{1}{2} \cdot (*PDPMem)[iSC - 1] -> gData(3, tShift, InterpType, InterpUtilValue)$

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, PDPMem, TDelay, TimeStore, tShift, and Usable.

Referenced by main().

10.41.3.2 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.

10.41.3.3 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\(\)](#).

10.41.3.4 double TDI_InterData::gettDelayCompute () [inline]

Returns [tShift](#) attribute.

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

References tShift.

10.41.3.5 double TDI_InterData::getTimeStore () [inline]

Returns [TimeStore](#) attribue .

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

References TimeStore.

10.41.3.6 double TDI_InterData::gettStep () [inline]

Returns first [Eta](#) ref step attribue .

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

References Eta.

10.41.3.7 bool TDI_InterData::getUsable () [inline]

Returns Usable attribute.

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

References Usable.

Referenced by TDI::Compute().

10.41.3.8 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.

10.41.4 Member Data Documentation

10.41.4.1 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().

10.41.4.2 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().

10.41.4.3 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().

10.41.4.4 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().

10.41.4.5 vector<Memory *>* TDI_InterData::PDPMem [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().

10.41.4.6 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().

10.41.4.7 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().

10.41.4.8 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().

10.41.4.9 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](#)

10.42 TDITools Class Reference

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

10.42.1 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.

10.42.2 Constructor & Destructor Documentation

10.42.2.1 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`.

10.42.2.2 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`.

10.42.2.3 TDITools::~TDITools () [virtual]

Destructor.

Definition at line 49 of file LISACODE-TDITools.cpp.

10.42.3 Member Function Documentation

10.42.3.1 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()`, and `TDI::ComputeNoEta()`.

10.42.3.2 bool TDITools::getRapidOption () [inline]

Returns `RapidOption` attribute.

Definition at line 58 of file LISACODE-TDITools.h.

References `RapidOption`.

Referenced by `TDI::Compute()`, and `main()`.

10.42.3.3 void TDITools::RefreshDelay (double *tComputeDelay*)

Updates DelayMem attribute.

For each 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().

10.42.4 Member Data Documentation

10.42.4.1 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().

10.42.4.2 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().

10.42.4.3 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](#)

10.43 TrFctGW Class Reference

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

10.43.1 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.

10.43.2 Constructor & Destructor Documentation

10.43.2.1 TrFctGW::TrFctGW ()

Constructs an instance and initializes it with default values.

init method is called

Definition at line 21 of file LISACODE-TrFctGW.cpp.

References init().

10.43.2.2 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 38 of file LISACODE-TrFctGW.cpp.

References init().

10.43.2.3 TrFctGW::~TrFctGW ()

Definition at line 45 of file LISACODE-TrFctGW.cpp.

10.43.3 Member Function Documentation

10.43.3.1 double TrFctGW::deltanu (int *rec*, int *em*, int *order*, double *trec*)

Returns the fluctuation frequency due to [GW](#).

$$tem = trec + LISAGeo->gtdelay(em, rec, order, trec)$$

$$\overrightarrow{rre\vec{c}} = \frac{LISAGeo->gposition(rec, trec)}{C}$$

$$\overrightarrow{rem} = LISAGeo->gposition(em, trec)$$

$$\overrightarrow{dr} = \overrightarrow{rre\vec{c}} - \overrightarrow{rem}$$

$$\overrightarrow{n} = \frac{\overrightarrow{dr}}{\|\overrightarrow{dr}\|}$$

$$\delta_\nu = 0$$

For each source (iGWindex) in [GWSources](#),

$$hpr = (*GWSources)[iGW]->hp(trec - \overrightarrow{k[iGW]} \cdot \overrightarrow{rre\vec{c}})$$

$$hpe = (*GWSources)[iGW]->hp(tem - \overrightarrow{k[iGW]} \cdot \overrightarrow{rem})$$

$$hcr = (*GWSources)[iGW]->hc(trec - \overrightarrow{k[iGW]} \cdot \overrightarrow{rre\vec{c}})$$

$$hce = (*GWSources)[iGW]->hc(tem - \overrightarrow{k[iGW]} \cdot \overrightarrow{rem})$$

$$\delta_\nu = \delta_\nu + \frac{(hpe - hpr) \cdot \frac{(\overrightarrow{u[iGW]} \cdot \overrightarrow{n})^2 - (\overrightarrow{v[iGW]} \cdot \overrightarrow{n})^2}{2} + (hce - hcr) \cdot (\overrightarrow{u[iGW]} \cdot \overrightarrow{n}) \cdot (\overrightarrow{v[iGW]} \cdot \overrightarrow{n})}{1 - \frac{\overrightarrow{k[iGW]}.rrec - \overrightarrow{k[iGW]}.rem}{\|\overrightarrow{dr}\|}}$$

Definition at line 149 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()`.

10.43.3.2 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 $\phi = \lambda$.

$$\begin{aligned} \lambda &= \lambda_{GW} + \pi \\ \beta &= -\beta_{GW} + \pi \\ \psi &= AnglPol_{GW} \\ k &= \begin{pmatrix} \cos(\lambda) \cdot \cos(\beta) \\ \sin(\lambda) \cdot \cos(\beta) \\ \sin(\beta) \end{pmatrix} \\ u &= \begin{pmatrix} \sin(\beta) \cdot \cos(\lambda) \cdot \cos(\psi) + \sin(\lambda) \cdot \sin(\psi) \\ \sin(\beta) \cdot \sin(\lambda) \cdot \cos(\psi) - \cos(\lambda) \cdot \cos(\psi) \\ -\cos(\beta) \cdot \cos(\psi) \end{pmatrix} \\ v &= \begin{pmatrix} \sin(\beta) \cdot \cos(\lambda) \cdot \sin(\psi) - \sin(\lambda) \cdot \cos(\psi) \\ \sin(\beta) \cdot \sin(\lambda) \cdot \sin(\psi) + \cos(\lambda) \cdot \cos(\psi) \\ -\cos(\beta) \cdot \sin(\psi) \end{pmatrix} \end{aligned}$$

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

References `GWSources`, `k`, `LISAGeo`, `u`, and `v`.

Referenced by `LISA::LISA()`, and `TrFctGW()`.

10.43.4 Member Data Documentation

10.43.4.1 vector<`GW` *>* `TrFctGW::GWSources` [protected]

Gravitational Waves sources.

Definition at line 38 of file LISACODE-TrFctGW.h.

Referenced by `deltanu()`, and `init()`.

10.43.4.2 TrFctGW::k [protected]

(k,u,v) is direct trihedron.

Referenced by deltanu(), and init().

10.43.4.3 Geometry* TrFctGW::LISAGeo [protected]

LISA's geometry.

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

Referenced by deltanu(), and init().

10.43.4.4 TrFctGW::u [protected]

Unit transverse vector.

Referenced by deltanu(), and init().

10.43.4.5 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](#)

10.44 USOClock Class Reference

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

10.44.1 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 there is no noise, else TRUE.

10.44.2 Constructor & Destructor Documentation

10.44.2.1 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().

10.44.2.2 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().

10.44.2.3 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()`.

10.44.2.4 USOClock::~USOClock ()

Destructor.

Definition at line 58 of file LISACODE-USOClock.cpp.

10.44.3 Member Function Documentation

10.44.3.1 double USOClock::getDeriv () [inline]

Returns `DerivLinearCoef` attribute.

Definition at line 69 of file LISACODE-USOClock.h.

References `DerivLinearCoef`.

10.44.3.2 double USOClock::getNoise () [inline]

Returns `SigmaNoise` attribute.

Definition at line 71 of file LISACODE-USOClock.h.

References `SigmaNoise`.

10.44.3.3 double USOClock::getOffset () [inline]

Returns `Offset` attribute.

Definition at line 67 of file LISACODE-USOClock.h.

References `Offset`.

10.44.3.4 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().

10.44.3.5 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](#)().

10.44.3.6 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](#)().

10.44.4 Member Data Documentation

10.44.4.1 double USOClock::DerivLinearCoef [protected]

Slope.

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

Referenced by [getDeriv](#)(), [gGap](#)(), and [init](#)().

10.44.4.2 double USOClock::Offset [protected]

Offset.

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

Referenced by [getOffset](#)(), [gGap](#)(), and [init](#)().

10.44.4.3 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().

10.44.4.4 bool USOClock::USONoise [protected]

FALSE if there 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](#)

10.45 Vect Class Reference

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

10.45.1 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.

10.45.2 Constructor & Destructor Documentation

10.45.2.1 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.

10.45.2.2 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.

10.45.2.3 Vect::~Vect ()

Destructor.

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

10.45.3 Member Function Documentation

10.45.3.1 void Vect::display ()

Displays vector components.

Definition at line 49 of file LISACODE-Vect.cpp.

References p.

10.45.3.2 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().

10.45.3.3 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().

10.45.4 Friends And Related Function Documentation

10.45.4.1 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.

10.45.4.2 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.

10.45.4.3 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.

10.45.4.4 Vect operator+ (Vect *u*, Vect *v*) [friend]

Vectors addition : returns vector u+v.

Definition at line 101 of file LISACODE-Vect.cpp.

10.45.4.5 Vect operator- (Vect *u*, Vect *v*) [friend]

Vectors subtraction : returns vector u-v.

Definition at line 112 of file LISACODE-Vect.cpp.

10.45.4.6 Vect operator/ (Vect *u*, double *a*) [friend]

Vector and scalar division, returns a vector.

$$\text{returned value} = \frac{\vec{u}}{a}$$

Definition at line 171 of file LISACODE-Vect.cpp.

10.45.5 Member Data Documentation**10.45.5.1 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-MLDC::position(), GeometryFile::position(), GeometryAnalytic::position(), Geometry::position(), unit(), Vect(), Geometry::VectNormal(), GeometryMLDC::velocity(), GeometryFile::velocity(), Geometry-Analytic::velocity(), 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

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(), mlmod(), 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::gXMLdouble(), ConfigSim::gXMLFrequency(), ConfigSim::gXMLint(), ConfigSim::gXMLstring(), ConfigSim::gXMLTime(), ConfigSim::gXMLTimeSeries(), ConfigSim::ReadASCIIFile(), 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(), ConfigSim::ReadASCIIFile(), 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::ReadASCIIFile(), and 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::gXMLdouble()`, `ConfigSim::gXMLFrequency()`, `ConfigSim::gXMLInt()`, `ConfigSim::gXMLTime()`, `ConfigSim::gXMLTimeSeries()`, and `ConfigSim::ReadXMLFile()`.

11.4.1.11 EZXML_TXTM 0x40

attribute name and value are strdупed

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::gXMLdouble(), ConfigSim::gXMLFrequency(), ConfigSim::gXMLInt(), ConfigSim::gXMLstring(), ConfigSim::gXMLTime(), ConfigSim::gXMLTimeSeries(), ConfigSim::gXMLWord(), ConfigSim::ReadASCIIFile(), and ConfigSim::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::gXMLdouble(), ConfigSim::gXMLFrequency(), ConfigSim::gXMLInt(), ConfigSim::gXMLstring(), ConfigSim::gXMLTime(), ConfigSim::gXMLTimeSeries(), ConfigSim::ReadASCIIFile(), 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(), ConfigSim::ReadASCIIFile(), 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::ReadASCIIFile()`, and `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-GWSto.h"
#include "LISACODE-GWFastSpinBBH.h"
#include "LISACODE-GWCusp.h"
#include "LISACODE-GWNew.h"
#include "LISACODE-Background.h"
#include "LISACODE-BackgroundGalactic.h"
#include "LISACODE-Noise.h"
#include "LISACODE-NoiseWhite.h"
#include "LISACODE-NoiseFilter.h"
#include "LISACODE-NoiseFShape.h"
#include "LISACODE-NoiseOof.h"
#include "LISACODE-NoiseTwoFilter.h"
#include "LISACODE-NoiseFile.h"
#include "LISACODE-USOClock.h"
#include "LISACODE-Filter.h"
#include "LISACODE-Geometry.h"
#include "LISACODE-GeometryAnalytic.h"
#include "LISACODE-GeometryMLDC.h"
#include "LISACODE-GeometryFile.h"
```

Classes

- class [ConfigSim](#)
Class to configure LISA simulation, that is, LISACode execution.
- struct [NoiseSpec](#)
Noise specification structure.

11.12 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 .

11.12.1 Function Documentation

11.12.1.1 **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.

11.12.1.2 **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.

11.12.1.3 **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.

11.12.1.4 **Couple operator+ (Couple z1, Couple z2)**

2 couples addiction.

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

References Couple::x, and Couple::y.

11.12.1.5 **Couple operator- (Couple z1, Couple z2)**

2 couples subtraction.

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

References Couple::x, and Couple::y.

11.12.1.6 **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.

11.13 LISACODE-Couple.h File Reference

```
#include <stdexcept>
#include <iostream.h>
#include <math.h>
```

Classes

- class [Couple](#)

Couple management class.

11.14 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-GWSto.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, ..., 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 ≤ tmax with tStepMes timsetep.*
[TDI](#) is computed using [TDI_InterData::ComputeEta](#) method.
Delays are recorded.
Positions are recorded.

11.15 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 frequency and Cell (type **QuadCell**) inputs.*
- double **AbsRespFunctQuadCellChain** (double f, int NCells, **QuadCell** Cell[])
 *Elliptic cells chain frequency response modulus, depending on frequency, number of cell NCells and Cell (type **QuadCell**) inputs.*
- double **HmQuadCell** (**QuadCell** Cell)
 *Returns max |I/D(w)|, where D(w) is the denominator of an elliptic cell (type **QuadCell**).*
- double **KmQuadCell** (**QuadCell** Cell)
 *Returns max|Ell(w)|, where Ell(w) is the frequency response of an elliptic 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.

11.16 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 > [Im](#) (0, 1)
Pure imaginary=(0,I).
- 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 frequency and Cell (type `QuadCell`) inputs.
- `double AbsRespFunctQuadCellChain (double f, int NCells, QuadCell Cell[])`
Elliptic cells chain frequency response modulus, depending on frequency, number of cell NCells and Cell (type `QuadCell`) inputs.
- `double HmQuadCell (QuadCell Cell)`
Returns max |I/D(w)|, where D(w) is the denominator of an elliptic cell (type `QuadCell`).
- `double KmQuadCell (QuadCell Cell)`
Returns max|Ell(w)|, where Ell(w) is the frequency response of an elliptic 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-Filter.cpp File Reference

```
#include "LISACODE-Filter.h"
```

11.18 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.

11.19 LISACODE-Geometry.cpp File Reference

```
#include "LISACODE-Geometry.h"
```

11.20 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.

11.21 LISACODE-GeometryAnalytic.cpp File Reference

```
#include "LISACODE-GeometryAnalytic.h"
```

11.22 LISACODE-GeometryAnalytic.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"
#include "LISACODE-Geometry.h"
```

Classes

- class [GeometryAnalytic](#)

Compute spacecraft position from analytical formulations of the article : S.

11.23 LISACODE-GeometryFile.cpp File Reference

```
#include "LISACODE-GeometryFile.h"
```

11.24 LISACODE-GeometryFile.h File Reference

```
#include <stdexcept>
#include <iostream.h>
#include <math.h>
#include <vector.h>
#include <fstream.h>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-LISAConstants.h"
#include "LISACODE-Couple.h"
#include "LISACODE-Vect.h"
#include "LISACODE-Geometry.h"
```

Classes

- class [GeometryFile](#)

Compute spacecraft position from :.

11.25 LISACODE-GeometryMLDC.cpp File Reference

```
#include "LISACODE-GeometryMLDC.h"
```

11.26 LISACODE-GeometryMLDC.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"
#include "LISACODE-Geometry.h"
```

Classes

- class [GeometryMLDC](#)

Compute spacecraft position from :.

11.27 LISACODE-GW.cpp File Reference

```
#include "LISACODE-GW.h"
```

11.28 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.

11.29 LISACODE-GWBinary.cpp File Reference

```
#include "LISACODE-GWBinary.h"
```

11.30 LISACODE-GWBinary.h File Reference

```
#include <stdexcept>
#include <iostream.h>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "LISACODE-GW.h"
```

Classes

- class [GWBinary](#)

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

11.31 LISACODE-GWCusp.cpp File Reference

```
#include "LISACODE-GWCusp.h"
```

11.32 LISACODE-GWCusp.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "fftw3.h"
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-GW.h"
```

Classes

- class [GWCusp](#)

Gravitational Waves instantaneous parameters h_+ and h_\times are described in this class.

11.33 LISACODE-GWFastSpinBBH.cpp File Reference

```
#include "LISACODE-GWFastSpinBBH.h"
```

11.34 LISACODE-GWFastSpinBBH.h File Reference

```
#include <stdexcept>
#include <iostream.h>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-GW.h"
```

Classes

- class [GWFastSpinBBH](#)

Gravitational Waves parameters for a spinning black holes binary system are defined in this class.

11.35 LISACODE-GWFile.cpp File Reference

```
#include "LISACODE-GWFile.h"
```

11.36 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.

11.37 LISACODE-GWMono.cpp File Reference

```
#include "LISACODE-GWMono.h"
```

11.38 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.

11.39 LISACODE-GWNew.cpp File Reference

```
#include "LISACODE-GWNew.h"
```

11.40 LISACODE-GWNew.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 [GWNew](#)

Gravitational Waves instantaneous parameters h_{plus} and h_{cross} are described in this class.

11.41 LISACODE-GWNewton2.cpp File Reference

```
#include "LISACODE-GWNewton2.h"
```

11.42 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.

11.43 LISACODE-GWPeriGate.cpp File Reference

```
#include "LISACODE-GWPeriGate.h"
```

11.44 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.

11.45 LISACODE-GWSto.cpp File Reference

```
#include "LISACODE-GWSto.h"
```

11.46 LISACODE-GWSto.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-GW.h"
#include "LISACODE-Noise.h"
#include "LISACODE-NoiseWhite.h"
#include "LISACODE-NoiseFilter.h"
#include "LISACODE-NoiseFile.h"
#include "LISACODE-NoiseOof.h"
```

Classes

- class [GWSto](#)

Gravitational Waves instantaneous parameters h_{plus} and h_{cross} are described in this class.

11.47 LISACODE-LISA.cpp File Reference

```
#include "LISACODE-LISA.h"
```

11.48 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-NoiseFShape.h"
#include "LISACODE-NoiseOof.h"
#include "LISACODE-NoiseTwoFilter.h"
#include "LISACODE-TrFctGW.h"
#include "LISACODE-Geometry.h"
#include "LISACODE-GeometryAnalytic.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.

11.49 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-GWSto.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 will 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, ..., tMemTDI + tTDIShift with tStepMes timestep.*
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 ≤ tmax with tStepMes timestep.*
TDI is computed using [TDI_InterData::ComputeEta](#) method.
Delays are recorded.
Positions are recorded.

11.49.1 Function Documentation

11.49.1.1 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 $t = 0, \dots, tMemTDI + tTDIShift$ 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\(\)](#), [ConfigSim::getFileEncodingDelays\(\)](#), [ConfigSim::getFileEncodingPositions\(\)](#), [ConfigSim::getFileEncodingSig\(\)](#), [ConfigSim::getFileEncodingTDI\(\)](#), [ConfigSim::getFileNameDelays\(\)](#), [ConfigSim::getFileNamePositions\(\)](#), [ConfigSim::getFileNameSig\(\)](#), [ConfigSim::getFileNameTDI\(\)](#), [ConfigSim::getGenTDIPacks\(\)](#), [ConfigSim::getGenTDIPacksFact\(\)](#), [ConfigSim::getGlobalRandomSeed\(\)](#), [ConfigSim::getNameGenTDI\(\)](#), [ConfigSim::getNoNoise\(\)](#), [TDITools::getRapidOption\(\)](#), [getsd\(\)](#), [ConfigSim::gettDeltaTDIDelay\(\)](#), [ConfigSim::gettTDIDelayApprox\(\)](#), [ConfigSim::getTDIIInterp\(\)](#), [ConfigSim::getTDIIInterpUtilVal\(\)](#), [ConfigSim::gettDisplay\(\)](#), [ConfigSim::gettMax\(\)](#), [ConfigSim::gettMemPhasemeters\(\)](#), [ConfigSim::gettMemTDI\(\)](#), [ConfigSim::gettStartPhasemeters\(\)](#), [ConfigSim::gettStepMes\(\)](#), [ConfigSim::gettTDIShift\(\)](#), [LISA::gPosSC\(\)](#), [LCVersion](#), [LISA::MakeOneStepOfTime\(\)](#), [Memory::MakeTitles\(\)](#), [ConfigSim::NbGenTDI\(\)](#), [Vect::p](#), [Memory::ReceiveData\(\)](#), [Memory::RecordAccData\(\)](#), [TDITools::RefreshDelay\(\)](#), [setall\(\)](#), [MathUtils::TimeISO8601\(\)](#), [ConfigSim::tMaxDelay\(\)](#), and [ConfigSim::tMemNecInterpTDI\(\)](#).

11.50 LISACODE-LISACode.ep.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-GWSto.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"
```

11.51 LISACODE-LISAConstants.h File Reference

11.51.1 Detailed Description

Physical constants of [LISA](#) instrument.

