LISACode Reference Manual

1.2

Generated by Doxygen 1.3.4

Mon Sep 15 08:48:30 2008

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 . . . 93
10.5 ezxml Struct Reference . . . 96
10.6 ezxml_root Struct Reference . . . 99
10.7 Filter Class Reference . . . 101
10.8 Geometry Class Reference . . . 107
10.9 GeometryAnalytic Class Reference . . . 119
10.10 GeometryFile Class Reference . . . 135
10.11 GeometryMLDC Class Reference . . . 150
10.12 GW Class Reference . . . 165
10.13 GWBinary Class Reference . . . 173
10.14 GWCusp Class Reference . . . 184
10.15 GWFastSpinBBH Class Reference . . . 194
10.16 GWFile Class Reference . . . 223
10.17 GWMono Class Reference . . . 234
10.18 GWNew Class Reference . . . 244
10.19 GWNewton2 Class Reference . . . 254
10.20 GWPeriGate Class Reference . . . 282
10.21 GWSto Class Reference . . . 291

10.22 LISA Class Reference . . . 301
10.23 Mat Class Reference . . . 309
10.24 MathUtils Class Reference . . . 312
10.25 Memory Class Reference . . . 313
10.26 MemoryReadDisk Class Reference . . . 319
10.27 MemoryWriteDisk Class Reference . . . 327
10.28 Noise Class Reference . . . 336
10.29 NoiseFile Class Reference . . . 345
10.30 NoiseFilter Class Reference . . . 356
10.31 NoiseFShape Class Reference . . . 368
10.32 NoiseOof Class Reference . . . 380
10.33 NoiseSpec Struct Reference . . . 391
10.34 NoiseTwoFilter Class Reference . . . 393
10.35 NoiseWhite Class Reference . . . 404
10.36 PhoDetPhaMet Class Reference . . . 415
10.37 QuadCell Struct Reference . . . 429
10.38 Serie Class Reference . . . 431
10.39 SerieC Class Reference . . . 440
10.40 TDI Class Reference . . . 446
10.41 TDI_InterData Class Reference . . . 456
10.42 TDITools Class Reference . . . 463
10.43 TrFctGW Class Reference . . . 466
10.44 USOClock Class Reference . . . 470
10.45 Vect Class Reference . . . 475

11 LISACode File Documentation **479**

11.1 com.c File Reference . . . 479
11.2 Doxygen.bibliography File Reference . . . 484
11.3 ezxml.c File Reference . . . 485
11.4 ezxml.h File Reference . . . 494
11.5 linpack.c File Reference . . . 502
11.6 LISACODE-Background.cpp File Reference . . . 503
11.7 LISACODE-Background.h File Reference . . . 504
11.8 LISACODE-BackgroundGalactic.cpp File Reference . . . 505
11.9 LISACODE-BackgroundGalactic.h File Reference . . . 506
11.10 LISACODE-ConfigSim.cpp File Reference . . . 507
11.11 LISACODE-ConfigSim.h File Reference . . . 508

11.12LISACODE-Couple.cpp File Reference 510
11.13LISACODE-Couple.h File Reference 512
11.14LISACODE-DnonGW.cpp File Reference 513
11.15LISACODE-EllipticFilter.cpp File Reference 514
11.16LISACODE-EllipticFilter.h File Reference 516
11.17LISACODE-Filter.cpp File Reference 518
11.18LISACODE-Filter.h File Reference 519
11.19LISACODE-Geometry.cpp File Reference 520
11.20LISACODE-Geometry.h File Reference 521
11.21LISACODE-GeometryAnalytic.cpp File Reference 522
11.22LISACODE-GeometryAnalytic.h File Reference 523
11.23LISACODE-GeometryFile.cpp File Reference 524
11.24LISACODE-GeometryFile.h File Reference 525
11.25LISACODE-GeometryMLDC.cpp File Reference 526
11.26LISACODE-GeometryMLDC.h File Reference 527
11.27LISACODE-GW.cpp File Reference 528
11.28LISACODE-GW.h File Reference 529
11.29LISACODE-GWBinary.cpp File Reference 530
11.30LISACODE-GWBinary.h File Reference 531
11.31LISACODE-GWCusp.cpp File Reference 532
11.32LISACODE-GWCusp.h File Reference 533
11.33LISACODE-GWFastSpinBBH.cpp File Reference 534
11.34LISACODE-GWFastSpinBBH.h File Reference 535
11.35LISACODE-GWFile.cpp File Reference 536
11.36LISACODE-GWFile.h File Reference 537
11.37LISACODE-GWMono.cpp File Reference 538
11.38LISACODE-GWMono.h File Reference 539
11.39LISACODE-GWNew.cpp File Reference 540
11.40LISACODE-GWNew.h File Reference 541
11.41LISACODE-GWNewton2.cpp File Reference 542
11.42LISACODE-GWNewton2.h File Reference 543
11.43LISACODE-GWPeriGate.cpp File Reference 544
11.44LISACODE-GWPeriGate.h File Reference 545
11.45LISACODE-GWSto.cpp File Reference 546
11.46LISACODE-GWSto.h File Reference 547
11.47LISACODE-LISA.cpp File Reference 548

11.48LISACODE-LISA.h File Reference 549
11.49LISACODE-LISACode.cpp File Reference 550
11.50LISACODE-LISACode.ep.cpp File Reference 552
11.51LISACODE-LISAConstants.h File Reference 553
11.52LISACODE-Mat.cpp File Reference 556
11.53LISACODE-Mat.h File Reference 557
11.54LISACODE-MathUtils.h File Reference 558
11.55LISACODE-Memory.cpp File Reference 559
11.56LISACODE-Memory.h File Reference 560
11.57LISACODE-MemoryReadDisk.cpp File Reference 561
11.58LISACODE-MemoryReadDisk.h File Reference 562
11.59LISACODE-MemoryWriteDisk.cpp File Reference 563
11.60LISACODE-MemoryWriteDisk.h File Reference 564
11.61LISACODE-Noise.cpp File Reference 565
11.62LISACODE-Noise.h File Reference 566
11.63LISACODE-NoiseFile.cpp File Reference 567
11.64LISACODE-NoiseFile.h File Reference 568
11.65LISACODE-NoiseFilter.cpp File Reference 569
11.66LISACODE-NoiseFilter.h File Reference 570
11.67LISACODE-NoiseFShape.cpp File Reference 571
11.68LISACODE-NoiseFShape.h File Reference 572
11.69LISACODE-NoiseOof.cpp File Reference 573
11.70LISACODE-NoiseOof.h File Reference 574
11.71LISACODE-NoiseTwoFilter.cpp File Reference 575
11.72LISACODE-NoiseTwoFilter.h File Reference 576
11.73LISACODE-NoiseWhite.cpp File Reference 577
11.74LISACODE-NoiseWhite.h File Reference 578
11.75LISACODE-PhoDetPhaMet.cpp File Reference 579
11.76LISACODE-PhoDetPhaMet.h File Reference 580
11.77LISACODE-PhysicConstants.h File Reference 581
11.78LISACODE-Serie.cpp File Reference 588
11.79LISACODE-Serie.h File Reference 589
11.80LISACODE-TDI.cpp File Reference 590
11.81LISACODE-TDI.h File Reference 591
11.82LISACODE-TDI_InterData.cpp File Reference 592
11.83LISACODE-TDI_InterData.h File Reference 593

11.84 LISACODE-TDIApply.cpp File Reference 594
11.85 LISACODE-TDITools.cpp File Reference 595
11.86 LISACODE-TDITools.h File Reference 596
11.87 LISACODE-TrFctGW.cpp File Reference 597
11.88 LISACODE-TrFctGW.h File Reference 598
11.89 LISACODE-USOClock.cpp File Reference 599
11.90 LISACODE-USOClock.h File Reference 600
11.91 LISACODE-Vect.cpp File Reference 601
11.92 LISACODE-Vect.h File Reference 603
11.93 randlib.c File Reference 604
11.94 randlib.h File Reference 610

12 LISACode Page Documentation **615**
12.1 Introduction 615
12.2 A description of the Code 617
12.3 LISACode parameters 620
12.4 Bibliography 621
12.5 Todo List 622

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 . . . 93
- ezxml . . . 96
- ezxml_root . . . 99
- Filter . . . 101
- Geometry . . . 107
  - GeometryAnalytic . . . 119
  - GeometryFile . . . 135
  - GeometryMLDC . . . 150
- GW . . . 165
  - GWBinary . . . 173
  - GWCusp . . . 184
  - GWFastSpinBBH . . . 194
  - GWFile . . . 223
  - GWMono . . . 234
  - GWNew . . . 244
  - GWNewton2 . . . 254
  - GWPeriGate . . . 282
  - GWSto . . . 291
- LISA . . . 301
- Mat . . . 309
- MathUtils . . . 312
- Memory . . . 313
  - MemoryReadDisk . . . 319
  - MemoryWriteDisk . . . 327
- Noise . . . 336
  - NoiseFile . . . 345
  - NoiseFilter . . . 356
  - NoiseFShape . . . 368
  - NoiseOof . . . 380

NoiseTwoFilter 393
NoiseWhite 404
NoiseSpec 391
PhoDetPhaMet 415
QuadCell 429
Serie 431
SerieC 440
TDI 446
TDI_InterData 456
TDITools 463
TrFctGW 466
USOClock 470
Vect 475

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) 93
ezxml 96
ezxml_root 99
Filter (Filter management class) 101
Geometry (Orbit geometry class) 107
GeometryAnalytic (Compute spacecraft position from analytical formulations of the article : S) 119
GeometryFile (Compute spacecraft position from :) 135
GeometryMLDC (Compute spacecraft position from :) 150
GW (Gravitational Waves parameters are described in this class) 165
GWBinary (Gravitational Waves parameters for a monochromatic binary system are defined in this class) 173
GWCusp (Gravitational Waves instantaneous parameters h_+ and $h_\times$ are described in this class) 184
GWFastSpinBBH (Gravitational Waves parameters for a spinning black holes binary system are defined in this class) 194
GWFile (Gravitational Waves file management) 223
GWMono (Gravitational Waves instantaneous parameters h_plus and h_cross are described in this class) 234
GWNew (Gravitational Waves instantaneous parameters h_plus and h_cross are described in this class) 244
GWNewton2 (Gravitational Waves binary system parameters computation) 254
GWPeriGate (Gravitational Waves periodic gate signal) 282
GWSto (Gravitational Waves instantaneous parameters h_plus and h_cross are described in this class) 291
LISA (This class contains and manages all the elements necessary to LISA satellites simulation) 301
Mat ((3x3) matrix management class) 309
MathUtils (Angle conversion class) 312
Memory (Memory management class) 313
MemoryReadDisk (Class to manage disk reading. This class (module) reads data in file and stores them) 319

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) 327
Noise (Noise base class) 336
NoiseFile (Noise derived class to treat files with noise data) 345
NoiseFilter (Noise derived class to treat noise filters) 356
NoiseFShape (Noise derived class to treat noise filters) 368
NoiseOof (Noise derived class to treat noise filters) 380
NoiseSpec (Noise specification structure) 391
NoiseTwoFilter (Noise derived class to treat noise filters) 393
NoiseWhite (Noise derived class to treat white noise) 404
PhoDetPhaMet (Phasemeter photodiode class) 415
QuadCell (Elliptic cell structure) 429
Serie (Serie interpolation class) 431
SerieC (Complex serie interpolation class) 440
TDI (Time Delay Interferometry combinaison class) 446
TDI_InterData (Time Delay Interferometry interpolated signal class) 456
TDITools (Time Delay Interferometry tools class) 463
TrFctGW (Gravitational Waves Transfer Function class) 466
USOClock (Ultra Stable Oscillator based satellite time is defined in this class) 470
Vect (3 components vector management class) 475

Chapter 6

LISACode File Index

6.1 LISACode File List

Here is a list of all files with brief descriptions:

com.c . . . 479
Doxygen.bibliography . . . 484
ezxml.c . . . 485
ezxml.h . . . 494
linpack.c . . . 502
LISACODE-Background.cpp . . . 503
LISACODE-Background.h . . . 504
LISACODE-BackgroundGalactic.cpp . . . 505
LISACODE-BackgroundGalactic.h . . . 506
LISACODE-ConfigSim.cpp . . . 507
LISACODE-ConfigSim.h . . . 508
LISACODE-Couple.cpp . . . 510
LISACODE-Couple.h . . . 512
LISACODE-DnonGW.cpp . . . 513
LISACODE-EllipticFilter.cpp . . . 514
LISACODE-EllipticFilter.h . . . 516
LISACODE-Filter.cpp . . . 518
LISACODE-Filter.h . . . 519
LISACODE-Geometry.cpp . . . 520
LISACODE-Geometry.h . . . 521
LISACODE-GeometryAnalytic.cpp . . . 522
LISACODE-GeometryAnalytic.h . . . 523
LISACODE-GeometryFile.cpp . . . 524
LISACODE-GeometryFile.h . . . 525
LISACODE-GeometryMLDC.cpp . . . 526
LISACODE-GeometryMLDC.h . . . 527
LISACODE-GW.cpp . . . 528
LISACODE-GW.h . . . 529
LISACODE-GWBinary.cpp . . . 530
LISACODE-GWBinary.h . . . 531
LISACODE-GWCusp.cpp . . . 532
LISACODE-GWCusp.h . . . 533
LISACODE-GWFastSpinBBH.cpp . . . 534

LISACODE-GWFastSpinBBH.h . . . 535
LISACODE-GWFile.cpp . . . 536
LISACODE-GWFile.h . . . 537
LISACODE-GWMono.cpp . . . 538
LISACODE-GWMono.h . . . 539
LISACODE-GWNew.cpp . . . 540
LISACODE-GWNew.h . . . 541
LISACODE-GWNewton2.cpp . . . 542
LISACODE-GWNewton2.h . . . 543
LISACODE-GWPeriGate.cpp . . . 544
LISACODE-GWPeriGate.h . . . 545
LISACODE-GWSto.cpp . . . 546
LISACODE-GWSto.h . . . 547
LISACODE-LISA.cpp . . . 548
LISACODE-LISA.h . . . 549
LISACODE-LISACode.cpp . . . 550
LISACODE-LISACode.ep.cpp . . . 552
LISACODE-LISAConstants.h (Physical constants of LISA instrument) . . . 553
LISACODE-Mat.cpp . . . 556
LISACODE-Mat.h . . . 557
LISACODE-MathUtils.h . . . 558
LISACODE-Memory.cpp . . . 559
LISACODE-Memory.h . . . 560
LISACODE-MemoryReadDisk.cpp . . . 561
LISACODE-MemoryReadDisk.h . . . 562
LISACODE-MemoryWriteDisk.cpp . . . 563
LISACODE-MemoryWriteDisk.h . . . 564
LISACODE-Noise.cpp . . . 565
LISACODE-Noise.h . . . 566
LISACODE-NoiseFile.cpp . . . 567
LISACODE-NoiseFile.h . . . 568
LISACODE-NoiseFilter.cpp . . . 569
LISACODE-NoiseFilter.h . . . 570
LISACODE-NoiseFShape.cpp . . . 571
LISACODE-NoiseFShape.h . . . 572
LISACODE-NoiseOof.cpp . . . 573
LISACODE-NoiseOof.h . . . 574
LISACODE-NoiseTwoFilter.cpp . . . 575
LISACODE-NoiseTwoFilter.h . . . 576
LISACODE-NoiseWhite.cpp . . . 577
LISACODE-NoiseWhite.h . . . 578
LISACODE-PhoDetPhaMet.cpp . . . 579
LISACODE-PhoDetPhaMet.h . . . 580
LISACODE-PhysicConstants.h (Physical constants, reference values and unit conversions) . . . 581
LISACODE-Serie.cpp . . . 588
LISACODE-Serie.h . . . 589
LISACODE-TDI.cpp . . . 590
LISACODE-TDI.h . . . 591
LISACODE-TDI_InterData.cpp . . . 592
LISACODE-TDI_InterData.h . . . 593
LISACODE-TDIApply.cpp . . . 594
LISACODE-TDITools.cpp . . . 595
LISACODE-TDITools.h . . . 596
LISACODE-TrFctGW.cpp . . . 597

LISACODE-TrFctGW.h 598
LISACODE-USOClock.cpp 599
LISACODE-USOClock.h 600
LISACODE-Vect.cpp 601
LISACODE-Vect.h 603
randlib.c 604
randlib.h 610

Chapter 7

LISACode Page Index

7.1 LISACode Related Pages

Here is a list of all related documentation pages:

Introduction 615
A description of the Code 617
LISACode parameters 620
Bibliography 621
Todo List 622

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 > I (0, 1)

  Pure imaginary=(0,1).

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

  Poles, zeros and elliptic cells coefficients computation.

- double ak (double y)

  Integral filter parameter computation.

- double cak (double x)

  Developped filter parameter computation.

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

  Recursive or direct coefficients computation.

- double FilterQuadCell (double xn, QuadCell *Cell)

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

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

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

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

  Elliptic cell Z transform.

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

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

- double AbsRespFunctQuadCell (double f, QuadCell Cell)

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

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

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

- double HmQuadCell (QuadCell Cell)

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

- double KmQuadCell (QuadCell Cell)

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

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

  Matches nearest poles for a chain of elliptic cells.

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

  Orders cells according to the the max of inf norm.

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

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

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

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

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

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

8.5.1 Define Documentation

8.5.1.1 #define alog(A) log(A)

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

Referenced by cak(), and elli().

8.5.1.2 #define alog10(A) log10(A)

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

8.5.2 Function Documentation

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

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

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

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

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

Referenced by AbsRespFunctQuadCellChain().

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

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

For each cell, AbsRespFunctQuadCell function is called.

Returned value is product of all cells frequency response modulus.

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

References AbsRespFunctQuadCell().

8.5.2.3 double ak (double *y*)

Integral filter parameter computation.

cak function is called.

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

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

References cak().

Referenced by elli().

8.5.2.4 double cak (double *x*)

Developped filter parameter computation.

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

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

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

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

References alog.

Referenced by ak(), and elli().

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

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

Inputs are:

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

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

Outputs are :

- FilterCellsOut : cells
- NCellsOut : number of cells

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

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

$$u = (0, 0)$$

$$a0 = 1$$

$$a1 = CoefB[i]$$

$$b1 = CoefD[i]$$

$$b2 = CoefC[i]$$

$$zero = Zeros[i]$$

$$pole = Poles[i]$$

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

CalcScalingFact result is returned.

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

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

Referenced by CalcEllipticFilter4LISACode().

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

Computes filter coefficients from LISA Code user specifications.

The global scale factor is included in the first cell.

Inputs are:

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

Outputs are :

- CellsCoef : cells coefficients
- NCellsOut : number of cells

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

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

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

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

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

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

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

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

First cell is rescaled :

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

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

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

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

References QuadCell::b1, and CalcEllipticFilter().

Referenced by Filter::Filter().

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

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

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

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

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

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

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

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

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

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

For last cell:

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

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

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

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

References HmQuadCell(), and KmQuadCell().

Referenced by CalcEllipticFilter().

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

Poles, zeros and elliptic cells coefficients computation.

Inputs are:

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

Outputs are :

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

Computations :

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

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

$T = 1/fe$

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

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

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

$ak1 = ak(dk)$ using ak function

$ak2 = ak(dk1)$ using ak function

$ak3 = ak(dkp)$ using ak function

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

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

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

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

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

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

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

Referenced by CalcEllipticFilter().

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

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

Computations :

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

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

Cell u memory attribute is updated :

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

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

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

Referenced by FilterQuadCellChain().

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

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

For each cell, FilterQuadCell function is called.

Cells are updated and returned value depends on last cell.

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

References FilterQuadCell().

8.5.2.11 double HmQuadCell (QuadCell *Cell*)

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

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

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

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

Referenced by CalcScalingFact(), and KmQuadCell().

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

Pure imaginary=(0,1).

8.5.2.13 double KmQuadCell (QuadCell *Cell*)

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

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

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

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

Referenced by CalcScalingFact(), and OrderCellMaxNorm().

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

Orders cells according to the the max of inf norm.

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

First, KmQuadCell function is called for each cell.

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

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

References KmQuadCell().

Referenced by CalcEllipticFilter().

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

Matches nearest poles for a chain of elliptic cells.

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

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

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

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

Referenced by CalcEllipticFilter().

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

Recursive or direct coefficients computation.

Inputs are:

- y
- A
- ak1
- ak3

Computations :

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

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

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

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

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

Referenced by elli().

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

Elliptic cell Z transform.

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

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

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

Referenced by TransfZQuadCellChain().

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

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

For each cell, TransfZQuadCell function is called.

Returned value is product of all cells Z transform.

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

References TransfZQuadCell().

8.6 Filter

Classes

- class Filter

  filter management class.

8.7 Matrix

Classes

- class Mat

  (3x3) matrix management class.

8.8 Mathematical Tools (directory Outils_Math)

Modules

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

8.9 Angles handling

Classes

- class MathUtils

  Angle conversion class.

Defines

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

8.9.1 Define Documentation

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

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

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

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

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

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

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

8.10 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* $t = 0, \ldots, tMemTDI + tTDIShift$ *with tStepMes timsetep.*
    Signals are stored.
    LISA::MakeOneStepOfTime method is called.
    Delays are recorded.
    Positions are recorded.
  - *Data processing second step : when there are enough data, TDI is computed and results are stored in file, while time* $t \leq tmax$ *with tStepMes timsetep.*
    TDI is computed using TDI_InterData::ComputeEta method.
    Delays are recorded.
    Positions are recorded.
    .

8.20.1 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 $t = 0, \ldots, tMemTDI + tTDIShift$ with tStepMes timsetep.

  Signals are stored.

  LISA::MakeOneStepOfTime method is called.

  Delays are recorded.

  Positions are recorded.

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

  TDI is computed using TDI_InterData::ComputeEta method.

  Delays are recorded.

Positions are recorded.

.

Definition at line 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 deltanu.

Protected Attributes

- Geometry * LISAGeo
  LISA orbit description.

10.1.2 Constructor & Destructor Documentation

10.1.2.1 Background::Background ()

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

- LISAGeo = Geometry instance with default attributes.

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

References LISAGeo.

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

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

- LISAGeo = LISAGeo_n input

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

References LISAGeo.

10.1.2.3 Background::~Background () [virtual]

Destructor.

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

10.1.3 Member Function Documentation

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

Gives deltanu.

Virtual unused method.

Returns:
0.0

Reimplemented in BackgroundGalactic.

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

Referenced by PhoDetPhaMet::gGWB().

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

Sets LISAGeo attribute with input.

- LISAGeo = *LISAGeo_n* input

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

References LISAGeo.

Referenced by LISA::LISA().

10.1.4 Member Data Documentation

10.1.4.1 Geometry* Background::LISAGeo [protected]

LISA orbit description.

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

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

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

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

10.2 BackgroundGalactic Class Reference

`#include <LISACODE-BackgroundGalactic.h>`

Inheritance diagram for BackgroundGalactic::

10.2.1 Detailed Description

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

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

Public Member Functions

- BackgroundGalactic ()

  Constructs an instance and initializes its attributes with default values.

- BackgroundGalactic (char ∗FileName, double Factor)

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

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

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

- ∼BackgroundGalactic ()

  Destructor.

- void ReadFile (char ∗FileName, double Factor_n)

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

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

  Gives frequency fluctuation.

- void setGeometry (Geometry ∗LISAGeo_n)

  Sets LISAGeo attribute with input.

Protected Attributes

- vector< double > TimeList

  Vector of time values associated to signal samples.

- vector< vector< double > > SignalList

  Set of signals received by each photometer.

- int NbData

  Number of data or samples.

- int iRead

  Last bin or index read.

- double tmp_t

  Last time value read.

- double tmp_ci

  Last bin time lower or equal to current time.

- double tmp_cip1

  First bin time greater than current time.

- vector< double > tmp_Sig_i

  Signal value corresponding to tmp_ci.

- vector< double > tmp_Sig_ip1

  Signal value corresponding to tmp_cip1.

- Geometry * LISAGeo

  LISA orbit description.

10.2.2 Constructor & Destructor Documentation

10.2.2.1 BackgroundGalactic::BackgroundGalactic ()

Constructs an instance and initializes its attributes with default values.

Default values are:

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

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

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

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

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

It calls ReadFile.

Parameters:

FileName: name of file containing signal values and associated times

Factor: scale factor applied to read signal values

Attributes are set as follows

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

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

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

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

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

It calls ReadFile.

Parameters:

LISAGeo_n: LISA orbit.

FileName: name of file containing signal values and associated times

Factor: scale factor applied to read signal values

Attributes will be set to next values:

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

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

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

10.2.2.4 BackgroundGalactic::~BackgroundGalactic ()

Destructor.

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

10.2.3 Member Function Documentation

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

Gives frequency fluctuation.

Parameters:

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

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

t: time

It computes *tGeo* to obtain deltanu:

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

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

Else:

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

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

$$+ \frac{tGeo - TimeList[iRead]}{TimeList[iRead+1] - TimeList[iRead]} \cdot tmp_Sig_ip1[iSC + 3 \cdot IndirSens - 1]$$

Reimplemented from Background.

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

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

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

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

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

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

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

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

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

Referenced by BackgroundGalactic(), and ReadFile().

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

Sets LISAGeo attribute with input.

- LISAGeo = *LISAGeo_n* input

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

References Background::LISAGeo.

Referenced by LISA::LISA().

10.2.4 Member Data Documentation

10.2.4.1 BackgroundGalactic::iRead [protected]

Last bin or index read.

Referenced by BackgroundGalactic(), and deltanu().

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

LISA orbit description.

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

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

10.2.4.3 BackgroundGalactic::NbData [protected]

Number of data or samples.

Referenced by BackgroundGalactic(), and ReadFile().

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

Set of signals received by each photometer.

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

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

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

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

Vector of time values associated to signal samples.

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

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

10.2.4.6 BackgroundGalactic::tmp_ci [protected]

Last bin time lower or equal to current time.

Referenced by deltanu().

10.2.4.7 BackgroundGalactic::tmp_cip1 [protected]

First bin time greater than current time.

Referenced by deltanu().

10.2.4.8 BackgroundGalactic::tmp_Sig_i [protected]

Signal value corresponding to tmp_ci.

Referenced by deltanu().

10.2.4.9 BackgroundGalactic::tmp_Sig_ip1 [protected]

Signal value corresponding to tmp_cip1.

Referenced by deltanu().

10.2.4.10 BackgroundGalactic::tmp_t [protected]

Last time value read.

Referenced by BackgroundGalactic(), and deltanu().

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

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

10.3 ConfigSim Class Reference

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

10.3.1 Detailed Description

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

Definition at line 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 gettDisplay ()

  It returns is the value of tDisplay attribute.

- double gettDeltaTDIDelay ()

It returns tDeltaTDIDelay attribute.

- INTERP getTDIInterp ()

  It returns TDIInterp attribute.

- double getTDIInterpUtilVal ()

  It returns TDIInterpUtilVal attribute.

- double getArmlength ()

  It returns the value of Armlength attribute.

- double getOrbStartTime ()

  It returns OrbStartTime attribute.

- double getOrbInitRot ()

  It returns OrbInitRot attribute.

- bool getTDIDelayApprox ()

  It returns TDIDelayApprox attribute.

- int getOrbMove ()

  It returns the value of OrbMove attribute.

- int getOrbOrder ()

  It returns the value of OrbOrder attribute.

- 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.

- 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 getTDIParamName ()

- string getTDIParamNameType ()
- string getSC1ParamName ()
- string getSC2ParamName ()
- string getSC3ParamName ()
- string getSimulator ()
- double getTimeOffset ()
- char * getFileNameSigSC1 ()
- char * getFileNameSigSC2 ()
- char * getFileNameSigSC3 ()
- int testbyteorder ()
- char * uppercase (const char *)
- void majuscule (char &)
- void ReadFile ()

  Reads configuration file.

- void ReadASCIIFile ()

  Reads ASCII configuration file.

- void ReadXMLFile ()

  Reads XML configuration file.

- void CreateXmlOutputFile ()

  Ceates XML Output file for MLDC.

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

  Character to double conversion.

- double gXMLTime (ezxml_t param)

  Extracts time parameter from XML sctructure.

- double gXMLAngle (ezxml_t param)

  Extracts angle parameter from XML sctructure.

- double gXMLFrequency (ezxml_t param)

  Extracts frequency parameter from XML sctructure.