Definition in file [LISACODE-LISAConstants.h](#).

```
#include <math.h>
#include "LISACODE-PhysicConstants.h"
```

Variables

- const char **LCVersion** [] = "LISACode v 1.4.4"
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.

11.51.2 Variable Documentation

11.51.2.1 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(), GeometryAnalytic::GeometryAnalytic(), GeometryFile::GeometryFile(), GeometryMLDC::GeometryMLDC(), and ConfigSim::NoisesCreation().

11.51.2.2 const double la0Laser_m = 1.064e-6

Nominal lasers wave length in meters.

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

11.51.2.3 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().

11.51.2.4 const char LCVersion[] = "LISACode v 1.4.4"

Simulator Version.

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

Referenced by main(), and MemoryWriteDisk::MakeTitles().

11.51.2.5 const double nu0Laser_Hz = c_SI/la0Laser_m

Nominal lasers frequency in Hz.

Definition at line 51 of file LISACODE-LISAConstants.h.

11.51.2.6 const double omega = 2.*M_PI/Yr_SI

Angular velocity.

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

Referenced by elli(), GeometryAnalytic::exanom(), GWNewton2::hc(), GWNewton2::hp(), ignpoi(), GeometryMLDC::position(), GeometryMLDC::velocity(), and GeometryAnalytic::velocity().

11.51.2.7 const double Rgc = au_m

Distance between the LISA barycenter and the Sun.

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

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

11.51.2.8 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::initGlobal().

11.51.2.9 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::initGlobal().

11.52 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.

11.52.1 Function Documentation

11.52.1.1 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.

11.52.1.2 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.

11.52.1.3 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.

11.52.1.4 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.

11.53 LISACODE-Mat.h File Reference

```
#include <stdexcept>
#include <iostream.h>
#include <math.h>
#include "LISACODE-Vect.h"
```

Classes

- class [Mat](#)

(3x3) matrix management class.

11.54 LISACODE-MathUtils.h File Reference

```
#include <math.h>
#include <vector.h>
#include <time.h>
#include <iostream>
```

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))

11.55 LISACODE-Memory.cpp File Reference

```
#include "LISACODE-Memory.h"
```

11.56 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.

11.57 LISACODE-MemoryReadDisk.cpp File Reference

```
#include "LISACODE-MemoryReadDisk.h"
```

11.58 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. This class (module) reads data in file and stores them.

11.59 LISACODE-MemoryWriteDisk.cpp File Reference

```
#include "LISACODE-MemoryWriteDisk.h"
```

11.60 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 writing. This class (module) manages and stores data supplied to its in a temporary memory and in a file. Index of first series is 0.

11.61 LISACODE-Noise.cpp File Reference

```
#include "LISACODE-Noise.h"
```

11.62 LISACODE-Noise.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <cstring>
#include "LISACODE-PhysicConstants.h"
#include "LISACODE-MathUtils.h"
#include "LISACODE-LISAConstants.h"
```

Namespaces

- namespace [std](#)

Classes

- class [Noise](#)

Noise base class.

11.63 LISACODE-NoiseFile.cpp File Reference

```
#include "LISACODE-NoiseFile.h"
```

11.64 LISACODE-NoiseFile.h File Reference

```
#include <stdexcept>
#include <iostream>
#include <vector.h>
#include <stdlib.h>
#include <math.h>
#include <fstream.h>
#include "randlib.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.

11.65 LISACODE-NoiseFilter.cpp File Reference

```
#include "LISACODE-NoiseFilter.h"
```

11.66 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.

11.67 LISACODE-NoiseFShape.cpp File Reference

```
#include "LISACODE-NoiseFShape.h"
```

11.68 LISACODE-NoiseFShape.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 [NoiseFShape](#)
Noise derived class to treat noise filters.

11.69 LISACODE-NoiseOof.cpp File Reference

```
#include "LISACODE-NoiseOof.h"
```

11.70 LISACODE-NoiseOof.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 [NoiseOof](#)

Noise derived class to treat noise filters.

11.71 LISACODE-NoiseTwoFilter.cpp File Reference

```
#include "LISACODE-NoiseTwoFilter.h"
```

11.72 LISACODE-NoiseTwoFilter.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 [NoiseTwoFilter](#)

Noise derived class to treat noise filters.

11.73 LISACODE-NoiseWhite.cpp File Reference

```
#include "LISACODE-NoiseWhite.h"
```

11.74 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.

11.75 LISACODE-PhoDetPhaMet.cpp File Reference

```
#include "LISACODE-PhoDetPhaMet.h"
```

11.76 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.

11.77 LISACODE-PhysicConstants.h File Reference

11.77.1 Detailed Description

Physical constants, reference values and unit conversions.

Definition in file [LISACODE-PhysicConstants.h](#).

```
#include <math.h>
#include <float.h>
#include <cstring>
```

Variables

- const double **c_SI** = 299792458
Light's speed in m · s⁻¹.
- const double **G_SI** = 6.67259e-11
Gravitational constant in m³ · Kg⁻¹ · s⁻².
- const double **k_SI** = 1.381e-23
Boltzmann's constant in joule · kelvin⁻¹.
- const double **h_SI** = 6.62620e-34
Planck's constant in joule · second.
- const double **hb_SI** = **h_SI**/(2*M_PI)
Planck's constant divided by 2 · π in joule · second.
- const double **S_SI** = 5.671e-8
Stefan's constant in Watt · meter⁻² · kelvin⁻⁴.
- const double **mu0_SI** = 4.e-7*M_PI
Permeability of free space or magnetic constant μ₀ in Henry · meter⁻¹.
- const double **eps0_SI** = 1.0/(4.e-7*M_PI*c_SI*c_SI)
Permittivity of free space or permittivity ε₀ in Farad · 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** = 0.57721566490153286060651209008240243104215933593994
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 **TSUN** = 4.92549232189886339689643862e-6
Sun's mass in Seconds.
- 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 Schwarzschild radius in $\frac{GM}{c^2}$.
- const double **au_m** = 1.49597870660e11
Astronomical unit in meters.
- const double **au_s** = **au_m/c_SI**
Astronomical unit in seconds.
- 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** = 3.08568025e19
Number of meters in a kiloparsec (kpc).

- const double **omegaYr** = 2.*M_PI/Yr_SI
Angular velocity of Earth.
- const double **PRECISION** = 100.0*DBL_EPSILON
Acceptable precision error on doubles.
- const double **LISA_L0_m** = 5.0e9
Armlength in meter.
- const double **LISA_L0_s** = **LISA_L0_m/c_SI**
Armlength in second.
- const double **gamma_u** = 1.
Post-Newtonian constant.

11.77.2 Variable Documentation

11.77.2.1 const double **au_m** = 1.49597870660e11

Astronomical unit in meters.

Definition at line 105 of file LISACODE-PhysicConstants.h.

Referenced by GeometryMLDC::position(), and GeometryMLDC::velocity().

11.77.2.2 const double **au_s** = **au_m/c_SI**

Astronomical unit in seconds.

Definition at line 108 of file LISACODE-PhysicConstants.h.

11.77.2.3 const double **c_SI** = 299792458

Light's speed in $m \cdot s^{-1}$.

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

Referenced by GWNewton2::commun(), TrFctGW::deltanu(), LISA::gArmLength(), GWNewton2::init(), GWFastSpinBBH::init(), GWBinary::init(), ConfigSim::ReadXMLFile(), Geometry::tdelay(), Geometry::tdelayOrderContribution(), ConfigSim::tMaxDelay(), and ConfigSim::tMinDelay().

11.77.2.4 const double **CE_RG** = 0.57721566490153286060651209008240243104215933593994

Euler's constant.

Definition at line 61 of file LISACODE-PhysicConstants.h.

Referenced by GWNewton2::init().

11.77.2.5 const double Dy_SI = 24.0*3600.0

Day duration in seconds.

Definition at line 92 of file LISACODE-PhysicConstants.h.

11.77.2.6 const double eps0_SI = 1.0/(4.e-7*M_PI*c_SI*c_SI)

Permittivity of free space or permittivity ϵ_0 in $\text{Farad} \cdot \text{meter}^{-1}$.

Definition at line 52 of file LISACODE-PhysicConstants.h.

11.77.2.7 const double G_SI = 6.67259e-11

Gravitational constant in $m^3 \cdot Kg^{-1} \cdot s^{-2}$.

Definition at line 32 of file LISACODE-PhysicConstants.h.

Referenced by GWNewton2::commun(), GWNewton2::init(), and GWBinary::init().

11.77.2.8 const double gamma_u = 1.

Post-Newtonian constant.

Definition at line 152 of file LISACODE-PhysicConstants.h.

Referenced by Geometry::tdelay(), and Geometry::tdelayOrderContribution().

11.77.2.9 const double H0_cgs = 0.7*3.24e-18

Hubble's constant in CGS.

Definition at line 58 of file LISACODE-PhysicConstants.h.

11.77.2.10 const double h_SI = 6.62620e-34

Planck's constant in $joule \cdot second$.

Definition at line 38 of file LISACODE-PhysicConstants.h.

11.77.2.11 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.

11.77.2.12 const double k_SI = 1.381e-23

Boltzmann's constant in $joule \cdot kelvin^{-1}$.

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

11.77.2.13 const double kpc_m = 3.08568025e19

Number of meters in a kiloparsec (kpc).

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

Referenced by GWBinary::GWBinary(), GWNewton2::GWNewton2(), and GWFastSpinBBH::init().

11.77.2.14 const double LISA_L0_m = 5.0e9

Armlength in meter.

Definition at line 146 of file LISACODE-PhysicConstants.h.

11.77.2.15 const double LISA_L0_s = LISA_L0_m/c_SI

Armlength in second.

Definition at line 149 of file LISACODE-PhysicConstants.h.

11.77.2.16 const double LS_SI = 3.83e26

Sun's energy flux in Watt.

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

11.77.2.17 const double ly_au = ly_m/au_m

Light year in astronomical units (63240.17695575401).

Definition at line 114 of file LISACODE-PhysicConstants.h.

11.77.2.18 const double ly_m = c_SI*365.25*24.0*3600.0

Light year in meters ($9.460730472580800 \cdot 10^{15}$).

Definition at line 111 of file LISACODE-PhysicConstants.h.

11.77.2.19 const double me_SI = 9.1091e-31

Electron rest mass in Kg.

Definition at line 68 of file LISACODE-PhysicConstants.h.

11.77.2.20 const double mn_SI = 1.6748e-27

Neutron rest mass in Kg.

Definition at line 74 of file LISACODE-PhysicConstants.h.

11.77.2.21 const double mp_SI = 1.6726e-27

Proton rest mass in Kg.

Definition at line 71 of file LISACODE-PhysicConstants.h.

11.77.2.22 const double MS_SI = 1.9889e30

Sun's mass in Kg.

Definition at line 77 of file LISACODE-PhysicConstants.h.

Referenced by GWBinary::GWBinary(), and GWNewton2::GWNewton2().

11.77.2.23 const double mu0_SI = 4.e-7*M_PI

Permeability of free space or magnetic constant μ_0 in $\text{Henry} \cdot \text{meter}^{-1}$.

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

11.77.2.24 const double Na_SI = 6.02252e-27

Avogadro's number in mol^{-1} .

Definition at line 55 of file LISACODE-PhysicConstants.h.

11.77.2.25 const double omegaYr = 2.*M_PI/Yr_SI

Angular velocity of Earth.

Definition at line 130 of file LISACODE-PhysicConstants.h.

11.77.2.26 const double pc_au = M_PI/(3600.0*180.0)

Parsec in astronomical unit (206265).

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

11.77.2.27 const double pc_ly = pc_au/ly_au

Parsec in light year (3.262).

Definition at line 123 of file LISACODE-PhysicConstants.h.

11.77.2.28 const double pc_m = pc_au*au_m

Parsec in meters ($3.086 \cdot 10^{16}$).

Definition at line 120 of file LISACODE-PhysicConstants.h.

11.77.2.29 const double PRECISION = 100.0*DBL_EPSILON

Acceptable precision error on doubles.

Definition at line 138 of file LISACODE-PhysicConstants.h.

Referenced by GWFastSpinBBH::AmpShared(), GWNewton2::commun(), BackgroundGalactic::deltanu(), Serie::gData(), Noise::getNoise(), NoiseFile::loadNoise(), Noise::Noise(), NoiseFile::NoiseFile(), NoiseFilter::NoiseFilter(), NoiseFShape::NoiseFShape(), NoiseOof::NoiseOof(), ConfigSim::NoisesCreation(), NoiseTwoFilter::NoiseTwoFilter(), NoiseWhite::NoiseWhite(), ConfigSim::ReadASCIIFile(), ConfigSim::ReadXMLFile(), GW::setDirProp(), Noise::settDurAdd(), Noise::settFirst(), and Noise::settLast().

11.77.2.30 const double RS_SI = 6.95e8

Sun's radius in meter.

Definition at line 83 of file LISACODE-PhysicConstants.h.

11.77.2.31 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 98 of file LISACODE-PhysicConstants.h.

Referenced by Geometry::tdelay(), and Geometry::tdelayOrderContribution().

11.77.2.32 const double S_SI = 5.671e-8

Stefan's constant in $\text{Watt} \cdot \text{meter}^{-2} \cdot \text{kelvin}^{-1}$.

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

11.77.2.33 const double TSUN = 4.92549232189886339689643862e-6

Sun's mass in Seconds.

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

Referenced by GWFastSpinBBH::AmpShared(), GWFastSpinBBH::Freq(), GWFastSpinBBH::init(), and GWFastSpinBBH::update().

11.77.2.34 const double Yr_SI = 3.15581498e7

Sidereal year in seconds.

Definition at line 89 of file LISACODE-PhysicConstants.h.

Referenced by ConfigSim::ReadXMLFile().

11.78 LISACODE-Serie.cpp File Reference

```
#include "LISACODE-Serie.h"
```

11.79 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.

11.80 LISACODE-TDI.cpp File Reference

```
#include "LISACODE-TDI.h"
```

11.81 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.

11.82 LISACODE-TDI_InterData.cpp File Reference

```
#include "LISACODE-TDI_InterData.h"
```

11.83 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.

11.84 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[])

11.84.1 Function Documentation

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

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

References Memory::AddSerieData(), TDI_InterData::ComputeEta(), ConfigSim::getFileEncodingDelays(), ConfigSim::getFileEncodingSig(), ConfigSim::getFileEncodingTDI(), ConfigSim::getFileNameDelays(), ConfigSim::getFileNameSig(), ConfigSim::getFileNameTDI(), ConfigSim::getGenTDIPacks(), ConfigSim::getGenTDIPacksFact(), ConfigSim::getNameGenTDI(), ConfigSim::getNoNoise(), TDITools::getRapidOption(), ConfigSim::gettDeltaTDIDelay(), ConfigSim::getTDIDelayApprox(), ConfigSim::getTDIInterp(), ConfigSim::getTDIInterpUtilVal(), ConfigSim::gettDisplay(), ConfigSim::gettMax(), ConfigSim::gettMemPhasemeters(), ConfigSim::gettMemTDI(), ConfigSim::gettStartPhasemeters(), ConfigSim::gettStepMes(), ConfigSim::gettTDIShift(), LCVersion, ConfigSim::NbGenTDI(), Memory::RecordAccData(), TDITools::RefreshDelay(), ConfigSim::tMaxDelay(), ConfigSim::tMemNecInterpTDI(), and ConfigSim::UseInternalPhasemeter().

11.85 LISACODE-TDITools.cpp File Reference

```
#include "LISACODE-TDITools.h"
```

11.86 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.

11.87 LISACODE-TrFctGW.cpp File Reference

```
#include "LISACODE-TrFctGW.h"  
#include <fstream.h>
```

11.88 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.

11.89 LISACODE-USOClock.cpp File Reference

```
#include "LISACODE-USOClock.h"
```

11.90 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.

11.91 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.

11.91.1 Function Documentation

11.91.1.1 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.

11.91.1.2 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.

11.91.1.3 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.

11.91.1.4 **Vect** operator+ (**Vect** *u*, **Vect** *v*)

Vectors addition : returns vector u+v.

Definition at line 101 of file LISACODE-Vect.cpp.

References Vect::p.

11.91.1.5 **Vect** operator- (**Vect** *u*, **Vect** *v*)

Vectors subtraction : returns vector u-v.

Definition at line 112 of file LISACODE-Vect.cpp.

References Vect::p.

11.91.1.6 **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.

11.92 LISACODE-Vect.h File Reference

```
#include <stdexcept>
#include <iostream.h>
#include <math.h>
```

Classes

- class [Vect](#)

3 components vector management class.

11.93 randlib.c File Reference

```
#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 **fnstop** (char *)
- float **genbet** (float aa, float bb)
- 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)
- 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 **ignbn** (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)

11.93.1 Define Documentation

11.93.1.1 #define ABS(x) ((x) >= 0 ? (x) : -(x))

Definition at line 5 of file randlib.c.

Referenced by ignbin().

11.93.1.2 #define expmax 87.49823

11.93.1.3 #define h 32768L

11.93.1.4 #define infnty 1.0E38

11.93.1.5 #define max(a, b) ((a) >= (b) ? (a) : (b))

Definition at line 7 of file randlib.c.

Referenced by ezxml_ampencode(), ezxml_str2utf8(), ezxml_toxml(), ezxml_toxml_r(), genbet(), and ignpoi().

11.93.1.6 #define maxnum 2147483561L

11.93.1.7 #define min(a, b) ((a) <= (b) ? (a) : (b))

Definition at line 6 of file randlib.c.

Referenced by genbet(), ignbin(), and ignpoi().

11.93.1.8 #define minlog 1.0E-37

11.93.1.9 #define numg 32L

11.93.2 Function Documentation

11.93.2.1 float fsign (float num, float sign)

Definition at line 2089 of file randlib.c.

Referenced by ignpoi(), and sgamma().

11.93.2.2 void ftnstop (char *)

Definition at line 2100 of file randlib.c.

Referenced by genmul(), ignbin(), and ignnbn().

11.93.2.3 float genbet (float aa, float bb)

Definition at line 9 of file randlib.c.

References genbet(), max, min, and ranf().

Referenced by genbet().

11.93.2.4 float genchi (float *df*)

Definition at line 244 of file randlib.c.

References genchi(), and sgamma().

Referenced by genchi().

11.93.2.5 float genexp (float *av*)

Definition at line 275 of file randlib.c.

References genexp(), and sexpo().

Referenced by genexp().

11.93.2.6 float genf (float *dfn*, float *dfd*)

Definition at line 309 of file randlib.c.

References genf(), and sgamma().

Referenced by genf().

11.93.2.7 float gengam (float *a*, float *r*)

Definition at line 365 of file randlib.c.

References gengam(), and sgamma().

Referenced by gengam().

11.93.2.8 void genmn (float **parm*, float **x*, float **work*)

Definition at line 410 of file randlib.c.

References snorm().

11.93.2.9 void genmul (long *n*, float **p*, long *ncat*, long **ix*)

Definition at line 461 of file randlib.c.

References fnstop(), and ignbin().

11.93.2.10 float gennch (float *df*, float *xnonc*)

Definition at line 522 of file randlib.c.

References gennch(), sgamma(), and snorm().

Referenced by gennch().

11.93.2.11 float gennf (float *dfn*, float *dfd*, float *xnonc*)

Definition at line 567 of file randlib.c.

References gennf(), sgamma(), and snorm().

Referenced by gennf().

11.93.2.12 float gennor (float *av*, float *sd*)

Definition at line 640 of file randlib.c.

References gennor(), and snorm().

Referenced by gennor().

11.93.2.13 void genprm (long * *iarray*, int *larray*)

Definition at line 674 of file randlib.c.

References ignuin().

11.93.2.14 float genunf (float *low*, float *high*)

Definition at line 695 of file randlib.c.

References genunf(), and ranf().

Referenced by NoiseFilter::generNoise(), NoiseFShape::generNoise(), NoiseOof::generNoise(), NoiseTwoFilter::generNoise(), NoiseWhite::generNoise(), genunf(), USOClock::gGap(), NoiseFile::loadNoise(), NoiseFilter::loadNoise(), NoiseFShape::loadNoise(), NoiseOof::loadNoise(), NoiseTwoFilter::loadNoise(), and NoiseWhite::loadNoise().

11.93.2.15 void gscgn (long *getset*, long * *g*)

Definition at line 718 of file randlib.c.

Referenced by advnst(), getsd(), ignlgi(), initgn(), setall(), setant(), and setsd().

11.93.2.16 void gsrgs (long *getset*, long * *qvalue*)

Definition at line 743 of file randlib.c.

Referenced by advnst(), getsd(), ignlgi(), initgn(), inrgcm(), setall(), setant(), and setsd().

11.93.2.17 void gssst (long *getset*, long * *qset*)

Definition at line 760 of file randlib.c.

Referenced by ignlgi(), and setall().

11.93.2.18 long ignbin (long *n*, float *pp*)

Definition at line 776 of file randlib.c.

References ABS, ftnstop(), ignbin(), min, and ranf().

Referenced by genmul(), and ignbin().

11.93.2.19 long ignnbn (long *n*, float *p*)

Definition at line 1043 of file randlib.c.

References ftnstop(), ignnbn(), ignpoi(), and sgamma().

Referenced by ignnbn().

11.93.2.20 long ignpoi (float *mu*)

Definition at line 1098 of file randlib.c.

References fsign(), ignpoi(), max, min, omega, ranf(), expo(), and snorm().

Referenced by ignnbn(), and ignpoi().

11.93.2.21 long ignuin (long *low*, long *high*)

Definition at line 1348 of file randlib.c.

References ignlgi(), and ignuin().

Referenced by genprm(), and ignuin().

11.93.2.22 long lennob (char * *str*)

Definition at line 1401 of file randlib.c.

Referenced by phrtsd().

11.93.2.23 long mltmod (long *a*, long *s*, long *m*)

Definition at line 1413 of file randlib.c.

References mltmod().

Referenced by advnst(), initgn(), mltmod(), and setall().

11.93.2.24 void phrtsd (char * *phrase*, long * *seed1*, long * *seed2*)

Definition at line 1507 of file randlib.c.

References lennob().

11.93.2.25 float ranf (void)

Definition at line 1569 of file randlib.c.

References ignlgi(), and ranf().

Referenced by genbet(), genunf(), ignbin(), ignpoi(), ranf(), expo(), sgamma(), and snorm().

11.93.2.26 void setgmn (float * *meanv*, float * *covm*, long *p*, float * *parm*)

Definition at line 1593 of file randlib.c.

References spofa().

11.93.2.27 float sexpo (void)

Definition at line 1658 of file randlib.c.

References ranf(), and sexpo().

Referenced by genexp(), ignpoi(), sexpo(), and sgamma().

11.93.2.28 float sgamma (float *a*)

Definition at line 1723 of file randlib.c.

References fsign(), ranf(), sexpo(), sgamma(), and snorm().

Referenced by genchi(), genf(), gengam(), gennch(), gennf(), ignnbn(), and sgamma().

11.93.2.29 float snorm (void)

Definition at line 1962 of file randlib.c.

References ranf(), and snorm().

Referenced by genmn(), gennch(), gennf(), gennor(), ignpoi(), sgamma(), and snorm().

11.94 randlib.h File Reference

Functions

- void [advnst](#) (long k)
- float [genbet](#) (float aa, float bb)
- 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)
- 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)

11.94.1 Function Documentation

11.94.1.1 void advnst (long *k*)

Definition at line 7 of file com.c.

References gscgn(), gsrgs(), mltmod(), setsd(), Xa1, Xa2, Xcg1, Xcg2, Xm1, and Xm2.

11.94.1.2 float genbet (float *aa*, float *bb*)

Definition at line 9 of file randlib.c.

References genbet(), max, min, and ranf().

Referenced by genbet().

11.94.1.3 float genchi (float *df*)

Definition at line 244 of file randlib.c.

References genchi(), and sgamma().

Referenced by genchi().

11.94.1.4 float genexp (float *av*)

Definition at line 275 of file randlib.c.

References genexp(), and sexpo().

Referenced by genexp().

11.94.1.5 float genf (float *dfn*, float *dfd*)

Definition at line 309 of file randlib.c.

References genf(), and sgamma().

Referenced by genf().

11.94.1.6 float gengam (float *a*, float *r*)

Definition at line 365 of file randlib.c.

References gengam(), and sgamma().

Referenced by gengam().

11.94.1.7 void genmn (float **parm*, float **x*, float **work*)

Definition at line 410 of file randlib.c.

References snorm().

11.94.1.8 void genmul (long *n*, float **p*, long *ncat*, long **ix*)

Definition at line 461 of file randlib.c.

References fnstop(), and ignbin().

11.94.1.9 float gennch (float *df*, float *xnonc*)

Definition at line 522 of file randlib.c.

References gennch(), sgamma(), and snorm().

Referenced by gennch().

11.94.1.10 float gennf (float *dfn*, float *dfd*, float *xnonc*)

Definition at line 567 of file randlib.c.

References gennf(), sgamma(), and snorm().

Referenced by gennf().

11.94.1.11 float gennor (float *av*, float *sd*)

Definition at line 640 of file randlib.c.

References gennor(), and snorm().

Referenced by gennor().

11.94.1.12 void genprm (long * *iarray*, int *larray*)

Definition at line 674 of file randlib.c.

References ignuin().

11.94.1.13 float genunf (float *low*, float *high*)

Definition at line 695 of file randlib.c.

References genunf(), and ranf().

Referenced by NoiseWhite::generNoise(), NoiseTwoFilter::generNoise(), NoiseOof::generNoise(), NoiseFShape::generNoise(), NoiseFilter::generNoise(), genunf(), USOClock::gGap(), NoiseWhite::loadNoise(), NoiseTwoFilter::loadNoise(), NoiseOof::loadNoise(), NoiseFShape::loadNoise(), NoiseFilter::loadNoise(), and NoiseFile::loadNoise().

11.94.1.14 void getsd (long * *iseed1*, long * *iseed2*)

Definition at line 52 of file com.c.

References gscgn(), gsrgs(), Xcg1, and Xcg2.

Referenced by main().

11.94.1.15 void gscgn (long *getset*, long * *g*)

Definition at line 718 of file randlib.c.

Referenced by advnst(), getsd(), ignlgi(), initgn(), setall(), setant(), and setsd().

11.94.1.16 long ignbin (long *n*, float *pp*)

Definition at line 776 of file randlib.c.

References ABS, ftnstop(), ignbin(), min, and ranf().

Referenced by genmul(), and ignbin().

11.94.1.17 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().

11.94.1.18 long ignnbn (long *n*, float *p*)

Definition at line 1043 of file randlib.c.

References fnstop(), ignnbn(), ignpoi(), and sgamma().

Referenced by ignnbn().

11.94.1.19 long ignpoi (float *mu*)

Definition at line 1098 of file randlib.c.

References fsign(), ignpoi(), max, min, omega, ranf(), expo(), and snorm().

Referenced by ignnbn(), and ignpoi().

11.94.1.20 long ignuin (long *low*, long *high*)

Definition at line 1348 of file randlib.c.

References ignlgi(), and ignuin().

Referenced by genprm(), and ignuin().

11.94.1.21 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().

11.94.1.22 long mltmod (long *a*, long *s*, long *m*)

Definition at line 1413 of file randlib.c.

References mltmod().

Referenced by advnst(), initgn(), mltmod(), and setall().

11.94.1.23 void phrtsd (char * *phrase*, long * *seed1*, long * *seed2*)

Definition at line 1507 of file randlib.c.

References lennob().

11.94.1.24 float ranf (void)

Definition at line 1569 of file randlib.c.

References ignlgi(), and ranf().

Referenced by genbet(), genunf(), ignbin(), ignpoi(), ranf(), expo(), sgamma(), and snorm().

11.94.1.25 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().

11.94.1.26 void setant (long *qvalue*)

Definition at line 296 of file com.c.

References gscgn(), gsrgs(), and Xqanti.

11.94.1.27 void setgmn (float * *meanv*, float * *covm*, long *p*, float * *parm*)

Definition at line 1593 of file randlib.c.

References spofa().

11.94.1.28 void setsd (long *iseed1*, long *iseed2*)

Definition at line 337 of file com.c.

References gscgn(), gsrgs(), initgn(), Xig1, and Xig2.

Referenced by advnst().

11.94.1.29 float expo (void)

Definition at line 1658 of file randlib.c.

References ranf(), and expo().

Referenced by genexp(), ignpoi(), expo(), and sgamma().

11.94.1.30 float sgamma (float *a*)

Definition at line 1723 of file randlib.c.

References fsign(), ranf(), expo(), sgamma(), and snorm().

Referenced by genchi(), genf(), gengam(), gennch(), gennf(), ignnbn(), and sgamma().

11.94.1.31 float snorm (void)

Definition at line 1962 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[Petiteau 2008] 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 and [Bender 1998]). 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] [Dhurandhar 2002]) 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.

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>).

12.1.2 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 [[Petiteau 2008](#)][[Petiteau 2006](#)]). Its main components are:

- a variety of [GW](#) inputs,
- a detailed description of the orbits [[Dhurandhar 2004](#)] 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 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 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.

12.2.4 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 [\[LISACode\]](#)). 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`

12.2.5 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.

12.2.6 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.

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 `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, int order) 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::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::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::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 NoiseOof::NoiseOof() Make a function to compute a int from a double. Is there a reason to no call rint ?

Member NoiseOof::NoiseOof(double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, double fmin, double
Make a function to compute a int from a double. Is there a reason to no call rint ?

Member NoiseTwoFilter::NoiseTwoFilter() loadNoise and generateNoise have confuse names in relation to the activities of the methods.

Member NoiseTwoFilter::NoiseTwoFilter() 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

- ~Background
 - Background, 52