- double gXMLAstroMass (ezxml_t param)

  Extracts AstroMass parameter from XML sctructure.

- double gXMLAstroDistance (ezxml_t param)

  Extracts AstroDistance parameter from XML sctructure.

- char * gXMLWord (ezxml_t param)
- double gXMLdouble (ezxml_t param)

  Extracts a value (double) parameter from XML sctructure.

- string gXMLstring (ezxml_t param)

  Extracts a string from XML structure.

- int gXMLint (ezxml_t param)

Extracts a value (double) parameter from XML sctructure.

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

  Extracts TimeSeries parameters from XML sctructure.

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

  Adds a noise specification structure.

- void NoisesCreation ()

  Creation of noises.

- double tMaxDelay ()

  Computes maximal time travel for one delay.

- double tMinDelay ()

  Computes minimal time travel for one delay.

- double tMemNecInterpTDI ()

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

- bool getNoNoise ()

  Checks if there are noises or not.

- bool FindTDIName (const char ∗generatorname, vector< int > &tmp_TDIPacks, vector< double > &tmp_TDIPacksFact)

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 TDIInterp

  TDI interpolator type.

- double TDIInterpUtilVal

  Value used for TDI interpolation.

- double Armlength

  Nominal LISA arm length.

- double OrbStartTime

  Orbits start time.

- double OrbInitRot

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

- int OrbOrder

  Order for time propagation computing.

- int OrbMove

  Satellites motion.

- bool TDIDelayApprox

  Approximated TDI delays computation or not.

- double LaserPower

  Laser Power in Watts.

- bool PhaMetFilterON

Phasemeter Filter (On, Off).

- vector< double > PhaMetFilterParam

  Phasemeter Filter Parameters.

- vector< GW * > GWs

  Gravitational Waves Parameters.

- Background * GWB

  Background signals

- vector< vector< NoiseSpec > > NoisesData

  noise specifications

- vector< Noise * > Noises

  satellites noise

- vector< USOClock > USOs

  satellites USO time

- char FileNameSigSC1 [256]

  FileName for 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::CreateXmlOutputFile ()

Ceates XML Output file for MLDC.

Definition at line 2953 of file LISACODE-ConfigSim.cpp.

References FichXML, getAuthor(), getFileEncodingSig(), getFileEncodingTDI(), getFileNameSigSC1(), getFileNameSigSC2(), getFileNameSigSC3(), getFileNameTDI(), getGenerationDate(), getGenerationType(), getSimulator(), getSystemEncoding(), getTDIParamName(), getTimeOffset(), gettMax(), gettStepMes(), and getXmlOutputFile().

Referenced by ReadXMLFile().

10.3.3.2 void ConfigSim::DefaultConfig (char * *NameConfigFile_n*)

It sets default simulation parameters.

The default parameters are:

- DefVectNoise = 0
- tStepPhy = 0.5
- tMax = 65736.0
- tStepMes = 1.0
- tMemNoiseFirst = 5.0
- tMemNoiseLast = -30.0
- tMemSig = 100.0
- tDisplay = 1000.0
- tDeltaTDIDelay = 0.0
- TDIInterp = LAG
- TDIInterpUtilVal = 20
- Armlength = L0_m_default
- OrbStartTime = 0.0
- OrbInitRot = 0.0
- OrbMove = 1
- OrbOrder = 2
- TDIDelayApprox = false
- LaserPower = LaserPower_W_default
- PhaMetFilterON = true
- Phasemeter Filter Parameters :
- attenuation : PhaMetFilterParam[0]= 180.0 [dB]
- oscillations in bandwidth : PhaMetFilterParam[1]= 0.1 [dB]
- low transition frequency in factor of frequency of measurment : PhaMetFilterParam[2]= 0.1
- high transition frequency in factor of frequency of measurment : PhaMetFilterParam[3]= 0.3
- filename for Space Craft 1 Signal : FileNameSigSC1= "None"
- filename for Space Craft 2 Signal : FileNameSigSC2= "None"
- filename for Space Craft 3 Signal : =FileNameSigSC3 "None"
- filename for delays : FileNameDelays= "None"
- fileNameP for positions : FileNamePositions= "None"

- filename for TDI generators : FileNameTDI= "Def_SignalTDI.txt"
- GWB = NULL
- NoisesData = 24 NULL vectors
- USOs = 3 USOClock instances set to 0.0
- NbMaxDelays = 0
- ConfigFileName = NameConfigFile_n input

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

References Armlength, ConfigFileName, Endian, FileEncodingDelays, FileEncodingPos, FileEncodingSC1, FileEncodingSC2, FileEncodingSC3, FileEncodingTDI, FileNameDelays, FileNamePositions, FileNameSigSC1, FileNameSigSC2, FileNameSigSC3, FileNameTDI, GlobalRandomSeed, GWB, L0_m_-default, LAG, LaserPower, LaserPower_W_default, NbMaxDelays, Noises, NoisesData, OrbInitRot, OrbMove, OrbOrder, OrbStartTime, PhaMetFilterON, PhaMetFilterParam, tDeltaTDIDelay, TDIDelayApprox, TDIInterp, TDIInterpUtilVal, tDisplay, tMax, tMemNoiseFirst, tMemNoiseLast, tMemSig, tStepMes, tStepPhy, USOs, and XmlOutputFile.

Referenced by ConfigSim().

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

Definition at line 3679 of file LISACODE-ConfigSim.cpp.

References NbMaxDelays.

Referenced by ReadASCIIFile(), and ReadXMLFile().

10.3.3.4 double ConfigSim::getArmlength () [inline]

It returns the value of Armlength attribute.

Definition at line 330 of file LISACODE-ConfigSim.h.

References Armlength.

10.3.3.5 string ConfigSim::getAuthor () [inline]

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

References Author.

Referenced by CreateXmlOutputFile().

10.3.3.6 int ConfigSim::getFileEncodingDelays () [inline]

It returns FileEncodingDelays attribute.

Definition at line 369 of file LISACODE-ConfigSim.h.

References FileEncodingDelays.

Referenced by main().

10.3.3.7 int ConfigSim::getFileEncodingPositions () [inline]

It returns #FileEncodingPositions attribute.

Definition at line 371 of file LISACODE-ConfigSim.h.

References FileEncodingPos.

Referenced by main().

10.3.3.8 int ConfigSim::getFileEncodingSig (int *iSC*)

Returns FileEncodingSig corresponding to iSC input.

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

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

References FileEncodingSC1, FileEncodingSC2, and FileEncodingSC3.

Referenced by CreateXmlOutputFile(), and main().

10.3.3.9 int ConfigSim::getFileEncodingTDI () [inline]

It returns FileEncodingTDI attribute.

Definition at line 367 of file LISACODE-ConfigSim.h.

References FileEncodingTDI.

Referenced by CreateXmlOutputFile(), and main().

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

It returns FileNameDelays attribute.

Definition at line 361 of file LISACODE-ConfigSim.h.

References FileNameDelays.

Referenced by main().

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

It returns FileNamePositions attribute.

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

References FileNamePositions.

Referenced by main().

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

Returns FileNameSig corresponding to iSC input.

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

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

References FileNameSigSC1, FileNameSigSC2, and FileNameSigSC3.

Referenced by main().

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

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

References FileNameSigSC1.

Referenced by CreateXmlOutputFile().

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

Definition at line 401 of file LISACODE-ConfigSim.h.

References FileNameSigSC2.

Referenced by CreateXmlOutputFile().

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

Definition at line 402 of file LISACODE-ConfigSim.h.

References FileNameSigSC3.

Referenced by CreateXmlOutputFile().

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

It returns FileNameTDI attribute.

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

References FileNameTDI.

Referenced by CreateXmlOutputFile(), and main().

10.3.3.17 string ConfigSim::getGenerationDate () [inline]

Definition at line 391 of file LISACODE-ConfigSim.h.

References GenerationDate.

Referenced by CreateXmlOutputFile().

10.3.3.18 string ConfigSim::getGenerationType () [inline]

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

References GenerationType.

Referenced by CreateXmlOutputFile().

10.3.3.19 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 251 of file LISACODE-ConfigSim.cpp.

References TDIsPacks.

Referenced by main().

10.3.3.20 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 264 of file LISACODE-ConfigSim.cpp.

References TDIsPacksFact.

Referenced by main().

10.3.3.21 void ConfigSim::getGeometry (Geometry *& *SCPos*)

Returns FileNameSig corresponding to iSC input.

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

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

References Armlength, OrbInitRot, OrbMove, OrbOrder, OrbStartTime, and tStepPhy.

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

10.3.3.22 int ConfigSim::getGlobalRandomSeed () [inline]

It returns the value of GlobalRandomSeed attribute.

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

References GlobalRandomSeed.

Referenced by main().

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

It returns the first element of the vector GWs.

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

References GWs.

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

It returns GWB attribute.

Definition at line 375 of file LISACODE-ConfigSim.h.

References GWB.

Referenced by LISA::LISA().

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

It returns a pointer to the GWs attribute.

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

References GWs.

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

10.3.3.26 double ConfigSim::getLaserPower () [inline]

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

Definition at line 344 of file LISACODE-ConfigSim.h.

References LaserPower.

10.3.3.27 char * ConfigSim::getNameGenTDI (int *iGen*)

Returns NameGenTDI corresponding to iGen input.

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

Definition at line 230 of file LISACODE-ConfigSim.cpp.

References TDIsName, and TDIsPacks.

Referenced by main().

10.3.3.28 int ConfigSim::getNbMaxDelays () [inline]

It returns NbMaxDelays attribute plus 1.

Definition at line 387 of file LISACODE-ConfigSim.h.

References NbMaxDelays.

Referenced by main().

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

It returns Noises attribute.

Definition at line 373 of file LISACODE-ConfigSim.h.

References Noises.

Referenced by LISA::LISA().

10.3.3.30 bool ConfigSim::getNoNoise ()

Checks if there are noises or not.

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

Definition at line 3641 of file LISACODE-ConfigSim.cpp.

References Noises.

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

10.3.3.31 double ConfigSim::getOrbInitRot () [inline]

It returns OrbInitRot attribute.

Definition at line 334 of file LISACODE-ConfigSim.h.

References OrbInitRot.

10.3.3.32 int ConfigSim::getOrbMove () [inline]

It returns the value of OrbMove attribute.

Definition at line 338 of file LISACODE-ConfigSim.h.

References OrbMove.

10.3.3.33 int ConfigSim::getOrbOrder () [inline]

It returns the value of OrbOrder attribute.

Definition at line 340 of file LISACODE-ConfigSim.h.

References OrbOrder.

10.3.3.34 double ConfigSim::getOrbStartTime () [inline]

It returns OrbStartTime attribute.

Definition at line 332 of file LISACODE-ConfigSim.h.

References OrbStartTime.

10.3.3.35 bool ConfigSim::getPhaMetFilter () [inline]

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

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

References PhaMetFilterON.

Referenced by LISA::LISA().

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

It returns PhaMetFilterParam attribute.

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

References PhaMetFilterParam.

Referenced by LISA::LISA().

10.3.3.37 string ConfigSim::getSC1ParamName () [inline]

Definition at line 395 of file LISACODE-ConfigSim.h.

References SC1ParamName.

10.3.3.38 string ConfigSim::getSC2ParamName () [inline]

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

References SC2ParamName.

10.3.3.39 string ConfigSim::getSC3ParamName () [inline]

Definition at line 397 of file LISACODE-ConfigSim.h.

References SC3ParamName.

10.3.3.40 string ConfigSim::getSimulator () [inline]

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

References Simulator.

Referenced by CreateXmlOutputFile().

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

It returns SystemEncoding , BigEndian or LittleEndian.

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

References SystemEncoding.

Referenced by CreateXmlOutputFile(), and ReadFile().

10.3.3.42 double ConfigSim::gettDeltaTDIDelay () [inline]

It returns tDeltaTDIDelay attribute.

Definition at line 324 of file LISACODE-ConfigSim.h.

References tDeltaTDIDelay.

Referenced by main().

10.3.3.43 bool ConfigSim::getTDIDelayApprox () [inline]

It returns TDIDelayApprox attribute.

Definition at line 336 of file LISACODE-ConfigSim.h.

References TDIDelayApprox.

Referenced by main().

10.3.3.44 INTERP ConfigSim::getTDIInterp () [inline]

It returns TDIInterp attribute.

Definition at line 326 of file LISACODE-ConfigSim.h.

References INTERP, and TDIInterp.

Referenced by main().

10.3.3.45 double ConfigSim::getTDIInterpUtilVal () [inline]

It returns TDIInterpUtilVal attribute.

Definition at line 328 of file LISACODE-ConfigSim.h.

References TDIInterpUtilVal.

Referenced by main().

10.3.3.46 string ConfigSim::getTDIParamName () [inline]

Definition at line 393 of file LISACODE-ConfigSim.h.

References TDIParamName.

Referenced by CreateXmlOutputFile().

10.3.3.47 string ConfigSim::getTDIParamNameType () [inline]

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

References TDIParamNameType.

10.3.3.48 double ConfigSim::gettDisplay () [inline]

It returns is the value of tDisplay attribute.

Definition at line 322 of file LISACODE-ConfigSim.h.

References tDisplay.

Referenced by main().

10.3.3.49 double ConfigSim::getTimeOffset () [inline]

Definition at line 399 of file LISACODE-ConfigSim.h.

References TimeOffset.

Referenced by CreateXmlOutputFile().

10.3.3.50 double ConfigSim::gettMax () [inline]

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

Definition at line 312 of file LISACODE-ConfigSim.h.

References tMax.

Referenced by CreateXmlOutputFile(), and main().

10.3.3.51 double ConfigSim::gettMemNoiseFirst () [inline]

It returns tMemNoiseFirst attribute.

Definition at line 316 of file LISACODE-ConfigSim.h.

References tMemNoiseFirst.

10.3.3.52 double ConfigSim::gettMemNoiseLast () [inline]

It returns tMemNoiseLast attribute.

Definition at line 318 of file LISACODE-ConfigSim.h.

References tMemNoiseLast.

10.3.3.53 double ConfigSim::gettMemSig () [inline]

It returns tMemSig attribute.

Definition at line 320 of file LISACODE-ConfigSim.h.

References tMemSig.

Referenced by LISA::LISA().

10.3.3.54 double ConfigSim::gettStepMes () [inline]

It returns tStepMes attribute.

Definition at line 314 of file LISACODE-ConfigSim.h.

References tStepMes.

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

10.3.3.55 double ConfigSim::gettStepPhy () [inline]

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

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

References tStepPhy.

Referenced by LISA::LISA().

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

It returns USOs attribute.

Definition at line 377 of file LISACODE-ConfigSim.h.

References USOs.

Referenced by LISA::LISA().

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

Definition at line 389 of file LISACODE-ConfigSim.h.

References XmlOutputFile.

Referenced by CreateXmlOutputFile().

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

Extracts angle parameter from XML sctructure.

Checks if unit is degree.

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

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

Referenced by ReadXMLFile().

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

Extracts AstroDistance parameter from XML sctructure.

Checks if unit is Parsec.

Definition at line 3176 of file LISACODE-ConfigSim.cpp.

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

Referenced by ReadXMLFile().

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

Extracts AstroMass parameter from XML sctructure.

Checks if unit is SolarMass.

Definition at line 3160 of file LISACODE-ConfigSim.cpp.

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

Referenced by ReadXMLFile().

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

Extracts a value (double) parameter from XML sctructure.

NO units

Definition at line 3226 of file LISACODE-ConfigSim.cpp.

References ezxml_attr(), ezxml_t, and ezxml_txt.

Referenced by ReadXMLFile().

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

Extracts frequency parameter from XML sctructure.

Checks if unit is MilliHertz.

Definition at line 3141 of file LISACODE-ConfigSim.cpp.

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

Referenced by ReadXMLFile().

10.3.3.63 int ConfigSim::gXMLint (ezxml_t *param*)

Extracts a value (double) parameter from XML sctructure.

NO units

Definition at line 3259 of file LISACODE-ConfigSim.cpp.

References ezxml_attr(), ezxml_t, and ezxml_txt.

Referenced by ReadXMLFile().

10.3.3.64 string ConfigSim::gXMLstring (ezxml_t *param*)

Extracts a string from XML structure.

NO units

Definition at line 3240 of file LISACODE-ConfigSim.cpp.

References ezxml_attr(), and ezxml_t.

Referenced by ReadXMLFile().

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

Extracts time parameter from XML sctructure.

Checks if unit is second.

Definition at line 3105 of file LISACODE-ConfigSim.cpp.

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

Referenced by ReadASCIIFile(), and ReadXMLFile().

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

Extracts TimeSeries parameters from XML sctructure.

Sets parameters :

- typedata get the type of record data : 0 for ASCII, 1 for Binary,LittleEndian and 2 for Binary,BigEndian (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 3277 of file LISACODE-ConfigSim.cpp.

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

Referenced by ReadASCIIFile(), and ReadXMLFile().

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

Character to double conversion.

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

Definition at line 3089 of file LISACODE-ConfigSim.cpp.

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

10.3.3.68 char * ConfigSim::gXMLWord (ezxml_t *param*)

Definition at line 3193 of file LISACODE-ConfigSim.cpp.

References ezxml_t.

Referenced by ReadXMLFile().

10.3.3.69 void ConfigSim::majuscule (char &)

Definition at line 3674 of file LISACODE-ConfigSim.cpp.

Referenced by uppercase().

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

It returns TDIsName attribute.

Definition at line 379 of file LISACODE-ConfigSim.h.

References TDIsName.

Referenced by main().

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

Adds a noise specification structure.

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

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

Definition at line 3360 of file LISACODE-ConfigSim.cpp.

References NoisesData.

Referenced by ReadXMLFile().

10.3.3.72 void ConfigSim::NoisesCreation ()

Creation of noises.

Number of created noises is given by size of NoisesData.

Definition at line 3388 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().

10.3.3.73 void ConfigSim::ReadASCIIFile ()

Reads ASCII configuration file.

Data are read :

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

Definition at line 343 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(), LAG, LaserPower, NbMaxDelays, ezxml::next, NoisesCreation(), NoisesData, NoiseSpec::NStr, NoiseSpec::NType, NoiseSpec::NVal0, NoiseSpec::NVal01, NoiseSpec::NVal1, NoiseSpec::NVal2, OrbInitRot, OrbMove, OrbOrder, OrbStartTime, PhaMetFilterON, PhaMetFilterParam, PRECISION, MathUtils::rad2deg(), tDeltaTDIDelay, TDIDelayApprox, TDIInterp, TDIInterpUtilVal, TDIsName, TDIsPacks, TDIsPacksFact, tDisplay, tMax, tMaxDelay(), tMemNoiseFirst, tMemNoiseLast, tMemSig, tStepMes, tStepPhy, and USOs.

Referenced by ReadFile().

10.3.3.74 void ConfigSim::ReadFile ()

Reads configuration file.

Calls ReadXMLFile or ReadASCIIFile depending on ConfigFileName type

Definition at line 279 of file LISACODE-ConfigSim.cpp.

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

Referenced by ConfigSim().

10.3.3.75 void ConfigSim::ReadXMLFile ()

Reads XML configuration file.

Data are read :

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

Definition at line 1441 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(), LAG, LaserPower, NbMaxDelays, ezxml::next, NoisePlace(), NoisesCreation(), NoiseSpec::NType, NoiseSpec::NVal0, NoiseSpec::NVal01, NoiseSpec::NVal1, NoiseSpec::NVal2, OrbInitRot, OrbMove, OrbOrder, OrbStartTime, PhaMetFilterON, PhaMetFilterParam, PRECISION, MathUtils::rad2deg(), Simulator, tDeltaTDIDelay, TDIDelayApprox, TDIInterp, TDIInterpUtilVal, TDIParamName, TDIsName, TDIsPacks, TDIsPacksFact, tDisplay, TimeOffset, tMax, tMaxDelay(), tMemNoiseFirst, tMemNoiseLast, tStepMes, tStepPhy, uppercase(), USOs, XmlOutputFile, and Yr_SI.

Referenced by ReadFile().

10.3.3.76 int ConfigSim::testbyteorder ()

Definition at line 3653 of file LISACODE-ConfigSim.cpp.

Referenced by ReadFile().

10.3.3.77 double ConfigSim::tMaxDelay ()

Computes maximal time travel for one delay.

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

Definition at line 3607 of file LISACODE-ConfigSim.cpp.

References Armlength, c_SI, and tStepMes.

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

10.3.3.78 double ConfigSim::tMemNecInterpTDI ()

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

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

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

Definition at line 3629 of file LISACODE-ConfigSim.cpp.

References LAG, TDIInterp, TDIInterpUtilVal, and tStepMes.

Referenced by main().

10.3.3.79 double ConfigSim::tMinDelay ()

Computes minimal time travel for one delay.

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

Definition at line 3616 of file LISACODE-ConfigSim.cpp.

References Armlength, c_SI, and tStepMes.

Referenced by main().

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

Definition at line 3663 of file LISACODE-ConfigSim.cpp.

References majuscule().

Referenced by gXMLTimeSeries(), and ReadXMLFile().

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 285 of file LISACODE-ConfigSim.h.

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

10.3.4.8 int ConfigSim::FileEncodingPos [private]

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

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

10.3.4.9 int ConfigSim::FileEncodingSC1 [private]

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

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

10.3.4.10 int ConfigSim::FileEncodingSC2 [private]

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

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

10.3.4.11 int ConfigSim::FileEncodingSC3 [private]

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

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

10.3.4.12 int ConfigSim::FileEncodingTDI [private]

Definition at line 285 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 220 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 218 of file LISACODE-ConfigSim.h.

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

10.3.4.24 double ConfigSim::LaserPower [private]

Laser Power in Watts.

It specifies the beam power at the laser output.

Definition at line 201 of file LISACODE-ConfigSim.h.

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

10.3.4.25 int ConfigSim::NbMaxDelays [private]

Maximum Delays Number.

Definition at line 293 of file LISACODE-ConfigSim.h.

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

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

satellites noise

Definition at line 224 of file LISACODE-ConfigSim.h.

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

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

noise specifications

Definition at line 222 of file LISACODE-ConfigSim.h.

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

10.3.4.28 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().

10.3.4.29 int ConfigSim::OrbMove [private]

Satellites motion.

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

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

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

10.3.4.30 int ConfigSim::OrbOrder [private]

Order for time propagation computing.

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

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

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

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

10.3.4.31 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().

10.3.4.32 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 207 of file LISACODE-ConfigSim.h.

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

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

Phasemeter Filter Parameters.

Phasemeter Filter Parameters :

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

Definition at line 216 of file LISACODE-ConfigSim.h.

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

10.3.4.34 string ConfigSim::SC1ParamName [private]

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

Referenced by getSC1ParamName().

10.3.4.35 string ConfigSim::SC2ParamName [private]

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

Referenced by getSC2ParamName().

10.3.4.36 string ConfigSim::SC3ParamName [private]

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

Referenced by getSC3ParamName().

10.3.4.37 string ConfigSim::Simulator [private]

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

Referenced by getSimulator(), and ReadXMLFile().

10.3.4.38 char* ConfigSim::SystemEncoding [private]

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

Referenced by getSystemEncoding(), and ReadFile().

10.3.4.39 int ConfigSim::SystemEncoding_int [private]

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

10.3.4.40 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().

10.3.4.41 bool ConfigSim::TDIDelayApprox [private]

Approximated TDI delays computation or not.

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

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

10.3.4.42 INTERP ConfigSim::TDIInterp [private]

TDI interpolator type.

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

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

10.3.4.43 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 tMemNecInterpTDI().

10.3.4.44 string ConfigSim::TDIParamName [private]

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

Referenced by getTDIParamName(), and ReadXMLFile().

10.3.4.45 string ConfigSim::TDIParamNameType [private]

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

Referenced by getTDIParamNameType().

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

Name of TDI.

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

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

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

Vector of TDI data.

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

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

10.3.4.48 vector< vector<double> > ConfigSim::TDIsPacksFact [private]

Vector of TDI data.

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

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

10.3.4.49 ConfigSim::tDisplay [private]

Time step for screen display.

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

10.3.4.50 double ConfigSim::TimeOffset [private]

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

Referenced by getTimeOffset(), and ReadXMLFile().

10.3.4.51 ConfigSim::tMax [private]

It is the maximal simulation duration time.

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

10.3.4.52 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().

10.3.4.53 ConfigSim::tMemNoiseLast [private]

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

later noises are eleminated

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

10.3.4.54 ConfigSim::tMemSig [private]

Duration of satellite memory for signal storage.

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

10.3.4.55 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().

10.3.4.56 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().

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

satellites USO time

Definition at line 226 of file LISACODE-ConfigSim.h.

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

10.3.4.58 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_ERRL]

10.6.1 Member Data Documentation

10.6.1.1 char*** ezxml_root::attr

Definition at line 51 of file ezxml.c.

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

10.6.1.2 ezxml_t ezxml_root::cur

Definition at line 44 of file ezxml.c.

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

10.6.1.3 char* ezxml_root::e

Definition at line 49 of file ezxml.c.

Referenced by ezxml_free(), and ezxml_parse_str().

10.6.1.4 char** ezxml_root::ent

Definition at line 50 of file ezxml.c.

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

10.6.1.5 char ezxml_root::err[EZXML_ERRL]

Definition at line 54 of file ezxml.c.

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

10.6.1.6 size_t ezxml_root::len

Definition at line 46 of file ezxml.c.

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

10.6.1.7 char* ezxml_root::m

Definition at line 45 of file ezxml.c.

Referenced by ezxml_free(), and ezxml_parse_str().

10.6.1.8 char*** ezxml_root::pi

Definition at line 52 of file ezxml.c.

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

10.6.1.9 char* ezxml_root::s

Definition at line 48 of file ezxml.c.

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

10.6.1.10 short ezxml_root::standalone

Definition at line 53 of file ezxml.c.

Referenced by ezxml_internal_dtd(), and ezxml_proc_inst().

10.6.1.11 char* ezxml_root::u

Definition at line 47 of file ezxml.c.

Referenced by ezxml_free(), and ezxml_parse_str().

10.6.1.12 struct ezxml ezxml_root::xml

Definition at line 43 of file ezxml.c.

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

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

- ezxml.c

10.7 Filter Class Reference

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

10.7.1 Detailed Description

filter management class.

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

Public Member Functions

- Filter ()

  Constructs an instance and initializes it with default values.

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

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

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

  Constructs an instance and initializes it with inputs.

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

  Constructs an instance and initializes it with inputs.

- ~Filter ()

  Destructor.

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

  Initializes an instance with default values and inputs.

- int getDepth ()

  Gives maximum of alpha or beta attribute size.

- int getNbDataStab ()

  Returns NbDataStab attribute.

- vector< vector< double > > getAlpha ()

  Returns alpha attribute.

- vector< vector< double > > getBeta ()

  Returns beta attribute.

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

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

Protected Attributes

- vector< vector< double > > alpha

  Alpha parameters list.

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

  Number of data for stabilization.

- vector< vector< double > > TmpData

  Temporary data.

10.7.2 Constructor & Destructor Documentation

10.7.2.1 Filter::Filter ()

Constructs an instance and initializes it with default values.

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

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

References alpha, beta, NbDataStab, and TmpData.

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

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

init method is called to set attributes :

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

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

References init().

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

Constructs an instance and initializes it with inputs.

init method is called to set attributes :

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

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

References init().

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

Constructs an instance and initializes it with inputs.

Parameters:

fe sampling frequency

at attenuation

bp oscillations in bandwidth

fb low transition frequency

fa high transition frequency

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

Then, CalcEllipticFilter4LISACode outputs are used :

- CellsCoef Cells coefficients
- NCellsOut number of cells

init method is called to set attributes :

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

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

References CalcEllipticFilter4LISACode(), and init().

10.7.2.5 Filter::~Filter ()

Destructor.

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

10.7.3 Member Function Documentation

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

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

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

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

for $i = 0, \ldots, StartBin$

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

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

Then last TmpData array is copied into FilterData.

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

References alpha, beta, and TmpData.

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

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

Returns alpha attribute.

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

References alpha.

Referenced by NoiseTwoFilter::getFilterAlpha(), NoiseOof::getFilterAlpha(), and NoiseFilter::getFilterAlpha().

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

Returns beta attribute.

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

References beta.

Referenced by NoiseTwoFilter::getFilterBeta(), NoiseOof::getFilterBeta(), and NoiseFilter::getFilterBeta().