- ~BackgroundGalactic
 - BackgroundGalactic, 56
- ~ConfigSim
 - ConfigSim, 68
- ~Couple
 - Couple, 97
- ~Filter
 - Filter, 106
- ~GW
 - GW, 172
- ~GWBinary
 - GWBinary, 181
- ~GWCusp
 - GWCusp, 191
- ~GWFastSpinBBH
 - GWFastSpinBBH, 206
- ~GWFile
 - GWFile, 232
- ~GWMono
 - GWMono, 243
- ~GWNew
 - GWNew, 253
- ~GWNewton2
 - GWNewton2, 268
- ~GWPeriGate
 - GWPeriGate, 290
- ~Geometry
 - Geometry, 113
- ~GeometryAnalytic
 - GeometryAnalytic, 128
- ~GeometryFile
 - GeometryFile, 144
- ~GeometryMLDC
 - GeometryMLDC, 159
- ~LISA
 - LISA, 310
- ~Mat
 - Mat, 316
- ~Memory
 - Memory, 321
- ~MemoryReadDisk
 - MemoryReadDisk, 327
- ~MemoryWriteDisk
 - MemoryWriteDisk, 336
- ~Noise
 - Noise, 345
- ~PhoDetPhaMet
 - PhoDetPhaMet, 426
- ~Serie
 - Serie, 440
- ~SerieC
 - SerieC, 448
- ~TDI
 - TDI, 456
- ~TDITools
 - TDITools, 470
- ~TDI_InterData
 - TDI_InterData, 465
- ~TrFctGW
 - TrFctGW, 473
- ~USOClock
 - USOClock, 478
- ~Vect
 - Vect, 482
- a0
 - QuadCell, 435
- a1
 - GWNewton2, 277
 - QuadCell, 435
- a11
 - GWNewton2, 277
- a1x
 - GWNewton2, 277
- A2
 - GWFastSpinBBH, 215
- a2
 - GWNewton2, 277
- a22
 - GWNewton2, 278
- A3
 - GWFastSpinBBH, 215
- a3
 - GWNewton2, 278
- A4
 - GWFastSpinBBH, 215
- a4
 - GWNewton2, 278

A5
 GWFastSpinBBH, 215

a5
 GWNewton2, 278

A6
 GWFastSpinBBH, 215

a6
 GWNewton2, 278

a7
 GWNewton2, 278

ABS
 randlib.c, 611

AbsRespFunctQuadCell
 ellipFilter, 22

AbsRespFunctQuadCellChain
 ellipFilter, 23

Ac
 GWBinary, 186

addData
 Serie, 440

addDataC
 SerieC, 449

addNoise
 Noise, 345
 NoiseFile, 354
 NoiseFilter, 367
 NoiseFShape, 377
 NoiseOof, 389
 NoiseTwoFilter, 402
 NoiseWhite, 413

AddSerieData
 Memory, 321
 MemoryReadDisk, 328
 MemoryWriteDisk, 337

AddTimeSeriesInXMLOutput
 ConfigSim, 68

advnst
 com.c, 486
 randlib.h, 616

ak
 ellipFilter, 23

AllocMem
 GWCusp, 192

AllocMemory
 GWFastSpinBBH, 206

alog
 ellipFilter, 22

alog10
 ellipFilter, 22

alpha
 Filter, 108
 Geometry, 119
 GeometryAnalytic, 135
 GeometryFile, 150

GeometryMLDC, 165
 NoiseOof, 394

AlreadyRecDat
 Memory, 324

MemoryReadDisk, 331
 MemoryWriteDisk, 340

Amp
 GWFastSpinBBH, 215

Amplhc
 GWMono, 247
 GWNew, 257
 GWPeriGate, 295

Amplhp
 GWMono, 247
 GWNew, 257
 GWPeriGate, 295

Amplitude
 GWCusp, 195

AmplPNorder
 GWFastSpinBBH, 215

AmpShared
 GWFastSpinBBH, 206

Angles handling, 34

AnglPol
 GW, 176
 GWBinary, 186
 GWCusp, 195
 GWFastSpinBBH, 215
 GWFile, 237
 GWMono, 248
 GWNew, 258
 GWNewton2, 278
 GWPeriGate, 295
 GWSto, 303

Ap
 GWBinary, 187

App
 Filter, 107

Approx
 GeometryMLDC, 165

Armlength
 ConfigSim, 86

ArmVelocity
 Geometry, 113
 GeometryAnalytic, 128
 GeometryFile, 144
 GeometryMLDC, 159

arot
 Geometry, 119
 GeometryAnalytic, 135
 GeometryFile, 150
 GeometryMLDC, 165

attr
 ezxml, 99

ezxml_root, 102
 au_m
 LISACODE-PhysicConstants.h, 589
 au_s
 LISACODE-PhysicConstants.h, 589
 Author
 ConfigSim, 86
 Ax
 GWFastSpinBBH, 216
 AzimuthalAngleOfSpin1
 GWFastSpinBBH, 216
 AzimuthalAngleOfSpin2
 GWFastSpinBBH, 216

 b1
 GWNewton2, 278
 QuadCell, 435
 b11
 GWNewton2, 279
 b1x
 GWNewton2, 279
 b2
 GWNewton2, 279
 QuadCell, 436
 B21
 GWFastSpinBBH, 216
 b3
 GWNewton2, 279
 B31
 GWFastSpinBBH, 216
 B32
 GWFastSpinBBH, 216
 b4
 GWNewton2, 279
 B41
 GWFastSpinBBH, 216
 B42
 GWFastSpinBBH, 216
 B43
 GWFastSpinBBH, 217
 B51
 GWFastSpinBBH, 217
 B52
 GWFastSpinBBH, 217
 B53
 GWFastSpinBBH, 217
 B54
 GWFastSpinBBH, 217
 B61
 GWFastSpinBBH, 217
 B62
 GWFastSpinBBH, 217
 B63
 GWFastSpinBBH, 217

 B64
 GWFastSpinBBH, 217
 B65
 GWFastSpinBBH, 218
 Background, 51
 ~Background, 52
 Background, 52
 deltanu, 52
 LISAGeo, 53
 setGeometry, 52
 Background (directory Background), 38
 BackgroundGalactic, 54
 BackgroundGalactic, 55, 56
 BackgroundGalactic
 ~BackgroundGalactic, 56
 BackgroundGalactic, 55, 56
 deltanu, 57
 iRead, 58
 LISAGeo, 58
 NbData, 58
 ReadFile, 57
 setGeometry, 57
 SignalList, 58
 TimeList, 58
 tmp_ci, 58
 tmp_cip1, 59
 tmp_Sig_i, 59
 tmp_Sig_ip1, 59
 tmp_t, 59
 Beta
 GW, 176
 GWBinary, 187
 GWCusp, 195
 GWFastSpinBBH, 218
 GWFile, 237
 GWMono, 248
 GWNew, 258
 GWNewton2, 279
 GWPeriGate, 295
 GWSto, 303
 beta
 Filter, 108
 betavec
 GWFastSpinBBH, 218
 betay2
 GWFastSpinBBH, 218
 BigEndian
 ConfigSim, 86
 BinHeader
 MemoryWriteDisk, 340
 C1
 GWFastSpinBBH, 218
 c1

GWNewton2, 279
c1x
 GWNewton2, 279
C2
 GWFastSpinBBH, 218
c2
 GWNewton2, 280
c2psi
 GWFastSpinBBH, 218
c2x
 GWNewton2, 280
C3
 GWFastSpinBBH, 219
c3
 GWNewton2, 280
C4
 GWFastSpinBBH, 219
C5
 GWFastSpinBBH, 219
C6
 GWFastSpinBBH, 219
c_SI
 LISACODE-PhysicConstants.h, 589
cak
 ellipFilter, 23
calcderrivvals
 GWFastSpinBBH, 207
CalcEllipticFilter
 ellipFilter, 23
CalcEllipticFilter4LISACode
 ellipFilter, 24
calcLcrossN
 GWFastSpinBBH, 207
calcLdotN
 GWFastSpinBBH, 207
calcLdotS1
 GWFastSpinBBH, 207
calcLdots2
 GWFastSpinBBH, 207
CalcScalingFact
 ellipFilter, 25
calcSdotsS
 GWFastSpinBBH, 207
CalculDirProp
 GW, 173
 GWBinary, 181
 GWCusp, 192
 GWFastSpinBBH, 208
 GWFile, 232
 GWMono, 243
 GWNew, 253
 GWNewton2, 268
 GWPeriGate, 291
 GWSto, 300
CE_RG
 LISACODE-PhysicConstants.h, 589
CentralTime
 GWCusp, 196
chi1
 GWFastSpinBBH, 219
chi2
 GWFastSpinBBH, 219
child
 ezxml, 99
ci
 GWFastSpinBBH, 219
 GWNewton2, 280
CloseFile
 MemoryWriteDisk, 337
cmass
 GWNewton2, 280
cmu
 Geometry, 119
 GeometryAnalytic, 135
 GeometryFile, 150
 GeometryMLDC, 165
com.c, 485
 advnst, 486
 getsd, 486
 ignlgi, 486
 initgn, 487
 inrgem, 487
 numg, 486
 setall, 487
 setant, 487
 setsd, 487
 Xa1, 487
 Xa1vw, 487
 Xa1w, 488
 Xa2, 488
 Xa2vw, 488
 Xa2w, 488
 Xcg1, 488
 Xcg2, 488
 Xig1, 488
 Xig2, 488
 Xlg1, 488
 Xlg2, 489
 Xm1, 489
 Xm2, 489
 Xqanti, 489
commun
 GWNewton2, 268
Compute
 TDI, 457
ComputeEta
 TDI_InterData, 465
ComputeNoEta

TDI, 457
 ConfigFileName
 ConfigSim, 86
 ConfigSim, 60
 ConfigSim, 68
 ConfigSim
 ~ConfigSim, 68
 AddTimeSeriesInXMLOutput, 68
 Armlength, 86
 Author, 86
 BigEndian, 86
 ConfigFileName, 86
 ConfigSim, 68
 CreateXmlOutputFile, 68
 DefaultConfig, 68
 Endian, 86
 FichXML, 87
 FileEncodingDelays, 87
 FileEncodingPos, 87
 FileEncodingSC1, 87
 FileEncodingSC2, 87
 FileEncodingSC3, 87
 FileEncodingTDI, 87
 FileNameDelays, 87
 FileNamePositions, 88
 FileNameSigSC1, 88
 FileNameSigSC2, 88
 FileNameSigSC3, 88
 FileNameTDI, 88
 FindTDIName, 70
 GenerationDate, 88
 GenerationType, 88
 getArmlength, 70
 getAuthor, 70
 getFileEncodingDelays, 70
 getFileEncodingPositions, 70
 getFileEncodingSig, 71
 getFileEncodingTDI, 71
 getFileNameDelays, 71
 getFileNamePositions, 71
 getFileNameSig, 71
 getFileNameSigSC1, 72
 getFileNameSigSC2, 72
 getFileNameSigSC3, 72
 getFileNameTDI, 72
 getGenerationDate, 72
 getGenerationType, 72
 getGenTDIPacks, 72
 getGenTDIPacksFact, 72
 getGeometry, 73
 getGlobalRandomSeed, 73
 getGW, 73
 getGWB, 73
 getGWs, 73
 getInternalPhasemeters, 74
 getLaserPower, 74
 getNameGenTDI, 74
 getNbMaxDelays, 74
 getNoises, 74
 getNoNoise, 74
 getOrbApprox, 75
 getOrbInitRot, 75
 getOrbOrder, 75
 getOrbStartTime, 75
 getOrbType, 75
 getPhaMetFilter, 75
 getPhaMetFilterParam, 75
 getSC1ParamName, 76
 getSC2ParamName, 76
 getSC3ParamName, 76
 getSimulator, 76
 getSystemEncoding, 76
 gettDeltaTDIDelay, 76
 getTDIDelayApprox, 76
 getTDIInterp, 77
 getTDIInterpUtilVal, 77
 getTDIParamName, 77
 getTDIParamNameType, 77
 gettDisplay, 77
 getTimeOffset, 77
 gettMax, 77
 gettMemNoiseFirst, 78
 gettMemNoiseLast, 78
 gettMemPhasemeters, 78
 gettMemSig, 78
 gettMemTDI, 78
 gettStartPhasemeters, 78
 gettStepMes, 79
 gettStepPhy, 79
 gettTDIShift, 79
 getUSOs, 79
 getXmlOutputFile, 79
 GlobalRandomSeed, 88
 GWB, 89
 GWs, 89
 gXMLAngle, 79
 gXMLAstroDistance, 80
 gXMLAstroMass, 80
 gXMLdouble, 80
 gXMLFrequency, 80
 gXMLLint, 80
 gXMLstring, 81
 gXMLTime, 81
 gXMLTimeSeries, 81
 gXMLUnit, 81
 gXMLWord, 82
 InternalPhasemeters, 89
 LaserPower, 89

majuscule, 82
NbGenTDI, 82
NbMaxDelays, 89
NoisePlace, 82
Noises, 89
NoisesCreation, 82
NoisesData, 89
OrbApprox, 90
OrbInitRot, 90
OrbOrder, 90
OrbStartTime, 90
OrbType, 91
PhaMetFilterON, 91
PhaMetFilterParam, 91
ReadASCIIFile, 83
ReadFile, 83
ReadXMLFile, 84
SC1ParamName, 91
SC2ParamName, 92
SC3ParamName, 92
scmp, 84
Simulator, 92
strcpy, 84
SystemEncoding, 92
SystemEncoding_int, 92
tDeltaTDIDelay, 92
TDIDelayApprox, 92
TDIInterp, 92
TDIInterpUtilVal, 93
TDIP paramName, 93
TDIP paramNameType, 93
TDIsName, 93
TDIsPacks, 93
TDIsPacksFact, 93
tDisplay, 93
testbyteorder, 85
TimeOffset, 94
tMax, 94
tMaxDelay, 85
tMemNecInterpTDI, 85
tMemNoiseFirst, 94
tMemNoiseLast, 94
tMemSig, 94
tMinDelay, 85
tStepMes, 94
tStepPhy, 94
uppercase, 85
upS, 86
UseInternalPhasemeter, 86
USOs, 95
XmlOutputFile, 95
Copy
 Filter, 107
costheta
 GWFastSpinBBH, 219
Couple, 20, 96
 ~Couple, 97
 Couple, 97
 operator *, 97
 operator+, 97
 operator-, 98
 operator/, 98
 x, 98
 y, 98
cphilvec
 GWFastSpinBBH, 219
cphily2
 GWFastSpinBBH, 220
CreateXmlOutputFile
 ConfigSim, 68
crot
 GeometryAnalytic, 135
 GeometryMLDC, 165
CUB
 serie, 35
cur
 ezxml_root, 102
D1
 GWFastSpinBBH, 220
d1
 GWNewton2, 280
d1x
 GWNewton2, 280
D2
 GWFastSpinBBH, 220
d2
 GWNewton2, 280
d2x
 GWNewton2, 281
D3
 GWFastSpinBBH, 220
d3x
 GWNewton2, 281
D4
 GWFastSpinBBH, 220
d4x
 GWNewton2, 281
D5
 GWFastSpinBBH, 220
d5x
 GWNewton2, 281
D6
 GWFastSpinBBH, 220
DefaultConfig
 ConfigSim, 68
deg2rad
 MathUtils, 318

DelayMem
 TDITools, 471
 DelayStore
 Geometry, 119
 GeometryAnalytic, 135
 GeometryFile, 150
 GeometryMLDC, 166
 delLastData
 Serie, 440
 delLastDataC
 SerieC, 449
 deltam
 GWNewton2, 281
 deltanu
 Background, 52
 BackgroundGalactic, 57
 TrFctGW, 473
 DeltaPhase2L
 GWFastSpinBBH, 220
 DerivLinearCoef
 USOClock, 479
 detect
 IM, 18
 LA, 18
 NOISEORIG, 18
 OB, 18
 OP, 18
 PDPMINTERF, 18
 S, 18
 TAU, 18
 Detector (directory Dectecteur), 18
 DirProp
 GW, 176
 GWBinary, 187
 GWCusp, 196
 GWFastSpinBBH, 220
 GWFile, 237
 GMono, 248
 GWNew, 258
 GWNewton2, 281
 GWPeriGate, 295
 GWSto, 303
 DispGeneralInfo
 Geometry, 114
 GeometryAnalytic, 128
 GeometryFile, 144
 GeometryMLDC, 159
 DispInfo
 Geometry, 114
 GeometryAnalytic, 129
 GeometryFile, 145
 GeometryMLDC, 160
 display
 Mat, 316
 Vect, 482
 DisplayStoredData
 PhoDetPhaMet, 426
 DispTempVal
 GW, 173
 GWBinary, 181
 GWCusp, 192
 GWFastSpinBBH, 208
 GWFile, 232
 GMono, 243
 GWNew, 253
 GWNewton2, 269
 GWPeriGate, 291
 GWSto, 300
 DL
 GWFastSpinBBH, 221
 dm
 GWFastSpinBBH, 221
 Doxygen.bibliography, 490
 dx
 Serie, 445
 SerieC, 451
 Dy_SI
 LISACODE-PhysicConstants.h, 589
 e
 ezxml_root, 102
 Geometry, 119
 GeometryAnalytic, 136
 GeometryFile, 150
 GeometryMLDC, 166
 e1
 GWNewton2, 281
 e1x
 GWNewton2, 281
 e2
 GWNewton2, 282
 e2x
 GWNewton2, 282
 e3
 GWNewton2, 282
 e3x
 GWNewton2, 282
 e4
 GWNewton2, 282
 e4x
 GWNewton2, 282
 e5
 GWNewton2, 282
 e5x
 GWNewton2, 282
 e6x
 GWNewton2, 282
 e7x

GWNewton2, 283
e8x
 GWNewton2, 283
e_mldc
 GeometryMLDC, 166
elli
 ellipFilter, 26
ellipFilter
 AbsRespFunctQuadCell, 22
 AbsRespFunctQuadCellChain, 23
 ak, 23
 alog, 22
 alog10, 22
 cak, 23
 CalcEllipticFilter, 23
 CalcEllipticFilter4LISACode, 24
 CalcScalingFact, 25
 elli, 26
 FilterQuadCell, 28
 FilterQuadCellChain, 28
 HmQuadCell, 28
 Im, 28
 KmQuadCell, 28
 OrderCellMaxNorm, 29
 PoleMatching, 29
 sn, 29
 TransfZQuadCell, 30
 TransfZQuadCellChain, 30
Elliptic Filter, 21
Endian
 ConfigSim, 86
ent
 ezxml_root, 102
eps0_SI
 LISACODE-PhysicConstants.h, 590
err
 ezxml_root, 102
Eta
 TDI, 459
 TDI_InterData, 467
eta
 GWFastSpinBBH, 221
etain
 GWFastSpinBBH, 221
exanom
 GeometryAnalytic, 129
 GeometryFile, 145
 GeometryMLDC, 160
expmax
 randlib.c, 611
ezxml, 99
 attr, 99
 child, 99
 flags, 100
name, 100
next, 100
off, 100
ordered, 100
parent, 100
sibling, 101
txt, 101
ezxml.c, 491
 ezxml_add_child, 493
 ezxml_ampencode, 493
 ezxml_attr, 494
 ezxml_char_content, 494
 ezxml_child, 494
 ezxml_close_tag, 494
 ezxml_decode, 494
 ezxml_ent_ok, 494
 ezxml_err, 495
 EZXML_ERRL, 493
 ezxml_error, 495
 ezxml_free, 495
 ezxml_free_attr, 495
 ezxml_get, 495
 ezxml_idx, 495
 ezxml_internal_dtd, 496
 ezxml_new, 496
 EZXML_NIL, 499
 ezxml_open_tag, 496
 ezxml_parse_fd, 496
 ezxml_parse_file, 496
 ezxml_parse_fp, 497
 ezxml_parse_str, 497
 ezxml_pi, 497
 ezxml_proc_inst, 497
 ezxml_remove, 497
 ezxml_root_t, 493
 ezxml_set_attr, 497
 ezxml_set_flag, 498
 ezxml_set_txt, 498
 ezxml_str2utf8, 498
 ezxml_toxml, 498
 ezxml_toxml_r, 498
 ezxml_vget, 498
 EZXML_WS, 493
ezxml.h, 500
 ezxml_add_child, 504
 ezxml_add_child_d, 502
 ezxml_attr, 504
 EZXML_BUFSIZE, 502
 ezxml_child, 504
 EZXML_DUP, 502
 ezxml_error, 504
 ezxml_free, 505
 ezxml_get, 505
 ezxml_idx, 505

ezxml_name, 502
EZXML_NAMEM, 502
ezxml_new, 505
ezxml_new_d, 503
ezxml_next, 503
ezxml_parse_fd, 505
ezxml_parse_file, 506
ezxml_parse_fp, 506
ezxml_parse_str, 506
ezxml_pi, 506
ezxml_remove, 506
ezxml_set_attr, 507
ezxml_set_attr_d, 503
ezxml_set_flag, 507
ezxml_set_txt, 507
ezxml_set_txt_d, 503
ezxml_t, 504
ezxml_toxml, 507
ezxml_txt, 503
EZXML_TXTM, 503
ezxml_add_child
 ezxml.c, 493
 ezxml.h, 504
ezxml_add_child_d
 ezxml.h, 502
ezxml_ampencode
 ezxml.c, 493
ezxml_attr
 ezxml.c, 494
 ezxml.h, 504
EZXML_BUFSIZE
 ezxml.h, 502
ezxml_char_content
 ezxml.c, 494
ezxml_child
 ezxml.c, 494
 ezxml.h, 504
ezxml_close_tag
 ezxml.c, 494
ezxml_decode
 ezxml.c, 494
EZXML_DUP
 ezxml.h, 502
ezxml_ent_ok
 ezxml.c, 494
ezxml_err
 ezxml.c, 495
EZXML_ERRL
 ezxml.c, 493
ezxml_error
 ezxml.c, 495
 ezxml.h, 504
ezxml_free
 ezxml.c, 495
 ezxml.h, 502
ezxml_get
 ezxml.c, 495
 ezxml.h, 505
ezxml_idx
 ezxml.c, 495
 ezxml.h, 505
ezxml_internal_dtd
 ezxml.c, 496
ezxml_name
 ezxml.h, 502
EZXML_NAMEM
 ezxml.h, 502
ezxml_new
 ezxml.c, 496
 ezxml.h, 505
ezxml_new_d
 ezxml.h, 503
ezxml_next
 ezxml.h, 503
EZXML NIL
 ezxml.c, 499
ezxml_open_tag
 ezxml.c, 496
ezxml_parse_fd
 ezxml.c, 496
 ezxml.h, 505
ezxml_parse_file
 ezxml.c, 496
 ezxml.h, 506
ezxml_parse_fp
 ezxml.c, 497
 ezxml.h, 506
ezxml_parse_str
 ezxml.c, 497
 ezxml.h, 506
ezxml_pi
 ezxml.c, 497
 ezxml.h, 506
ezxml_proc_inst
 ezxml.c, 497
ezxml_remove
 ezxml.c, 497
 ezxml.h, 506
ezxml_root, 102
 attr, 102
 cur, 102
 e, 102
 ent, 102
 err, 102
 len, 102
 m, 103

pi, 103
s, 103
standalone, 103
u, 103
xml, 103
ezxml_root_t
 ezxml.c, 493
ezxml_set_attr
 ezxml.c, 497
 ezxml.h, 507
ezxml_set_attr_d
 ezxml.h, 503
ezxml_set_flag
 ezxml.c, 498
 ezxml.h, 507
ezxml_set_txt
 ezxml.c, 498
 ezxml.h, 507
ezxml_set_txt_d
 ezxml.h, 503
ezxml_str2utf8
 ezxml.c, 498
ezxml_t
 ezxml.h, 504
ezxml_toxml
 ezxml.c, 498
 ezxml.h, 507
ezxml_toxml_r
 ezxml.c, 498
ezxml_txt
 ezxml.h, 503
EZXML_TXTM
 ezxml.h, 503
ezxml_vget
 ezxml.c, 498
EZXML_WS
 ezxml.c, 493

f1
 GWNewton2, 283
f10
 GWNewton2, 283
f3
 GWNewton2, 283
f5
 GWNewton2, 283
f6
 GWNewton2, 283
f7
 GWNewton2, 283
f8
 GWNewton2, 283
f9
 GWNewton2, 284

Fac_Hc
 GWSto, 303
Fac Hp
 GWSto, 303
Fac_norm
 GWSto, 303
Fact
 TDI, 459
FactF0
 NoiseFShape, 381
FactFm
 NoiseFShape, 382
FactFp
 NoiseFShape, 382
FactMult
 NoiseFile, 359
fe
 GWNewton2, 269
FEncoding
 MemoryReadDisk, 331
 MemoryWriteDisk, 340
fh
 GWCusp, 196
FichMem
 MemoryReadDisk, 331
 MemoryWriteDisk, 340
FichXML
 ConfigSim, 87
FileEncoding
 GWFile, 237
FileEncodingDelays
 ConfigSim, 87
FileEncodingPos
 ConfigSim, 87
FileEncodingSC1
 ConfigSim, 87
FileEncodingSC2
 ConfigSim, 87
FileEncodingSC3
 ConfigSim, 87
FileEncodingTDI
 ConfigSim, 87
FileName
 GWFile, 237
 NoiseFile, 359
FileNameDelays
 ConfigSim, 87
FileNamePositions
 ConfigSim, 88
FileNameSigSC1
 ConfigSim, 88
FileNameSigSC2
 ConfigSim, 88
FileNameSigSC3
 ConfigSim, 88

ConfigSim, 88
 FileNameTDI
 ConfigSim, 88
 Filter, 31, 104
 ~Filter, 106
 alpha, 108
 App, 107
 beta, 108
 Copy, 107
 Filter, 105, 106
 getAlpha, 107
 getBeta, 107
 getDepth, 107
 getNbDataStab, 108
 init, 108
 NbDataStab, 109
 operator=, 108
 TmpData, 109
 FilterFm
 NoiseFShape, 382
 FilterFp
 NoiseFShape, 382
 FilterON
 PhoDetPhaMet, 431
 FilterParam
 PhoDetPhaMet, 431
 FilterPhyData
 PhoDetPhaMet, 431
 FilterQuadCell
 ellipFilter, 28
 FilterQuadCellChain
 ellipFilter, 28
 FindTDIName
 ConfigSim, 70
 Fknee
 GWSto, 304
 flags
 ezxml, 100
 Flow
 GWCusp, 196
 fmax
 NoiseOof, 394
 FmData
 NoiseFShape, 382
 Fmin
 GWSto, 304
 fmin
 NoiseOof, 394
 forb
 GWBinary, 187
 FpData
 NoiseFShape, 382
 Freq
 GWFastSpinBBH, 208
 GWMono, 248
 GWNew, 258
 GWPeriGate, 295
 fsign
 randlib.c, 611
 fnstop
 randlib.c, 611
 FTRevPlan
 GWCusp, 196
 g1
 GWNewton2, 284
 G_SI
 LISACODE-PhysicConstants.h, 590
 gamma0
 GWFastSpinBBH, 221
 gamma_u
 LISACODE-PhysicConstants.h, 590
 gArmLength
 LISA, 310
 gData
 Memory, 321
 MemoryReadDisk, 328
 MemoryWriteDisk, 337
 Serie, 440
 TDI_InterData, 465, 466
 gDelayT
 LISA, 311
 genbet
 randlib.c, 611
 randlib.h, 616
 genchi
 randlib.c, 612
 randlib.h, 616
 GenerationDate
 ConfigSim, 88
 GenerationType
 ConfigSim, 88
 generNoise
 Noise, 345
 NoiseFile, 354
 NoiseFilter, 367
 NoiseFShape, 377
 NoiseOof, 389
 NoiseTwoFilter, 402
 NoiseWhite, 413
 genexp
 randlib.c, 612
 randlib.h, 617
 genf
 randlib.c, 612
 randlib.h, 617
 gengam
 randlib.c, 612

randlib.h, 617
genmn
 randlib.c, 612
 randlib.h, 617
genmul
 randlib.c, 612
 randlib.h, 617
gennch
 randlib.c, 612
 randlib.h, 617
gennf
 randlib.c, 612
 randlib.h, 617
gennor
 randlib.c, 613
 randlib.h, 618
genprm
 randlib.c, 613
 randlib.h, 618
genunf
 randlib.c, 613
 randlib.h, 618
Geometry, 110
 ~Geometry, 113
 alpha, 119
 ArmVelocity, 113
 arot, 119
 cmu, 119
 DelayStore, 119
 DispGeneralInfo, 114
 DispInfo, 114
 e, 119
 Geometry, 112, 113
 getL0, 114
 gett0, 114
 gettRangeStoreDelay, 114
 gettRangeStorePos, 115
 gposition, 115
 gtdelay, 115
 initGlobal, 115
 L0m, 120
 move, 120
 nu, 120
 order_default, 120
 position, 116
 rot0, 120
 SCposStore, 120
 sett0, 116
 settRangeStoreDelay, 116
 settRangeStorePos, 117
 smu, 120
 sqree, 121
 t0, 121
 tdelay, 117
 tdelayOrderContribution, 117
 tmu, 121
 tRangeStoreDelay, 121
 tRangeStorePos, 121
 tStoreDelay, 121
 tStorePos, 121
 VectNormal, 118
 velocity, 118
 Geometry (directory Orbitographie), 44
 GeometryAnalytic, 123
 GeometryAnalytic, 126, 127
 GeometryAnalytic
 ~GeometryAnalytic, 128
 alpha, 135
 ArmVelocity, 128
 arot, 135
 cmu, 135
 crot, 135
 DelayStore, 135
 DispGeneralInfo, 128
 DispInfo, 129
 e, 136
 exanom, 129
 GeometryAnalytic, 126, 127
 getL0, 129
 gett0, 129
 gettRangeStoreDelay, 129
 gettRangeStorePos, 130
 gposition, 130
 gtdelay, 130
 init, 130
 initGlobal, 131
 L0m, 136
 move, 136
 nu, 136
 order_default, 136
 position, 132
 rot, 137
 rot0, 137
 SCposStore, 137
 sett0, 132
 settRangeStoreDelay, 133
 settRangeStorePos, 133
 smu, 137
 sqree, 137
 srot, 137
 t0, 137
 tdelay, 133
 tdelayOrderContribution, 133
 tmu, 138
 tRangeStoreDelay, 138
 tRangeStorePos, 138
 tStoreDelay, 138
 tStorePos, 138

VectNormal, 134
 velocity, 134
 GeometryFile, 139
 GeometryFile, 142, 143
 GeometryFile
 ~GeometryFile, 144
 alpha, 150
 ArmVelocity, 144
 arot, 150
 cmu, 150
 DelayStore, 150
 DispGeneralInfo, 144
 DispInfo, 145
 e, 150
 exanom, 145
 GeometryFile, 142, 143
 getL0, 145
 gett0, 145
 gettRangeStoreDelay, 145
 gettRangeStorePos, 145
 gposition, 145
 gtdelay, 146
 init, 146
 initGlobal, 146
 InputFileName, 150
 L0m, 151
 move, 151
 NbData, 151
 nu, 151
 order_default, 151
 position, 147
 rot0, 151
 SCposStore, 151
 sett0, 147
 settRangeStoreDelay, 147
 settRangeStorePos, 148
 smu, 152
 sqrtee, 152
 t0, 152
 tdelay, 148
 tdelayOrderContribution, 148
 Time_sec, 152
 tmu, 152
 tRangeStoreDelay, 152
 tRangeStorePos, 152
 tStoreDelay, 153
 tStorePos, 153
 VectNormal, 149
 velocity, 149
 XYZ, 153
 GeometryMLDC, 154
 GeometryMLDC, 157, 158
 GeometryMLDC
 ~GeometryMLDC, 159
 alpha, 165
 Approx, 165
 ArmVelocity, 159
 arot, 165
 cmu, 165
 crot, 165
 DelayStore, 166
 DispGeneralInfo, 159
 DispInfo, 160
 e, 166
 e_mldc, 166
 exanom, 160
 GeometryMLDC, 157, 158
 getL0, 160
 gett0, 160
 gettRangeStoreDelay, 160
 gettRangeStorePos, 160
 gposition, 160
 gtdelay, 161
 init, 161
 initGlobal, 162
 L0m, 166
 move, 166
 nu, 167
 order_default, 167
 position, 162
 rot, 167
 rot0, 167
 SCposStore, 167
 sett0, 163
 settRangeStoreDelay, 163
 settRangeStorePos, 163
 smu, 167
 sqrt_3, 167
 sqrtee, 168
 srot, 168
 t0, 168
 tdelay, 163
 tdelayOrderContribution, 164
 tmu, 168
 tRangeStoreDelay, 168
 tRangeStorePos, 168
 tStoreDelay, 168
 tStorePos, 168
 VectNormal, 164
 velocity, 165
 getAlpha
 Filter, 107
 getAmplhc
 GWMono, 243
 GWNew, 253
 GWPeriGate, 291
 getAmplhp
 GWMono, 243

GWNew, 253
GWPeriGate, 291
getAnglPol
 GW, 173
 GWBinary, 182
 GWCusp, 192
 GWFastSpinBBH, 208
 GWFile, 232
 GWMono, 243, 244
 GWNew, 254
 GWNewton2, 269
 GWPeriGate, 291
 GWSto, 300
getArmlength
 ConfigSim, 70
getAuthor
 ConfigSim, 70
getBeta
 Filter, 107
 GW, 173
 GWBinary, 182
 GWCusp, 192
 GWFastSpinBBH, 208
 GWFile, 233
 GWMono, 244
 GWNew, 254
 GWNewton2, 269
 GWPeriGate, 291
 GWSto, 300
getBinValue
 Serie, 441
getBinValueC
 SerieC, 449
getCountInterDelay
 TDI, 457
getCountInterEta
 TDI, 457
getDelay
 TDITools, 470
getDepth
 Filter, 107
getDeriv
 USOClock, 478
getDirProp
 GW, 173
 GWBinary, 182
 GWCusp, 192
 GWFastSpinBBH, 208
 GWFile, 233
 GWMono, 244
 GWNew, 254
 GWNewton2, 269
 GWPeriGate, 291
 GWSto, 300
getDistance
 GWBinary, 182
 GWFastSpinBBH, 209
 GWNewton2, 269
getFileEncodingDelays
 ConfigSim, 70
getFileEncodingPositions
 ConfigSim, 70
getFileEncodingSig
 ConfigSim, 71
getFileEncodingTDI
 ConfigSim, 71
getFileName
 GWFile, 233
 NoiseFile, 355
getFileNameDelays
 ConfigSim, 71
getFileNamePositions
 ConfigSim, 71
getFileNameSig
 ConfigSim, 71
getFileNameSigSC1
 ConfigSim, 72
getFileNameSigSC2
 ConfigSim, 72
getFileNameSigSC3
 ConfigSim, 72
getFileNameTDI
 ConfigSim, 72
getFilterAlpha
 NoiseFilter, 367
 NoiseOof, 390
 NoiseTwoFilter, 402
getFilterBeta
 NoiseFilter, 367
 NoiseOof, 390
 NoiseTwoFilter, 403
getForb
 GWBinary, 182
getFreq
 GWMono, 244
 GWNew, 254
 GWPeriGate, 292
getGenerationDate
 ConfigSim, 72
getGenerationType
 ConfigSim, 72
getGenTDIPacks
 ConfigSim, 72
getGenTDIPacksFact
 ConfigSim, 72
getGeometry
 ConfigSim, 73
getGlobalRandomSeed

ConfigSim, 73
 getGW
 ConfigSim, 73
 getGWB
 ConfigSim, 73
 getGWS
 ConfigSim, 73
 getInc
 GWBinary, 182
 GWNewton2, 269
 getIndirectDir
 PhoDetPhaMet, 427
 getInternalPhasemeters
 ConfigSim, 74
 getiSC
 PhoDetPhaMet, 427
 getL0
 Geometry, 114
 GeometryAnalytic, 129
 GeometryFile, 145
 GeometryMLDC, 160
 getLambda
 GW, 173
 GWBinary, 182
 GWCusp, 192
 GWFastSpinBBH, 209
 GWFile, 233
 GWMono, 244
 GWNew, 254
 GWNewton2, 270
 GWPeriGate, 292
 GWSto, 300
 getLaserPower
 ConfigSim, 74
 getM1
 GWBinary, 182
 GWNewton2, 270
 getM2
 GWBinary, 183
 GWNewton2, 270
 getMass1
 GWFastSpinBBH, 209
 getMass2
 GWFastSpinBBH, 209
 getNameGenTDI
 ConfigSim, 74
 getNbBinAdd
 Noise, 346
 NoiseFile, 355
 NoiseFilter, 368
 NoiseFShape, 378
 NoiseOof, 390
 NoiseTwoFilter, 403
 NoiseWhite, 414
 getNbDataStab
 Filter, 108
 getNbDataStored
 NoiseFile, 355
 getNbMaxDelays
 ConfigSim, 74
 getNbSerie
 Memory, 322
 MemoryReadDisk, 328
 MemoryWriteDisk, 337
 getNbStored
 GWFile, 233
 getNbVal
 Serie, 441
 getNbValC
 SerieC, 449
 getNoise
 Noise, 346
 NoiseFile, 355, 356
 NoiseFilter, 368
 NoiseFShape, 378
 NoiseOof, 390, 391
 NoiseTwoFilter, 403
 NoiseWhite, 414
 USOClock, 478
 getNoises
 ConfigSim, 74
 getNoNoise
 ConfigSim, 74
 PhoDetPhaMet, 427
 getNParam
 GW, 173
 GWBinary, 183
 GWCusp, 193
 GWFastSpinBBH, 209
 GWFile, 233
 GWMono, 244
 GWNew, 254
 GWNewton2, 270
 GWPeriGate, 292
 GWSto, 300
 getOffset
 USOClock, 478
 getOrbApprox
 ConfigSim, 75
 getOrbInitRot
 ConfigSim, 75
 getOrbOrder
 ConfigSim, 75
 getOrbStartTime
 ConfigSim, 75
 getOrbType
 ConfigSim, 75
 getParam

GW, 174
GWBinary, 183
GWCusp, 193
GWFastSpinBBH, 209
GWFile, 233
GWMono, 244
GWNew, 254
GWNewton2, 270
GWPeriGate, 292
GWSto, 300
getPhaMetFilter
 ConfigSim, 75
getPhaMetFilterParam
 ConfigSim, 75
getPhCoal
 GWNewton2, 270
getPhi0
 GWBinary, 183
getPhi0hc
 GWMono, 244
 GWNew, 255
getPhi0hp
 GWMono, 245
 GWNew, 255
getPSD
 NoiseWhite, 415
getRapidOption
 TDITools, 470
getRef
 Serie, 441
getRefC
 SerieC, 449
getRefStep
 Serie, 441
getRefStepC
 SerieC, 450
getSC1ParamName
 ConfigSim, 76
getSC2ParamName
 ConfigSim, 76
getSC3ParamName
 ConfigSim, 76
getsd
 com.c, 486
 randlib.h, 618
getSimulator
 ConfigSim, 76
getSqPSD
 NoiseWhite, 415
getSystemEncoding
 ConfigSim, 76
gett0
 Geometry, 114
 GeometryAnalytic, 129
 GeometryFile, 145
 GeometryMLDC, 160
gettcoal
 GWNewton2, 270
gettDelayCompute
 TDI_InterData, 466
gettDeltaTDIDelay
 ConfigSim, 76
gettTDIDelayApprox
 ConfigSim, 76
gettTDIInterp
 ConfigSim, 77
gettTDIInterpUtilVal
 ConfigSim, 77
gettTDIParamName
 ConfigSim, 77
gettTDIParamNameType
 ConfigSim, 77
gettDisplay
 ConfigSim, 77
gettDurAdd
 Noise, 347
 NoiseFile, 356
 NoiseFilter, 369
 NoiseFShape, 379
 NoiseOof, 391
 NoiseTwoFilter, 404
 NoiseWhite, 415
gettFirst
 Noise, 347
 NoiseFile, 356
 NoiseFilter, 369
 NoiseFShape, 379
 NoiseOof, 391
 NoiseTwoFilter, 404
 NoiseWhite, 415
getTimeOffset
 ConfigSim, 77
getTimeStore
 TDI_InterData, 466
gettLast
 Noise, 347
 NoiseFile, 356
 NoiseFilter, 369
 NoiseFShape, 379
 NoiseOof, 392
 NoiseTwoFilter, 404
 NoiseWhite, 416
gettMax
 ConfigSim, 77
 Memory, 322
 MemoryReadDisk, 328
 MemoryWriteDisk, 338
gettMemNoiseFirst

ConfigSim, 78
 gettMemNoiseLast
 ConfigSim, 78
 gettMemPhasemeters
 ConfigSim, 78
 gettMemSig
 ConfigSim, 78
 gettMemTDI
 ConfigSim, 78
 gettRangeStoreDelay
 Geometry, 114
 GeometryAnalytic, 129
 GeometryFile, 145
 GeometryMLDC, 160
 gettRangeStorePos
 Geometry, 115
 GeometryAnalytic, 130
 GeometryFile, 145
 GeometryMLDC, 160
 gettStab
 PhoDetPhaMet, 427
 gettStartPhasemeters
 ConfigSim, 78
 gettStep
 Noise, 347
 NoiseFile, 357
 NoiseFilter, 369
 NoiseFShape, 379
 NoiseOof, 392
 NoiseTwoFilter, 404
 NoiseWhite, 416
 TDI_InterData, 466
 gettStepMes
 ConfigSim, 79
 gettStepPhy
 ConfigSim, 79
 gettStepRecord
 Memory, 322
 MemoryReadDisk, 329
 MemoryWriteDisk, 338
 gettStoreData
 Memory, 322
 MemoryReadDisk, 329
 MemoryWriteDisk, 338
 getTDIShift
 ConfigSim, 79
 getUsable
 TDI_InterData, 467
 getUSOs
 ConfigSim, 79
 getXmlOutputFile
 ConfigSim, 79
 gGap
 USOClock, 478
 gGWB
 PhoDetPhaMet, 427
 GlobalRandomSeed
 ConfigSim, 88
 gN
 PhoDetPhaMet, 428
 gposition
 Geometry, 115
 GeometryAnalytic, 130
 GeometryFile, 145
 GeometryMLDC, 160
 gPosSC
 LISA, 311
 Gravitational waves (directory Ondes_Gravit), 37
 gscgn
 randlib.c, 613
 randlib.h, 618
 gsrgs
 randlib.c, 613
 gssst
 randlib.c, 613
 gtdelay
 Geometry, 115
 GeometryAnalytic, 130