10.7.3.4 int Filter::getDepth ()

Gives maximum of alpha or beta attribute size.

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

References alpha, beta, and MAX.

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

10.7.3.5 int Filter::getNbDataStab ()

Returns NbDataStab attribute.

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

References NbDataStab.

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

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

Initializes an instance with default values and inputs.

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

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

References alpha, beta, NbDataStab, and TmpData.

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

10.7.4 Member Data Documentation

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

Alpha parameters list.

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

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

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

Beta parameters list.

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

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

10.7.4.3 int Filter::NbDataStab [protected]

Number of data for stabilization.

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

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

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

Temporary data.

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

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

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

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

10.8 Geometry Class Reference

`#include <LISACODE-Geometry.h>`

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)

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

- double getL0 ()

  Returns L0m attribute.

- double gett0 ()

  Returns t0 attribute.

- 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, $arot = \frac{2 \cdot \pi}{3}$.

- 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:

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

Definition at line 363 of file LISACODE-Geometry.cpp.

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

10.8.3.2 double Geometry::getL0 () [inline]

Returns L0m attribute.

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

References L0m.

10.8.3.3 double Geometry::gett0 () [inline]

Returns t0 attribute.

Definition at line 108 of file LISACODE-Geometry.h.

References t0.

Referenced by BackgroundGalactic::deltanu().

10.8.3.4 double Geometry::gettRangeStoreDelay () [inline]

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

References tRangeStoreDelay.

10.8.3.5 double Geometry::gettRangeStorePos () [inline]

Definition at line 110 of file LISACODE-Geometry.h.

References tRangeStorePos.

10.8.3.6 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 418 of file LISACODE-Geometry.cpp.

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

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

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

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

tdelay method is used to set DelayStore attribute

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

If (abs(trec-tStoreDelay)>tRangeStorePos)

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

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

Definition at line 443 of file LISACODE-Geometry.cpp.

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

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

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

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

Inputs are checked.

tdelay method is used to set DelayStore attribute

position method is used to set SCposStore attribute

Attributes are set :

- t0 = t0_n input (expected to be positive or null)
- rot0 = rot0_n input
- L0m = L0m_n input (expected to be positive or null)
- order_default = order_default_n input
- move = move_n input (expected to be 0 or 1)
- 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, and tRangeStorePos_default.

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

10.8.3.9 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 153 of file LISACODE-Geometry.cpp.

References Vect::p.

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

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

Definition at line 125 of file LISACODE-Geometry.cpp.

References t0.

10.8.3.11 void Geometry::settRangeStoreDelay (double *tRangeStoreDelay_n*)

Definition at line 136 of file LISACODE-Geometry.cpp.

References tRangeStoreDelay.

10.8.3.12 void Geometry::settRangeStorePos (double *tRangeStorePos_n*)

Definition at line 131 of file LISACODE-Geometry.cpp.

References tRangeStorePos.

10.8.3.13 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:
$delay = \sum_{i=0}^{order} c_i$,
where :
ri is computed using position method with rec and trec inputs,
vi is computed using velocity method with rec and trec inputs, then divided by c_SI,
rj is computed using position method with em and trec inputs,
vj is computed using velocity method with em and trec inputs, then divided by c_SI,
$\overrightarrow{r_{ij}} = \overrightarrow{r_j} - \overrightarrow{r_i}$,
$c_0 = \frac{\left\|\overrightarrow{r_{ij}}\right\|}{C}$,

$\overrightarrow{n} = -\frac{\overrightarrow{r_{ij}}}{\|\overrightarrow{r_{ij}}\|}$,
$c_1 = t_{ij} \cdot \overrightarrow{n} \cdot \overrightarrow{v_j}$,
$c_2 = c_{21} + c_{22} + c_{23}$,
$c_{21} = \frac{t_{ij}}{2} \cdot (\overrightarrow{v_j}^2 + (\overrightarrow{n} \cdot \overrightarrow{v_j})^2)$,
$c_{22} = -\frac{{t_{ij}}^2}{2} \cdot C \cdot RSchw$,
$c_{23} = -\frac{RSchw}{C} \cdot (1+\gamma_u) \cdot log\left(\frac{\sqrt{(C \cdot t_{ij} \cdot \overrightarrow{n} \cdot \overrightarrow{r_{ij}})^2 + {r_i}^2 + C \cdot t_{ij} + \overrightarrow{n} \cdot \overrightarrow{r_i}}}{\overrightarrow{n} \cdot \overrightarrow{r_i} + \|\overrightarrow{r_i}\|}\right)$.
c_SI and gamma_u constants are used.

Definition at line 224 of file LISACODE-Geometry.cpp.

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

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

10.8.3.14 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 :

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

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

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

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

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

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

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

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

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

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

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

c_SI and gamma_u constants are used

Definition at line 301 of file LISACODE-Geometry.cpp.

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

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

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

position method is used to set SCposStore attribute

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

Definition at line 392 of file LISACODE-Geometry.cpp.

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

Referenced by main().

10.8.3.16 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 187 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, $arot = \frac{2\cdot\pi}{3}$.

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 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 GeometryAnalytic::exanom(), 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 initGlobal(), and tdelay().

10.8.4.10 Geometry::rot0 [protected]

Initial rotation (radians).

Referenced by 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 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 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 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().

10.8.4.19 double Geometry::tStorePos [protected]

last position computation time.

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

Referenced by gposition().

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

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

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

- 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)

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

- double getL0 ()

  Returns L0m attribute.

- double gett0 ()

  Returns t0 attribute.

- 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

  Orbital parameter.

- double nu

  Orbital angular constant.

- double tmu

  Orbital constant.

- double cmu

  Orbital constant.

- double smu

  Orbital constant.

- double e

  Eccentricity.

- double sqrtee

  Eccentricity derived constant.

- double arot

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

- 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:

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

Definition at line 363 of file LISACODE-Geometry.cpp.

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

10.9.3.2 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 :

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

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

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

Referenced by position(), and velocity().

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

Returns L0m attribute.

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

References Geometry::L0m.

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

Returns t0 attribute.

Definition at line 108 of file LISACODE-Geometry.h.

References Geometry::t0.

Referenced by BackgroundGalactic::deltanu().

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

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

References Geometry::tRangeStoreDelay.

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

Definition at line 110 of file LISACODE-Geometry.h.

References Geometry::tRangeStorePos.

10.9.3.7 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 418 of file LISACODE-Geometry.cpp.

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

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

10.9.3.8 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)

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

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

Definition at line 443 of file LISACODE-Geometry.cpp.

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

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

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

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

Inputs are checked.

tdelay method is used to set DelayStore attribute

position method is used to set SCposStore attribute

Attributes are set :

- t0 = t0_n input (expected to be positive or null)
- rot0 = rot0_n input
- L0m = L0m_n input (expected to be positive or null)
- order_default = order_default_n input
- move = move_n input (expected to be 0 or 1)
- alpha = $\frac{L0m}{2 \cdot Rgc}$
- nu = $\frac{\pi}{3} + \frac{5 \cdot \alpha}{8}$

- tmu =$\frac{\alpha \cdot sin(\nu)}{sin(\frac{\pi}{3})} + \alpha \cdot cos(\nu)$
- cmu = $\frac{1}{\sqrt{1+t_\mu^2}}$
- smu =$t_\mu \cdot c_\mu$
- e =$\sqrt{1 + \frac{4}{\sqrt{3}} \cdot cos(\nu) \cdot \alpha + \frac{4}{3}\alpha^2} - 1$
- sqrtee =$\sqrt{1 - e^2}$
- arot =$\frac{2 \cdot \pi}{3}$
- rot :$rot[i] = i \cdot arot \cdot rot0$ for i=1,2,3
- crot : $crot[i] = cos(rot_i)$ for i=1,2,3
- srot : $srot[i = 0, 1, 2] = sin(rot_i)$ for i=1,2,3
- SCposStore :$SCposStore[i] = position(i, 0)$ for i=1,2,3,4
- DelayStore : for i=1,2,3,4 $\begin{cases} DelayStore[i-1] = tdelay(i, i, mod((i+1), 3) + 1, 2, 0) \\ DelayStore[i+2] = tdelay(i, i, mod(i, 3) + 1, 2, 0) \end{cases}$
- 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().

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

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

Inputs are checked.

tdelay method is used to set DelayStore attribute

position method is used to set SCposStore attribute

Attributes are set :

- t0 = t0_n input (expected to be positive or null)
- rot0 = rot0_n input
- L0m = L0m_n input (expected to be positive or null)
- order_default = order_default_n input
- move = move_n input (expected to be 0 or 1)
- 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, and tRangeStorePos_default.

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

10.9.3.11 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 233 of file LISACODE-GeometryAnalytic.cpp.

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

Referenced by init().

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

Definition at line 125 of file LISACODE-Geometry.cpp.

References Geometry::t0.

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

Definition at line 136 of file LISACODE-Geometry.cpp.

References Geometry::tRangeStoreDelay.

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

Definition at line 131 of file LISACODE-Geometry.cpp.

References Geometry::tRangeStorePos.

10.9.3.15 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:
$delay = \sum_{i=0}^{order} c_i$,
where :
ri is computed using position method with rec and trec inputs,
vi is computed using velocity method with rec and trec inputs, then divided by c_SI,
rj is computed using position method with em and trec inputs,
vj is computed using velocity method with em and trec inputs, then divided by c_SI,
$\overrightarrow{r_{ij}} = \overrightarrow{r_j} - \overrightarrow{r_i}$,
$c_0 = \frac{\left\|\overrightarrow{r_{ij}}\right\|}{C}$,
$\overrightarrow{n} = -\frac{\overrightarrow{r_{ij}}}{\left\|\overrightarrow{r_{ij}}\right\|}$,
$c_1 = t_{ij} \cdot \overrightarrow{n} \cdot \overrightarrow{v_j}$,
$c_2 = c_{21} + c_{22} + c_{23}$,
$c_{21} = \frac{t_{ij}}{2} \cdot (\overrightarrow{v_j}^2 + (\overrightarrow{n} \cdot \overrightarrow{v_j})^2)$,
$c_{22} = -\frac{{t_{ij}}^2}{2} \cdot C \cdot RSchw$,
$c_{23} = -\frac{RSchw}{C} \cdot (1 + \gamma_u) \cdot log\left(\frac{\sqrt{(C \cdot t_{ij} \cdot \overrightarrow{n} \cdot \overrightarrow{r_{ij}})^2 + {r_i}^2 + C \cdot t_{ij} + \overrightarrow{n} \cdot \overrightarrow{r_i}}}{\overrightarrow{n} \cdot \overrightarrow{r_i} + \left\|\overrightarrow{r_i}\right\|}\right)$.
c_SI and gamma_u constants are used.

Definition at line 224 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().

10.9.3.16 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 :

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

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

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

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

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

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

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

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

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

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

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

c_SI and gamma_u constants are used

Definition at line 301 of file LISACODE-Geometry.cpp.

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

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

Definition at line 392 of file LISACODE-Geometry.cpp.

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

Referenced by main().

10.9.3.18 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: $\overrightarrow{v} = \begin{pmatrix} \dot{\psi} \cdot (-spsi \cdot cmu \cdot crot_{nb-1} - sqrtee \cdot cpsi \cdot srot_{nb-1}) \\ \dot{\psi} \cdot (-spsi \cdot cmu \cdot srot_{nb-1} + sqrtee \cdot cpsi \cdot crot_{nb-1}) \\ \dot{\psi} \cdot smu \cdot spsi \end{pmatrix}$,

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

Reimplemented from Geometry.

Definition at line 270 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]

Orbital parameter.

Reimplemented from Geometry.

Referenced by init().

10.9.4.2 GeometryAnalytic::arot [protected]

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

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 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::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 exanom(), Geometry::initGlobal(), GeometryMLDC::position(), GeometryFile::position(), GeometryMLDC::velocity(), and GeometryFile::velocity().

10.9.4.9 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::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 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 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::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::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().

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().

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.

- 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)

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

- double getL0 ()

  Returns L0m attribute.

- double gett0 ()

  Returns t0 attribute.

- 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, $arot = \frac{2 \cdot \pi}{3}$.

- 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:

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

Definition at line 363 of file LISACODE-Geometry.cpp.

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

10.10.3.2 Couple GeometryFile::exanom (int *nb*, double *ts*)

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

Returns L0m attribute.

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

References Geometry::L0m.

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

Returns t0 attribute.

Definition at line 108 of file LISACODE-Geometry.h.

References Geometry::t0.

Referenced by BackgroundGalactic::deltanu().

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

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

References Geometry::tRangeStoreDelay.

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

Definition at line 110 of file LISACODE-Geometry.h.

References Geometry::tRangeStorePos.

10.10.3.7 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 418 of file LISACODE-Geometry.cpp.

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

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

10.10.3.8 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)

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

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

Definition at line 443 of file LISACODE-Geometry.cpp.

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

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

10.10.3.9 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().

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

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

Inputs are checked.

tdelay method is used to set DelayStore attribute

position method is used to set SCposStore attribute

Attributes are set :

- t0 = t0_n input (expected to be positive or null)
- rot0 = rot0_n input
- L0m = L0m_n input (expected to be positive or null)
- order_default = order_default_n input
- move = move_n input (expected to be 0 or 1)
- 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, and tRangeStorePos_default.

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

10.10.3.11 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 201 of file LISACODE-GeometryFile.cpp.

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

Referenced by init().

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

Definition at line 125 of file LISACODE-Geometry.cpp.

References Geometry::t0.

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

Definition at line 136 of file LISACODE-Geometry.cpp.

References Geometry::tRangeStoreDelay.

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

Definition at line 131 of file LISACODE-Geometry.cpp.

References Geometry::tRangeStorePos.

10.10.3.15 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:

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

Definition at line 224 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().

10.10.3.16 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 :

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

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

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

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

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

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

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

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

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

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

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

c_SI and gamma_u constants are used

Definition at line 301 of file LISACODE-Geometry.cpp.

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

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

Definition at line 392 of file LISACODE-Geometry.cpp.

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

Referenced by main().

10.10.3.18 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 254 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, $arot = \frac{2 \cdot \pi}{3}$.

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::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 GeometryAnalytic::exanom(), Geometry::initGlobal(), GeometryMLDC::position(), position(), GeometryMLDC::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::initGlobal(), and Geometry::tdelay().

10.10.4.12 Geometry::rot0 [protected, inherited]

Initial rotation (radians).

Referenced by 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 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::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::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().

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().

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

- 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)

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

- double getL0 ()

  Returns L0m attribute.

- double gett0 ()

  Returns t0 attribute.

- 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

- 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

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

- 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:

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

Definition at line 363 of file LISACODE-Geometry.cpp.

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

10.11.3.2 Couple GeometryMLDC::exanom (int *nb*, double *ts*)

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

Returns L0m attribute.

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

References Geometry::L0m.

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

Returns t0 attribute.

Definition at line 108 of file LISACODE-Geometry.h.

References Geometry::t0.

Referenced by BackgroundGalactic::deltanu().

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

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

References Geometry::tRangeStoreDelay.

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

Definition at line 110 of file LISACODE-Geometry.h.

References Geometry::tRangeStorePos.

10.11.3.7 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 418 of file LISACODE-Geometry.cpp.

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

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

10.11.3.8 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)

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

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

Definition at line 443 of file LISACODE-Geometry.cpp.

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

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

10.11.3.9 void GeometryMLDC::init (double *t0_n*, double *rot0_n*, double *L0m_n*, int *order_default_n*, int *move_n*, double *tStep*)

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

Inputs are checked.

tdelay method is used to set DelayStore attribute

position method is used to set SCposStore attribute

Attributes are set :

- 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 131 of file LISACODE-GeometryMLDC.cpp.

References Geometry::arot, crot, Geometry::DelayStore, e_mldc, Geometry::initGlobal(), position(), rot, Geometry::rot0, Geometry::SCposStore, sqrt_3, srot, and Geometry::tdelay().

Referenced by GeometryMLDC().

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

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

Inputs are checked.

tdelay method is used to set DelayStore attribute

position method is used to set SCposStore attribute

Attributes are set :

- t0 = t0_n input (expected to be positive or null)

- rot0 = rot0_n input
- L0m = L0m_n input (expected to be positive or null)
- order_default = order_default_n input
- move = move_n input (expected to be 0 or 1)
- 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, and tRangeStorePos_default.

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

10.11.3.11 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 173 of file LISACODE-GeometryMLDC.cpp.

References Geometry::alpha, au_m, crot, e_mldc, Geometry::move, omega, Vect::p, rot, sqrt_3, srot, and Geometry::t0.

Referenced by init().

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

Definition at line 125 of file LISACODE-Geometry.cpp.

References Geometry::t0.

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

Definition at line 136 of file LISACODE-Geometry.cpp.

References Geometry::tRangeStoreDelay.

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

Definition at line 131 of file LISACODE-Geometry.cpp.

References Geometry::tRangeStorePos.

10.11.3.15 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:
$delay = \sum_{i=0}^{order} c_i$,
where :
ri is computed using position method with rec and trec inputs,
vi is computed using velocity method with rec and trec inputs, then divided by c_SI,
rj is computed using position method with em and trec inputs,
vj is computed using velocity method with em and trec inputs, then divided by c_SI,
$\overrightarrow{r_{ij}} = \overrightarrow{r_j} - \overrightarrow{r_i}$,
$c_0 = \frac{\|\overrightarrow{r_{ij}}\|}{C}$,
$\overrightarrow{n} = -\frac{\overrightarrow{r_{ij}}}{\|\overrightarrow{r_{ij}}}$,
$c_1 = t_{ij} \cdot \overrightarrow{n} \cdot \overrightarrow{v_j}$,
$c_2 = c_{21} + c_{22} + c_{23}$,
$c_{21} = \frac{t_{ij}}{2} \cdot (\overrightarrow{v_j}^2 + (\overrightarrow{n} \cdot \overrightarrow{v_j})^2)$,
$c_{22} = -\frac{{t_{ij}}^2}{2} \cdot C \cdot RSchw$,
$c_{23} = -\frac{RSchw}{C} \cdot (1 + \gamma_u) \cdot log\left(\frac{\sqrt{(C \cdot t_{ij} \cdot \overrightarrow{n} \cdot \overrightarrow{r_{ij}})^2 + {r_i}^2 + C \cdot t_{ij} + \overrightarrow{n} \cdot \overrightarrow{r_i}}}{\overrightarrow{n} \cdot \overrightarrow{r_i} + \|\overrightarrow{r_i}\|}\right)$.
c_SI and gamma_u constants are used.

Definition at line 224 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().

10.11.3.16 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 :

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

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

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

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

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

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

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

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

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

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

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

c_SI and gamma_u constants are used

Definition at line 301 of file LISACODE-Geometry.cpp.

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

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

Definition at line 392 of file LISACODE-Geometry.cpp.

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

Referenced by main().

10.11.3.18 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 205 of file LISACODE-GeometryMLDC.cpp.

References Geometry::alpha, 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 Geometry::arot [protected, inherited]

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

Reimplemented in GeometryAnalytic.

Referenced by init().

10.11.4.3 Geometry::cmu [protected, inherited]

Orbital constant.

Reimplemented in GeometryAnalytic.

10.11.4.4 GeometryMLDC::crot [protected]

Satellites phase cosinus.

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

10.11.4.5 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(), init(), GeometryFile::init(), and GeometryAnalytic::init().

10.11.4.6 Geometry::e [protected, inherited]

Eccentricity.

Reimplemented in GeometryAnalytic.

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

10.11.4.7 double GeometryMLDC::e_mldc [protected]

excentricity for MLDC orbits

Definition at line 54 of file LISACODE-GeometryMLDC.h.

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

10.11.4.8 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::getL0(), GeometryAnalytic::init(), and Geometry::initGlobal().

10.11.4.9 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 GeometryAnalytic::exanom(), Geometry::initGlobal(), position(), GeometryFile::position(), velocity(), and GeometryFile::velocity().

10.11.4.10 Geometry::nu [protected, inherited]

Orbital angular constant.

Reimplemented in GeometryAnalytic.

10.11.4.11 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::initGlobal(), and Geometry::tdelay().

10.11.4.12 GeometryMLDC::rot [protected]

Satellites phase.

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

10.11.4.13 Geometry::rot0 [protected, inherited]

Initial rotation (radians).

Referenced by init(), GeometryAnalytic::init(), and Geometry::initGlobal().

10.11.4.14 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().

10.11.4.15 Geometry::smu [protected, inherited]

Orbital constant.

Reimplemented in GeometryAnalytic.

10.11.4.16 double GeometryMLDC::sqrt_3 [protected]

sqrt(3) for MLDC orbits

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

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

10.11.4.17 Geometry::sqrtee [protected, inherited]

Eccentricity derived constant.

Reimplemented in GeometryAnalytic.

10.11.4.18 GeometryMLDC::srot [protected]

Satellites phase sinus.

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

10.11.4.19 Geometry::t0 [protected, inherited]

Initial time (seconds).

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

10.11.4.20 Geometry::tmu [protected, inherited]

Orbital constant.

Reimplemented in GeometryAnalytic.

10.11.4.21 double Geometry::tRangeStoreDelay [protected, inherited]

Last delay storage time.

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

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

10.11.4.22 double Geometry::tRangeStorePos [protected, inherited]

last position storage time.

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

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

10.11.4.23 double Geometry::tStoreDelay [protected, inherited]

Last delay computation time.

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

Referenced by Geometry::gtdelay().

10.11.4.24 double Geometry::tStorePos [protected, inherited]

last position computation time.

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

Referenced by Geometry::gposition().

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:

$$\beta_n \in [-\frac{\pi}{2}, -\frac{\pi}{2}]$$
$$\lambda_n \in [0, 2 \cdot \pi]$$

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

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

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

10.12.2.3 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:

$$\beta_n \in [-\frac{\pi}{2}, -\frac{\pi}{2}]$$
$$\lambda_n \in [0, 2 \cdot \pi]$$
$$AnglPol_n \in [0, 2 \cdot \pi]$$

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

Definition at line 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.

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

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, GWMono, GWNew, GWNewton2, GWPeriGate, and GWSto.

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, GWMono, GWNew, GWNewton2, GWPeriGate, and GWSto.

Definition at line 105 of file LISACODE-GW.cpp.

References Beta, and Lambda.

10.12.3.9 double GW::hc (double *t*) [virtual]

Return $h_\times(t)$.

Virtual unused method. Returned value is constant : 0.

Reimplemented in GWBinary, GWCusp, GWFastSpinBBH, GWFile, GWMono, GWNew, GWNewton2, GWPeriGate, and GWSto.

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, GWMono, GWNew, GWNewton2, GWPeriGate, and GWSto.

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, GWMono, GWNew, GWNewton2, GWPeriGate, and GWSto.

Definition at line 200 of file LISACODE-GW.cpp.

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

$*$ Input is checked:

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

Reimplemented in GWCusp, GWMono, and GWNew.

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

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

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, GWMono, GWNew, GWNewton2, GWPeriGate, and GWSto.

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 GWCusp, GWMono, GWNew, and GWPeriGate.

Definition at line 51 of file LISACODE-GW.h.

Referenced by getAnglPol(), GWNewton2::getParam(), GWFile::getParam(), GWBinary::getParam(), GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), setAnglPol(), GWNewton2::setParam(), GWFile::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(), GWSto::getParam(), GWPeriGate::getParam(), GWNewton2::getParam(), GWNew::getParam(), GWMono::getParam(), GWFile::getParam(), GWFastSpinBBH::getParam(), GWCusp::getParam(), GWBinary::getParam(), getParam(), GW(), GWBinary::GWBinary(), GWFastSpinBBH::GWFastSpinBBH(), GWNewton2::GWNewton2(), GWFastSpinBBH::init(), setBeta(), setDirProp(), GWSto::setParam(), GWPeriGate::setParam(), GWNewton2::setParam(), GWNew::setParam(), GWMono::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(), GWSto::getParam(), GWPeriGate::getParam(), GWNewton2::getParam(), GWNew::getParam(), GWMono::getParam(), GWFile::getParam(), GWFastSpinBBH::getParam(), GWCusp::getParam(), GWBinary::getParam(), getParam(), GW(), GWBinary::GWBinary(), GWFastSpinBBH::GWFastSpinBBH(), GWNewton2::GWNewton2(), GWFastSpinBBH::init(), setDirProp(), setLambda(), GWSto::setParam(), GWPeriGate::setParam(), GWNewton2::setParam(), GWNew::setParam(), GWMono::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(), 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

$$\beta = \beta_n \text{ rad}$$

$$\lambda = \lambda_n \text{ rad}$$

$$AnglPol = AnglPol_n \text{ rad}$$

$$M_1 = bM_1 \cdot M_{Sun} \text{ kg}$$

$$M_2 = bM_2 \cdot M_{Sun} \text{ kg, where } M_{Sun} = 1.9889 \cdot 10^{30} \text{ kg}$$

$$f_{orb} = bf_{orb} \text{ Hz}$$

$$inc = binc \text{ rad}$$

$$\phi_0 = b\phi_0 \text{ rad}$$

$$r = br \cdot kpc_m \text{ m, where } kpc_m = 3.0856675807 \cdot 10^{19} \text{ m}$$

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

Definition at line 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 :

$$A_p = A \cdot (1 + cos^2(inc))$$

$$A_c = 2 \cdot A \cdot cos(inc)$$

where binary components orbital separation is :

$$R = (\frac{G \cdot (M_1 + M_2)}{(2 \cdot \pi \cdot f_{orb})^2})^{\frac{1}{3}} \text{ m}$$

and gravitational wave intrinsic amplitude is:

$$A = \frac{2 \cdot G^2 \cdot M_1 \cdot M_2}{C^4 \cdot r \cdot R}$$

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:

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

Reimplemented in GWCusp, 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

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

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

References GW::Beta, GW::DirProp, GW::Lambda, and PRECISION.

10.13.3.21 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 GWCusp, GWMono, GWNew, and GWPeriGate.

Definition at line 51 of file LISACODE-GW.h.

Referenced by GW::getAnglPol(), GWNewton2::getParam(), GWFile::getParam(), getParam(), GW::GW(), GWBinary(), GWNewton2::GWNewton2(), GW::setAnglPol(), GWNewton2::setParam(), GWFile::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(), GWFastSpinBBH::GWFastSpinBBH(), 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(), GWBi-

nary(), GWFastSpinBBH::GWFastSpinBBH(), 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(), 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 36 of file LISACODE-GWCusp.h.

Public Member Functions

- GWCusp ()
- GWCusp (double Beta_n, double Lambda_n, double AnglPol_n, double Freq_n, double Amplhp_n, double Amplhc_n)
- GWCusp (double Beta_n, double Lambda_n, double AnglPol_n, double Freq_n, double Amplhp_n, double Amplhc_n, double Phi0hp_n, double Phi0hc_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 -> 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_{x} polarisation component amplitude

- double AnglPol

  Polarisation angle (rad).

- double Phi0hp

  Initial phase of h_plus polarisation component.

- double Phi0hc

  Initial phase of h_cross polarisation component.

- double Beta

  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.

10.14.2 Constructor & Destructor Documentation

10.14.2.1 GWCusp::GWCusp ()

- Freq = 0
- Amplhp = 0
- Amplhc = 0
- AnglPol = 0
- Phi0hp = 0
- Phi0hc = 0

Definition at line 28 of file LISACODE-GWCusp.cpp.

References Amplhc, Amplhp, AnglPol, Freq, Phi0hc, and Phi0hp.

10.14.2.2 GWCusp::GWCusp (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 47 of file LISACODE-GWCusp.cpp.

References Amplhc, Amplhp, Freq, Phi0hc, and Phi0hp.

10.14.2.3 GWCusp::GWCusp (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 74 of file LISACODE-GWCusp.cpp.

References Amplhc, Amplhp, Freq, Phi0hc, and Phi0hp.

10.14.2.4 GWCusp::~GWCusp ()

Definition at line 93 of file LISACODE-GWCusp.cpp.

10.14.3 Member Function Documentation

10.14.3.1 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().

10.14.3.2 void GWCusp::DispTempVal (double *t*, ostream * *OutDisp*) [virtual]

Reimplemented from GW.

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

10.14.3.3 double GWCusp::getAmplhc () const [inline]

Definition at line 93 of file LISACODE-GWCusp.h.

References Amplhc.

10.14.3.4 double GWCusp::getAmplhp () const [inline]

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

References Amplhp.