 GeometryFile, 146
 GeometryMLDC, 161
 gTime
 USOClock, 479
 GW, 170
 ~GW, 172
 AnglPol, 176
 Beta, 176
 CalculDirProp, 173
 DirProp, 176
 DispTempVal, 173
 getAnglPol, 173
 getBeta, 173
 getDirProp, 173
 getLambda, 173
 getNParam, 173
 getParam, 174
 GW, 172
 hc, 174
 hp, 174
 init, 174
 Lambda, 176
 NParam, 176
 setAnglPol, 174
 setBeta, 175
 setDirProp, 175
 setLambda, 175
 setParam, 175
 gw
 GWNewton2, 284

GWB
 ConfigSim, 89
 LISA, 313
 PhoDetPhaMet, 431
GWBinary, 178
 ~GWBinary, 181
 Ac, 186
 AnglPol, 186
 Ap, 187
 Beta, 187
 CalculDirProp, 181
 DirProp, 187
 DispTempVal, 181
 forb, 187
 getAnglPol, 182
 getBeta, 182
 getDirProp, 182
 getDistance, 182
 getForb, 182
 getInc, 182
 getLambda, 182
 getM1, 182
 getM2, 183
 getNParam, 183
 getParam, 183
 getPhi0, 183
 GWBinary, 180
 hc, 183
 hp, 183
 inc, 187
 init, 184
 Lambda, 187
 M1, 188
 M2, 188
 NParam, 188
 phi0, 188
 r, 188
 setAnglPol, 184
 setBeta, 184
 setDirProp, 184
 setDistance, 185
 setForb, 185
 setInc, 185
 setLambda, 185
 setM1, 185
 setM2, 185
 setParam, 186
 setPhi0, 186
GWCusp, 189
 ~GWCusp, 191
 AllocMem, 192
 Amplitude, 195
 AnglPol, 195
 Beta, 195
 CalculDirProp, 192
 CentralTime, 196
 DirProp, 196
 DispTempVal, 192
 fh, 196
 Flow, 196
 FTRevPlan, 196
 getAnglPol, 192
 getBeta, 192
 getDirProp, 192
 getLambda, 192
 getNParam, 193
 getParam, 193
 GWCusp, 191
 hc, 193
 hp, 193
 init, 193
 initConst, 193
 Lambda, 196
 m43, 197
 MaximumFrequency, 197
 mTPI, 197
 NfDat, 197
 NParam, 197
 NtDat, 197
 setAnglPol, 194
 setBeta, 194
 setDirProp, 194
 setLambda, 194
 setParam, 195
 T0, 197
 Tback, 198
 Tburst, 198
 th, 198
 Tobs, 198
 Tpad, 198
 Tstep, 198
 GWFastSpinBBH, 199
 GWFastSpinBBH, 206
GWFastSpinBBH
 ~GWFastSpinBBH, 206
 A2, 215
 A3, 215
 A4, 215
 A5, 215
 A6, 215
 AllocMemory, 206
 Amp, 215
 AmplPNorder, 215
 AmpShared, 206
 AnglPol, 215
 Ax, 216
 AzimuthalAngleOfSpin1, 216
 AzimuthalAngleOfSpin2, 216

B21, 216
 B31, 216
 B32, 216
 B41, 216
 B42, 216
 B43, 217
 B51, 217
 B52, 217
 B53, 217
 B54, 217
 B61, 217
 B62, 217
 B63, 217
 B64, 217
 B65, 218
 Beta, 218
 betavec, 218
 betay2, 218
 C1, 218
 C2, 218
 c2psi, 218
 C3, 219
 C4, 219
 C5, 219
 C6, 219
 calcdervvals, 207
 calcLcrossN, 207
 calcLdotN, 207
 calcLdotS1, 207
 calcLdotS2, 207
 calcSdotS, 207
 CalculDirProp, 208
 chi1, 219
 chi2, 219
 ci, 219
 costheta, 219
 cphilvec, 219
 cphily2, 220
 D1, 220
 D2, 220
 D3, 220
 D4, 220
 D5, 220
 D6, 220
 DeltaPhase2L, 220
 DirProp, 220
 DispTempVal, 208
 DL, 221
 dm, 221
 eta, 221
 etain, 221
 Freq, 208
 gamma0, 221
 getAnglPol, 208
 getBeta, 208
 getDirProp, 208
 getDistance, 209
 getLambda, 209
 getMass1, 209
 getMass2, 209
 getNParam, 209
 getParam, 209
 GWFastSpinBBH, 206
 hc, 209
 hp, 210
 idxcur, 221
 idxm, 221
 init, 210
 InitialAzimuthalAngleL, 222
 InitialPolarAngleL, 222
 initNULL, 210
 initRK, 210
 krk, 222
 Lambda, 222
 m1, 222
 m2, 222
 Mchirp, 223
 Mtot, 223
 mu, 223
 mulvec, 223
 muly2, 223
 NParam, 223
 p15, 223
 p150, 223
 p20, 224
 p200, 224
 phi, 224
 Phi0, 224
 Phi10, 224
 phic, 224
 PolarAngleOfSpin1, 224
 PolarAngleOfSpin2, 224
 psi, 225
 RK_EPS, 225
 RK_H, 225
 RK_VECLENGTH, 225
 RK_VECLENGTH_Old, 225
 rkckstep, 210
 Rmin, 225
 S1xdot, 211
 S1ydot, 211
 S1zdot, 211
 s2psi, 225
 S2xdot, 211
 S2ydot, 211
 S2zdot, 212
 setAmplPNorder, 212
 setAnglPol, 212

setBeta, 212
setDirProp, 212
setDistance, 213
setLambda, 213
setMass1, 213
setMass2, 213
setParam, 213
sf, 225
shc, 226
shp, 226
sigmavec, 226
sigmay2, 226
sintheta, 226
sphilvec, 226
sphily2, 226
spline, 214
sx, 226
TaperApplied, 227
TaperSteepness, 227
tc, 227
thomasvec, 227
thomasy2, 227
timeCur, 227
timevec, 227
tmax, 227
Tobs, 228
Toffset, 228
update, 214
uspline, 228
xmax, 228
xold, 228
GWFile, 229
~GWFile, 232
AnglPol, 237
Beta, 237
CalculDirProp, 232
DirProp, 237
DispTempVal, 232
FileEncoding, 237
FileName, 237
getAnglPol, 232
getBeta, 233
getDirProp, 233
getFileName, 233
getLambda, 233
getNbStored, 233
getParam, 233
GWFile, 231, 232
hc, 233
hList, 237
hp, 234
init, 234
Interpol, 234
Lambda, 238
LastUsedBin, 238
Length, 238
NbDat, 238
NParam, 238
ReadASCIIFile, 235
ReadBinaryFile, 235
Records, 238
setAnglPol, 235
setBeta, 235
setDirProp, 235
setFileName, 236
setLambda, 236
setParam, 236
TimeList, 238
TimeOffset, 239
TimeStep, 239
GWMono, 240
~GWMono, 243
Amplhc, 247
Amplhp, 247
AnglPol, 248
Beta, 248
CalculDirProp, 243
DirProp, 248
DispTempVal, 243
Freq, 248
getAmplhc, 243
getAmplhp, 243
getAnglPol, 243, 244
getBeta, 244
getDirProp, 244
getFreq, 244
getLambda, 244
getNParam, 244
getParam, 244
getPhi0hc, 244
getPhi0hp, 245
GWMono, 242
hc, 245
hp, 245
init, 245
Lambda, 248
NParam, 249
Phi0hc, 249
Phi0hp, 249
setAmplhc, 245
setAmplhp, 245
setAnglPol, 246
setBeta, 246
setDirProp, 246
setFreq, 246
setLambda, 246
setParam, 247

setPhi0hc, 247
 setPhi0hp, 247
 GWNew, 250
 ~GWNew, 253
 Amplhc, 257
 Amplhp, 257
 AnglPol, 258
 Beta, 258
 CalculDirProp, 253
 DirProp, 258
 DispTempVal, 253
 Freq, 258
 getAmplhc, 253
 getAmplhp, 253
 getAnglPol, 254
 getBeta, 254
 getDirProp, 254
 getFreq, 254
 getLambda, 254
 getNParam, 254
 getParam, 254
 getPhi0hc, 255
 getPhi0hp, 255
 GWNew, 252
 hc, 255
 hp, 255
 init, 255
 Lambda, 258
 NParam, 259
 Phi0hc, 259
 Phi0hp, 259
 setAmplhc, 255
 setAmplhp, 255
 setAnglPol, 256
 setBeta, 256
 setDirProp, 256
 setFreq, 256
 setLambda, 256
 setParam, 257
 setPhi0hc, 257
 setPhi0hp, 257
 GWNewton2, 260
 ~GWNewton2, 268
 a1, 277
 a11, 277
 a1x, 277
 a2, 277
 a22, 278
 a3, 278
 a4, 278
 a5, 278
 a6, 278
 a7, 278
 AnglPol, 278
 b1, 278
 b11, 279
 b1x, 279
 b2, 279
 b3, 279
 b4, 279
 Beta, 279
 c1, 279
 c1x, 279
 c2, 280
 c2x, 280
 c3, 280
 CalculDirProp, 268
 ci, 280
 cmass, 280
 commun, 268
 d1, 280
 d1x, 280
 d2, 280
 d2x, 281
 d3x, 281
 d4x, 281
 d5x, 281
 deltam, 281
 DirProp, 281
 DispTempVal, 269
 e1, 281
 e1x, 281
 e2, 282
 e2x, 282
 e3, 282
 e3x, 282
 e4, 282
 e4x, 282
 e5, 282
 e5x, 282
 e6x, 282
 e7x, 283
 e8x, 283
 f1, 283
 f10, 283
 f3, 283
 f5, 283
 f6, 283
 f7, 283
 f8, 283
 f9, 284
 fe, 269
 g1, 284
 getAnglPol, 269
 getBeta, 269
 getDirProp, 269
 getDistance, 269
 getInc, 269

getLambda, 270
getM1, 270
getM2, 270
getNParam, 270
getParam, 270
getPhCoal, 270
getTcoal, 270
gw, 284
GWNewton2, 267
hc, 270
hint, 284
hp, 271
inc, 284
init, 272
Lambda, 284
lambda, 284
m1, 285
m2, 285
mtot, 285
mu, 285
NParam, 285
nu, 285
omega, 285
omega0, 286
phase, 274
phcoal, 286
phi, 286
psi, 286
rdist, 286
setAnglPol, 274
setBeta, 275
setDirProp, 275
setDistance, 275
setInc, 275
setLambda, 276
setM1, 276
setM2, 276
setParam, 276
setPhCoal, 277
setTcoal, 277
si, 286
taud, 286
taud0, 287
tcoal, 287
teta, 287
time_encour, 287
type, 287
GWPeriGate, 288
 GWPeriGate, 290
GWPeriGate
 ~GWPeriGate, 290
 Amplhc, 295
 Amplhp, 295
 AnglPol, 295
Beta, 295
CalculDirProp, 291
DirProp, 295
DispTempVal, 291
Freq, 295
getAmplhc, 291
getAmplhp, 291
getAnglPol, 291
getBeta, 291
getDirProp, 291
getFreq, 292
getLambda, 292
getNParam, 292
getParam, 292
GWPeriGate, 290
hc, 292
hp, 292
init, 293
Lambda, 295
NParam, 296
setAmplhc, 293
setAmplhp, 293
setAnglPol, 293
setBeta, 293
setDirProp, 293
setFreq, 294
setLambda, 294
setParam, 294
GWs
 ConfigSim, 89
GWSources
 TrFctGW, 474
GWSto, 297
 AnglPol, 303
 Beta, 303
 CalculDirProp, 300
 DirProp, 303
 DispTempVal, 300
 Fac_Hc, 303
 Fac_Hp, 303
 Fac_norm, 303
 Fknee, 304
 Fmin, 304
 getAnglPol, 300
 getBeta, 300
 getDirProp, 300
 getLambda, 300
 getNParam, 300
 getParam, 300
 GWSto, 299
 hc, 301
 hp, 301
 init, 301
 Lambda, 304

Nb_Ageing, 304
NFhc, 304
NFhp, 304
NParam, 304
OrderLa, 305
setAnglPol, 301
setBeta, 301
setDirProp, 302
setLambda, 302
setParam, 302
Slope, 305
tder_hc, 305
tder_hp, 305
tDurAdd, 305
tFirst, 305
tLast, 305
Tsample, 305
tStep, 305
gXMLAngle
ConfigSim, 79
gXMLAstroDistance
ConfigSim, 80
gXMLAstroMass
ConfigSim, 80
gXMLdouble
ConfigSim, 80
gXMLFrequency
ConfigSim, 80
gXMLint
ConfigSim, 80
gXMLstring
ConfigSim, 81
gXMLTime
ConfigSim, 81
gXMLTimeSeries
ConfigSim, 81
gXMLUnit
ConfigSim, 81
gXMLWord
ConfigSim, 82

h
randlib.c, 611

H0_cgs
LISACODE-PhysicConstants.h, 590

h_SI
LISACODE-PhysicConstants.h, 590

hb_SI
LISACODE-PhysicConstants.h, 590

hc
GW, 174
GWBinary, 183
GWCusp, 193
GWFastSpinBBH, 209

GWFile, 233
GWMono, 245
GWNew, 255
GWNewton2, 270
GWPeriGate, 292
GWSto, 301

hint
GWNewton2, 284

hList
GWFile, 237

HmQuadCell
ellipFilter, 28

hp
GW, 174
GWBinary, 183
GWCusp, 193
GWFastSpinBBH, 210
GWFile, 234
GWMono, 245
GWNew, 255
GWNewton2, 271
GWPeriGate, 292
GWSto, 301

idxcur
GWFastSpinBBH, 221

idxm
GWFastSpinBBH, 221

ignbin
randlib.c, 613
randlib.h, 618

ignlgi
com.c, 486
randlib.h, 618

ignnbn
randlib.c, 614
randlib.h, 619

ignpoi
randlib.c, 614
randlib.h, 619

ignuin
randlib.c, 614
randlib.h, 619

IM
detect, 18

Im
ellipFilter, 28

inc
GWBinary, 187
GWNewton2, 284

IndexDelay
TDI, 459

IndexEta
TDI, 459

IndexInReadData
 MemoryReadDisk, 331

IndirectDir
 PhoDetPhaMet, 432

infnty
 randlib.c, 611

init
 Filter, 108
 GeometryAnalytic, 130
 GeometryFile, 146
 GeometryMLDC, 161
 GW, 174
 GWBinary, 184
 GWCusp, 193
 GWFastSpinBBH, 210
 GWFile, 234
 GWMono, 245
 GWNew, 255
 GWNewton2, 272
 GWPeriGate, 293
 GWSto, 301
 PhoDetPhaMet, 428
 TrFctGW, 474
 USOClock, 479

initConst
 GWCusp, 193

initGlobal
 Geometry, 115
 GeometryAnalytic, 131
 GeometryFile, 146
 GeometryMLDC, 162

initgn
 com.c, 487
 randlib.h, 619

InitialAzimuthalAngleL
 GWFastSpinBBH, 222

InitialPolarAngleL
 GWFastSpinBBH, 222

initNULL
 GWFastSpinBBH, 210

initRK
 GWFastSpinBBH, 210

Input data (directory Input_data), 19

InputFileName
 GeometryFile, 150

inrgcm
 com.c, 487

IntegrateSignal
 PhoDetPhaMet, 429

InterCubic
 Serie, 441

InterfPhyData
 PhoDetPhaMet, 432

InterfType

 PhoDetPhaMet, 432

InterHermite
 Serie, 442

InterLagrange
 Serie, 442

InterLinear
 Serie, 443

InternalPhasemeters
 ConfigSim, 89

INTERP
 serie, 35

Interpol
 GWFile, 234

InterpType
 TDI_InterData, 467

InterpUtilValue
 TDI_InterData, 467

iRead
 BackgroundGalactic, 58

iSC
 PhoDetPhaMet, 432

iSerie
 TDI, 459

k
 TrFctGW, 474

k_SI
 LISACODE-PhysicConstants.h, 590

KmQuadCell
 ellipFilter, 28

kpc_m
 LISACODE-PhysicConstants.h, 590

krk
 GWFastSpinBBH, 222

L0_m_default
 LISACODE-LISAConstants.h, 559

L0m
 Geometry, 120
 GeometryAnalytic, 136
 GeometryFile, 151
 GeometryMLDC, 166

LA
 detect, 18

la0Laser_m
 LISACODE-LISAConstants.h, 559

LAG
 serie, 35

Lambda
 GW, 176
 GWBinary, 187
 GWCusp, 196
 GWFastSpinBBH, 222
 GWFile, 238

GWMono, 248
 GWNew, 258
 GWNewton2, 284
 GWPeriGate, 295
 GWSto, 304
 lambda
 GWNewton2, 284
 LaserPower
 ConfigSim, 89
 LaserPower_W_default
 LISACODE-LISAConstants.h, 560
 LastUsedBin
 GWFile, 238
 LCVersion
 LISACODE-LISAConstants.h, 560
 len
 ezxml_root, 102
 Length
 GWFile, 238
 lennob
 randlib.c, 614
 LIN
 serie, 35
 linpack.c, 508
 sdot, 508
 spofa, 508
 LISA, 307
 ~LISA, 310
 gArmLength, 310
 gDelayT, 311
 gPosSC, 311
 GWB, 313
 LISA, 308, 309
 MakeOneStepOfTime, 311
 NoisePointers, 313
 PhotoDetects, 313
 PresentMeanNoise, 312
 RecordPDPM, 313
 SCPos, 313
 sGW, 313
 Stabilization, 312
 tMemRAM, 313
 tStepMes, 314
 tStepPhy, 314
 USOs, 314
 LISA_L0_m
 LISACODE-PhysicConstants.h, 591
 LISA_L0_s
 LISACODE-PhysicConstants.h, 591
 LISACODE-Background.cpp, 509
 LISACODE-Background.h, 510
 LISACODE-BackgroundGalactic.cpp, 511
 LISACODE-BackgroundGalactic.h, 512
 LISACODE-ConfigSim.cpp, 513
 LISACODE-ConfigSim.h, 514
 LISACODE-Couple.cpp, 516
 operator *, 516
 operator+, 516
 operator-, 517
 operator/, 517
 LISACODE-Couple.h, 518
 LISACODE-DnonGW.cpp, 519
 LISACODE-EllipticFilter.cpp, 520
 LISACODE-EllipticFilter.h, 522
 LISACODE-Filter.cpp, 524
 LISACODE-Filter.h, 525
 LISACODE-Geometry.cpp, 526
 LISACODE-Geometry.h, 527
 LISACODE-GeometryAnalytic.cpp, 528
 LISACODE-GeometryAnalytic.h, 529
 LISACODE-GeometryFile.cpp, 530
 LISACODE-GeometryFile.h, 531
 LISACODE-GeometryMLDC.cpp, 532
 LISACODE-GeometryMLDC.h, 533
 LISACODE-GW.cpp, 534
 LISACODE-GW.h, 535
 LISACODE-GWBinary.cpp, 536
 LISACODE-GWBinary.h, 537
 LISACODE-GWCusp.cpp, 538
 LISACODE-GWCusp.h, 539
 LISACODE-GWFastSpinBBH.cpp, 540
 LISACODE-GWFastSpinBBH.h, 541
 LISACODE-GWFile.cpp, 542
 LISACODE-GWFile.h, 543
 LISACODE-GWMono.cpp, 544
 LISACODE-GWMono.h, 545
 LISACODE-GWNew.cpp, 546
 LISACODE-GWNew.h, 547
 LISACODE-GWNewton2.cpp, 548
 LISACODE-GWNewton2.h, 549
 LISACODE-GWPeriGate.cpp, 550
 LISACODE-GWPeriGate.h, 551
 LISACODE-GWSto.cpp, 552
 LISACODE-GWSto.h, 553
 LISACODE-LISA.cpp, 554
 LISACODE-LISA.h, 555
 LISACODE-LISACode.cpp, 556
 main, 557
 LISACODE-LISACode.ep.cpp, 558
 LISACODE-LISAConstants.h, 559
 L0_m_default, 559
 la0Laser_m, 559
 LaserPower_W_default, 560
 LCVersion, 560
 nu0Laser_Hz, 560
 omega, 560
 Rgc, 560
 tRangeStoreDelay_default, 560

tRangeStorePos_default, 560
LISACODE-Mat.cpp, 562
operator *, 562
operator+, 562
operator-, 562
LISACODE-Mat.h, 563
LISACODE-MathUtils.h, 564
LISACODE-Memory.cpp, 565
LISACODE-Memory.h, 566
LISACODE-MemoryReadDisk.cpp, 567
LISACODE-MemoryReadDisk.h, 568
LISACODE-MemoryWriteDisk.cpp, 569
LISACODE-MemoryWriteDisk.h, 570
LISACODE-Noise.cpp, 571
LISACODE-Noise.h, 572
LISACODE-NoiseFile.cpp, 573
LISACODE-NoiseFile.h, 574
LISACODE-NoiseFilter.cpp, 575