10.14.3.5 double GW::getAnglPol () [inline, inherited]

Definition at line 83 of file LISACODE-GW.h.

References GW::AnglPol.

10.14.3.6 double GWCusp::getAnglPol () const [inline]

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

References AnglPol.

10.14.3.7 double GW::getBeta () const [inline, inherited]

Definition at line 76 of file LISACODE-GW.h.

References GW::Beta.

10.14.3.8 vector<double> GW::getDirProp () const [inline, inherited]

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

References GW::DirProp.

10.14.3.9 double GWCusp::getFreq () const [inline]

Definition at line 91 of file LISACODE-GWCusp.h.

References Freq.

10.14.3.10 double GW::getLambda () const [inline, inherited]

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

References GW::Lambda.

10.14.3.11 int GW::getNParam () [inline, inherited]

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

References GW::NParam.

10.14.3.12 double GWCusp::getParam (int *iP*) [virtual]

Return parameters specified by iP (see function for setParam the iP code).

Reimplemented from GW.

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

References Amplhc, Amplhp, AnglPol, GW::Beta, Freq, GW::Lambda, Phi0hc, and Phi0hp.

10.14.3.13 double GWCusp::getPhi0hc () const [inline]

Definition at line 96 of file LISACODE-GWCusp.h.

References Phi0hc.

10.14.3.14 double GWCusp::getPhi0hp () const [inline]

Definition at line 95 of file LISACODE-GWCusp.h.

References Phi0hp.

10.14.3.15 double GWCusp::hc (double *t*) [virtual]

Return $h_\times(t)$.

$$\text{returned value} = Amplhc \cdot sin(2 \cdot \pi \cdot Freq \cdot t + Phi0hc)$$

Reimplemented from GW.

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

References Amplhc, Freq, and Phi0hc.

10.14.3.16 double GWCusp::hp (double *t*) [virtual]

Return $h_+(t)$.

$$\text{returned value} = Amplhp \cdot sin(2 \cdot \pi \cdot Freq \cdot t + Phi0hp)$$

Reimplemented from GW.

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

References Amplhp, Freq, and Phi0hp.

10.14.3.17 void GWCusp::init () [virtual]

Initialization : ...

Reimplemented from GW.

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

10.14.3.18 void GWCusp::setAmplhc (double *Amplhc_n*)

Definition at line 179 of file LISACODE-GWCusp.cpp.

References Amplhc.

10.14.3.19 void GWCusp::setAmplhp (double *Amplhp_n*)

Definition at line 174 of file LISACODE-GWCusp.cpp.

References Amplhp.

10.14.3.20 void GWCusp::setAnglPol (double *AnglPol_n*)

∗ Input is checked:

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

Reimplemented from GW.

10.14.3.21 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().

10.14.3.22 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

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

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

References GW::Beta, GW::DirProp, GW::Lambda, and PRECISION.

10.14.3.23 void GWCusp::setFreq (double *Freq_n*)

Input is checked:*Freq_n* must be positive or null

Definition at line 167 of file LISACODE-GWCusp.cpp.

References Freq.

10.14.3.24 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.

10.14.3.25 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 -> 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 102 of file LISACODE-GWCusp.cpp.

References Amplhc, Amplhp, AnglPol, GW::Beta, Freq, GW::Lambda, Phi0hc, and Phi0hp.

10.14.3.26 void GWCusp::setPhi0hc (double *Phi0hc_n*) [inline]

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

References Phi0hc.

10.14.3.27 void GWCusp::setPhi0hp (double *Phi0hp_n*) [inline]

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

References Phi0hp.

10.14.4 Member Data Documentation

10.14.4.1 double GWCusp::Amplhc [protected]

h_{x} polarisation component amplitude

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

Referenced by getAmplhc(), getParam(), GWCusp(), hc(), setAmplhc(), and setParam().

10.14.4.2 double GWCusp::Amplhp [protected]

h_{+} polarisation component amplitude

Definition at line 42 of file LISACODE-GWCusp.h.

Referenced by getAmplhp(), getParam(), GWCusp(), hp(), setAmplhp(), and setParam().

10.14.4.3 double GWCusp::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-GWCusp.h.

Referenced by getAnglPol(), getParam(), GWCusp(), and setParam().

10.14.4.4 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(), getParam(), GWBinary::getParam(), GW::getParam(), GW::GW(), GWBinary::GWBinary(), GWFastSpinBBH::GWFastSpinBBH(), GWNewton2::GWNewton2(), GWFastSpinBBH::init(), GW::setBeta(), GW::setDirProp(), GWSto::setParam(), GWPeriGate::setParam(), GWNewton2::setParam(), GWNew::setParam(), GWMono::setParam(), GWFile::setParam(), GWFastSpinBBH::setParam(), setParam(), GWBinary::setParam(), and GW::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 double GWCusp::Freq [protected]

Frequency.

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

Referenced by getFreq(), getParam(), GWCusp(), hc(), hp(), setFreq(), and setParam().

10.14.4.7 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(), GWFile::getParam(), GWFastSpinBBH::getParam(), getParam(), GWBinary::getParam(), GW::getParam(), GW::GW(), GWBinary::GWBinary(), GWFastSpinBBH::GWFastSpinBBH(), GWNewton2::GWNewton2(), GWFast-

SpinBBH::init(), GW::setDirProp(), GW::setLambda(), GWSto::setParam(), GWPeriGate::setParam(), GWNewton2::setParam(), GWNew::setParam(), GWMono::setParam(), GWFile::setParam(), GWFastSpinBBH::setParam(), setParam(), GWBinary::setParam(), and GW::setParam().

10.14.4.8 int GW::NParam [protected, inherited]

Number of parameters.

Definition at line 53 of file LISACODE-GW.h.

Referenced by GW::getNParam(), GWBinary::GWBinary(), and GWFastSpinBBH::GWFastSpinBBH().

10.14.4.9 double GWCusp::Phi0hc [protected]

Initial phase of h_cross polarisation component.

Definition at line 55 of file LISACODE-GWCusp.h.

Referenced by getParam(), getPhi0hc(), GWCusp(), hc(), setParam(), and setPhi0hc().

10.14.4.10 double GWCusp::Phi0hp [protected]

Initial phase of h_plus polarisation component.

Definition at line 53 of file LISACODE-GWCusp.h.

Referenced by getParam(), getPhi0hp(), GWCusp(), hp(), setParam(), and setPhi0hp().

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 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)*
  - *11 -> AzimuthalAngleOfSpin2 : φ_{S_2} : Azimuthal angle of spin S2 (in radians)*
  - *12 -> phic : ϕ_c : Phase at coalescence (in raidans)*
  - *12 -> InitialPolarAngleL : $\theta_{L,0}$: Initial polar angle of orbital momentum (in raidans)*

- *14 -> 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 setAmplPNorder (double AmplPNorder_n)
- void initRK ()

  Set Runge-Kutta parameters.

- void init ()

  Initialisation (integration of precession and spline).

- void AmpShared (double t)

  Compute shared data for calcultion 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 tegration 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 calcderivvals (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
- int RK_VECLENGTH

  Runge-Kutta parameters.

- 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 AnglPol

  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 AmplPNorder, AzimuthalAngleOfSpin1, AzimuthalAngleOfSpin2, GW::Beta, GW::CalculDirProp(), chi1, chi2, DL, init(), InitialAzimuthalAngleL, InitialPolarAngleL, initRK(), GW::Lambda, m1, m2, GW::NParam, phic, PolarAngleOfSpin1, PolarAngleOfSpin2, Rmin, TaperApplied, TaperSteepness, tc, Tobs, and Toffset.

10.15.2.2 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 57 of file LISACODE-GWFastSpinBBH.cpp.

References AzimuthalAngleOfSpin1, AzimuthalAngleOfSpin2, chi1, chi2, init(), InitialAzimuthalAngleL, InitialPolarAngleL, initRK(), GW::NParam, phic, PolarAngleOfSpin1, PolarAngleOfSpin2, Rmin, setAmplPNorder(), setDistance(), setMass1(), setMass2(), TaperApplied, TaperSteepness, tc, Tobs, and Toffset.

10.15.2.3 GWFastSpinBBH::~GWFastSpinBBH ()

Destructor.

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

References betavec, betay2, cphilvec, cphily2, mulvec, muly2, sigmavec, sigmay2, sphilvec, sphily2, thomasvec, thomasy2, and timevec.

10.15.3 Member Function Documentation

10.15.3.1 void GWFastSpinBBH::AmpShared (double *t*)

Compute shared data for calcultion of amplitude.

Definition at line 645 of file LISACODE-GWFastSpinBBH.cpp.

References Amp, AmplPNorder, 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, sintheta, sphilvec, sphily2, sx, TaperApplied, TaperSteepness, tc, thomasvec, thomasy2, timeCur, timevec, tmax, TSUN, and xold.

Referenced by hc(), and hp().

10.15.3.2 void GWFastSpinBBH::calcderivvals (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 960 of file LISACODE-GWFastSpinBBH.cpp.

References S1xdot(), S1ydot(), S1zdot(), S2xdot(), S2ydot(), and S2zdot().

Referenced by init(), and update().

10.15.3.3 void GWFastSpinBBH::calcLcrossN (double *output*[ ], double *inputs*[ ], double *nvec*[ ])

Calculate scalar product LxN.

Definition at line 1029 of file LISACODE-GWFastSpinBBH.cpp.

Referenced by init(), and update().

10.15.3.4 double GWFastSpinBBH::calcLdotN (double *inputs*[ ], double *nvec*[ ])

Calculate scalar product L.N.

Definition at line 1024 of file LISACODE-GWFastSpinBBH.cpp.

Referenced by init(), and update().

10.15.3.5 double GWFastSpinBBH::calcLdotS1 (double *inputs*[ ])

Calculate scalar product L.S1.

Definition at line 1009 of file LISACODE-GWFastSpinBBH.cpp.

Referenced by init(), S2xdot(), S2ydot(), S2zdot(), and update().

10.15.3.6 double GWFastSpinBBH::calcLdotS2 (double *inputs*[ ])

Calculate scalar product L.S1.

Definition at line 1014 of file LISACODE-GWFastSpinBBH.cpp.

Referenced by init(), S1ydot(), S1zdot(), and update().

10.15.3.7 double GWFastSpinBBH::calcSdotS (double *inputs*[ ])

Calculate scalar product L.S2.

Definition at line 1019 of file LISACODE-GWFastSpinBBH.cpp.

Referenced by init(), and update().

10.15.3.8 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().

10.15.3.9 void GWFastSpinBBH::DispTempVal (double *t*, ostream * *OutDisp*) [virtual]

Display temporary values.

Reimplemented from GW.

Definition at line 833 of file LISACODE-GWFastSpinBBH.cpp.

References Ax, ci, psi, sf, and sx.

10.15.3.10 double GWFastSpinBBH::Freq (double *t*, double *Mtot*, double *Mchirp*, double *eta*, double *beta*, double *sigma*, double *tc*)

Compute frequency.

Definition at line 849 of file LISACODE-GWFastSpinBBH.cpp.

References TSUN.

Referenced by AmpShared(), init(), and update().

10.15.3.11 double GW::getAnglPol () [inline, inherited]

Definition at line 83 of file LISACODE-GW.h.

References GW::AnglPol.

10.15.3.12 double GW::getBeta () const [inline, inherited]

Definition at line 76 of file LISACODE-GW.h.

References GW::Beta.

10.15.3.13 vector<double> GW::getDirProp () const [inline, inherited]

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

References GW::DirProp.

10.15.3.14 double GWFastSpinBBH::getDistance () [inline]

Definition at line 195 of file LISACODE-GWFastSpinBBH.h.

References DL.

10.15.3.15 double GW::getLambda () const [inline, inherited]

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

References GW::Lambda.

10.15.3.16 double GWFastSpinBBH::getMass1 () [inline]

Definition at line 193 of file LISACODE-GWFastSpinBBH.h.

References m1.

10.15.3.17 double GWFastSpinBBH::getMass2 () [inline]

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

References m2.

10.15.3.18 int GW::getNParam () [inline, inherited]

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

References GW::NParam.

10.15.3.19 double GWFastSpinBBH::getParam (int *iP*) [virtual]

Return parameters specified by iP (see function for setParam the iP code).

Reimplemented from GW.

Definition at line 183 of file LISACODE-GWFastSpinBBH.cpp.

References AzimuthalAngleOfSpin1, AzimuthalAngleOfSpin2, GW::Beta, chi1, chi2, DL, InitialAzimuthalAngleL, InitialPolarAngleL, GW::Lambda, m1, m2, phic, PolarAngleOfSpin1, PolarAngleOfSpin2, and tc.

10.15.3.20 double GWFastSpinBBH::hc (double *t*) [virtual]

Compute wave amplitude component hcross.

Reimplemented from GW.

Definition at line 826 of file LISACODE-GWFastSpinBBH.cpp.

References AmpShared(), c2psi, s2psi, shc, and shp.

10.15.3.21 double GWFastSpinBBH::hp (double *t*) [virtual]

Compute wave amplitude component hplus.

Reimplemented from GW.

Definition at line 819 of file LISACODE-GWFastSpinBBH.cpp.

References AmpShared(), c2psi, s2psi, shc, and shp.

10.15.3.22 void GWFastSpinBBH::init () [virtual]

Initialisation (integration of precession and spline).

Reimplemented from GW.

Definition at line 322 of file LISACODE-GWFastSpinBBH.cpp.

References Amp, AzimuthalAngleOfSpin1, AzimuthalAngleOfSpin2, GW::Beta, betavec, betay2, c_SI, calcderivvals(), calcLcrossN(), calcLdotN(), calcLdotS1(), calcLdotS2(), calcSdotS(), chi1, chi2, costheta, cphilvec, cphily2, DL, dm, eta, etain, Freq(), gamma0, idxcur, idxm, InitialAzimuthalAngleL, InitialPolarAngleL, kpc_m, GW::Lambda, m1, m2, Mchirp, Mtot, mu, mulvec, muly2, p15, p150, p20, p200, phi, Phi0, Phi10, 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().

10.15.3.23 void GWFastSpinBBH::initRK ()

Set Runge-Kutta parameters.

Definition at line 275 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().

10.15.3.24 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 866 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, and update().

Referenced by init().

10.15.3.25 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 calcderivvals().

10.15.3.26 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 980 of file LISACODE-GWFastSpinBBH.cpp.

References calcLdotS2().

Referenced by calcderivvals().

10.15.3.27 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 986 of file LISACODE-GWFastSpinBBH.cpp.

References calcLdotS2().

Referenced by calcderivvals().

10.15.3.28 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 992 of file LISACODE-GWFastSpinBBH.cpp.

References calcLdotS1().

Referenced by calcderivvals().

10.15.3.29 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 998 of file LISACODE-GWFastSpinBBH.cpp.

References calcLdotS1().

Referenced by calcderivvals().

10.15.3.30 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 1004 of file LISACODE-GWFastSpinBBH.cpp.

References calcLdotS1().

Referenced by calcderivvals().

10.15.3.31 void GWFastSpinBBH::setAmplPNorder (double *AmplPNorder_n*)

Definition at line 261 of file LISACODE-GWFastSpinBBH.cpp.

References AmplPNorder.

Referenced by GWFastSpinBBH().

10.15.3.32 void GW::setAnglPol (double *AnglPol_n*) [inherited]

$*$ Input is checked:

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

Reimplemented in GWCusp, GWMono, and GWNew.

Definition at line 187 of file LISACODE-GW.cpp.

References GW::AnglPol.

10.15.3.33 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().

10.15.3.34 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

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

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

References GW::Beta, GW::DirProp, GW::Lambda, and PRECISION.

10.15.3.35 void GWFastSpinBBH::setDistance (double *r_n*)

Definition at line 252 of file LISACODE-GWFastSpinBBH.cpp.

References DL.

Referenced by GWFastSpinBBH().

10.15.3.36 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.

10.15.3.37 void GWFastSpinBBH::setMass1 (double *M1_n*)

Definition at line 236 of file LISACODE-GWFastSpinBBH.cpp.

References m1.

Referenced by GWFastSpinBBH().

10.15.3.38 void GWFastSpinBBH::setMass2 (double *M2_n*)

Definition at line 244 of file LISACODE-GWFastSpinBBH.cpp.

References m2.

Referenced by GWFastSpinBBH().

10.15.3.39 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 raidans)
- 12 -> InitialPolarAngleL : $\theta_{L,0}$: Initial polar angle of orbital momentum (in raidans)
- 14 -> InitialPolarAngleL : $\varphi_{L,0}$: Initial azimuthal angle of orbital momentum (in radians).

Reimplemented from GW.

Definition at line 131 of file LISACODE-GWFastSpinBBH.cpp.

References AzimuthalAngleOfSpin1, AzimuthalAngleOfSpin2, GW::Beta, chi1, chi2, DL, InitialAzimuthalAngleL, InitialPolarAngleL, GW::Lambda, m1, m2, phic, PolarAngleOfSpin1, PolarAngleOfSpin2, and tc.

10.15.3.40 void GWFastSpinBBH::spline (double * *x*, double * *y*, int *n*, double *yp1*, double *ypn*, double * *y2*)

Spline data : compute y2.

Definition at line 1036 of file LISACODE-GWFastSpinBBH.cpp.

Referenced by init().

10.15.3.41 void GWFastSpinBBH::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.

Definition at line 912 of file LISACODE-GWFastSpinBBH.cpp.

References calcderivvals(), 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 133 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.2 double GWFastSpinBBH::A3 [protected]

Definition at line 133 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.3 double GWFastSpinBBH::A4 [protected]

Definition at line 133 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.4 double GWFastSpinBBH::A5 [protected]

Definition at line 133 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.5 double GWFastSpinBBH::A6 [protected]

Definition at line 133 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 GWCusp, GWMono, GWNew, and GWPeriGate.

Definition at line 51 of file LISACODE-GW.h.

Referenced by GW::getAnglPol(), GWNewton2::getParam(), GWFile::getParam(), GWBinary::getParam(), GW::GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), GW::setAnglPol(), GWNewton2::setParam(), GWFile::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(), GWFastSpinBBH(), 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(), GWFastSpinBBH(), init(), and setParam().

10.15.4.12 double GWFastSpinBBH::B21 [protected]

Definition at line 134 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.13 double GWFastSpinBBH::B31 [protected]

Definition at line 134 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.14 double GWFastSpinBBH::B32 [protected]

Definition at line 134 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.15 double GWFastSpinBBH::B41 [protected]

Definition at line 134 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.16 double GWFastSpinBBH::B42 [protected]

Definition at line 134 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.17 double GWFastSpinBBH::B43 [protected]

Definition at line 134 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.18 double GWFastSpinBBH::B51 [protected]

Definition at line 134 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.19 double GWFastSpinBBH::B52 [protected]

Definition at line 134 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.20 double GWFastSpinBBH::B53 [protected]

Definition at line 134 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.21 double GWFastSpinBBH::B54 [protected]

Definition at line 134 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.22 double GWFastSpinBBH::B61 [protected]

Definition at line 134 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.23 double GWFastSpinBBH::B62 [protected]

Definition at line 134 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.24 double GWFastSpinBBH::B63 [protected]

Definition at line 134 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.25 double GWFastSpinBBH::B64 [protected]

Definition at line 134 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.26 double GWFastSpinBBH::B65 [protected]

Definition at line 134 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(), GWFastSpinBBH(), 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 AmpShared(), init(), and ~GWFastSpinBBH().

10.15.4.29 double* GWFastSpinBBH::betay2 [protected]

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

Referenced by AmpShared(), init(), and ~GWFastSpinBBH().

10.15.4.30 double GWFastSpinBBH::C1 [protected]

Definition at line 135 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.31 double GWFastSpinBBH::C2 [protected]

Definition at line 135 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 135 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.34 double GWFastSpinBBH::C4 [protected]

Definition at line 135 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.35 double GWFastSpinBBH::C5 [protected]

Definition at line 135 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.36 double GWFastSpinBBH::C6 [protected]

Definition at line 135 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(), GWFastSpinBBH(), 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(), GWFastSpinBBH(), 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 AmpShared(), init(), and ~GWFastSpinBBH().

10.15.4.42 double* GWFastSpinBBH::cphily2 [protected]

Definition at line 126 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), init(), and ~GWFastSpinBBH().

10.15.4.43 double GWFastSpinBBH::D1 [protected]

Definition at line 136 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.44 double GWFastSpinBBH::D2 [protected]

Definition at line 136 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.45 double GWFastSpinBBH::D3 [protected]

Definition at line 136 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.46 double GWFastSpinBBH::D4 [protected]

Definition at line 136 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.47 double GWFastSpinBBH::D5 [protected]

Definition at line 136 of file LISACODE-GWFastSpinBBH.h.

Referenced by initRK(), and rkckstep().

10.15.4.48 double GWFastSpinBBH::D6 [protected]

Definition at line 136 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(), GWFastSpinBBH(), 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 init().

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(), and init().

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 init().

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(), GWFastSpinBBH(), 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(), GWFastSpinBBH(), init(), and setParam().

10.15.4.60 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(), GWNewton2::getParam(), GWNew::getParam(), GWMono::getParam(), GWFile::getParam(), getParam(), GWCusp::getParam(), GWBinary::getParam(), GW::getParam(), GW::GW(), GWBinary::GWBinary(), GWFastSpinBBH(), 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().

10.15.4.61 double GWFastSpinBBH::m1 [protected]

Mass of object 1 (in solar masses).

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

Referenced by getMass1(), getParam(), GWFastSpinBBH(), init(), setMass1(), and setParam().

10.15.4.62 double GWFastSpinBBH::m2 [protected]

Mass of object 2 (in solar masses).

Definition at line 42 of file LISACODE-GWFastSpinBBH.h.

Referenced by getMass2(), getParam(), GWFastSpinBBH(), init(), setMass2(), and setParam().

10.15.4.63 double GWFastSpinBBH::Mchirp [protected]

Chirp mass.

Definition at line 89 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.64 double GWFastSpinBBH::Mtot [protected]

Total mass.

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

Referenced by AmpShared(), and init().

10.15.4.65 double GWFastSpinBBH::mu [protected]

Reduced mass.

Definition at line 85 of file LISACODE-GWFastSpinBBH.h.

Referenced by init().

10.15.4.66 double* GWFastSpinBBH::mulvec [protected]

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

Referenced by AmpShared(), init(), and ~GWFastSpinBBH().

10.15.4.67 double* GWFastSpinBBH::muly2 [protected]

Definition at line 124 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), init(), and ~GWFastSpinBBH().

10.15.4.68 int GW::NParam [protected, inherited]

Number of parameters.

Definition at line 53 of file LISACODE-GW.h.

Referenced by GW::getNParam(), GWBinary::GWBinary(), and GWFastSpinBBH().

10.15.4.69 double GWFastSpinBBH::p15 [protected]

Definition at line 96 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.70 double GWFastSpinBBH::p150 [protected]

Definition at line 96 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.71 double GWFastSpinBBH::p20 [protected]

Definition at line 96 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.72 double GWFastSpinBBH::p200 [protected]

Definition at line 96 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.73 double GWFastSpinBBH::phi [protected]

Definition at line 95 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.74 double GWFastSpinBBH::Phi0 [protected]

Definition at line 96 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.75 double GWFastSpinBBH::Phi10 [protected]

Definition at line 96 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and init().

10.15.4.76 double GWFastSpinBBH::phic [protected]

Phase at coalescence.

Definition at line 60 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), getParam(), GWFastSpinBBH(), and setParam().

10.15.4.77 double GWFastSpinBBH::PolarAngleOfSpin1 [protected]

Polar angle of spin S1.

Definition at line 52 of file LISACODE-GWFastSpinBBH.h.

Referenced by getParam(), GWFastSpinBBH(), init(), and setParam().

10.15.4.78 double GWFastSpinBBH::PolarAngleOfSpin2 [protected]

Polar angle of spin S2.

Definition at line 54 of file LISACODE-GWFastSpinBBH.h.

Referenced by getParam(), GWFastSpinBBH(), init(), and setParam().

10.15.4.79 double GWFastSpinBBH::psi [protected]

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

Referenced by AmpShared(), and DispTempVal().

10.15.4.80 double GWFastSpinBBH::RK_EPS [protected]

Definition at line 137 of file LISACODE-GWFastSpinBBH.h.

Referenced by init(), and initRK().

10.15.4.81 double GWFastSpinBBH::RK_H [protected]

Definition at line 137 of file LISACODE-GWFastSpinBBH.h.

Referenced by init(), and initRK().

10.15.4.82 int GWFastSpinBBH::RK_VECLENGTH [protected]

Runge-Kutta parameters.

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

Referenced by init(), and initRK().

10.15.4.83 double GWFastSpinBBH::Rmin [protected]

Definition at line 74 of file LISACODE-GWFastSpinBBH.h.

Referenced by GWFastSpinBBH(), and init().

10.15.4.84 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().

10.15.4.85 double GWFastSpinBBH::sf [protected]

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

Referenced by AmpShared(), and DispTempVal().

10.15.4.86 double GWFastSpinBBH::shc [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().

10.15.4.87 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().

10.15.4.88 double* GWFastSpinBBH::sigmavec [protected]

Definition at line 121 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), init(), and ~GWFastSpinBBH().

10.15.4.89 double* GWFastSpinBBH::sigmay2 [protected]

Definition at line 128 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), init(), and ~GWFastSpinBBH().

10.15.4.90 double GWFastSpinBBH::sintheta [protected]

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

Referenced by AmpShared(), and init().

10.15.4.91 double* GWFastSpinBBH::sphilvec [protected]

Definition at line 118 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), init(), and ~GWFastSpinBBH().

10.15.4.92 double* GWFastSpinBBH::sphily2 [protected]

Definition at line 125 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), init(), and ~GWFastSpinBBH().

10.15.4.93 double GWFastSpinBBH::sx [protected]

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

Referenced by AmpShared(), and DispTempVal().

10.15.4.94 double GWFastSpinBBH::TaperApplied [protected]

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

Referenced by AmpShared(), and GWFastSpinBBH().

10.15.4.95 double GWFastSpinBBH::TaperSteepness [protected]

Definition at line 76 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), and GWFastSpinBBH().

10.15.4.96 double GWFastSpinBBH::tc [protected]

Time of coalescence (in seconds).

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

Referenced by AmpShared(), getParam(), GWFastSpinBBH(), init(), and setParam().

10.15.4.97 double* GWFastSpinBBH::thomasvec [protected]

Definition at line 122 of file LISACODE-GWFastSpinBBH.h.

Referenced by AmpShared(), init(), and ~GWFastSpinBBH().

10.15.4.98 double* GWFastSpinBBH::thomasy2 [protected]

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

Referenced by AmpShared(), init(), and ~GWFastSpinBBH().

10.15.4.99 double GWFastSpinBBH::timeCur [protected]

Last time of compute amplitude.

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

Referenced by AmpShared(), and init().

10.15.4.100 double* GWFastSpinBBH::timevec [protected]

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

Referenced by AmpShared(), init(), and ~GWFastSpinBBH().

10.15.4.101 double GWFastSpinBBH::tmax [protected]

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

Referenced by AmpShared(), and init().

10.15.4.102 double GWFastSpinBBH::Tobs [protected]

Duration of observation.

Definition at line 72 of file LISACODE-GWFastSpinBBH.h.

Referenced by GWFastSpinBBH(), and init().

10.15.4.103 double GWFastSpinBBH::Toffset [protected]

Time offset.

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

Referenced by GWFastSpinBBH(), and init().

10.15.4.104 double GWFastSpinBBH::xmax [protected]

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

Referenced by init().

10.15.4.105 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, int Records)

  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 ∗ getFileName ()
- 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_plus,h_cross)=(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(), 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 54 of file LISACODE-GWFile.cpp.

References FileEncoding, FileName, init(), TimeOffset, and TimeStep.

10.16.2.4 GWFile::~GWFile ()

Destructor.

Definition at line 75 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 367 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 101 of file LISACODE-GWFile.cpp.

References GW::AnglPol, GW::Beta, and GW::Lambda.

10.16.3.11 double GWFile::hc (double *t*) [virtual]

Return $h_\times(t)$.

Returns:
Result of Interpol method with *t* input time and 2 as arguments.

Reimplemented from GW.

Definition at line 360 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 350 of file LISACODE-GWFile.cpp.

References Interpol().

10.16.3.13 void GWFile::init () [virtual]

Initialization : ...

Reimplemented from GW.

Definition at line 129 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:

$$returnedvalue = \begin{cases} \sum_{k=kmin}^{kmax} hList[k].x \cdot P_k & \text{if type=1} \\ \sum_{k=kmin}^{kmax} hList[k].y \cdot P_k & \text{if type=2} \end{cases}$$

$$\text{where } P_k = \prod_{j=kmin, j\neq k}^{kmax} \frac{t - TimeList[j]}{TimeList[k] - TimeList[j]}$$

LastUsedBin is updated, so that

$$TimeList[LastUsedBin] \leq t \leq TimeList[LastUsedBin + 1]$$

$$ordermin = floor(\frac{order + 1}{2}) \text{ where } order = 3$$

$$kmin = min(0, LastUsedBin - ordermin + 1)$$

$$kmax = max(LastUsedBin + order + 1 - ordermin, sizeof(TimeList) - 1)$$

Definition at line 292 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 specifie in arguments *TimeOffset* and *TimeStep*. If *TimeStep* is -1.0, it is means that the time informations are not specifed. Header (lines begining with "#" charected) are ignored. For each line, time and 2 components are read and pushed back into TimeList and hList attributes.

Definition at line 148 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 specifie in arguments *TimeOffset* and *TimeStep*. If *TimeStep* is -1.0, it is means that the time informations are not specifed. Header (lines begining with "#" charected) are ignored. For each line, time and 2 components are read and pushed back into TimeList and hList attributes.

Definition at line 234 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 GWCusp, 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

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

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

References GW::Beta, GW::DirProp, GW::Lambda, and PRECISION.

10.16.3.20 void GWFile::setFileName (char * *FileName_n*)

Definition at line 119 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 85 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 GWCusp, GWMono, GWNew, and GWPeriGate.

Definition at line 51 of file LISACODE-GW.h.

Referenced by GW::getAnglPol(), GWNewton2::getParam(), getParam(), GWBinary::getParam(), GW::GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), GW::setAnglPol(), GWNewton2::setParam(), 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(), GWPeriGate::getParam(), GWNewton2::getParam(), GWNew::getParam(), GWMono::getParam(), getParam(), GWFastSpinBBH::getParam(), GWCusp::getParam(), GWBinary::getParam(), GW::getParam(), GW::GW(), GWBinary::GWBinary(), GWFastSpinBBH::GWFastSpinBBH(), GWNewton2::GWNewton2(), GWFastSpinBBH::init(), GW::setBeta(), GW::setDirProp(), GWSto::setParam(), GWPeriGate::setParam(), GWNewton2::setParam(), GWNew::setParam(), GWMono::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(), GWFastSpinBBH::GWFastSpinBBH(), 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 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(), 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 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_{x} polarisation component amplitude

- double AnglPol

  Polarisation angle (rad).

- double Phi0hp

  Initial phase of h_plus polarisation component.

- double Phi0hc

  Initial phase of h_cross polarisation component.

- double Beta

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 $h_\times(t)$.

$$\text{returned value} = Amplhc \cdot sin(2 \cdot \pi \cdot Freq \cdot t + Phi0hc)$$

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 $h_+(t)$.

$$\text{returned value} = Amplhp \cdot sin(2 \cdot \pi \cdot Freq \cdot t + Phi0hp)$$

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

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

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 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.

10.17.3.25 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, GW::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(), GWCusp::getParam(), GWBinary::getParam(), GW::getParam(), GW::GW(), GWBinary::GWBinary(), GWFastSpinBBH::GWFastSpinBBH(), GWNewton2::GWNewton2(), GWFastSpinBBH::init(), GW::setBeta(), GW::setDirProp(), GWSto::setParam(), GWPeriGate::setParam(), GWNewton2::setParam(), GWNew::setParam(), setParam(), GWFile::setParam(), GWFastSpinBBH::setParam(), GWCusp::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(), GWCusp::getParam(), GWBinary::getParam(), GW::getParam(), GW::GW(), GWBinary::GWBinary(), GWFastSpinBBH::GWFastSpinBBH(), GWNewton2::GWNewton2(), GWFast-

SpinBBH::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(), 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 Amplhc

  h_{x} polarisation component amplitude

- double AnglPol

  Polarisation angle (rad).

- double Phi0hp

  Initial phase of h_plus polarisation component.

- double Phi0hc

  Initial phase of h_cross polarisation component.

- double Beta

  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.

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

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 $h_\times(t)$.

$$\text{returned value} = Amplhc \cdot sin(2 \cdot \pi \cdot Freq \cdot t + Phi0hc)$$

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 $h_+(t)$.

$$\text{returned value} = Amplhp \cdot sin(2 \cdot \pi \cdot Freq \cdot t + Phi0hp)$$

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

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

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(), GWFastSpinBBH::GWFastSpinBBH(), 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(), GWFastSpinBBH::GWFastSpinBBH(), GWNewton2::GWNewton2(), GWFast-

SpinBBH::init(), GW::setDirProp(), GW::setLambda(), GWSto::setParam(), GWPeriGate::setParam(), GWNewton2::setParam(), setParam(), GWMono::setParam(), GWFile::setParam(), GWFastSpin-BBH::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(), 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)

  Qets 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(inc)

- double si

  sin(inc)

- double cmass

  Chirp mass.

- double time_encour

  Current computation time.

- double hint

  1PN amplitude (depending on time)

- double omega

  Pulsation (depending on time).

- double phi

  Phase (depending on time).

- double teta

  System constant.

- double lambda

  System constant.

- double taud

  2.5 PN approximation parameter.

- double taud0

  System parameter.

- double omega0

  System parameter.

- double gw

  System parameter.

- double psi

  2.5 PN parameter (depending on time).

- double a1

  2.5 PN approximation parameter.

- double a2

  2.5 PN approximation parameter.

- double a3

  2.5 PN approximation parameter.

- double a4

  2.5 PN approximation parameter.

- double a5

  2.5 PN approximation parameter.

- double a6

  2.5 PN approximation parameter.

- double b1

  2.5 PN approximation parameter.

- double b2

  2.5 PN approximation parameter.

- double b3

  2.5 PN approximation parameter.

- double b4

  2.5 PN approximation parameter.

- double a11

  2.5 PN approximation parameter.

- double a22

  2.5 PN approximation parameter.

- double b11

  2.5 PN approximation parameter.

- double c1

  2.5 PN approximation parameter.

- double c2

  2.5 PN approximation parameter.

- double c3

  2.5 PN approximation parameter.

- double d1

  2.5 PN approximation parameter.

- double d2

  2.5 PN approximation parameter.

- double e1

  2.5 PN approximation parameter.

- double e2

  2.5 PN approximation parameter.

- double e3

  2.5 PN approximation parameter.

- double e4

  2.5 PN approximation parameter.

- double e5

2.5 PN approximation parameter.

- double a7

  2.5 PN approximation parameter.

- double f1

  2.5 PN approximation parameter.

- double f3

  2.5 PN approximation parameter.

- double f5

  2.5 PN approximation parameter.

- double f6

  2.5 PN approximation parameter.

- double f7

  2.5 PN approximation parameter.

- double f8

  2.5 PN approximation parameter.

- double f9

  2.5 PN approximation parameter.

- double f10

  2.5 PN approximation parameter.

- double a1x

  2.5 PN approximation parameter.

- double b1x

  2.5 PN approximation parameter.

- double c1x

  2.5 PN approximation parameter.

- double c2x

  2.5 PN approximation parameter.

- double d1x

  2.5 PN approximation parameter.

- double d2x

  2.5 PN approximation parameter.

- double d3x

  2.5 PN approximation parameter.

- double d4x

  2.5 PN approximation parameter.

- double d5x

  2.5 PN approximation parameter.

- double e1x

  2.5 PN approximation parameter.

- double e2x

  2.5 PN approximation parameter.

- double e3x

  2.5 PN approximation parameter.

- double e4x

  2.5 PN approximation parameter.

- double e5x

  2.5 PN approximation parameter.

- double e6x

  2.5 PN approximation parameter.

- double e7x

  2.5 PN approximation parameter.

- double e8x

  2.5 PN approximation parameter.

- double g1

  2.5 PN approximation parameter.

- double Beta

  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.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 GWNewton2::GWNewton2 ()

Constructs an instance and initializes it with default values.

$$\beta = 0.917311 - \frac{\pi}{2} \text{ rad}$$

$$\lambda = 2.97378 - \pi \text{ rad}$$

$$AnglPol = 2.46531 \text{ rad}$$

$$type = 1$$

$$m_1 = 10^5 \cdot M_{Sun} \text{ kg}$$

$$m_2 = 10^5 \cdot M_{Sun} \text{ kg, where } M_{Sun} = 1.9889 \cdot 10^{30} \text{ kg}$$

$$rdist = 10^6 \cdot kpc_m \text{ m, where } kpc_m = 3.0856675807 \cdot 10^{19} \text{ m}$$

$$\phi_{coal} = 0 \text{ rad}$$

$$\tau_{d0} = 0$$

$$\omega_0 = 0$$

$$gw = 1$$

$$inc = \frac{\pi}{2} \text{ rad}$$

$$t_{coal} = 60000 \text{ s}$$

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

$$m1 = mm1 \cdot M_{Sun} \text{ kg, where } M_{Sun} = 1.9889 \cdot 10^{30} \text{ kg}$$

$$m2 = mm2 \cdot M_{Sun} \text{ kg, where } M_{Sun} = 1.9889 \cdot 10^{30} \text{ kg}$$

$$rdist = \cdot kpc_m \text{ m, where } kpc_m = 3.0856675807 \cdot 10^{19} \text{ m}$$

- phcoal = pphcoal
- taud0 = ttaud0
- omega0 = oomega0

- gw = ggw
- inc = iinc
- tcoal = ttcoal 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* $\neq$ time_encour , time dependant attributes must be updated and time_encour is set to *temps* input :

- if type=1

$$\phi = \phi_{coal} - \left(C^3 * \frac{t_{coal} - temps}{5 \cdot G \cdot cmass} \right)^{\frac{5}{8}}$$

$$\omega = \frac{C^3}{8 \cdot G \cdot cmass} \cdot * \left(\frac{C^3 \left(t_{coal} - temps \right)}{5 \cdot G \cdot cmass} \right)^{-\frac{3}{8}}$$

$$hint = -\frac{2 \cdot G \cdot mtot \cdot \nu}{C^2 \cdot rdist} \cdot \left(\frac{G \cdot mtot \cdot \omega}{C^3} \right)^{\frac{2}{3}}$$

- if type=2

$$\tau_d = \nu \cdot C^3 \cdot \frac{t_{coal} - temps}{5 \cdot G \cdot mtot}$$

$$\phi = \frac{-1}{\nu} \left(\tau_d^{\frac{5}{8}} + a1 \cdot \tau_d^{\frac{3}{8}} + a2 \cdot \tau_d^{\frac{1}{4}} + a3 \cdot \tau_d^{\frac{1}{8}} + a4 \cdot log(\frac{\tau_d}{\tau_{d0}}) + \left(a5 \cdot log(\frac{\tau_d}{256}) + a6 \right) \cdot \tau_d^{-\frac{1}{8}} + a7 \cdot \tau_d^{-\frac{1}{4}} \right)$$

$$\omega = b1 \cdot (\tau_d^{-\frac{3}{8}} + b2 \cdot \tau_d^{-\frac{5}{8}} + b3 \cdot \tau_d^{-\frac{4}{8}} + b4 \cdot \tau_d^{-\frac{7}{8}})$$
$$\psi = \phi - \frac{2 \cdot G \cdot mtot \cdot \omega}{C^3} \cdot log(\frac{\omega}{\omega_0})$$

Definition at line 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 $h_\times(t)$.

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
$$hc = a11 \cdot (a22 \cdot \omega)^{\frac{2}{3}} \cdot$$
$$(a1x \cdot \sin(2 \cdot \psi)$$
$$+\left(\frac{g1 \cdot \omega}{gw}\right)^{\frac{1}{3}} (b1x \cdot (\sin(\psi) - 3 \cdot \sin(3 \cdot \psi)))$$
$$+\left(\frac{g1 \cdot \omega}{gw}\right)^{\frac{2}{3}} \cdot (c1x \cdot \sin(2 \cdot \psi) + c2x \cdot \sin(4 \cdot \psi))$$
$$+\frac{g1 \cdot \omega}{gw} (d1x \cdot (d2x \cdot \sin(\psi) + d3x \cdot \sin(3 \cdot \psi) + d4x \cdot \sin(5 \cdot \psi)) + d5x \cdot \sin(2 \cdot \psi))$$
$$+\left(\frac{g1 \cdot \omega}{gw}\right)^{\frac{4}{3}} \cdot (e1x \sin(2 \cdot \psi) + e2x \sin(4 \cdot \psi) + e3x \sin(6 \cdot \psi)$$
$$+e4x \cdot ((e5x \cdot \cos(\psi) + e6x \cdot \sin(\psi) + e7x) \cdots \sin(3 \cdot \psi) - e8x \sin(2 \cdot \psi))))$$

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
$$hp = a11 \cdot (a22 \cdot \omega)^{\frac{2}{3}} \cdot (b11 \cdot \cos(2 \cdot \psi)$$
$$+c1 \cdot \left(\frac{g1 \cdot \omega}{gw}\right)^{\frac{1}{3}} \cdot (c2 \cdot \cos(\psi) - c3 \cdot \cos(3 \cdot \psi))$$
$$+\left(\frac{g1 \cdot \omega}{gw}\right)^{\frac{2}{3}} \cdot (d1 \cdot \cos(2. \cdot \psi) - d2 \cdot \cos(4 \cdot \psi))$$

$$+\frac{g1\cdot\omega}{gw}\cdot(e1\,(e2\cdot cos(\cdot\psi)-e3\cdot cos(3\cdot\psi)+e4\cdot\cos(5\cdot\psi))-e5\cdot\cos(2\cdot\psi))$$
$$+\left(\frac{g1\cdot\omega}{gw}\right)^{\frac{2}{3}}\cdot(f1\cdot\cos(2\cdot\psi)+f3\cdot\cos(4\cdot\psi)-f5\cdot\cos(6\cdot\psi)$$
$$+f6\cdot(f7\sin(\psi)+f8\cdot\cos(\psi)-f9\cdot\sin(3\cdot\psi)+f10\cdot\cos(3\cdot\psi))))$$

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 : ...

$$m_{tot}=m1+m2$$
$$deltam=m1-m2$$
$$\mu=\frac{m_1*m_2}{m_1+m_2}$$
$$\nu=\frac{\mu}{m_{tot}}$$
$$ci=cos(inc)$$
$$si=sin(inc)$$
$$c_{mass}=\mu^{\frac{3}{5}}\cdot m_{tot}^{\frac{2}{5}}$$
$$time_{encour}=-1000$$

- If (type=2)

$$\theta=-\frac{11831}{9240}$$
$$\lambda=\frac{3}{7}\cdot(\theta-\frac{1039}{4620})$$
$$a1=\frac{55\cdot\nu}{96}+\frac{3715}{8064}$$
$$a2=-\frac{3\cdot\pi}{4}$$
$$a3=\frac{9275495}{14450688}+\frac{284875}{258048}\cdot\nu+\frac{1855}{2048})\cdot\nu^2$$
$$a4=(-\frac{38645}{172032}+\frac{65}{2048}\cdot\nu)\cdot\pi$$
$$a5=\frac{831032450749357}{57682522275840}-\frac{53}{40}\cdot\pi^2-\frac{107}{56}\cdot CE_{RG}+\frac{107}{448}\text{ where }CE_{RG}\text{ is Euler constant}$$
$$a6=(-\frac{123292747421}{4161798144}+\frac{2255}{2048}\cdot\pi^2+\frac{385}{48}\cdot\lambda-\frac{55}{16}\cdot\theta)\cdot\nu+\frac{154565}{1835008}\cdot\nu^2-\frac{1179625}{1769472}\cdot\nu^3$$
$$a7=(\frac{188516689}{173408256}+\frac{488825}{516096}\cdot\nu-\frac{141769}{516096}\cdot\nu^2)\cdot\pi$$

$$b1 = \frac{C^3}{8 \cdot G \cdot mtot}$$

$$b2 = \frac{743}{2688} + \frac{11}{32} \cdot \nu$$

$$b3 = -\frac{3 \cdot \pi}{10}$$

$$b4 = \frac{1855099}{14450688} + \frac{56975}{258048} \cdot \nu + \frac{371}{2048} \cdot \nu^2$$

$$a11 = \frac{2 \cdot G \cdot mtot \cdot \nu}{rdist \cdot C^2}$$

$$a22 = \frac{G \cdot mtot}{C^3}$$

$$b11 = -(1 + ci^2)$$

$$c1 = -\frac{si}{8} \cdot \frac{deltam}{mtot}$$

$$c2 = 5 + ci^2$$

$$c3 = 9 \cdot (1 + ci^2)$$

$$d1 = \frac{1}{6} \cdot (19 + 9 \cdot ci^2 - 2 \cdot ci^4 - \nu \cdot (19 - 11 \cdot ci^2 - 6 \cdot ci^4))$$

$$d2 = \frac{4}{3} \cdot si^2 \cdot (1 + ci^2) \cdot (1 - 3 \cdot \nu)$$

$$e1 = \frac{si}{192} \cdot \frac{deltam}{mtot}$$

$$e2 = (57 + 60 \cdot ci^2 - ci^4) - 2 \cdot \nu \cdot (49 - 12 \cdot ci^2 - ci^4)$$

$$e3 = \frac{27}{2} \cdot ((73 + 40 \cdot ci^2 - 9 \cdot ci^4) - 2 \cdot \nu \cdot (25 - 8 \cdot ci^2 - 9 \cdot ci^4))$$

$$e4 = \frac{625}{2} \cdot (1 - 2 \cdot \nu) \cdot si^2 \cdot (1 + ci^2)$$

$$e5 = 2 \cdot \pi \cdot (1 + ci^2)$$

$$f1 = \frac{1}{120} \cdot (22 + 396 \cdot ci^2 + 145 \cdot ci^4 - 5 \cdot ci^6 + \frac{5}{3} \cdot \nu \cdot (706 - 216 \cdot ci^2 - 251 \cdot ci^4 + 15 \cdot ci^6) - 5 \cdot \nu^2 \cdot (98 - 108 \cdot ci^2 + 7 \cdot ci^4 + 5 \cdot ci^6))$$

$$f3 = \frac{2}{15} \cdot si^2 \cdot ((59 + 35 \cdot ci^2 - 8 \cdot ci^4) - \frac{5}{3} \cdot \nu \cdot (131 + 59 \cdot ci^2 - 24 \cdot ci^4) + 5 \cdot \nu^2 \cdot (21 - 3 \cdot ci^2 - 8 \cdot ci^4))$$

$$f5 = \frac{81}{40} \cdot (1 - 5 \cdot \nu + 5 \cdot \nu^2) \cdot si^4 \cdot (1 + ci^2)$$

$$f6 = \frac{si}{40} \cdot \frac{deltam}{mtot}$$

$$f7 = 11 + 7 \cdot ci^2 + 10 \cdot (5 + ci^2) \cdot log(2)$$

$$f8 = 5 \cdot \pi \cdot (5 + ci^2)$$

$$f9 = 27 \cdot (7 - 10 \cdot log(\frac{3}{2})) \cdot (1 + ci^2)$$

$$f10 = 135 \cdot \pi \cdot (1 + ci^2)$$

$$a1x = -2 \cdot ci$$

$$b1x = -\frac{3}{4} \cdot si \cdot ci \cdot \frac{deltam}{mtot}$$

$$c1x = \frac{ci}{3} \cdot ((17 - 4 \cdot ci^2) - \cdot\nu \cdot (13 - 12 \cdot ci^2))$$

$$c2x = -\frac{8}{3} \cdot (1 - 3 \cdot \nu) \cdot ci \cdot si^2$$

$$d1x = \frac{si \cdot ci}{96} \cdot \frac{deltam}{mtot}$$

$$d2x = 63 - 5 \cdot ci^2 - 2 \cdot \nu \cdot (23 - 5 \cdot ci^2)$$

$$d3x = -\frac{27}{2} \cdot (67 - 15 \cdot ci^2 - 2 \cdot \nu \cdot (19 - 15 \cdot ci^2))$$

$$d4x = \frac{625}{2} \cdot (1 - 2 \cdot \nu) \cdot si^2$$

$$d5x = -4 \cdot \pi \cdot ci$$

$$e1x = \frac{ci}{60} \cdot (68 + 226 \cdot ci^2 - 15 \cdot ci^4 + \frac{5}{3} \cdot \nu \cdot (572 - 490 \cdot ci^2 + 45 \cdot ci^4) - 5 \cdot \nu^2 \cdot (56 - 70 \cdot ci^2 + 15 \cdot ci^4))$$

$$e2x = \frac{4}{15} \cdot ci \cdot si^2 \cdot (55 - 12 \cdot ci^2 - \frac{5}{3} \cdot \nu \cdot (119 - 36 \cdot ci^2) + 5 \cdot \nu^2 \cdot (17 - 12 \cdot ci^2))$$

$$e3x = -\frac{81}{20} \cdot (1 - 5 \cdot \nu + 5 \cdot \nu^2) \cdot ci \cdot si^4$$

$$e4x = -\frac{3}{20} \cdot si \cdot ci \cdot \frac{deltam}{mtot}$$

$$e5x = 3 + 10 \cdot log(2)$$

$$e6x = 5 \cdot \pi$$

$$e7x = -9 \cdot (7 - 10 \cdot log(\frac{3}{2}))$$

$$e8x = -45 \cdot \pi$$

$$g1 = \frac{\cdot G \cdot mtot}{C^3}$$

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:

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

Reimplemented in GWCusp, 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

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

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

References GW::Beta, GW::DirProp, GW::Lambda, and PRECISION.

10.19.3.24 void GWNewton2::setDistance (double *rdist_n*)

Sets rdist attribute.

Input is checked:

$$rdist_n \geq 0$$

Definition at line 244 of file LISACODE-GWNewton2.cpp.

References rdist.

10.19.3.25 void GWNewton2::setInc (double *inc_n*)

Qets 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 GWCusp, GWMono, GWNew, and GWPeriGate.

Definition at line 51 of file LISACODE-GW.h.

Referenced by GW::getAnglPol(), getParam(), GWFile::getParam(), GWBinary::getParam(), GW::GW(), GWBinary::GWBinary(), GWNewton2(), GW::setAnglPol(), setParam(), GWFile::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(), GWNew::getParam(), GWMono::getParam(), GWFile::getParam(), GWFastSpinBBH::getParam(), GWCusp::getParam(), GWBinary::getParam(), GW::getParam(), GW::GW(), GWBinary::GWBinary(), GWFastSpinBBH::GWFastSpinBBH(), 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 GWNewton2::e1x [protected]

2.5 PN approximation parameter.

Referenced by hc(), and init().

10.19.4.37 GWNewton2::e2 [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.38 GWNewton2::e2x [protected]

2.5 PN approximation parameter.

Referenced by hc(), and init().

10.19.4.39 GWNewton2::e3 [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.40 GWNewton2::e3x [protected]

2.5 PN approximation parameter.

Referenced by hc(), and init().

10.19.4.41 GWNewton2::e4 [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.42 GWNewton2::e4x [protected]

2.5 PN approximation parameter.

Referenced by hc(), and init().

10.19.4.43 GWNewton2::e5 [protected]

2.5 PN approximation parameter.

Referenced by hp(), and init().

10.19.4.44 GWNewton2::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(), GWNew::getParam(), GWMono::getParam(), GWFile::getParam(), GWFastSpinBBH::getParam(), GWCusp::getParam(), GWBinary::getParam(), GW::getParam(), GW::GW(), GWBinary::GWBinary(), GWFastSpinBBH::GWFastSpinBBH(), 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 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(), and GWFastSpinBBH::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 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 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 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 componenrs are modelised as periodic gate signals.

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

Public Member Functions

- GWPeriGate ()

  Constructs an instance and initializes it with default values.

- GWPeriGate (double Beta_n, double Lambda_n, double AnglPol_n, double Freq_n, double Amplhp_n, double Amplhc_n)

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

- ~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::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().

10.20.3.2 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 $h_\times(t)$.

$$\text{returned value} = \begin{cases} Amplhc & \text{if } \frac{Freq \cdot t}{1} - ceil(\frac{Freq \cdot t}{1}) < 0.5 \\ 0 & else \end{cases}$$

Reimplemented from GW.

Definition at line 166 of file LISACODE-GWPeriGate.cpp.

References Amplhc, and Freq.

10.20.3.13 double GWPeriGate::hp (double *t*) [virtual]

Return $h_+(t)$.

$$\text{returned value} = \begin{cases} Amplhp & \text{if } \frac{Freq \cdot t}{1} - ceil(\frac{Freq \cdot t}{1}) < 0.5 \\ 0 & else \end{cases}$$

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 GWCusp, 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

- $$\beta = asin(DirProp_{norm}[2])$$

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

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

References GW::Beta, GW::DirProp, GW::Lambda, and PRECISION.

10.20.3.20 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(), GWFile::getParam(), GWFastSpinBBH::getParam(), GWCusp::getParam(), GWBinary::getParam(), GW::getParam(), GW::GW(), GWBinary::GWBinary(), GWFastSpinBBH::GWFastSpinBBH(), GWNewton2::GWNewton2(), GWFastSpinBBH::init(), GW::setBeta(), GW::setDirProp(), GWSto::setParam(), setParam(), GWNewton2::setParam(), GWNew::setParam(), GWMono::setParam(), GWFile::setParam(), GWFastSpinBBH::setParam(), GWCusp::setParam(), GWBinary::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(), GWFastSpinBBH::GWFastSpinBBH(), 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(), 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_plus and h_cross are described in this class.

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

Public Member Functions

- 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 longtude*
  - *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.*

- 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 Tsample_n, double Fknee_n, double Fmin_n, int Nb_Ageing_n, double Fac_Hp_n, double Fac_Hc_n)
- 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 -> 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 AnglPol

  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 longtude
- 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 *Tsample_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, Tsample, and tStep.

10.21.3 Member Function Documentation

10.21.3.1 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().

10.21.3.2 void GWSto::DispTempVal (double *t*, ostream * *OutDisp*) [virtual]

Reimplemented from GW.

Definition at line 248 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 $h_\times(t)$.

Reimplemented from GW.

Definition at line 222 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 196 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, and tStep.

Referenced by GWSto().

10.21.3.12 void GW::setAnglPol (double *AnglPol_n*) [inherited]

$*$ Input is checked:

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

Reimplemented in GWCusp, 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

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

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

References GW::Beta, GW::DirProp, GW::Lambda, and PRECISION.

10.21.3.15 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 GWCusp, GWMono, GWNew, and GWPeriGate.

Definition at line 51 of file LISACODE-GW.h.

Referenced by GW::getAnglPol(), GWNewton2::getParam(), GWFile::getParam(), GWBinary::getParam(), GW::GW(), GWBinary::GWBinary(), GWNewton2::GWNewton2(), GW::setAnglPol(), GWNewton2::setParam(), GWFile::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(), GWFile::getParam(), GWFastSpinBBH::getParam(), GWCusp::getParam(), GWBinary::getParam(), GW::getParam(), GW::GW(), GWBinary::GWBinary(), GWFastSpinBBH::GWFastSpinBBH(), GWNewton2::GWNewton2(), GWFastSpinBBH::init(), GW::setBeta(), GW::setDirProp(), setParam(), GWPeriGate::setParam(), GWNewton2::setParam(), GWNew::setParam(), GWMono::setParam(), GWFile::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(), GWFastSpinBBH::GWFastSpinBBH(), 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(), and GWFastSpinBBH::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().

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 satelites position (SCPos), the set of photodiodes-phasemeters (PhotoDetects) and the gravitational waves transfert fonstion (sGW). It is related to memory object to which data are sent (RecordPDPM).

Definition at line 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 for on input i index, IndirectDir, and t time.

- Vect gPosSC (int i, double t)

  Gives satellite position depending at a given time t.

- void Stabilization ()

  It does phasemeters stabilization (signal and noise) for a given LISA geometry.

- void MakeOneStepOfTime (double t)

  Makes one step in time after noises loading.

- vector< double > PresentMeanNoise (double t)

  Returns the list of noises mean values, depending on t time.

Protected Attributes

- Geometry * SCPos

  Geometry pointer : spacecraft orbitography structure.