LISACODE-NoiseFilter.h, 576
LISACODE-NoiseFShape.cpp, 577
LISACODE-NoiseFShape.h, 578
LISACODE-NoiseOof.cpp, 579
LISACODE-NoiseOof.h, 580
LISACODE-NoiseTwoFilter.cpp, 581
LISACODE-NoiseTwoFilter.h, 582
LISACODE-NoiseWhite.cpp, 583
LISACODE-NoiseWhite.h, 584
LISACODE-PhoDetPhaMet.cpp, 585
LISACODE-PhoDetPhaMet.h, 586
LISACODE-PhysicConstants.h, 587
LISACODE-PhysicConstants.h
 au_m, 589
 au_s, 589
 c_SI, 589
 CE_RG, 589
 Dy_SI, 589
 eps0_SI, 590
 G_SI, 590
 gamma_u, 590
 H0_cgs, 590
 h_SI, 590
 hb_SI, 590
 k_SI, 590
 kpc_m, 590
 LISA_L0_m, 591
 LISA_L0_s, 591
 LS_SI, 591
 ly_au, 591
 ly_m, 591
 me_SI, 591
 mn_SI, 591
 mp_SI, 591
 MS_SI, 592
 mu0_SI, 592
 Na_SI, 592
 omegaYr, 592
 pc_au, 592
 pc_ly, 592
 pc_m, 592
 PRECISION, 592
 RS_SI, 593
 RSchw, 593
 S_SI, 593
 TSUN, 593
 Yr_SI, 593
LISACODE-Serie.cpp, 594
LISACODE-Serie.h, 595
LISACODE-TDI.cpp, 596
LISACODE-TDI.h, 597
LISACODE-TDI_InterData.cpp, 598
LISACODE-TDI_InterData.h, 599
LISACODE-TDIAppl.cpp, 600
 main, 600
LISACODE-TDITools.cpp, 601
LISACODE-TDITools.h, 602
LISACODE-TrFctGW.cpp, 603
LISACODE-TrFctGW.h, 604
LISACODE-USOClock.cpp, 605
LISACODE-USOClock.h, 606
LISACODE-Vect.cpp, 607
 operator *, 607
 operator+, 608
 operator-, 608
 operator/, 608
LISACODE-Vect.h, 609
LISAGeo
 Background, 53
 BackgroundGalactic, 58
 TrFctGW, 475
ListTmpData
 Memory, 324
 MemoryReadDisk, 331
 MemoryWriteDisk, 340
loadNoise
 Noise, 347
 NoiseFile, 357
 NoiseFilter, 369
 NoiseFShape, 380
 NoiseOof, 392
 NoiseTwoFilter, 404
 NoiseWhite, 416
LS_SI
 LISACODE-PhysicConstants.h, 591
ly_au
 LISACODE-PhysicConstants.h, 591
ly_m
 LISACODE-PhysicConstants.h, 591

m
ezxml_root, 103

M1
GWBinary, 188

m1
GWFastSpinBBH, 222
GWNewton2, 285

M2
GWBinary, 188

m2
GWFastSpinBBH, 222
GWNewton2, 285

m43
GWCusp, 197

main
LISACODE-LISACode.cpp, 557
LISACODE-TDIApply.cpp, 600
main, 45

Main (directory Main), 45

majuscule
ConfigSim, 82

MakeOneStepOfTime
LISA, 311

MakeTitles
Memory, 322
MemoryReadDisk, 329
MemoryWriteDisk, 338

Mat, 315
~Mat, 316
display, 316
Mat, 316
operator *, 316
operator+, 316
operator-, 317
p, 317

Mathematical Tools (directory Outils_Math), 33

MathUtils, 318

MathUtils
deg2rad, 318
rad2deg, 318
TimeISO8601, 318

mathUtils
MAX, 34
MIN, 34
SWAP, 34

Matrix, 32

MAX
mathUtils, 34

max
randlib.c, 611

MaximumFrequency
GWCusp, 197

maxnum
randlib.c, 611

Mchirp
GWFastSpinBBH, 223

me_SI
LISACODE-PhysicConstants.h, 591

Memory, 319
~Memory, 321
AddSerieData, 321
AlreadyRecDat, 324
gData, 321
getNbSerie, 322
gettMax, 322
gettStepRecord, 322
gettStoreData, 322
ListTmpData, 324
MakeTitles, 322
Memory, 320
ReceiveData, 322
RecordAccData, 323
settStepRecord, 323
settStoreData, 323
tStepRecord, 324
tStoreData, 324
unusable, 323

Memory (directory Memoire), 47

MemoryReadDisk, 325
MemoryReadDisk, 327

MemoryReadDisk
~MemoryReadDisk, 327
AddSerieData, 328
AlreadyRecDat, 331
FEncoding, 331
FichMem, 331
gData, 328
getNbSerie, 328
gettMax, 328
gettStepRecord, 329
gettStoreData, 329
IndexInReadData, 331
ListTmpData, 331
MakeTitles, 329
MemoryReadDisk, 327
NomFichMem, 331
ReadASCIIFile, 329
ReadBinaryFile, 329
ReadData, 331
ReceiveData, 329
RecordAccData, 330
settStepRecord, 330
settStoreData, 330
TitlesReadData, 332
tStepRecord, 332
tStoreData, 332
unusable, 330

MemoryWriteDisk, 333

MemoryWriteDisk, 335, 336
MemoryWriteDisk
 ~MemoryWriteDisk, 336
 AddSerieData, 337
 AlreadyRecDat, 340
 BinHeader, 340
 CloseFile, 337
 FEncoding, 340
 FichMem, 340
 gData, 337
 getNbSerie, 337
 gettMax, 338
 gettStepRecord, 338
 gettStoreData, 338
 ListTmpData, 340
 MakeTitles, 338
 MemoryWriteDisk, 335, 336
 NomFichMem, 340
 ReceiveData, 338
 RecordAccData, 339
 SCSerie, 340
 settStepRecord, 339
 settStoreData, 339
 TimeEnd, 341
 TimeOffset, 341
 TitleSerie, 341
 tStepRecord, 341
 tStoreData, 341
 unusable, 339

MIN
 mathUtils, 34

min
 randlib.c, 611

minlog
 randlib.c, 611

mltmod
 randlib.c, 614
 randlib.h, 619

mn_SI
 LISACODE-PhysicConstants.h, 591

move
 Geometry, 120
 GeometryAnalytic, 136
 GeometryFile, 151
 GeometryMLDC, 166

mp_SI
 LISACODE-PhysicConstants.h, 591

MS_SI
 LISACODE-PhysicConstants.h, 592

Mtot
 GWFastSpinBBH, 223

mtot
 GWNewton2, 285

mTPI

mu
 GWCusp, 197
 GWFastSpinBBH, 223
 GWNewton2, 285

mu0_SI
 LISACODE-PhysicConstants.h, 592

mulvec
 GWFastSpinBBH, 223

muly2
 GWFastSpinBBH, 223

Na_SI
 LISACODE-PhysicConstants.h, 592

name
 ezxml, 100

Nb_Ageing
 GWSto, 304
 NoiseOof, 394

NbBinAdd
 Noise, 349
 NoiseFile, 359
 NoiseFilter, 371
 NoiseFShape, 382
 NoiseOof, 394
 NoiseTwoFilter, 406
 NoiseWhite, 418

NbDat
 GWFile, 238

NbData
 BackgroundGalactic, 58
 GeometryFile, 151
 Noise, 349
 NoiseFile, 359
 NoiseFilter, 371
 NoiseFShape, 382
 NoiseOof, 394
 NoiseTwoFilter, 406
 NoiseWhite, 418

NbDataAdd
 PhoDetPhaMet, 432

NbDataStab
 Filter, 109

NbDataStored
 NoiseFile, 359
 PhoDetPhaMet, 432

NbDelayMax
 TDI, 458

NbGenTDI
 ConfigSim, 82

NbMaxDelays
 ConfigSim, 89

next
 ezxml, 100

NfDat

GWCusp, 197
 NFhc
 GWSto, 304
 NFhp
 GWSto, 304
 NFilter
 NoiseFilter, 372
 NoiseOof, 395
 NoiseTwoFilter, 407
 NFilter_2
 NoiseTwoFilter, 407
 NFm
 NoiseFShape, 383
 NFp
 NoiseFShape, 383
 Noise, 342
 ~Noise, 345
 addNoise, 345
 generNoise, 345
 getNbBinAdd, 346
 getNoise, 346
 gettDurAdd, 347
 gettFirst, 347
 gettLast, 347
 gettStep, 347
 loadNoise, 347
 NbBinAdd, 349
 NbData, 349
 Noise, 344
 NoiseData, 349
 NoiseType, 349
 settDurAdd, 347
 settFirst, 348
 settLast, 348
 settStep, 348
 tDurAdd, 350
 TestType, 349
 tFirst, 350
 tLast, 350
 tStep, 350
 Noise (directory Bruits), 17
 NoiseData
 Noise, 349
 NoiseFile, 359
 NoiseFilter, 372
 NoiseFShape, 383
 NoiseOof, 395
 NoiseTwoFilter, 407
 NoiseWhite, 418
 NoiseData_tmp1
 NoiseTwoFilter, 407
 NoiseData_tmp2
 NoiseTwoFilter, 407
 NoiseFile, 351
 NoiseFile, 353, 354
 NoiseFile
 addNoise, 354
 FactMult, 359
 FileName, 359
 generNoise, 354
 getFileName, 355
 getNbBinAdd, 355
 getNbDataStored, 355
 getNoise, 355, 356
 gettDurAdd, 356
 gettFirst, 356
 gettLast, 356
 gettStep, 357
 loadNoise, 357
 NbBinAdd, 359
 NbData, 359
 NbDataStored, 359
 NoiseData, 359
 NoiseFile, 353, 354
 NoiseType, 360
 ReadBin, 360
 setFileName, 357
 settDurAdd, 357
 settFirst, 357
 settLast, 358
 settStep, 358
 StoredData, 360
 tDurAdd, 360
 TestType, 358
 tFirst, 360
 tLast, 360
 tStep, 361
 NoiseFilter, 362
 NoiseFilter, 364–366
 NoiseFilter
 addNoise, 367
 generNoise, 367
 getFilterAlpha, 367
 getFilterBeta, 367
 getNbBinAdd, 368
 getNoise, 368
 gettDurAdd, 369
 gettFirst, 369
 gettLast, 369
 gettStep, 369
 loadNoise, 369
 NbBinAdd, 371
 NbData, 371
 NFilter, 372
 NoiseData, 372
 NoiseFilter, 364–366
 NoiseType, 372
 settDurAdd, 370

settFirst, 370
settLast, 370
settStep, 371
tDurAdd, 372
TestType, 371
tFirst, 372
tLast, 372
tStep, 373
WhiteData, 373
NoiseFShape, 374
 NoiseFShape, 376, 377
NoiseFShape
 addNoise, 377
 FactF0, 381
 FactFm, 382
 FactFp, 382
 FilterFm, 382
 FilterFp, 382
 FmData, 382
 FpData, 382
 generNoise, 377
 getNbBinAdd, 378
 getNoise, 378
 gettDurAdd, 379
 gettFirst, 379
 gettLast, 379
 gettStep, 379
 loadNoise, 380
 NbBinAdd, 382
 NbData, 382
 NFm, 383
 NFp, 383
 NoiseData, 383
 NoiseFShape, 376, 377
 NoiseType, 383
 settDurAdd, 380
 settFirst, 380
 settLast, 381
 settStep, 381
 tDurAdd, 383
 TestType, 381
 tFirst, 384
 tLast, 384
 tmpData_m, 384
 tmpData_p, 384
 tStep, 384
 WhiteData, 384
NoiseOof, 386
 NoiseOof, 388, 389
NoiseOof
 addNoise, 389
 alpha, 394
 fmax, 394
 fmin, 394
 generNoise, 389
 getFilterAlpha, 390
 getFilterBeta, 390
 getNbBinAdd, 390
 getNoise, 390, 391
 gettDurAdd, 391
 gettFirst, 391
 gettLast, 392
 gettStep, 392
 loadNoise, 392
 Nb_Ageing, 394
 NbBinAdd, 394
 NbData, 394
 NFfilter, 395
 NoiseData, 395
 NoiseOof, 388, 389
 NoiseType, 395
 settDurAdd, 392
 settFirst, 393
 settLast, 393
 settStep, 393
 tDurAdd, 395
 TestType, 393
 tFirst, 395
 tLast, 396
 tStep, 396
 WhiteData, 396
NOISEORIG
 detect, 18
NoisePlace
 ConfigSim, 82
NoisePointers
 LISA, 313
Noises
 ConfigSim, 89
NoisesCreation
 ConfigSim, 82
NoisesData
 ConfigSim, 89
NoiseSpec, 397
NoiseSpec
 NStr, 397
 NType, 397
 NVal0, 398
 NVal01, 398
 NVal1, 398
 NVal2, 398
NoiseTwoFilter, 399
 NoiseTwoFilter, 401
NoiseTwoFilter
 addNoise, 402
 generNoise, 402
 getFilterAlpha, 402
 getFilterBeta, 403

getNbBinAdd, 403
 getNoise, 403
 gettDurAdd, 404
 gettFirst, 404
 gettLast, 404
 gettStep, 404
 loadNoise, 404
 NbBinAdd, 406
 NbData, 406
 NFilter, 407
 NFilter_2, 407
 NoiseData, 407
 NoiseData_tmp1, 407
 NoiseData_tmp2, 407
 NoiseTwoFilter, 401
 NoiseType, 407
 settDurAdd, 405
 settFirst, 405
 settLast, 406
 settStep, 406
 tDurAdd, 408
 TestType, 406
 tFirst, 408
 tLast, 408
 tStep, 408
 WhiteData, 408
 WhiteData2, 408
 NoiseType
 Noise, 349
 NoiseFile, 360
 NoiseFilter, 372
 NoiseFShape, 383
 NoiseOof, 395
 NoiseTwoFilter, 407
 NoiseWhite, 418
 NoiseWhite, 410
 NoiseWhite, 412, 413
 NoiseWhite
 addNoise, 413
 generNoise, 413
 getNbBinAdd, 414
 getNoise, 414
 getPSD, 415
 getSqPSD, 415
 gettDurAdd, 415
 gettFirst, 415
 gettLast, 416
 gettStep, 416
 loadNoise, 416
 NbBinAdd, 418
 NbData, 418
 NoiseData, 418
 NoiseType, 418
 NoiseWhite, 412, 413
 setSqPSD, 416
 settDurAdd, 416
 settFirst, 417
 settLast, 417
 settStep, 417
 Sigma, 419
 tDurAdd, 419
 TestType, 418
 tFirst, 419
 tLast, 419
 tStep, 419
 NomFichMem
 MemoryReadDisk, 331
 MemoryWriteDisk, 340
 NoNoise
 PhoDetPhaMet, 432
 TDI_InterData, 467
 norme
 Vect, 482
 NParam
 GW, 176
 GWBinary, 188
 GWCusp, 197
 GWFastSpinBBH, 223
 GWFile, 238
 GWMono, 249
 GWNew, 259
 GWNewton2, 285
 GWPeriGate, 296
 GWSto, 304
 NPs
 PhoDetPhaMet, 433
 NStr
 NoiseSpec, 397
 NtDat
 GWCusp, 197
 NType
 NoiseSpec, 397
 nu
 Geometry, 120
 GeometryAnalytic, 136
 GeometryFile, 151
 GeometryMLDC, 167
 GWNewton2, 285
 nu0Laser_Hz
 LISACODE-LISAConstants.h, 560
 numg
 com.c, 486
 randlib.c, 611
 NVal0
 NoiseSpec, 398
 NVal01
 NoiseSpec, 398
 NVal1

NoiseSpec, 398
NVal2
 NoiseSpec, 398

OB
 detect, 18
off
 ezxml, 100
Offset
 USOClock, 479
omega
 GWNewton2, 285
 LISACODE-LISAConstants.h, 560
omega0
 GWNewton2, 286
omegaYr
 LISACODE-PhysicConstants.h, 592
OP
 detect, 18
operator *
 Couple, 97
 LISACODE-Couple.cpp, 516
 LISACODE-Mat.cpp, 562
 LISACODE-Vect.cpp, 607
 Mat, 316
 Vect, 483
operator+
 Couple, 97
 LISACODE-Couple.cpp, 516
 LISACODE-Mat.cpp, 562
 LISACODE-Vect.cpp, 608
 Mat, 316
 Vect, 483
operator-
 Couple, 98
 LISACODE-Couple.cpp, 517
 LISACODE-Mat.cpp, 562
 LISACODE-Vect.cpp, 608
 Mat, 317
 Vect, 484
operator/
 Couple, 98
 LISACODE-Couple.cpp, 517
 LISACODE-Vect.cpp, 608
 Vect, 484
operator=

 Filter, 108
OrbApprox
 ConfigSim, 90
OrbInitRot
 ConfigSim, 90
OrbOrder
 ConfigSim, 90
OrbStartTime

 ConfigSim, 90
 OrbType
 ConfigSim, 91
order_default
 Geometry, 120
 GeometryAnalytic, 136
 GeometryFile, 151
 GeometryMLDC, 167
OrderCellMaxNorm
 ellipFilter, 29
ordered
 ezxml, 100
OrderLa
 GWSto, 305
OutFile
 TDI, 459
OutFileEncoding
 TDI, 460

p

 Mat, 317
 Vect, 484

p15
 GWFastSpinBBH, 223

p150
 GWFastSpinBBH, 223

p20
 GWFastSpinBBH, 224

p200
 GWFastSpinBBH, 224

parent
 ezxml, 100

PBFilter
 PhoDetPhaMet, 433

pc_au
 LISACODE-PhysicConstants.h, 592

pc_ly
 LISACODE-PhysicConstants.h, 592

pc_m
 LISACODE-PhysicConstants.h, 592

PDPMINTERF
 detect, 18

PDPMMem
 TDI_InterData, 467

PhaMetFilterON
 ConfigSim, 91

PhaMetFilterParam
 ConfigSim, 91

phase
 GWNewton2, 274

phcoal
 GWNewton2, 286

phi
 GWFastSpinBBH, 224

GWNewton2, 286
 Phi0
 GWFastSpinBBH, 224
 phi0
 GWBinary, 188
 Phi0hc
 GWMono, 249
 GWNew, 259
 Phi0hp
 GWMono, 249
 GWNew, 259
 Phi10
 GWFastSpinBBH, 224
 phic
 GWFastSpinBBH, 224
 PhoDetPhaMet, 421
 PhoDetPhaMet, 423–425
 PhoDetPhaMet
 ~PhoDetPhaMet, 426
 DisplayStoredData, 426
 FilterON, 431
 FilterParam, 431
 FilterPhyData, 431
 getIndirectDir, 427
 getISc, 427
 getNoNoise, 427
 gettStab, 427
 gGWB, 427
 gN, 428
 GWB, 431
 IndirectDir, 432
 init, 428
 IntegrateSignal, 429
 InterfPhyData, 432
 InterfType, 432
 iSC, 432
 NbDataAdd, 432
 NbDataStored, 432
 NoNoise, 432
 NPs, 433
 PBFILTER, 433
 PhoDetPhaMet, 423–425
 ReceiveSignal, 430
 RecordData, 433
 SCPos, 433
 sGW, 433
 tStepMes, 433
 tStepPhy, 433
 USO, 434
 PhotoDetects
 LISA, 313
 phrtsd
 randlib.c, 614
 randlib.h, 619
 pi
 ezxml_root, 103
 PolarAngleOfSpin1
 GWFastSpinBBH, 224
 PolarAngleOfSpin2
 GWFastSpinBBH, 224
 pole
 QuadCell, 436
 PoleMatching
 ellipFilter, 29
 position
 Geometry, 116
 GeometryAnalytic, 132
 GeometryFile, 147
 GeometryMLDC, 162
 PRECISION
 LISACODE-PhysicConstants.h, 592
 PresentMeanNoise
 LISA, 312
 psi
 GWFastSpinBBH, 225
 GWNewton2, 286
 QuadCell, 435
 QuadCell
 a0, 435
 a1, 435
 b1, 435
 b2, 436
 pole, 436
 u, 436
 zero, 436
 r
 GWBinary, 188
 rad2deg
 MathUtils, 318
 randlib.c, 610
 ABS, 611
 expmax, 611
 fsign, 611
 ftnstop, 611
 genbet, 611
 genchi, 612
 genexp, 612
 genf, 612
 gengam, 612
 genmn, 612
 genmul, 612
 gennch, 612
 gennf, 612
 gennor, 613
 genprm, 613
 genunf, 613

gscgn, 613
gsrgs, 613
gssst, 613
h, 611
ignbin, 613
ignnbn, 614
ignpoi, 614
ignuin, 614
infnty, 611
lennob, 614
max, 611
maxnum, 611
min, 611
minlog, 611
mltmod, 614
numg, 611
phrtsd, 614
ranf, 614
setgmn, 614
sexpo, 615
sgamma, 615
snorm, 615
randlib.h, 616
advnst, 616
genbet, 616
genchi, 616
genexp, 617
genf, 617
gengam, 617
genmn, 617
genmul, 617
gennch, 617
gennf, 617
gennor, 618
genprm, 618
genunf, 618
getsd, 618
gscgn, 618
ignbin, 618
ignlgi, 618
ignnbn, 619
ignpoi, 619
ignuin, 619
initgn, 619
mltmod, 619
phrtsd, 619
ranf, 619
setall, 620
setant, 620
setgmn, 620
setsd, 620
sexpo, 620
sgamma, 620
snorm, 620
ranf
 randlib.c, 614
 randlib.h, 619
RapidOption
 TDITools, 471
rdist
 GWNewton2, 286
ReadASCIIFile
 ConfigSim, 83
 GWFile, 235
 MemoryReadDisk, 329
ReadBin
 NoiseFile, 360
ReadBinaryFile
 GWFile, 235
 MemoryReadDisk, 329
ReadData
 MemoryReadDisk, 331
ReadFile
 BackgroundGalactic, 57
 ConfigSim, 83
ReadSignEtaDelays
 TDI, 458
ReadXMLFile
 ConfigSim, 84
ReceiveData
 Memory, 322
 MemoryReadDisk, 329
 MemoryWriteDisk, 338
ReceiveSignal
 PhoDetPhaMet, 430
RecordAccData
 Memory, 323
 MemoryReadDisk, 330
 MemoryWriteDisk, 339
RecordAndReturnResult
 TDI, 458
RecordData
 PhoDetPhaMet, 433
RecordPDPM
 LISA, 313
Records
 GWFile, 238
RefreshDelay
 TDITools, 470
rfile
 Serie, 443
rfileC
 SerieC, 450
Rgc
 LISACODE-LISAConstants.h, 560
RK_EPS
 GWFastSpinBBH, 225
RK_H

GWFastSpinBBH, 225
 RK_VECLENGTH
 GWFastSpinBBH, 225
 RK_VECLENGTH_Old
 GWFastSpinBBH, 225
 rkckstep
 GWFastSpinBBH, 210
 Rmin
 GWFastSpinBBH, 225
 rot
 GeometryAnalytic, 137
 GeometryMLDC, 167
 rot0
 Geometry, 120
 GeometryAnalytic, 137
 GeometryFile, 151
 GeometryMLDC, 167
 RS_SI
 LISACODE-PhysicConstants.h, 593
 RSchw
 LISACODE-PhysicConstants.h, 593

 S
 detect, 18
 s
 ezxml_root, 103
 S1xdot
 GWFastSpinBBH, 211
 S1ydot
 GWFastSpinBBH, 211
 S1zdot
 GWFastSpinBBH, 211
 s2psi
 GWFastSpinBBH, 225
 S2xdot
 GWFastSpinBBH, 211
 S2ydot
 GWFastSpinBBH, 211
 S2zdot
 GWFastSpinBBH, 212
 S_SI
 LISACODE-PhysicConstants.h, 593
 SC1ParamName
 ConfigSim, 91
 SC2ParamName
 ConfigSim, 92
 SC3ParamName
 ConfigSim, 92
 scmp
 ConfigSim, 84
 SCPos
 LISA, 313
 PhoDetPhaMet, 433
 SCposStore

Geometry, 120
 GeometryAnalytic, 137
 GeometryFile, 151
 GeometryMLDC, 167
 SC Serie
 MemoryWriteDisk, 340
 SCSig
 TDI, 460
 sdot
 linpack.c, 508
 Serie, 35, 437
 ~Serie, 440
 addData, 440
 delLastData, 440
 dx, 445
 gData, 440
 getBinValue, 441
 getNbVal, 441
 getRef, 441
 getRefStep, 441
 InterCubic, 441
 InterHermite, 442
 InterLagrange, 442
 InterLinear, 443
 rfile, 443
 Serie, 439
 setBinValue, 443
 setNbVal, 444
 setRefStart, 444
 setRefStep, 444
 TruncVal, 444
 wfile, 444
 x0, 445
 ys, 445
 serie
 CUB, 35
 INTERP, 35
 LAG, 35
 LIN, 35
 SIN, 35
 TRU, 35
 SerieC, 446
 SerieC, 447, 448
 SerieC
 ~SerieC, 448
 addDataC, 449
 delLastDataC, 449
 dx, 451
 getBinValueC, 449
 getNbValC, 449
 getRefC, 449
 getRefStepC, 450
 rfileC, 450
 SerieC, 447, 448

setBinValueC, 450
setNbValC, 450
setRefStartC, 450
setRefStepC, 450
wfileC, 451
x0, 451
ys, 451
setall
 com.c, 487
 randlib.h, 620
setAmplhc
 GWMono, 245
 GWNew, 255
 GWPeriGate, 293
setAmplhp
 GWMono, 245
 GWNew, 255
 GWPeriGate, 293
setAmplPNorder
 GWFastSpinBBH, 212
setAngIPol
 GW, 174
 GWBinary, 184
 GWCusp, 194
 GWFastSpinBBH, 212
 GWFile, 235
 GWMono, 246
 GWNew, 256
 GWNewton2, 274
 GWPeriGate, 293
 GWSto, 301
setant
 com.c, 487
 randlib.h, 620
setBeta
 GW, 175
 GWBinary, 184
 GWCusp, 194
 GWFastSpinBBH, 212
 GWFile, 235
 GWMono, 246
 GWNew, 256
 GWNewton2, 275
 GWPeriGate, 293
 GWSto, 301
setBinValue
 Serie, 443
setBinValueC
 SerieC, 450
setDirProp
 GW, 175
 GWBinary, 184
 GWCusp, 194
 GWFastSpinBBH, 212
 GWFile, 235
 GWMono, 246
 GWNew, 256
 GWNewton2, 275
 GWPeriGate, 293
 GWSto, 302
setDistance
 GWBinary, 185
 GWFastSpinBBH, 213
 GWNewton2, 275
setFileName
 GWFile, 236
 NoiseFile, 357
setForb
 GWBinary, 185
setFreq
 GWMono, 246
 GWNew, 256
 GWPeriGate, 294
setGeometry
 Background, 52
 BackgroundGalactic, 57
setgmn
 randlib.c, 614
 randlib.h, 620
setInc
 GWBinary, 185
 GWNewton2, 275
setLambda
 GW, 175
 GWBinary, 185
 GWCusp, 194
 GWFastSpinBBH, 213
 GWFile, 236
 GWMono, 246
 GWNew, 256
 GWNewton2, 276
 GWPeriGate, 294
 GWSto, 302
setM1
 GWBinary, 185
 GWNewton2, 276
setM2
 GWBinary, 185
 GWNewton2, 276
setMass1
 GWFastSpinBBH, 213
setMass2
 GWFastSpinBBH, 213
setNbVal
 Serie, 444
setNbValC
 SerieC, 450
setParam

GW, 175
 GWBinary, 186
 GWCusp, 195
 GWFastSpinBBH, 213
 GWFile, 236
 GWMono, 247
 GWNew, 257
 GWNewton2, 276
 GWPeriGate, 294
 GWSto, 302
 setPhCoal
 GWNewton2, 277
 setPhi0
 GWBinary, 186
 setPhi0hc
 GWMono, 247
 GWNew, 257
 setPhi0hp
 GWMono, 247
 GWNew, 257
 setRefStart
 Serie, 444
 setRefStartC
 SerieC, 450
 setRefStep
 Serie, 444
 setRefStepC
 SerieC, 450
 setsd
 com.c, 487
 randlib.h, 620
 setSqPSD
 NoiseWhite, 416
 sett0
 Geometry, 116
 GeometryAnalytic, 132
 GeometryFile, 147
 GeometryMLDC, 163
 setTcoal
 GWNewton2, 277
 settDurAdd
 Noise, 347
 NoiseFile, 357
 NoiseFilter, 370
 NoiseFShape, 380
 NoiseOof, 392
 NoiseTwoFilter, 405
 NoiseWhite, 416
 settFirst
 Noise, 348
 NoiseFile, 357
 NoiseFilter, 370
 NoiseFShape, 380
 NoiseOof, 393
 NoiseTwoFilter, 405
 NoiseWhite, 417
 setTimeStore
 TDI_InterData, 467
 settLast
 Noise, 348
 NoiseFile, 358
 NoiseFilter, 370
 NoiseFShape, 381
 NoiseOof, 393
 NoiseTwoFilter, 406
 NoiseWhite, 417
 settRangeStoreDelay
 Geometry, 116
 GeometryAnalytic, 133
 GeometryFile, 147
 GeometryMLDC, 163
 settRangeStorePos
 Geometry, 117
 GeometryAnalytic, 133
 GeometryFile, 148
 GeometryMLDC, 163
 settStep
 Noise, 348
 NoiseFile, 358
 NoiseFilter, 371
 NoiseFShape, 381
 NoiseOof, 393
 NoiseTwoFilter, 406
 NoiseWhite, 417
 settStepRecord
 Memory, 323
 MemoryReadDisk, 330
 MemoryWriteDisk, 339
 settStoreData
 Memory, 323
 MemoryReadDisk, 330
 MemoryWriteDisk, 339
 expo
 randlib.c, 615
 randlib.h, 620
 sf
 GWFastSpinBBH, 225
 sgamma
 randlib.c, 615
 randlib.h, 620
 sGW
 LISA, 313
 PhoDetPhaMet, 433
 shc
 GWFastSpinBBH, 226
 shp
 GWFastSpinBBH, 226
 si

GWNewton2, 286
sibling
 ezxml, 101
Sigma
 NoiseWhite, 419
SigmaNoise
 USOClock, 479
sigmavec
 GWFastSpinBBH, 226
sigmay2
 GWFastSpinBBH, 226
Sign
 TDI, 460
SignalList
 BackgroundGalactic, 58
Simulator
 ConfigSim, 92
SIN
 serie, 35
sintheta
 GWFastSpinBBH, 226
Slope
 GWSto, 305
smu
 Geometry, 120
 GeometryAnalytic, 137
 GeometryFile, 152
 GeometryMLDC, 167
sn
 ellipFilter, 29
snorm
 randlib.c, 615
 randlib.h, 620
sphilvec
 GWFastSpinBBH, 226
sphily2
 GWFastSpinBBH, 226
spline
 GWFastSpinBBH, 214
spofa
 linpack.c, 508
sqrt_3
 GeometryMLDC, 167
sqrtree
 Geometry, 121
 GeometryAnalytic, 137
 GeometryFile, 152
 GeometryMLDC, 168
srot
 GeometryAnalytic, 137
 GeometryMLDC, 168
Stabilization
 LISA, 312
standalone
 ezxml_root, 103
 std, 49
 StoredData
 NoiseFile, 360
 stripcopy
 ConfigSim, 84
 SWAP
 mathUtils, 34
 sx
 GWFastSpinBBH, 226
SystemEncoding
 ConfigSim, 92
SystemEncoding_int
 ConfigSim, 92
T0
 GWCusp, 197
t0
 Geometry, 121
 GeometryAnalytic, 137
 GeometryFile, 152
 GeometryMLDC, 168
TaperApplied
 GWFastSpinBBH, 227
TaperSteepness
 GWFastSpinBBH, 227
TAU
 detect, 18
taud
 GWNewton2, 286
taud0
 GWNewton2, 287
Tback
 GWCusp, 198
Tburst
 GWCusp, 198
tc
 GWFastSpinBBH, 227
tcoal
 GWNewton2, 287
TDelay
 TDI, 460
 TDI_InterData, 468
 TDITools, 471
tdelay
 Geometry, 117
 GeometryAnalytic, 133
 GeometryFile, 148
 GeometryMLDC, 163
tdelayOrderContribution
 Geometry, 117
 GeometryAnalytic, 133
 GeometryFile, 148
 GeometryMLDC, 164

tDeltaTDIDelay
 ConfigSim, 92
tder_hc
 GWSto, 305
tder_hp
 GWSto, 305
TDI, 41, 452
 ~TDI, 456
 Compute, 457
 ComputeNoEta, 457
 Eta, 459
 Fact, 459
 getCountInterDelay, 457
 getCountInterEta, 457
 IndexDelay, 459
 IndexEta, 459
 iSerie, 459
 NbDelayMax, 458
 OutFile, 459
 OutFileEncoding, 460
 ReadSignEtaDelays, 458
 RecordAndReturnResult, 458
 SCSig, 460
 Sign, 460
 TDelay, 460
 TDI, 454–456
 TDIQuickMod, 460
 tmpCountInterDelay, 460
 tmpCountInterEta, 460
TDI handling (directory TDI), 40
TDI_InterData, 42, 462
 TDI_InterData, 463, 464
TDI_InterData
 ~TDI_InterData, 465
 ComputeEta, 465
 Eta, 467
 gData, 465, 466
 gettDelayCompute, 466
 getTimeStore, 466
 gettStep, 466
 getUsable, 467
 InterpType, 467
 InterpUtilValue, 467
 NoNoise, 467
 PDPMMem, 467
 setTimeStore, 467
 TDelay, 468
 TDI_InterData, 463, 464
 TimeStore, 468
 tShift, 468
 Usable, 468
TDIDelayApprox
 ConfigSim, 92
TDIInterp

ConfigSim, 92
TDIInterpUtilVal
 ConfigSim, 93
TDIParamName
 ConfigSim, 93
TDIParamNameType
 ConfigSim, 93
TDIQuickMod
 TDI, 460
TDIsName
 ConfigSim, 93
TDIsPacks
 ConfigSim, 93
TDIsPacksFact
 ConfigSim, 93
tDisplay
 ConfigSim, 93
TDITools, 43, 469
 ~TDITools, 470
 DelayMem, 471
 getDelay, 470
 getRapidOption, 470
 RapidOption, 471
 RefreshDelay, 470
 TDelay, 471
 TDITools, 469, 470
tDurAdd
 GWSto, 305
 Noise, 350
 NoiseFile, 360
 NoiseFilter, 372
 NoiseFShape, 383
 NoiseOof, 395
 NoiseTwoFilter, 408
 NoiseWhite, 419
testbyteorder
 ConfigSim, 85
TestType
 Noise, 349
 NoiseFile, 358
 NoiseFilter, 371
 NoiseFShape, 381
 NoiseOof, 393
 NoiseTwoFilter, 406
 NoiseWhite, 418
teta
 GWNewton2, 287
tFirst
 GWSto, 305
 Noise, 350
 NoiseFile, 360
 NoiseFilter, 372
 NoiseFShape, 384
 NoiseOof, 395

NoiseTwoFilter, 408
NoiseWhite, 419
th
 GWCusp, 198
thomasvec
 GWFastSpinBBH, 227
thomasy2
 GWFastSpinBBH, 227
time_encour
 GWNewton2, 287
Time_sec
 GeometryFile, 152
timeCur
 GWFastSpinBBH, 227
TimeEnd
 MemoryWriteDisk, 341
TimeISO8601
 MathUtils, 318
TimeList
 BackgroundGalactic, 58
 GWFile, 238
TimeOffset
 ConfigSim, 94
 GWFile, 239
 MemoryWriteDisk, 341
TimeStep
 GWFile, 239
TimeStore
 TDI_InterData, 468
timevec
 GWFastSpinBBH, 227
TitleSerie
 MemoryWriteDisk, 341
TitlesReadData
 MemoryReadDisk, 332
tLast
 GWSto, 305
 Noise, 350
 NoiseFile, 360
 NoiseFilter, 372
 NoiseFShape, 384
 NoiseOof, 396
 NoiseTwoFilter, 408
 NoiseWhite, 419
tMax
 ConfigSim, 94
tmax
 GWFastSpinBBH, 227
tMaxDelay
 ConfigSim, 85
tMemNecInterpTDI
 ConfigSim, 85
tMemNoiseFirst
 ConfigSim, 94
tMemNoiseLast
 ConfigSim, 94
tMemRAM
 LISA, 313
tMemSig
 ConfigSim, 94
tMinDelay
 ConfigSim, 85
tmp_ci
 BackgroundGalactic, 58
tmp_cip1
 BackgroundGalactic, 59
tmp_Sig_i
 BackgroundGalactic, 59
tmp_Sig_ip1
 BackgroundGalactic, 59
tmp_t
 BackgroundGalactic, 59
tmpCountInterDelay
 TDI, 460
tmpCountInterEta
 TDI, 460
TmpData
 Filter, 109
tmpData_m
 NoiseFShape, 384
tmpData_p
 NoiseFShape, 384
tmu
 Geometry, 121
 GeometryAnalytic, 138
 GeometryFile, 152
 GeometryMLDC, 168
Tobs
 GWCusp, 198
 GWFastSpinBBH, 228
Toffset
 GWFastSpinBBH, 228
Tpad
 GWCusp, 198
tRangeStoreDelay
 Geometry, 121
 GeometryAnalytic, 138
 GeometryFile, 152
 GeometryMLDC, 168
tRangeStoreDelay_default
 LISACODE-LISAConstants.h, 560
tRangeStorePos
 Geometry, 121
 GeometryAnalytic, 138
 GeometryFile, 152
 GeometryMLDC, 168
tRangeStorePos_default
 LISACODE-LISAConstants.h, 560

TransfZQuadCell
 ellipFilter, 30
 TransfZQuadCellChain
 ellipFilter, 30
 TrFctGW, 472
 TrFctGW, 473
 TrFctGW
 ~TrFctGW, 473
 deltanu, 473
 GWSources, 474
 init, 474
 k, 474
 LISAGeo, 475
 TrFctGW, 473
 u, 475
 v, 475
 TRU
 serie, 35
 TruncVal
 Serie, 444
 Tsample
 GWSto, 305
 tShift
 TDI_InterData, 468
 Tstep
 GWCusp, 198
 tStep
 GWSto, 305
 Noise, 350
 NoiseFile, 361
 NoiseFilter, 373
 NoiseFShape, 384
 NoiseOof, 396
 NoiseTwoFilter, 408
 NoiseWhite, 419
 tStepMes
 ConfigSim, 94
 LISA, 314
 PhoDetPhaMet, 433
 tStepPhy
 ConfigSim, 94
 LISA, 314
 PhoDetPhaMet, 433
 tStepRecord
 Memory, 324
 MemoryReadDisk, 332
 MemoryWriteDisk, 341
 tStoreData
 Memory, 324
 MemoryReadDisk, 332
 MemoryWriteDisk, 341
 tStoreDelay
 Geometry, 121
 GeometryAnalytic, 138
 GeometryFile, 153
 GeometryMLDC, 168
 tStorePos
 Geometry, 121
 GeometryAnalytic, 138
 GeometryFile, 153
 GeometryMLDC, 168
 TSUN
 LISACODE-PhysicConstants.h, 593
 txt
 ezxml, 101
 type
 GWNewton2, 287
 u
 ezxml_root, 103
 QuadCell, 436
 TrFctGW, 475
 unit
 Vect, 483
 unusable
 Memory, 323
 MemoryReadDisk, 330
 MemoryWriteDisk, 339
 update
 GWFastSpinBBH, 214
 uppercase
 ConfigSim, 85
 upS
 ConfigSim, 86
 Usable
 TDI_InterData, 468
 UseInternalPhasemeter
 ConfigSim, 86
 USO
 PhoDetPhaMet, 434
 USO clock (directory USO_Temps), 39
 USOClock, 476
 ~USOClock, 478
 DerivLinearCoef, 479
 getDeriv, 478
 getNoise, 478
 getOffset, 478
 gGap, 478
 gTime, 479
 init, 479
 Offset, 479
 SigmaNoise, 479
 USOClock, 477
 USONoise, 480
 USONoise
 USOClock, 480
 USOs
 ConfigSim, 95

LISA, 314
uspline
 GWFastSpinBBH, 228

v
 TrFctGW, 475

Vect, 481
 ~Vect, 482
 display, 482
 norme, 482
 operator *, 483
 operator+, 483
 operator-, 484
 operator/, 484
 p, 484
 unit, 483
 Vect, 482

VectNormal
 Geometry, 118
 GeometryAnalytic, 134
 GeometryFile, 149
 GeometryMLDC, 164

Vector, 36
velocity
 Geometry, 118
 GeometryAnalytic, 134
 GeometryFile, 149
 GeometryMLDC, 165

wfile
 Serie, 444

wfileC
 SerieC, 451

WhiteData
 NoiseFilter, 373
 NoiseFShape, 384
 NoiseOof, 396
 NoiseTwoFilter, 408

WhiteData2
 NoiseTwoFilter, 408

x
 Couple, 98

x0
 Serie, 445
 SerieC, 451

Xa1
 com.c, 487

Xa1vw
 com.c, 487

Xa1w
 com.c, 488

Xa2
 com.c, 488

Xa2vw
 com.c, 488

Xa2w
 com.c, 488

Xcg1
 com.c, 488

Xcg2
 com.c, 488

Xig1
 com.c, 488

Xig2
 com.c, 488

Xlg1
 com.c, 488

Xlg2
 com.c, 489

Xm1
 com.c, 489

Xm2
 com.c, 489

xmax
 GWFastSpinBBH, 228

xml
 ezxml_root, 103

XmlOutputFile
 ConfigSim, 95

xold
 GWFastSpinBBH, 228

Xqanti
 com.c, 489

XYZ
 GeometryFile, 153

y
 Couple, 98

Yr_SI
 LISACODE-PhysicConstants.h, 593

ys
 Serie, 445
 SerieC, 451

zero
 QuadCell, 436