- vector< Noise * > NoisePointers

  Vector of noise pointers.

- vector< PhoDetPhaMet > PhotoDetects

  Vector of photodetector-phasemeters.

- vector< Memory * > * RecordPDPM

  Vector of memory objects where the photodetector-phasemeters signals are stored.

- TrFctGW sGW

  Transfer fonction to compute the LISA answer to gravitationnals waves.

- Background * GWB

  Background pointer : gravitational background noise (small mass binaries signal received by phasemeters).

- double tStepPhy

  Time step to simulate continuous physical process.

- double tMemRAM

  Duration (time) of photodetector-phasemeters signals stored.

- double tStepMes

  Time step for phasemeters measurement.

- vector< USOClock > USOs

  Vector of LISA USO clocks.

10.22.2 Constructor & Destructor Documentation

10.22.2.1 LISA::LISA ()

Constructs an instance and initializes it with default values.

- tStepPhy = 0.01
- tStepMes = 1.0
- tMemRAM = 60.0
- SCPos = Geometry instance with default values
- sGW = TrFctGW initialized with default values
- GWB = NULL
- RecordPDPM = 3 Memory elements intitalized with tMemRAM and tStepMes attributes

- USOs = 3 USOClock elements intitalized with 0.0 value
- NoisePointers = 18 NULL elements
- PhotoDetects = 12 elements (2 directions, 3 spacecrafts, 2 interference types : 6 first are S interference type, 6 last are TAU interference type)

Definition at line 30 of file LISACODE-LISA.cpp.

References GWB, TrFctGW::init(), NoisePointers, PhotoDetects, RecordPDPM, S, SCPos, sGW, Stabilization(), TAU, tMemRAM, tStepMes, tStepPhy, and USOs.

10.22.2.2 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 intitalized with 0.0 value
- NoisePointers = 6 white noises (NULL if *PSDLaser* input = 0.0), 6 opticals benchs noises (NULL if *PSDOptBench* input = 0.0), 6 inertials masses noises (NULL if *PSDInertialMass* input = 0.0).
- PhotoDetects = 12 elements (2 directions, 3 spacecrafts, 2 interference types : 6 first are S interference type, 6 last are TAU interference type). All noises are initialized using NoiseWhite (ex. NoiseWhite(tStepPhy, tStepMes, *tMemNoiseFirst_n*, *tMemNoiseLast_n*, {PSDLaser,PSDOptBench,PSDInertialMass}).

Definition at line 123 of file LISACODE-LISA.cpp.

References GWB, TrFctGW::init(), NoisePointers, PhotoDetects, RecordPDPM, S, SCPos, Background::setGeometry(), sGW, Stabilization(), TAU, tMemRAM, tStepMes, tStepPhy, and USOs.

10.22.2.3 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 intitalized with *ConfigLISA* input value
- NoisePointers = extracted from *ConfigLISA* input
- PhotoDetects = 12 elements (2 directions, 3 spacecrafts, 2 interference types : 6 first are S interference type, 6 last are TAU interference type)

Definition at line 252 of file LISACODE-LISA.cpp.

References ConfigSim::getGeometry(), ConfigSim::getGWB(), ConfigSim::getGWs(), ConfigSim::getNoises(), ConfigSim::getNoNoise(), ConfigSim::getPhaMetFilter(), ConfigSim::getPhaMetFilterParam(), ConfigSim::gettMemSig(), ConfigSim::gettStepMes(), ConfigSim::gettStepPhy(), ConfigSim::getUSOs(), GWB, TrFctGW::init(), NoisePointers, PhotoDetects, RecordPDPM, S, SCPos, Background::setGeometry(), sGW, Stabilization(), TAU, tMemRAM, tStepMes, tStepPhy, and USOs.

10.22.2.4 LISA::~LISA ()

Destructor.

Definition at line 339 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 405 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 365 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 426 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 493 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 522 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 454 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 35 of file LISACODE-MathUtils.h.

Static Public Member Functions

- double deg2rad (double angle_)
  Angle conversion (from degrees to radians).
- double rad2deg (double angle_)
  Angle conversion (from radians to degrees).

10.24.2 Member Function Documentation

10.24.2.1 double MathUtils::deg2rad (double *angle_*) [inline, static]

Angle conversion (from degrees to radians).

Definition at line 40 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 47 of file LISACODE-MathUtils.h.

Referenced by ConfigSim::ReadASCIIFile(), and ConfigSim::ReadXMLFile().

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)
- ∼MemoryReadDisk ()

  Destructor.

- double gettMax ()

  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)

  Adds series in ListTmpData attribute, updates AlreadyRecDat attribute, fills IndexInReadData[SerieNumber] attribute.

- void MakeTitles (char ∗FileNameHead="")

  Empty method.

- void ReceiveData (int SerieNumber, double Data)

  Sets first value of ListTmpData[SerieNumber] to data *input value and sets AlreadyRecDat[SerieNumber] to TRUE.*

- void RecordAccData (double tStep, double t)

  Records received data.

- int getNbSerie ()

  Gets the number of the serie.

- double gettStoreData ()

  Gets tStoreData attribute.

- void settStoreData (double tStoreData_n)

  Sets tStoreData attribute using tStoreData_n *input.*

- double gettStepRecord ()

  Gets tStepRecord attribute.

- void settStepRecord (double tStepRecord_n)

  Sets tStepRecord attribute using tStepRecord_n *input.*

- double gData (int SerieNumber, double delay) const

  *Returns the data of specified serie (*SerieNumber*) delayed by* tDelay *delay.*

- double gData (int SerieNumber, double delay, INTERP InterpolType, double InterpUtilValue) const

  *Returns the data of specified serie (*SerieNumber*), delayed by* tDelay *and interpolated using interpolation parameters (*InterpolType, InterpUtilValue*).*

- int unusable (double tSinceFirstReception) const

  Returns 0 if there are enough stored data to use them.

Protected Attributes

- char * NomFichMem

  File name on which data are read.

- 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 gettMax(), 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(), gettMax(), 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::gettMax(), Memory::gettMax(), Memory::gettStoreData(), Memory::Memory(), MemoryWriteDisk::RecordAccData(), RecordAccData(), Memory::RecordAccData(), Memory::settStoreData(), and Memory::unusable().

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

- LISACODE-MemoryReadDisk.h
- LISACODE-MemoryReadDisk.cpp

10.27 MemoryWriteDisk Class Reference

`#include <LISACODE-MemoryWriteDisk.h>`

Inheritance diagram for MemoryWriteDisk::

10.27.1 Detailed Description

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.

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 gettMax ()

  Gets maximum time.

- void AddSerieData (int SerieNumber, char ∗TypeName, int IndirectDirName, int iSCName)

  Adds series in ListTmpData, updates corresponding elements in AlreadyRecDat, fills TitleSerie[SerieNumber] attribute and sets SCSerie[SerieNumber].

- void MakeTitles (char ∗FileNameHead="")

  Copies header from FileNameHead *file to FichMem.*

- void RecordAccData (double tStep, double t)

  Records received data in file FichMem (make it between each step of time).

- void CloseFile ()

  Closes FichMem.

- int getNbSerie ()

  Gets the number of the serie.

- double gettStoreData ()

  Gets tStoreData attribute.

- void settStoreData (double tStoreData_n)

  Sets tStoreData attribute using tStoreData_n *input.*

- double gettStepRecord ()

  Gets tStepRecord attribute.

- void settStepRecord (double tStepRecord_n)

  Sets tStepRecord attribute using tStepRecord_n *input.*

- void ReceiveData (int SerieNumber, double data)

  Sets first value of ListTmpData[SerieNumber] to data *input value and sets AlreadyRecDat[SerieNumber] to TRUE.*

- double gData (int SerieNumber, double delay) const

  *Returns the data of specified serie (*SerieNumber*) delayed by* tDelay *delay.*

- double gData (int SerieNumber, double delay, INTERP InterpolType, double InterpUtilValue) const

  *Returns the data of specified serie (*SerieNumber*), delayed by* tDelay *and interpolated using interpolation parameters (*InterpolType, InterpUtilValue*).*

- int unusable (double tSinceFirstReception) const

  Returns 0 if there are enough stored data to use them.

Protected Attributes

- char * NomFichMem

  File name on which data are recorded.

- 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, 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 gettMax(), Memory::gettMax(), Memory::gettStoreData(), Memory::Memory(), RecordAccData(), MemoryReadDisk::RecordAccData(), Memory::RecordAccData(), Memory::settStoreData(), and Memory::unusable().

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

- LISACODE-MemoryWriteDisk.h
- LISACODE-MemoryWriteDisk.cpp

10.28 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 46 of file LISACODE-Noise.h.

Public Member Functions

- Noise ()

  Base constructor.

- Noise (double tStep_n, double tDurAdd_n)

  Constructor.

- Noise (double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n)

  Constructor.

- virtual ~Noise ()

  Destructor.

- bool TestType (char *SubmitType)

  It verifies the type of the noise (NoiseType).

- double gettStep () const

  It returns the time step between two saved data, that is the value of tStep attribute.

- void settStep (double tStep_n)

  It sets tStep value.

- double gettDurAdd () const

  It returns the duration for a noise addition, that is the value of tDurAdd attribute.

- void settDurAdd (double tDurAdd_n)

  It sets tDurAdd value and NbBinAdd.

- double gettFirst () const

  It returns the delay of the first data in data vector, that is the value of tFirst attribute.

- void settFirst (double tFirst_n)

  It sets tFirst.

- double gettLast () const

  It returns the delay of the last data in data vector, that is the value of tLast attribute.

- void settLast (double tLast_n)

  It sets tLast. It verifies that the input argument is a positif factor of tStep, otherwise it shows an error message.

- int getNbBinAdd () const

  It returns the number of bins added, that is the value of NbBinAdd attribute.

- virtual void loadNoise ()

  It initializes noise vector NoiseData.

- virtual void generNoise (int StartBin)

  Default noise generation.

- void addNoise ()

  Appends null noise data corresponding to one measurement into NoiseData attribute.

- double getNoise (double tDelay) const

  It returns the noise value for the specified delay.

- 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 97 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 85 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 90 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 94 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 81 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 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 NoiseType.

10.28.4 Member Data Documentation

10.28.4.1 int Noise::NbBinAdd [protected]

Number of bins (for each measurement).

Definition at line 58 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 60 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 62 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 64 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 52 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 54 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 56 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 50 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 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

- 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

  Multtplication 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.

$$\text{for i=0,...,Startbin } NoiseData[StartBin - i] = \begin{cases} StoredData[i] & \text{if } i \leq NbDataStored \\ StoredData[0] & \text{else} \end{cases}$$

Reimplemented from Noise.

Definition at line 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 97 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 85 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 90 of file LISACODE-Noise.h.

References Noise::tFirst.

10.29.3.10 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 94 of file LISACODE-Noise.h.

References Noise::tLast.

10.29.3.11 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 81 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]

Multtplication 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 58 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 60 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 62 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 64 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 52 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 54 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 56 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 50 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 filtererd white noise. The filter is given by the next expression:

$$y[n] = \sum_{k=1}^{N_a} alpha[k]y[n-k] + \sum_{k=0}^{N_b} beta[k]x[n-k]$$

where N_a is the number of poles and N_b is the number of zeros in the Z-transform.

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

Public Member Functions

- NoiseFilter ()

  Base constructor.

- NoiseFilter (double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n)

  Constructor.

- NoiseFilter (double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, Filter NFilter_n)

  Constructor.

- NoiseFilter (double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, vector< vector< double > > FilterAlpha_n, vector< vector< double > > FilterBeta_n)

  Constructor.

- NoiseFilter (double tStep_n, double tDurAdd_n, double tFirst_n, double tLast_n, vector< vector< double > > FilterAlpha_n, vector< vector< double > > FilterBeta_n, int FilterNbDataStab_n)

  Constructor.

- vector< vector< double > > getFilterAlpha ()

  It returns the list of the alpha parameters (see Filter::alpha) of the filter NFilter.

- vector< vector< double > > getFilterBeta ()

  It returns the list of the beta parameters (see Filter::beta) of the filter NFilter.

- void loadNoise ()

Initialization.

- void generNoise (int StartBin)

  Noise generation (for one measurement), using Startbin input as beginning index.

- bool TestType (char *SubmitType)

  It verifies the type of the noise (NoiseType).

- double gettStep () const

  It returns the time step between two saved data, that is the value of tStep attribute.

- void settStep (double tStep_n)

  It sets tStep value.

- double gettDurAdd () const

  It returns the duration for a noise addition, that is the value of tDurAdd attribute.

- void settDurAdd (double tDurAdd_n)

  It sets tDurAdd value and NbBinAdd.

- double gettFirst () const

  It returns the delay of the first data in data vector, that is the value of tFirst attribute.

- void settFirst (double tFirst_n)

  It sets tFirst.

- double gettLast () const

  It returns the delay of the last data in data vector, that is the value of tLast attribute.

- void settLast (double tLast_n)

  It sets tLast. It verifies that the input argument is a positif factor of tStep, otherwise it shows an error message.

- int getNbBinAdd () const

  It returns the number of bins added, that is the value of NbBinAdd attribute.

- void addNoise ()

  Appends null noise data corresponding to one measurement into NoiseData attribute.

- double getNoise (double tDelay) const

  It returns the noise value for the specified delay.

- 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 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 :

$$\text{for i=0,...,Startbin } WhiteData[i] = \sqrt{-\frac{log(r1)}{tStep} \cdot cos(2 \cdot \pi \cdot r1)}$$

where r1 and r2 are random values between 0 and 1 (using genunf).

Noise data are generated using Filter::App method with StartBin, WhiteData, and NoiseData arguments.

Then last data are deleted : WhiteData size and NoiseData size are set to NbData.

Reimplemented from Noise.

Definition at line 243 of file LISACODE-NoiseFilter.cpp.

References Filter::App(), genunf(), Noise::NbBinAdd, Noise::NbData, NFilter, Noise::NoiseData, Noise::tStep, and WhiteData.

10.30.3.3 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 97 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 85 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 90 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 94 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 81 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 :

$$\text{for i=0,...,Startbin } WhiteData[i] = \sqrt{-\frac{log(r1)}{tStep} \cdot cos(2 \cdot \pi \cdot r1)}$$

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 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.

10.30.4 Member Data Documentation

10.30.4.1 int Noise::NbBinAdd [protected, inherited]

Number of bins (for each measurement).

Definition at line 58 of file LISACODE-Noise.h.

Referenced by Noise::addNoise(), 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 60 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 62 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 64 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 52 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 54 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 56 of file LISACODE-Noise.h.

Referenced by Noise::gettLast(), 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 50 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 of f.

- vector< vector< double > > tmpData_m

  Temporary data for negative power of f.

- int NFp

  Number of positive power of f (f^0, f^1, f^2, ...).

- int NFm

  Number of negative power of f (f^-1, f^-2, ...).

- vector< vector< double > > FpData

Data filtered in f, f^2, f^3, ...

- vector< vector< double > > FmData

  Data filtered in f^-1, f^-2, f^-3, ...

- 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 :

$$\text{for i=0,...,Startbin } WhiteData[i] = \sqrt{-\frac{log(r1)}{tStep} \cdot cos(2 \cdot \pi \cdot r1)}$$

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 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 97 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 85 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 90 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 94 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 81 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 :

$$NbDataStab = 2 \cdot NbData + NbDataStab_{NFilter} + Max(size(\alpha_{NFilter}), size(\beta_{NFilter}))$$

WhiteData size and NoiseData size are set to NbDataStab.

WhiteData is generated as a standard gaussian :

$$\text{for i=0,...,Startbin } WhiteData[i] = \sqrt{-\frac{log(r1)}{tStep} \cdot cos(2 \cdot \pi \cdot r1)}$$

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, ...

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, ...

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 58 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 60 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, ...).

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, ...).

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 62 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 64 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 52 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 54 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 56 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 50 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 filtererd white noise. The filter is given by the next expression:

$$y[n] = \sum_{k=1}^{N_a} alpha[k]y[n-k] + \sum_{k=0}^{N_b} beta[k]x[n-k]$$

where N_a is the number of poles and N_b is the number of zeros in the Z-transform.

Definition at line 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

  Powerof 1/f –> 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 :

$$\text{for i=0,...,Startbin } WhiteData[i] = \sqrt{-\frac{log(r1)}{tStep} \cdot cos(2 \cdot \pi \cdot r1)}$$

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 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 97 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 85 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 90 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 94 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 81 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 :

$$\text{for i=0,...,Startbin } WhiteData[i] = \sqrt{-\frac{log(r1)}{tStep} \cdot cos(2 \cdot \pi \cdot r1)}$$

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 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.

10.32.4 Member Data Documentation

10.32.4.1 double NoiseOof::alpha [protected]

Powerof 1/f –> 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 58 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 60 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 62 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 64 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 52 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 54 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 56 of file LISACODE-Noise.h.

Referenced by Noise::gettLast(), 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 50 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]
  Filename (used if NType=2).

10.33.2 Member Data Documentation

10.33.2.1 char NoiseSpec::NStr[256]

Filename (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 filtererd white noise. The filter is given by the next expression:

$$y[n] = \sum_{k=1}^{N_a} alpha[k]y[n-k] + \sum_{k=0}^{N_b} beta[k]x[n-k]$$

where N_a is the number of poles and N_b is the number of zeros in the Z-transform.

Definition at line 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 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 :

$$\text{for i=0,...,Startbin } WhiteData[i] = \sqrt{-\frac{log(r1)}{tStep} \cdot cos(2 \cdot \pi \cdot r1)}$$

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 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 97 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 85 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 90 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 94 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 81 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 :

$$NbDataStab = 2 \cdot NbData + NbDataStab_{NFilter} + Max(size(\alpha_{NFilter}), size(\beta_{NFilter}))$$

WhiteData size and NoiseData size are set to NbDataStab.

WhiteData is generated as a standard gaussian :

$$\text{for i=0,...,Startbin } WhiteData[i] = \sqrt{-\frac{log(r1)}{tStep} \cdot cos(2 \cdot \pi \cdot r1)}$$

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(), 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 58 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 60 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 62 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 64 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 52 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 54 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 56 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 50 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(σ) is computed by setSqPSD from the root square of the PSD set to 1.0e-13.

- tStep = 0.01
- tDurAdd = 1.0
- tFirst = 5.0
- tLast = -20.0
- NoiseType = White

Todo

Make a function to compute a int from a double. Is there a reason to no call rint ?

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

References loadNoise(), Noise::NbData, Noise::NoiseType, PRECISION, setSqPSD(), Noise::tDurAdd, Noise::tFirst, Noise::tLast, and Noise::tStep.

10.35.2.2 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(σ) is computed from a by default PSD value.

NoiseType attribute is set to `White`.

Sigma(σ) is computed by setSqPSD from the root square of the PSD set to 1.0e-13.

Parameters:

tStep_n Value of tStep.

tDurAdd_n Value of tDurAdd.

tFirst_n Value of tFirst.

tLast_n Value of tLast.

Todo

Make a function to compute a int from a double. Is there a reason to no call rint ?

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

References loadNoise(), Noise::NbData, Noise::NoiseType, PRECISION, setSqPSD(), Noise::settDurAdd(), Noise::settFirst(), Noise::settLast(), Noise::settStep(), Noise::tFirst, Noise::tLast, and Noise::tStep.

10.35.2.3 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 :

$$\text{for i=0,...,StartBin } WhiteData[i] = \sigma \cdot \sqrt{-2 \cdot log(r2)} \cdot cos(2 \cdot \pi \cdot r1)$$

where r1 and r2 are random values between 0 and 1 (using genunf).

Reimplemented from Noise.

Definition at line 170 of file LISACODE-NoiseWhite.cpp.

References genunf(), Noise::NoiseData, and Sigma.

10.35.3.3 int Noise::getNbBinAdd () const [inline, inherited]

It returns the number of bins added, that is the value of NbBinAdd attribute.

Definition at line 97 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 85 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 90 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 94 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 81 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 :

$$\text{for i=0,...,NbData } WhiteData[i] = \sigma \cdot \sqrt{-2 \cdot log(r2)} \cdot cos(2 \cdot \pi \cdot r1)$$

where r1 and r2 are random values between 0 and 1 (using genunf).

Reimplemented from Noise.

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

References genunf(), Noise::NbData, Noise::NoiseData, and Sigma.

Referenced by NoiseWhite().

10.35.3.13 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:

$$\sigma = \frac{SqPSD}{\sqrt{2.0 \cdot tStep}}$$

where $tStep$ tStep is the time between to samples. If $SqPSD$ is negatif a message is shown.

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

References Sigma, and Noise::tStep.

Referenced by NoiseWhite().

10.35.3.14 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 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.

10.35.4 Member Data Documentation

10.35.4.1 int Noise::NbBinAdd [protected, inherited]

Number of bins (for each measurement).

Definition at line 58 of file LISACODE-Noise.h.

Referenced by Noise::addNoise(), 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 60 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 62 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 64 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 52 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 54 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 56 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 50 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, substraction, delay).

Gravitational Wave contribution is added.

Noises sampling is equal to physical time step.

Phasemeter output is phase difference between 2 beams received by photodiode; its sampling is equal to measurement time step.

Low pass filter keeps only frequencies lower than half measurement frequency.

Simulator time step is equal to measurement step, so that 1 phasemeter signal data is stored in memory when 1 signal data is received.

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

Public Member Functions

- PhoDetPhaMet ()

  Constructs an instance and initializes it with default values.

- PhoDetPhaMet (PDPMINTERF InterfType_n, int IndirectDir_n, int iSC_n, Geometry ∗SCPos_n, vector< Noise ∗ > ∗NPs_n, USOClock ∗USO_n, Memory ∗RecordData_n, double tStepPhy_n, double tStepMes_n)

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

- PhoDetPhaMet (PDPMINTERF InterfType_n, int IndirectDir_n, int iSC_n, Geometry ∗SCPos_n, vector< Noise ∗ > ∗NPs_n, USOClock ∗USO_n, Memory ∗RecordData_n, TrFctGW ∗sGW_n, Background ∗GWB_n, double tStepPhy_n, double tStepMes_n)

  *Constructs an instance and initializes it with default values and inputs. It is like previous constructor with added inputs for the transfer function (*sGW_n*) and the background (*GWB_n*).*

- PhoDetPhaMet (PDPMINTERF InterfType_n, int IndirectDir_n, int iSC_n, Geometry ∗SCPos_n, vector< Noise ∗ > ∗NPs_n, USOClock ∗USO_n, Memory ∗RecordData_n, TrFctGW ∗sGW_n, Background ∗GWB_n, double tStepPhy_n, double tStepMes_n, bool FilterON_n, vector< double > FilterParam_n)

  Constructs an instance and initializes it with inputs. It is like previous constructor with added inputs for filter description.

- ∼PhoDetPhaMet ()

  Destructor.

- void init (PDPMINTERF InterfType_n, int IndirectDir_n, int iSC_n, Geometry ∗SCPos_n, vector< Noise ∗ > ∗NPs_n, USOClock ∗USO_n, Memory ∗RecordData_n, TrFctGW ∗sGW_n, Background ∗GWB_n, double tStepPhy_n, double tStepMes_n, bool FilterON_n, vector< double > FilterParam_n)

Initializes an instance with default values and inputs.

- double getIndirectDir () const

  Returns IndirectDir attribute.

- double getiSC () const

  Returns iSC attribute.

- void DisplayStoredData ()

  Displays stored data.

- double gettStab ()

  Gets stabilization time.

- double gN (NOISEORIG OrigN, int iSC, int IndirectDir, double tDelay)

  Returns value of specified noise after delay computation.

- double gGWB (int iSC, int IndirectDir, double t)

  Returns value of Gravitationnal Wave Background (iSC={1,2,3}, IndirectDir={0,1}).

- void ReceiveSignal (double t)

  Computes the signal received by photodetector-phasemeters.

- void IntegrateSignal (double t)

  Stores the result in memory (one measurement).

- bool getNoNoise ()

  Indicates if noises are present or not. It returns true if there are no noises.

Protected Attributes

- PDPMINTERF InterfType

  Type of interferences made by the photodetector-phasemeter.

- int IndirectDir

  Direction flag : 0 if the optical bench is in the direct direction, else 1.

- int iSC

  Spacecraft index corresponding to the photodetector-phasemeter.

- Geometry * SCPos

  Pointer to LISA geometry.

- vector< Noise * > * NPs

  Noise pointers vector.

- USOClock * USO

  Pointers to spacecrafts USO clocks.

- Memory * RecordData

  Pointer to the storage memory of the photodetector-phasemeters signal.

- TrFctGW * sGW

  Pointer to the transfer function.

- Background * GWB

  Background pointer : confusion whites dwarfs background.

- double tStepPhy

  Physical simulation time step.

- double tStepMes

  Measurement time step.

- vector< double > InterfPhyData

  Physical data vector.

- vector< double > FilterPhyData

  Filtered physical data vector.

- int NbDataStored

  Number of data stored (in InterfPhyData and FilterPhyData).

- int NbDataAdd

  Number of data added (in InterfPhyData and FilterPhyData) for each measurement.

- Filter * PBFilter

  Filter pointer to a low pass filter.

- bool FilterON

  Filter flag : If true, the filter is applied.

- vector< double > FilterParam

  Filter parameters : attenuation [dB], oscillations in bandwidth [dB], low transition frequency divided by measurement frequency, high transition frequency divided by measurement frequency.

- bool NoNoise

  Noise flag : if true, there is no noise.

10.36.2 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 :

$$indexNPs = \begin{cases} 3 \cdot IndirectDir + iSC - 1 & \text{if OrigN=LA} \\ 3 \cdot IndirectDir + iSC - 1 + 6 & \text{if OrigN=OB} \\ 3 \cdot IndirectDir + iSC - 1 + 12 & \text{if OrigN=IM} \\ 3 \cdot IndirectDir + iSC - 1 + 18 & \text{if OrigN=OP} \\ 24 & \text{else} \end{cases}$$

Then, noise corresponding to that index and tDelay time is given by Noise::getNoise method and its value is returned.

Definition at line 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

InterfPhyData = NbDataStored null elements, with $NbDataAdd = 2 \cdot (int)(\frac{tStepMes}{tStepPhy} + \frac{1}{2})$

if (FilterON) $\begin{cases} \text{PBFilter = new Filter(1/tStepPhy, FilterParam[0], FilterParam[1], FilterParam[2]/tStepMes, FilterParam[} \\ \text{FilterPhyData = NbDataStored null elements} \end{cases}$

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

References Memory::AddSerieData(), FilterON, FilterParam, FilterPhyData, Filter::getDepth(), Filter::getNbDataStab(), getNoNoise(), GWB, IndirectDir, InterfPhyData, InterfType, iSC, MAX, NbDataAdd, NbDataStored, NoNoise, NPs, PBFilter, RecordData, S, SCPos, sGW, TAU, tStepMes, tStepPhy, and USO.

Referenced by PhoDetPhaMet().

10.36.3.9 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 :

$s_1 = s_1^{GW} + d_1^{OPN} + D_3 p_2' - p_1 - 2 * d_1^{IM} ...(+D_3 d'^{OB}_2 + d_1^{OB})$

$s_1' = s'^{GW}_1 + d'^{OPN}_1 + D'_2 p_3 - p'_1 + 2 * d'^{IM}_1 ...(-D_2 d_3 3^{OB} - d'^{OB}_1)$

Phasemeter S : Between distant and local laser beam :

$\tau_1 = p'_1 - p_1 - 2 * d_1^{IM} ...(+2 d'^{OB}_1)$

$\tau_1' = p_1 - p'_1 + 2 * d_1^{IM} ...(-2 d_1^{OB})$

Computations :

- if (InterfType=S)

  $GWSignal = sGW->deltanu(iSC, modulo(iSC + 1 + IndirectDir, 3), 2, t)$

  $+gGWB(iSC, IndirectDir, t)$

  for $i = 0, \ldots, NbDataAdd - 1$,

  $$\begin{cases} \text{if (NoNoise)} & InterfPhyData[i] = GWSignal \\ \text{else} & InterfPhyData[i] = GWSignal + noise_i \end{cases}$$

  $noise_i = gN(OP, iSC, IndirectDir, tDPhy_i)$

  $+gN(LA, iSCpe1, 1 - IndirectDir, tDPhy_i$

$+(*SCPos).gtdelay(modulo(iSC+1+IndirectDir,3),iSC,2,t+tDPhy_i))$

$+gN(OB,iSCpe1,1-IndirectDir,tDPhy_i$

$+(*SCPos).gtdelay(modulo(iSC+1+IndirectDir,3),iSC,2,t+tDPhy_i))$

$-2\cdot gN(IM,iSC,0,tDPhy_i)$

$-gN(LA,iSC,IndirectDir,tDPhy_i)$

$+gN(OB,iSC,IndirectDir,tDPhy_i)$

$tDPhy_i = USO->gGap(t,tStepPhy)-i\cdot tStepPhy$

- if (InterfType=TAU)

  for $i=0,\ldots,NbDataAdd-1$

  $InterfPhyData[i]=gN(LA,iSC,1,tDPhy_i)$

  $-2\cdot gN(IM,iSC,1-IndirectDir,tDPhy_i)$

  $+2\cdot gN(OB,iSC,1-IndirectDir,tDPhy_i)$

  $-gN(LA,iSC,IndirectDir,tDPhy_i)$

PhoDetPhaMet::gN , TrFctGW::deltanu USOClock::gGap and Geometry::gtdelay methods are called.

Definition at line 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 betwwen 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:

$$\begin{cases} TruncVal(x) & \text{if } modulo(\frac{x-x_0}{dx}, 1) \approx 0 \\ TruncVal(x) & \text{if } InterpolType = TRU \\ InterLinear(x) & \text{if } InterpolType = LIN \\ InterCubic(x) & \text{if } InterpolType = CUB \\ InterLagrange(x, int(InterpUtilValue)) & \text{if } InterpolType = LAG \end{cases}$$

Definition at line 259 of file LISACODE-Serie.cpp.

References CUB, dx, InterCubic(), InterLagrange(), InterLinear(), LAG, LIN, PRECISION, TRU, TruncVal(), and x0.

10.38.3.5 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:
$x0 + dx \cdot bin$

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 :

$bin_1 = floor(\frac{x-x_0}{dx})$

$bin_2 = bin_1 + 1$

$bin_0 = bin_1 - 1$

$bin_3 = bin_2 + 1$

Indices are checked ; bin_0 must be positive or null, and bin_3 must be lower than ys attribute size.

Returns:
$\mu^3 \cdot (ys[bin_3] - ys[bin_2] - ys[bin_0] + ys[bin_1])$
$+\mu^2 \cdot (2 \cdot ys[bin_0] - 2 \cdot ys[bin_1] + ys[bin_2] - ys[bin_3])$
$+\mu \cdot (ys[bin_2] - ys[bin_0])$
$+ys[bin_3]$,
where :
$\mu = \frac{x-x_0}{dx} - floor(\frac{x-x_0}{dx})$

Definition at line 349 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

Referenced by gData().

10.38.3.10 double Serie::InterHermite (double *x*, double *tension*, double *bias*) const

Returns hermite interpolation result, depending on x, tension and bias inputs.

Indices are computed :

$bin_1 = floor(\frac{x-x_0}{dx})$

$bin_2 = bin_1 + 1$

$bin_0 = bin_1 - 1$

$bin_3 = bin_2 + 1$

Indices are checked ; bin_0 must be positive or null, and bin_3 must be lower than ys attribute size.

Returns:
$(\mu^3 - 2 \cdot \mu^2 + \mu) \cdot ys[bin_1] + (\mu^3 - 2 \cdot \mu^2 + \mu)) \cdot ((ys[bin_1] - ys[bin_0]) \cdot (1 + bias) \cdot \frac{1-tension}{2} + (ys[bin_2] - ys[bin_1]) \cdot (1 - bias) \cdot \frac{1-tension}{2})$,
where :
$\mu = \frac{x-x_0}{dx} - floor(\frac{x-x_0}{dx})$

Definition at line 394 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

10.38.3.11 double Serie::InterLagrange (double *x*, int *order*) const

Returns Lagrange interpolation result.

Indices are computed :

$bin = floor(\frac{x-x_0}{dx})$

$kmin = bin - ordermin + 1$, where $ordermin = floor(\frac{order+1}{2})$

$kmax = bin + order + 1 - ordermin$

Indices are checked :

- bin must be positive or null, and lower than (ys attribute size -1)
- kmin must be positive or null
- kmax must be and lower than (ys attribute size -1)

Returns:
$\sum_{k=kmin}^{kmax} ys[k] \cdot P_k$, where $P_k = \prod_{j=kmin, j \neq k}^{kmax} \frac{x-x_0-j \cdot dx}{(k-j) \cdot dx}$

Definition at line 445 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

Referenced by gData().

10.38.3.12 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 :

$$bin = floor(\frac{x - x_0}{dx})$$

Then bin is checked ; it must be positive or null, and lower than ys attribute size.

Returns:

$$ys[bin]$$

Definition at line 300 of file LISACODE-Serie.cpp.

References dx, x0, and ys.

Referenced by gData().

10.38.3.19 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 *lenght*)

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

- x0 = start
- dx = delta
- ys = lenght elements, with 0 values

Definition at line 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:
$x0 + dx \cdot bin$

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 *lenght*)

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
- TDIQuickMod = empty

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

References Eta, Fact, IndexDelay, IndexEta, iSerie, OutFile, OutFileEncoding, SCSig, Sign, TDelay, and TDIQuickMod.

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
- TDIQuickMod = empty

Definition at line 61 of file LISACODE-TDI.cpp.

References Eta, Fact, IndexDelay, IndexEta, iSerie, OutFile, OutFileEncoding, SCSig, Sign, TDelay, and TDIQuickMod.

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 * *TDIQuickMod_n*)

Constructs an instance and initializes it using TDelay_n, Eta_n, OutFile_n, iSerie_n, Sign_n, IndexEta_n, IndexDelay_n and TDIQuickMod_n inputs.

Attributes are :

- TDelay = TDelay_n
- Eta = Eta_n
- OutFile = OutFile_n
- iSerie = iSerie_n
- Sign = Sign_n
- IndexEta = IndexEta_n
- IndexDelay = IndexDelay_n
- TDIQuickMod = TDIQuickMod_n
- tmpCountInterDelay = 0
- tmpCountInterEta = 0

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

References Eta, Fact, IndexDelay, IndexEta, iSerie, OutFile, OutFileEncoding, SCSig, Sign, TDelay, TDIQuickMod, 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 * *TDIQuickMod_n*)

Constructs an instance and initializes it using TDelay_n, Eta_n, OutFile_n, iSerie_n, SignEtaDelays and TDIQuickMod_n inputs.

- TDelay = TDelay_n
- Eta = Eta_n
- OutFile = OutFile_n
- iSerie = iSerie_n
- Sign : set by ReadSignEtaDelays, using SignEtaDelays input
- IndexEta : set by ReadSignEtaDelays, using SignEtaDelays input
- IndexDelay : set by ReadSignEtaDelays, using SignEtaDelays input

- TDIQuickMod = TDIQuickMod_n
- tmpCountInterDelay = 0
- tmpCountInterEta = 0

Definition at line 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*, 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.

- 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

$$TotalDelay = \sum_{iPack=0}^{size(IndexEta)-1} \sum_{iDelay=0}^{size(IndexDelay[iPack])-1}$$

$$TDIQuickMod->getDelay((IndexDelay[iPack])[iDelay])$$

else

$$TotalDelay = \sum_{iPack=0}^{size(IndexEta)-1} \sum_{iDelay=0}^{size(IndexDelay[iPack])-1}$$

$$TDelay->gData((IndexDelay[iPack])[iDelay]-1, TotalDelay, LAG, 6)$$

TDITools::getDelay and Memory::gData methods are called.

Returns:

if enough data are stored

$$\sum_{iPack=0}^{size(IndexEta)-1} \sum_{iDelay=0}^{size(IndexDelay[iPack])-1}$$

$$Sign[iPack] \cdot Eta->gData(IndexEta[iPack], TotalDelay)$$

else 0

Definition at line 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 TDIQuickMod.

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, LAG, SCSig, Sign, and TDIQuickMod.

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 456 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 $iPack = 0, \ldots, size(SignEtaDelays) - 1$:

  SignEtaDelays[iPack] is checked : $SignEtaDelays[iPack] \neq 0$

  $$\begin{cases} \text{if } (SignEtaDelays[iPack] > 0) & +1 \text{ is pushed back in Sign attribute} \\ \text{if } (SignEtaDelays[iPack] < 0) & -1 \text{ is pushed back in Sign attribute} \end{cases}$$

  $$TmpInfo = abs(SignEtaDelays[iPack])$$

  $$TmpIndexEta = ceil(10 \cdot (\frac{TmpInfo}{10} - floor(\frac{TmpInfo}{10}) + 10^{-}6))$$

  TmpIndexEta is checked : $1 \leq TmpIndexEta \leq 6$

  TmpIndexEta is pushed back in IndexEta attribute

- $TmpInfo = ceil(\frac{TmpInfo}{10})$; while $tmpInfo \neq 0$:

  $$TmpIndexDelay = ceil(10 \cdot (\frac{TmpInfo}{10} - floor(\frac{TmpInfo}{10}) + 10^{-}6))$$

  TmpIndexDelay is checked :$1 \leq TmpIndexDelay \leq 6$

  TmpIndexDelay is pushed back in IndexDelay attribute

  $$TmpInfo = ceil(\frac{TmpInfo}{10})$$

Definition at line 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 442 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 ()

  Desctrutor.

- double getTimeStore ()

  Returns TimeStore attribue .

- void setTimeStore (double TimeStored_n)

  Sets TimeStore attribute using TimeStore_n input.

- double gettStep ()

  Returns first Eta ref step attribue .

- bool getUsable ()

  Returns Usable attribute.

- double gettDelayCompute ()

  Returns tShift attribute.

- void ComputeEta ()

  Computes Eta using TimeStore and NoNoise attributes.

- double gData (int iSC, int IndirectDir, double Delay)

  Returns value interpolated for the delay (iSC=[1,3] and Indirect=[0,1].

- double gData (int iSerie, double Delay)

  Returns value interpolated for the delay (iSerie=[1,6]).

Protected Attributes

- Memory * TDelay

  Pointer to the list of delay's lengths.

- vector< Memory * > * PDPMMem

  Memory where the signals of photodetector-phasemeter are stored.

- double TimeStore

  Time during which the values Eta are stored.

- vector< Serie > Eta

  List of the memory where Eta's values are stored.

- bool Usable

  True if enough data are stored.

- INTERP InterpType

  Type of interpolation used to obtain and to get data.

- double InterpUtilValue

  Value used for interpolation.

- double tShift

  Delay between the compute of Eta and the signals.

- bool NoNoise

  If true, there are no noise.

10.41.2 Constructor & Destructor Documentation

10.41.2.1 TDI_InterData::TDI_InterData ()

Constructs an instance and initializes it with default values.

Attributes are :

- TDelay = NULL
- PDPMMem = NULL
- TimeStore = 30.0
- tShift = 0.0
- Eta = 6 elements initialzed with tShift and 1.0 arguments
- Usable = false
- InterpType = LAG
- InterpUtilValue = 6

- NoNoise = false

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

References Eta, InterpType, InterpUtilValue, LAG, NoNoise, PDPMMem, TDelay, TimeStore, tShift, and Usable.

10.41.2.2 TDI_InterData::TDI_InterData (Memory * *TDelay_n*, vector< Memory * > * *PDPMMem_n*)

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

Attributes are :

- TDelay = TDelay_n
- PDPMMem = NULL
- TimeStore = 30.0
- tShift = 0.0
- Eta = 6 elements initialzed with tShift and 1.0 arguments
- Usable = false
- InterpType = LAG
- InterpUtilValue = 6
- NoNoise = false

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

References Eta, InterpType, InterpUtilValue, LAG, NoNoise, PDPMMem, TDelay, TimeStore, tShift, and Usable.

10.41.2.3 TDI_InterData::TDI_InterData (Memory * *TDelay_n*, vector< Memory * > * *PDPMMem_n*, double *TimeStore_n*, double *tShift_n*, bool *NoNoise_n*, INTERP *InterpType_n* = LAG, double *InterpUtilValue_n* = 6)

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

Attributes are :

- TDelay = TDelay_n
- PDPMMem = PDPMMem_n
- TimeStore = TimeStore_n (checked; expected positive or null value)
- tShift = tShift_n
- Eta = 6 elements initialzed with tShift and step (from PDPMMem) arguments
- Usable = false
- InterpType = InterpUtilValue_n

- InterpUtilValue = 6
- NoNoise = false

Definition at line 91 of file LISACODE-TDI_InterData.cpp.

References Eta, InterpType, InterpUtilValue, NoNoise, PDPMMem, TDelay, TimeStore, tShift, and Usable.

10.41.2.4 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

  $Eta[iSC-1].addData((*PDPMMem)[iSC-1]->gData(0, tShift, InterpType, InterpUtilValue))$

  $Eta[iSC+2].addData((*PDPMMem)[iSC-1]->gData(1, tShift, InterpType, InterpUtilValue))$

  using Memory::gData method
- else the following value is added at the begining of Eta[iSC-1] serie, using Serie::addData method

$$(*PDPMMem)[iSC-1]->gData(0, tShift, InterpType, InterpUtilValue)$$
$$-\frac{1}{2}\cdot(*PDPMMem)[mod(iSC+1,3)]->gData(2, tShift+TDelay->gData(mod(iSC+2,3), tShift), InterpType, InterpUtilValue)$$
$$-\frac{1}{2}\cdot(*PDPMMem)[mod(iSC+1,3)]->gData(3, tShift+TDelay->gData(mod(iSC+2,3), tShift), InterpType, InterpUtilValue)$$

  and the following value is added at the begining of Eta[iSC+2] serie, using Serie::addData method

$$(*PDPMMem)[iSC-1]->gData(1, tShift, InterpType, InterpUtilValue)$$
$$+\frac{1}{2}\cdot(*PDPMMem)[iSC-1]->gData(2, tShift, InterpType, InterpUtilValue)$$
$$-\frac{1}{2}\cdot(*PDPMMem)[iSC-1]->gData(3, tShift, InterpType, InterpUtilValue)$$

Last data of series Eta[iSC+2] and Eta[iSC-1], while x reference is greater than TimeStore input, using Serie::delLastData method.

Definition at line 161 of file LISACODE-TDI_InterData.cpp.

References Eta, Memory::gData(), gData(), InterpType, InterpUtilValue, NoNoise, PDPMMem, TDelay, TimeStore, tShift, and Usable.

Referenced by main().

10.41.3.2 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(), ComputeEta(), and main().

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(), and main().

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::PDPMMem [protected]

Memory where the signals of photodetector-phasemeter are stored.

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

Referenced by ComputeEta(), and TDI_InterData().

10.41.4.6 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 ech delay (0,1,2), Memory::gdata method is used with tComputeDelay input delay, truncated interpolation type, and zero InterpUtilValue.

Definition at line 73 of file LISACODE-TDITools.cpp.

References DelayMem, Memory::gData(), TDelay, and TRU.

Referenced by main().

10.42.4 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.

$$tem = trec + LISAGeo->gtdelay(em, rec, order, trec)$$

$$\overrightarrow{rrec} = \frac{LISAGeo->gposition(rec, trec)}{C}$$

$$\overrightarrow{rem} = LISAGeo->gposition(em, trec)$$

$$\overrightarrow{dr} = \overrightarrow{rrec} - \overrightarrow{rem}$$

$$\overrightarrow{n} = \frac{\overrightarrow{dr}}{\|\overrightarrow{dr}\|}$$

$$\delta_\nu = 0$$

For each source (iGWindex) in GWSources,

$$hpr = (*GWSources)[iGW]->hp(trec - \overrightarrow{k[iGW]} \cdot \overrightarrow{rrec})$$

$$hpe = (*GWSources)[iGW]->hp(tem - \overrightarrow{k[iGW]} \cdot \overrightarrow{rem})$$

$$hcr = (*GWSources)[iGW]->hc(trec - \overrightarrow{k[iGW]} \cdot \overrightarrow{rrec})$$

$$hce = (*GWSources)[iGW]->hc(tem - \overrightarrow{k[iGW]} \cdot \overrightarrow{rem})$$

$$\delta_\nu = \delta_\nu + \frac{(hpe - hpr) \cdot \frac{(\overrightarrow{u[iGW]} \cdot \overrightarrow{n})^2 - (\overrightarrow{v[iGW]} \cdot \overrightarrow{n})^2}{2} + (hce - hcr) \cdot (\overrightarrow{u[iGW]} \cdot \overrightarrow{n}) \cdot (\overrightarrow{v[iGW]} \cdot \overrightarrow{n})}{1 - \frac{\overrightarrow{k[iGW]} \cdot \overrightarrow{rrec} - \overrightarrow{k[iGW]} \cdot \overrightarrow{rem}}{\|\overrightarrow{dr}\|}}$$

Definition at line 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 $\phi = \lambda$.

$$\lambda = \lambda_{GW} + \pi$$

$$\beta = -\beta_{GW} + \pi$$

$$\psi = AnglPol_{GW}$$

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

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

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

Definition at line 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 the 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 the is no noise, else TRUE.

Definition at line 53 of file LISACODE-USOClock.h.

Referenced by gGap(), and init().

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

- LISACODE-USOClock.h
- LISACODE-USOClock.cpp

10.45 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.

$$\text{returned value} = \frac{\overrightarrow{u}}{a}$$

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/(), GeometryMLDC::position(), GeometryFile::position(), GeometryAnalytic::position(), Geometry::position(), unit(), Vect(), Geometry::VectNormal(), GeometryMLDC::velocity(), GeometryFile::velocity(), GeometryAnalytic::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
- #define numg 32L
- #define numg 32L
- #define numg 32L
- #define numg 32L
- #define numg 32L
- #define numg 32L
- #define numg 32L

Functions

- void advnst (long k)
- void getsd (long *iseed1, long *iseed2)
- long ignlgi (void)
- void initgn (long isdtyp)
- void inrgcm (void)
- void setall (long iseed1, long iseed2)
- void setant (long qvalue)
- void setsd (long iseed1, long iseed2)

Variables

- long Xm1
- long Xm2

- long Xa1
- long Xa2
- long Xcg1 [32]
- long Xcg2 [32]
- long Xa1w
- long Xa2w
- long Xig1 [32]
- long Xig2 [32]
- long Xlg1 [32]
- long Xlg2 [32]
- long Xa1vw
- long Xa2vw
- long Xqanti [32]

11.1.1 Define Documentation

11.1.1.1 #define numg 32L

11.1.1.2 #define numg 32L

11.1.1.3 #define numg 32L

11.1.1.4 #define numg 32L

11.1.1.5 #define numg 32L

11.1.1.6 #define numg 32L

11.1.1.7 #define numg 32L

11.1.1.8 #define numg 32L

11.1.2 Function Documentation

11.1.2.1 void advnst (long *k*)

Definition at line 7 of file com.c.

References gscgn(), gsrgs(), mltmod(), setsd(), Xa1, Xa2, Xcg1, Xcg2, Xm1, and Xm2.

11.1.2.2 void getsd (long ∗ *iseed1*, long ∗ *iseed2*)

Definition at line 52 of file com.c.

References gscgn(), gsrgs(), Xcg1, and Xcg2.

Referenced by main().

11.1.2.3 long ignlgi (void)

Definition at line 89 of file com.c.

References gscgn(), gsrgs(), gssst(), ignlgi(), inrgcm(), setall(), Xa1, Xa2, Xcg1, Xcg2, Xm1, Xm2, and Xqanti.

Referenced by ignlgi(), ignuin(), and ranf().

11.1.2.4 void initgn (long *isdtyp*)

Definition at line 143 of file com.c.

References gscgn(), gsrgs(), mltmod(), Xa1w, Xa2w, Xcg1, Xcg2, Xig1, Xig2, Xlg1, Xlg2, Xm1, and Xm2.

Referenced by setall(), and setsd().

11.1.2.5 void inrgcm (void)

Definition at line 203 of file com.c.

References gsrgs(), Xa1, Xa1vw, Xa1w, Xa2, Xa2vw, Xa2w, Xm1, Xm2, and Xqanti.

Referenced by ignlgi(), and setall().

11.1.2.6 void setall (long *iseed1*, long *iseed2*)

Definition at line 245 of file com.c.

References gscgn(), gsrgs(), gssst(), initgn(), inrgcm(), mltmod(), Xa1vw, Xa2vw, Xig1, Xig2, Xm1, and Xm2.

Referenced by ignlgi(), and main().

11.1.2.7 void setant (long *qvalue*)

Definition at line 296 of file com.c.

References gscgn(), gsrgs(), and Xqanti.

11.1.2.8 void setsd (long *iseed1*, long *iseed2*)

Definition at line 337 of file com.c.

References gscgn(), gsrgs(), initgn(), Xig1, and Xig2.

Referenced by advnst().

11.1.3 Variable Documentation

11.1.3.1 long Xa1 [static]

Definition at line 4 of file com.c.

Referenced by advnst(), ignlgi(), and inrgcm().

11.1.3.2 long Xa1vw [static]

Definition at line 4 of file com.c.

Referenced by inrgcm(), and setall().

11.1.3.3 long Xa1w [static]

Definition at line 4 of file com.c.

Referenced by initgn(), and inrgcm().

11.1.3.4 long Xa2 [static]

Definition at line 4 of file com.c.

Referenced by advnst(), ignlgi(), and inrgcm().

11.1.3.5 long Xa2vw [static]

Definition at line 4 of file com.c.

Referenced by inrgcm(), and setall().

11.1.3.6 long Xa2w [static]

Definition at line 4 of file com.c.

Referenced by initgn(), and inrgcm().

11.1.3.7 long Xcg1[32] [static]

Definition at line 4 of file com.c.

Referenced by advnst(), getsd(), ignlgi(), and initgn().

11.1.3.8 long Xcg2[32] [static]

Definition at line 4 of file com.c.

Referenced by advnst(), getsd(), ignlgi(), and initgn().

11.1.3.9 long Xig1[32] [static]

Definition at line 4 of file com.c.

Referenced by initgn(), setall(), and setsd().

11.1.3.10 long Xig2[32] [static]

Definition at line 4 of file com.c.

Referenced by initgn(), setall(), and setsd().

11.1.3.11 long Xlg1[32] [static]

Definition at line 4 of file com.c.

Referenced by initgn().

11.1.3.12 long Xlg2[32] [static]

Definition at line 4 of file com.c.

Referenced by initgn().

11.1.3.13 long Xm1 [static]

Definition at line 4 of file com.c.

Referenced by advnst(), ignlgi(), initgn(), inrgcm(), and setall().

11.1.3.14 long Xm2 [static]

Definition at line 4 of file com.c.

Referenced by advnst(), ignlgi(), initgn(), inrgcm(), and setall().

11.1.3.15 long Xqanti[32] [static]

Definition at line 6 of file com.c.

Referenced by ignlgi(), inrgcm(), and setant().

11.2 Doxygen.bibliography File Reference

11.3 ezxml.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <ctype.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include "ezxml.h"
```

Classes

- struct ezxml_root

Defines

- #define EZXML_WS "\t\r\n "
- #define EZXML_ERRL 128

Typedefs

- typedef ezxml_root * ezxml_root_t

Functions

- ezxml_t ezxml_child (ezxml_t xml, const char *name)

  Returns the first child tag (one level deeper) with the given name or NULL if not found.

- ezxml_t ezxml_idx (ezxml_t xml, int idx)

  Returns the Nth tag with the same name in the same section at the same depth or NULL if not found. An index of 0 returns the tag given.

- const char * ezxml_attr (ezxml_t xml, const char *attr)

  Returns the value of the requested tag attribute, or NULL if not found.

- ezxml_t ezxml_vget (ezxml_t xml, va_list ap)
- ezxml_t ezxml_get (ezxml_t xml,...)

  Traverses the ezxml sturcture to retrieve a specific subtag.

- const char ** ezxml_pi (ezxml_t xml, const char *target)

  Returns a NULL terminated array of processing instructions for the given target.

- ezxml_t ezxml_err (ezxml_root_t root, char *s, const char *err,...)
- char * ezxml_decode (char *s, char **ent, char t)
- void ezxml_open_tag (ezxml_root_t root, char *name, char **attr)
- void ezxml_char_content (ezxml_root_t root, char *s, size_t len, char t)
- ezxml_t ezxml_close_tag (ezxml_root_t root, char *name, char *s)
- int ezxml_ent_ok (char *name, char *s, char **ent)
- void ezxml_proc_inst (ezxml_root_t root, char *s, size_t len)
- short ezxml_internal_dtd (ezxml_root_t root, char *s, size_t len)
- char * ezxml_str2utf8 (char **s, size_t *len)
- void ezxml_free_attr (char **attr)
- ezxml_t ezxml_parse_str (char *s, size_t len)

  Given a string of xml data and its length, parses it and creates an ezxml structure. For efficiency, modifies the data by adding null terminators and decoding ampersand sequences. If you don't want this, copy the data and pass in the copy. Returns NULL on failure.

- ezxml_t ezxml_parse_fp (FILE *fp)

  Wrapper for ezxml_parse_str() that accepts a file stream. Reads the entire stream into memory and then parses it. For xml files, use ezxml_parse_file() or ezxml_parse_fd().

- ezxml_t ezxml_parse_fd (int fd)

  A wrapper for ezxml_parse_str() that accepts a file descriptor. First attempts to mem map the file. Failing that, reads the file into memory. Returns NULL on failure.

- ezxml_t ezxml_parse_file (const char *file)

  a wrapper for ezxml_parse_fd() that accepts a file name

- char * ezxml_ampencode (const char *s, size_t len, char **dst, size_t *dlen, size_t *max, short a)
- char * ezxml_toxml_r (ezxml_t xml, char **s, size_t *len, size_t *max, size_t start, char ***attr)
- char * ezxml_toxml (ezxml_t xml)

  Converts an ezxml structure back to xml. Returns a string of xml data that must be freed.

- void ezxml_free (ezxml_t xml)

  Frees the memory allocated for an ezxml structure.

- const char * ezxml_error (ezxml_t xml)

  Returns parser error message or empty string if none.

- ezxml_t ezxml_new (const char *name)

  Returns a new empty ezxml structure with the given root tag name.

- ezxml_t ezxml_add_child (ezxml_t xml, const char *name, size_t off)

  Adds a child tag. off is the offset of the child tag relative to the start of the parent tag's character content. Returns the child tag.

- ezxml_t ezxml_set_txt (ezxml_t xml, const char *txt)

  Sets the character content for the given tag and returns the tag.

- void ezxml_set_attr (ezxml_t xml, const char *name, const char *value)

  Sets the given tag attribute or adds a new attribute if not found. A value of NULL will remove the specified attribute.

- ezxml_t ezxml_set_flag (ezxml_t xml, short flag)

  Sets a flag for the given tag and returns the tag.

- void ezxml_remove (ezxml_t xml)

  Removes a tag along with all its subtags.

Variables

- char * EZXML_NIL [ ] = { NULL }

11.3.1 Define Documentation

11.3.1.1 #define EZXML_ERRL 128

Definition at line 39 of file ezxml.c.

Referenced by ezxml_err().

11.3.1.2 #define EZXML_WS "\t\r\n "

Definition at line 38 of file ezxml.c.

Referenced by ezxml_internal_dtd(), ezxml_parse_str(), and ezxml_proc_inst().

11.3.2 Typedef Documentation

11.3.2.1 typedef struct ezxml_root* ezxml_root_t

Definition at line 41 of file ezxml.c.

Referenced by ezxml_attr(), ezxml_char_content(), ezxml_close_tag(), ezxml_err(), ezxml_error(), ezxml_free(), ezxml_internal_dtd(), ezxml_new(), ezxml_open_tag(), ezxml_parse_fd(), ezxml_parse_-fp(), ezxml_parse_str(), ezxml_pi(), ezxml_proc_inst(), and ezxml_toxml().

11.3.3 Function Documentation

11.3.3.1 ezxml_t ezxml_add_child (ezxml_t *xml*, const char * *name*, size_t *off*)

Adds a child tag. off is the offset of the child tag relative to the start of the parent tag's character content. Returns the child tag.

Definition at line 850 of file ezxml.c.

References ezxml::attr, ezxml::child, EZXML_NIL, ezxml_t, ezxml::name, ezxml::next, ezxml::off, ezxml::ordered, ezxml::parent, ezxml::sibling, and ezxml::txt.

Referenced by ezxml_open_tag().

11.3.3.2 char* ezxml_ampencode (const char * *s*, size_t *len*, char ** *dst*, size_t * *dlen*, size_t * *max*, short *a*)

Definition at line 667 of file ezxml.c.

References EZXML_BUFSIZE, and max.

Referenced by ezxml_toxml_r().

11.3.3.3 const char* ezxml_attr (ezxml_t *xml*, const char * *attr*)

Returns the value of the requested tag attribute, or NULL if not found.

Definition at line 76 of file ezxml.c.

References ezxml_root::attr, ezxml::attr, ezxml_root_t, ezxml_t, ezxml::name, ezxml::parent, and ezxml_-root::xml.

Referenced by ezxml_toxml_r(), ConfigSim::gXMLAngle(), ConfigSim::gXMLAstroDistance(), ConfigSim::gXMLAstroMass(), ConfigSim::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 strduped

Definition at line 45 of file ezxml.h.

Referenced by ezxml_char_content(), ezxml_free(), ezxml_free_attr(), ezxml_parse_str(), ezxml_set_attr(), and ezxml_set_txt().

11.4.2 Typedef Documentation

11.4.2.1 typedef struct ezxml* ezxml_t

XML manipulation structure.

Definition at line 52 of file ezxml.h.

Referenced by ezxml_add_child(), ezxml_attr(), ezxml_char_content(), ezxml_child(), ezxml_close_tag(), ezxml_err(), ezxml_error(), ezxml_free(), ezxml_get(), ezxml_idx(), ezxml_new(), ezxml_open_tag(), ezxml_parse_fd(), ezxml_parse_file(), ezxml_parse_fp(), ezxml_parse_str(), ezxml_pi(), ezxml_remove(), ezxml_set_attr(), ezxml_set_flag(), ezxml_set_txt(), ezxml_toxml(), ezxml_toxml_r(), ezxml_vget(), ConfigSim::gXMLAngle(), ConfigSim::gXMLAstroDistance(), ConfigSim::gXMLAstroMass(), ConfigSim::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* $t = 0, \ldots, tMemTDI + tTDIShift$ *with tStepMes timsetep.*
    Signals are stored.
    LISA::MakeOneStepOfTime method is called.
    Delays are recorded.
    Positions are recorded.
  - *Data processing second step : when there are enough data, TDI is computed and results are stored in file, while time* $t \leq tmax$ *with tStepMes timsetep.*
    TDI is computed using TDI_InterData::ComputeEta method.
    Delays are recorded.
    Positions are recorded.
    .

11.15 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 f frequency and Cell (type QuadCell) inputs.

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

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

- double HmQuadCell (QuadCell Cell)

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

- double KmQuadCell (QuadCell Cell)

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

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

  Matches nearest poles for a chain of elliptic cells.

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

Orders cells according to the the max of inf norm.

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

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

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

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

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

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

11.16 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 > I (0, 1)

  Pure imaginary=(0,1).

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

  Poles, zeros and elliptic cells coefficients computation.

- double ak (double y)

  Integral filter parameter computation.

- double cak (double y)

  Developped filter parameter computation.

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

  Recursive or direct coefficients computation.

- double FilterQuadCell (double xn, QuadCell *Cell)

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

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

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

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

  Elliptic cell Z transform.

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

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

- double AbsRespFunctQuadCell (double f, QuadCell Cell)

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

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

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

- double HmQuadCell (QuadCell Cell)

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

- double KmQuadCell (QuadCell Cell)

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

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

  Matches nearest poles for a chain of elliptic cells.

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

  Orders cells according to the the max of inf norm.

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

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

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

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

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

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

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

11.49.1 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 $t = 0, \ldots, tMemTDI + tTDIShift$ with tStepMes timsetep.

  Signals are stored.

  LISA::MakeOneStepOfTime method is called.

  Delays are recorded.

  Positions are recorded.

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

  TDI is computed using TDI_InterData::ComputeEta method.

  Delays are recorded.

  Positions are recorded.

  .

Definition at line 146 of file LISACODE-LISACode.cpp.

References Memory::AddSerieData(), TDI_InterData::ComputeEta(), TDI_InterData::gData(), 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::getNbMaxDelays(), ConfigSim::getNoNoise(), TDITools::getRapidOption(), getsd(), ConfigSim::gettDeltaTDIDelay(), ConfigSim::getTDIDelayApprox(), ConfigSim::getTDIInterp(), ConfigSim::getTDIInterpUtilVal(), ConfigSim::gettDisplay(), ConfigSim::gettMax(), ConfigSim::gettStepMes(), TDI_InterData::getUsable(), LISA::gPosSC(), LCVersion, LISA::MakeOneStepOfTime(), Memory::MakeTitles(), MAX, ConfigSim::NbGenTDI(), Vect::p, Memory::ReceiveData(), Memory::RecordAccData(), TDITools::RefreshDelay(), setall(), ConfigSim::tMaxDelay(), ConfigSim::tMemNecInterpTDI(), and ConfigSim::tMinDelay().

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 1.4.2"
  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 1.4.2"

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>
```

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 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.

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 "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>
```

Variables

- const double c_SI = 299792458
  Light's speed in $m \cdot s^{-1}$.
- const double G_SI = 6.67259e-11
  Gravitational constant in $m^3 \cdot Kg^{-1} \cdot s^{-2}$.
- const double k_SI = 1.381e-23
  Boltzmann's constant in $joule \cdot kelvin^{-1}$.
- const double h_SI = 6.62620e-34
  Planck's constant in $joule \cdot second$.
- const double hb_SI = h_SI/(2∗M_PI)
  Planck's constant divided by $2 \cdot \pi$ *in* $joule \cdot second$.
- const double S_SI = 5.671e-8
  Stefan's constant in $Watt \cdot meter^{-2} \cdot kelvin^{-1}$.
- const double mu0_SI = 4.e-7∗M_PI
  Permeability of free space or magnetic constant μ_0 *in* $Henry \cdot meter^{-1}$.
- const double eps0_SI = 1.0/(4.e-7∗M_PI∗c_SI∗c_SI)
  Permittivity of free space or permittivity ϵ_0 *in* $Farad \cdot meter^{-1}$.
- const double Na_SI = 6.02252e-27
  Avogadro's number in mol^{-1}.
- const double H0_cgs = 0.7∗3.24e-18
  Hubble's constant in CGS.
- const double CE_RG = 0.5772156649015328606065120900824024310421593359399 4
  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 = 4.925492321898863396896438 62e-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 Schwarzhild 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 104 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 107 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 28 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 = 0.5772156649015328606065120900824024310421593359399 4

Euler's constant.

Definition at line 60 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 91 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 $Farad \cdot meter^{-1}$.

Definition at line 51 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 31 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 151 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 57 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 37 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 40 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 34 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 126 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 145 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 148 of file LISACODE-PhysicConstants.h.

11.77.2.16 const double LS_SI = 3.83e26

Sun's energy flux in Watt.

Definition at line 85 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 113 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 110 of file LISACODE-PhysicConstants.h.

11.77.2.19 const double me_SI = 9.1091e-31

Electron rest mass in Kg.

Definition at line 67 of file LISACODE-PhysicConstants.h.

11.77.2.20 const double mn_SI = 1.6748e-27

Neutron rest mass in Kg.

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

11.77.2.21 const double mp_SI = 1.6726e-27

Proton rest mass in Kg.

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

11.77.2.22 const double MS_SI = 1.9889e30

Sun's mass in Kg.

Definition at line 76 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 $Henry \cdot meter^{-1}$.

Definition at line 47 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 54 of file LISACODE-PhysicConstants.h.

11.77.2.25 const double omegaYr = 2.∗M_PI/Yr_SI

Angular velocity of Earth.

Definition at line 129 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 116 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 122 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 119 of file LISACODE-PhysicConstants.h.

11.77.2.29 const double PRECISION = 100.0*DBL_EPSILON

Acceptable precision error on doubles.

Definition at line 137 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 82 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 97 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 $Watt \cdot meter^{-2} \cdot kelvin^{-1}$.

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

11.77.2.33 const double TSUN = 4.925492321898863396896438 62e-6

Sun's mass in Seconds.

Definition at line 79 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 88 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::getNbMaxDelays(), ConfigSim::getNoNoise(), TDITools::getRapidOption(), ConfigSim::getTDIDelayApprox(), ConfigSim::getTDIInterp(), ConfigSim::getTDIInterpUtilVal(), ConfigSim::gettDisplay(), ConfigSim::gettMax(), ConfigSim::gettStepMes(), LCVersion, MAX, ConfigSim::NbGenTDI(), Memory::RecordAccData(), TDITools::RefreshDelay(), ConfigSim::tMaxDelay(), ConfigSim::tMemNecInterpTDI(), and ConfigSim::tMinDelay().

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 ftnstop (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 ignnbn (long n, float p)
- long ignpoi (float mu)
- long ignuin (long low, long high)
- long lennob (char *str)
- long mltmod (long a, long s, long m)
- void phrtsd (char *phrase, long *seed1, long *seed2)
- float ranf (void)
- void setgmn (float *meanv, float *covm, long p, float *parm)
- float sexpo (void)
- float sgamma (float a)
- float snorm (void)
- float fsign (float num, float sign)

11.93.1 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 ftnstop(), 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(), sexpo(), 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(), sexpo(), 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 ftnstop(), 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 ftnstop(), 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(), sexpo(), 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(), sexpo(), 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 sexpo (void)

Definition at line 1658 of file randlib.c.

References ranf(), and sexpo().

Referenced by genexp(), ignpoi(), sexpo(), and sgamma().

11.94.1.30 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().

11.94.1.31 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 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, 67
~Couple
Couple, 94
~Filter
Filter, 103
~GW
GW, 167
~GWBinary
GWBinary, 176
~GWCusp
GWCusp, 187
~GWFastSpinBBH
GWFastSpinBBH, 201
~GWFile
GWFile, 226
~GWMono
GWMono, 237
~GWNew
GWNew, 247
~GWNewton2
GWNewton2, 262
~GWPeriGate
GWPeriGate, 284
~Geometry
Geometry, 110
~GeometryAnalytic
GeometryAnalytic, 124
~GeometryFile
GeometryFile, 140
~GeometryMLDC
GeometryMLDC, 155
~LISA
LISA, 304
~Mat
Mat, 310
~Memory
Memory, 315
~MemoryReadDisk
MemoryReadDisk, 321
~MemoryWriteDisk
MemoryWriteDisk, 330
~Noise
Noise, 339
~PhoDetPhaMet
PhoDetPhaMet, 420
~Serie
Serie, 434
~SerieC
SerieC, 442
~TDI
TDI, 450
~TDITools
TDITools, 464
~TDI_InterData
TDI_InterData, 459
~TrFctGW
TrFctGW, 467
~USOClock
USOClock, 472
~Vect
Vect, 476

a0
QuadCell, 429
a1
GWNewton2, 271
QuadCell, 429
a11
GWNewton2, 271
a1x
GWNewton2, 271
A2
GWFastSpinBBH, 209
a2
GWNewton2, 271
a22
GWNewton2, 272
A3
GWFastSpinBBH, 209
a3
GWNewton2, 272
A4
GWFastSpinBBH, 209
a4
GWNewton2, 272

A5
GWFastSpinBBH, 209
a5
GWNewton2, 272
A6
GWFastSpinBBH, 209
a6
GWNewton2, 272
a7
GWNewton2, 272
ABS
randlib.c, 605
AbsRespFunctQuadCell
ellipFilter, 22
AbsRespFunctQuadCellChain
ellipFilter, 23
Ac
GWBinary, 181
addData
Serie, 434
addDataC
SerieC, 443
addNoise
Noise, 339
NoiseFile, 348
NoiseFilter, 361
NoiseFShape, 371
NoiseOof, 383
NoiseTwoFilter, 396
NoiseWhite, 407
AddSerieData
Memory, 315
MemoryReadDisk, 322
MemoryWriteDisk, 331
advnst
com.c, 480
randlib.h, 610
ak
ellipFilter, 23
alog
ellipFilter, 22
alog10
ellipFilter, 22
alpha
Filter, 105
Geometry, 115
GeometryAnalytic, 131
GeometryFile, 145
GeometryMLDC, 161
NoiseOof, 388
AlreadyRecDat
Memory, 318
MemoryReadDisk, 325
MemoryWriteDisk, 334
Amp
GWFastSpinBBH, 209
Amplhc
GWCusp, 191
GWMono, 241
GWNew, 251
GWPeriGate, 289
Amplhp
GWCusp, 191
GWMono, 241
GWNew, 251
GWPeriGate, 289
AmplPNorder
GWFastSpinBBH, 209
AmpShared
GWFastSpinBBH, 201
Angles handling, 34
AnglPol
GW, 171
GWBinary, 181
GWCusp, 192
GWFastSpinBBH, 209
GWFile, 231
GWMono, 242
GWNew, 252
GWNewton2, 272
GWPeriGate, 289
GWSto, 297
Ap
GWBinary, 182
App
Filter, 104
Armlength
ConfigSim, 83
ArmVelocity
Geometry, 110
GeometryAnalytic, 124
GeometryFile, 140
GeometryMLDC, 155
arot
Geometry, 115
GeometryAnalytic, 131
GeometryFile, 145
GeometryMLDC, 161
attr
ezxml, 96
ezxml_root, 99
au_m
LISACODE-PhysicConstants.h, 583
au_s
LISACODE-PhysicConstants.h, 583
Author
ConfigSim, 83
Ax

GWFastSpinBBH, 210
AzimuthalAngleOfSpin1
GWFastSpinBBH, 210
AzimuthalAngleOfSpin2
GWFastSpinBBH, 210

b1
GWNewton2, 272
QuadCell, 429
b11
GWNewton2, 273
b1x
GWNewton2, 273
b2
GWNewton2, 273
QuadCell, 430
B21
GWFastSpinBBH, 210
b3
GWNewton2, 273
B31
GWFastSpinBBH, 210
B32
GWFastSpinBBH, 210
b4
GWNewton2, 273
B41
GWFastSpinBBH, 210
B42
GWFastSpinBBH, 210
B43
GWFastSpinBBH, 211
B51
GWFastSpinBBH, 211
B52
GWFastSpinBBH, 211
B53
GWFastSpinBBH, 211
B54
GWFastSpinBBH, 211
B61
GWFastSpinBBH, 211
B62
GWFastSpinBBH, 211
B63
GWFastSpinBBH, 211
B64
GWFastSpinBBH, 211
B65
GWFastSpinBBH, 212
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, 171
GWBinary, 182
GWCusp, 192
GWFastSpinBBH, 212
GWFile, 231
GWMono, 242
GWNew, 252
GWNewton2, 273
GWPeriGate, 289
GWSto, 297
beta
Filter, 105
betavec
GWFastSpinBBH, 212
betay2
GWFastSpinBBH, 212
BigEndian
ConfigSim, 84
BinHeader
MemoryWriteDisk, 334

C1
GWFastSpinBBH, 212
c1
GWNewton2, 273
c1x
GWNewton2, 273
C2
GWFastSpinBBH, 212
c2
GWNewton2, 274
c2psi

GWFastSpinBBH, 212
c2x
GWNewton2, 274
C3
GWFastSpinBBH, 213
c3
GWNewton2, 274
C4
GWFastSpinBBH, 213
C5
GWFastSpinBBH, 213
C6
GWFastSpinBBH, 213
c_SI
LISACODE-PhysicConstants.h, 583
cak
ellipFilter, 23
calcderivvals
GWFastSpinBBH, 201
CalcEllipticFilter
ellipFilter, 23
CalcEllipticFilter4LISACode
ellipFilter, 24
calcLcrossN
GWFastSpinBBH, 201
calcLdotN
GWFastSpinBBH, 202
calcLdotS1
GWFastSpinBBH, 202
calcLdotS2
GWFastSpinBBH, 202
CalcScalingFact
ellipFilter, 25
calcSdotS
GWFastSpinBBH, 202
CalculDirProp
GW, 168
GWBinary, 176
GWCusp, 187
GWFastSpinBBH, 202
GWFile, 226
GWMono, 237
GWNew, 247
GWNewton2, 262
GWPeriGate, 285
GWSto, 294
CE_RG
LISACODE-PhysicConstants.h, 583
chi1
GWFastSpinBBH, 213
chi2
GWFastSpinBBH, 213
child
ezxml, 96
ci
GWFastSpinBBH, 213
GWNewton2, 274
CloseFile
MemoryWriteDisk, 331
cmass
GWNewton2, 274
cmu
Geometry, 115
GeometryAnalytic, 131
GeometryFile, 146
GeometryMLDC, 161
com.c, 479
advnst, 480
getsd, 480
ignlgi, 480
initgn, 481
inrgcm, 481
numg, 480
setall, 481
setant, 481
setsd, 481
Xa1, 481
Xa1vw, 481
Xa1w, 482
Xa2, 482
Xa2vw, 482
Xa2w, 482
Xcg1, 482
Xcg2, 482
Xig1, 482
Xig2, 482
Xlg1, 482
Xlg2, 483
Xm1, 483
Xm2, 483
Xqanti, 483
commun
GWNewton2, 262
Compute
TDI, 451
ComputeEta
TDI_InterData, 459
ComputeNoEta
TDI, 451
ConfigFileName
ConfigSim, 84
ConfigSim, 60
ConfigSim, 67
ConfigSim
~ConfigSim, 67
Armlength, 83
Author, 83
BigEndian, 84

ConfigFileName, 84
ConfigSim, 67
CreateXmlOutputFile, 67
DefaultConfig, 67
Endian, 84
FichXML, 84
FileEncodingDelays, 84
FileEncodingPos, 84
FileEncodingSC1, 84
FileEncodingSC2, 84
FileEncodingSC3, 85
FileEncodingTDI, 85
FileNameDelays, 85
FileNamePositions, 85
FileNameSigSC1, 85
FileNameSigSC2, 85
FileNameSigSC3, 85
FileNameTDI, 86
FindTDIName, 69
GenerationDate, 86
GenerationType, 86
getArmlength, 69
getAuthor, 69
getFileEncodingDelays, 69
getFileEncodingPositions, 69
getFileEncodingSig, 70
getFileEncodingTDI, 70
getFileNameDelays, 70
getFileNamePositions, 70
getFileNameSig, 70
getFileNameSigSC1, 71
getFileNameSigSC2, 71
getFileNameSigSC3, 71
getFileNameTDI, 71
getGenerationDate, 71
getGenerationType, 71
getGenTDIPacks, 71
getGenTDIPacksFact, 72
getGeometry, 72
getGlobalRandomSeed, 72
getGW, 72
getGWB, 72
getGWs, 73
getLaserPower, 73
getNameGenTDI, 73
getNbMaxDelays, 73
getNoises, 73
getNoNoise, 73
getOrbInitRot, 74
getOrbMove, 74
getOrbOrder, 74
getOrbStartTime, 74
getPhaMetFilter, 74
getPhaMetFilterParam, 74
getSC1ParamName, 74
getSC2ParamName, 75
getSC3ParamName, 75
getSimulator, 75
getSystemEncoding, 75
gettDeltaTDIDelay, 75
getTDIDelayApprox, 75
getTDIInterp, 75
getTDIInterpUtilVal, 76
getTDIParamName, 76
getTDIParamNameType, 76
gettDisplay, 76
getTimeOffset, 76
gettMax, 76
gettMemNoiseFirst, 77
gettMemNoiseLast, 77
gettMemSig, 77
gettStepMes, 77
gettStepPhy, 77
getUSOs, 77
getXmlOutputFile, 77
GlobalRandomSeed, 86
GWB, 86
GWs, 86
gXMLAngle, 78
gXMLAstroDistance, 78
gXMLAstroMass, 78
gXMLdouble, 78
gXMLFrequency, 78
gXMLint, 79
gXMLstring, 79
gXMLTime, 79
gXMLTimeSeries, 79
gXMLUnit, 80
gXMLWord, 80
LaserPower, 86
majuscule, 80
NbGenTDI, 80
NbMaxDelays, 87
NoisePlace, 80
Noises, 87
NoisesCreation, 81
NoisesData, 87
OrbInitRot, 87
OrbMove, 87
OrbOrder, 87
OrbStartTime, 88
PhaMetFilterON, 88
PhaMetFilterParam, 88
ReadASCIIFile, 81
ReadFile, 82
ReadXMLFile, 82
SC1ParamName, 88
SC2ParamName, 89

SC3ParamName, 89
Simulator, 89
SystemEncoding, 89
SystemEncoding_int, 89
tDeltaTDIDelay, 89
TDIDelayApprox, 89
TDIInterp, 89
TDIInterpUtilVal, 90
TDIParamName, 90
TDIParamNameType, 90
TDIsName, 90
TDIsPacks, 90
TDIsPacksFact, 90
tDisplay, 90
testbyteorder, 82
TimeOffset, 91
tMax, 91
tMaxDelay, 82
tMemNecInterpTDI, 83
tMemNoiseFirst, 91
tMemNoiseLast, 91
tMemSig, 91
tMinDelay, 83
tStepMes, 91
tStepPhy, 91
uppercase, 83
USOs, 92
XmlOutputFile, 92
costheta
GWFastSpinBBH, 213
Couple, 20, 93
∼Couple, 94
Couple, 94
operator ∗, 94
operator+, 94
operator-, 95
operator/, 95
x, 95
y, 95
cphilvec
GWFastSpinBBH, 213
cphily2
GWFastSpinBBH, 214
CreateXmlOutputFile
ConfigSim, 67
crot
GeometryAnalytic, 131
GeometryMLDC, 161
CUB
serie, 35
cur
ezxml_root, 99

D1
GWFastSpinBBH, 214
d1
GWNewton2, 274
d1x
GWNewton2, 274
D2
GWFastSpinBBH, 214
d2
GWNewton2, 274
d2x
GWNewton2, 275
D3
GWFastSpinBBH, 214
d3x
GWNewton2, 275
D4
GWFastSpinBBH, 214
d4x
GWNewton2, 275
D5
GWFastSpinBBH, 214
d5x
GWNewton2, 275
D6
GWFastSpinBBH, 214
DefaultConfig
ConfigSim, 67
deg2rad
MathUtils, 312
DelayMem
TDITools, 465
DelayStore
Geometry, 115
GeometryAnalytic, 131
GeometryFile, 146
GeometryMLDC, 161
delLastData
Serie, 434
delLastDataC
SerieC, 443
deltam
GWNewton2, 275
deltanu
Background, 52
BackgroundGalactic, 57
TrFctGW, 467
DeltaPhase2L
GWFastSpinBBH, 214
DerivLinearCoef
USOClock, 473
detect
IM, 18
LA, 18
NOISEORIG, 18

OB, 18
OP, 18
PDPMINTERF, 18
S, 18
TAU, 18
Detector (directory Dectecteur), 18
DirProp
GW, 171
GWBinary, 182
GWCusp, 192
GWFastSpinBBH, 214
GWFile, 231
GWMono, 242
GWNew, 252
GWNewton2, 275
GWPeriGate, 289
GWSto, 297
display
Mat, 310
Vect, 476
DisplayStoredData
PhoDetPhaMet, 420
DispTempVal
GW, 168
GWBinary, 176
GWCusp, 187
GWFastSpinBBH, 202
GWFile, 226
GWMono, 237
GWNew, 247
GWNewton2, 263
GWPeriGate, 285
GWSto, 294
DL
GWFastSpinBBH, 215
dm
GWFastSpinBBH, 215
Doxygen.bibliography, 484
dx
Serie, 439
SerieC, 445
Dy_SI
LISACODE-PhysicConstants.h, 583

e
ezxml_root, 99
Geometry, 116
GeometryAnalytic, 131
GeometryFile, 146
GeometryMLDC, 161
e1
GWNewton2, 275
e1x
GWNewton2, 275
e2
GWNewton2, 276
e2x
GWNewton2, 276
e3
GWNewton2, 276
e3x
GWNewton2, 276
e4
GWNewton2, 276
e4x
GWNewton2, 276
e5
GWNewton2, 276
e5x
GWNewton2, 276
e6x
GWNewton2, 276
e7x
GWNewton2, 277
e8x
GWNewton2, 277
e_mldc
GeometryMLDC, 162
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
I, 28
KmQuadCell, 28
OrderCellMaxNorm, 29
PoleMatching, 29
sn, 29
TransfZQuadCell, 30
TransfZQuadCellChain, 30
Elliptic Filter, 21
Endian
ConfigSim, 84
ent
ezxml_root, 99
eps0_SI
LISACODE-PhysicConstants.h, 584

err
    ezxml_root, 99
Eta
    TDI, 453
    TDI_InterData, 461
eta
    GWFastSpinBBH, 215
etain
    GWFastSpinBBH, 215
exanom
    GeometryAnalytic, 124
    GeometryFile, 140
    GeometryMLDC, 155
expmax
    randlib.c, 605
ezxml, 96
    attr, 96
    child, 96
    flags, 97
    name, 97
    next, 97
    off, 97
    ordered, 97
    parent, 97
    sibling, 98
    txt, 98
ezxml.c, 485
    ezxml_add_child, 487
    ezxml_ampencode, 487
    ezxml_attr, 488
    ezxml_char_content, 488
    ezxml_child, 488
    ezxml_close_tag, 488
    ezxml_decode, 488
    ezxml_ent_ok, 488
    ezxml_err, 489
    EZXML_ERRL, 487
    ezxml_error, 489
    ezxml_free, 489
    ezxml_free_attr, 489
    ezxml_get, 489
    ezxml_idx, 489
    ezxml_internal_dtd, 490
    ezxml_new, 490
    EZXML_NIL, 493
    ezxml_open_tag, 490
    ezxml_parse_fd, 490
    ezxml_parse_file, 490
    ezxml_parse_fp, 491
    ezxml_parse_str, 491
    ezxml_pi, 491
    ezxml_proc_inst, 491
    ezxml_remove, 491
    ezxml_root_t, 487
    ezxml_set_attr, 491
    ezxml_set_flag, 492
    ezxml_set_txt, 492
    ezxml_str2utf8, 492
    ezxml_toxml, 492
    ezxml_toxml_r, 492
    ezxml_vget, 492
    EZXML_WS, 487
ezxml.h, 494
    ezxml_add_child, 498
    ezxml_add_child_d, 496
    ezxml_attr, 498
    EZXML_BUFSIZE, 496
    ezxml_child, 498
    EZXML_DUP, 496
    ezxml_error, 498
    ezxml_free, 499
    ezxml_get, 499
    ezxml_idx, 499
    ezxml_name, 496
    EZXML_NAMEM, 496
    ezxml_new, 499
    ezxml_new_d, 497
    ezxml_next, 497
    ezxml_parse_fd, 499
    ezxml_parse_file, 500
    ezxml_parse_fp, 500
    ezxml_parse_str, 500
    ezxml_pi, 500
    ezxml_remove, 500
    ezxml_set_attr, 501
    ezxml_set_attr_d, 497
    ezxml_set_flag, 501
    ezxml_set_txt, 501
    ezxml_set_txt_d, 497
    ezxml_t, 498
    ezxml_toxml, 501
    ezxml_txt, 497
    EZXML_TXTM, 497
ezxml_add_child
    ezxml.c, 487
    ezxml.h, 498
ezxml_add_child_d
    ezxml.h, 496
ezxml_ampencode
    ezxml.c, 487
ezxml_attr
    ezxml.c, 488
    ezxml.h, 498
EZXML_BUFSIZE
    ezxml.h, 496
ezxml_char_content
    ezxml.c, 488
ezxml_child

ezxml.c, 488
ezxml.h, 498
ezxml_close_tag
ezxml.c, 488
ezxml_decode
ezxml.c, 488
EZXML_DUP
ezxml.h, 496
ezxml_ent_ok
ezxml.c, 488
ezxml_err
ezxml.c, 489
EZXML_ERRL
ezxml.c, 487
ezxml_error
ezxml.c, 489
ezxml.h, 498
ezxml_free
ezxml.c, 489
ezxml.h, 499
ezxml_free_attr
ezxml.c, 489
ezxml_get
ezxml.c, 489
ezxml.h, 499
ezxml_idx
ezxml.c, 489
ezxml.h, 499
ezxml_internal_dtd
ezxml.c, 490
ezxml_name
ezxml.h, 496
EZXML_NAMEM
ezxml.h, 496
ezxml_new
ezxml.c, 490
ezxml.h, 499
ezxml_new_d
ezxml.h, 497
ezxml_next
ezxml.h, 497
EZXML_NIL
ezxml.c, 493
ezxml_open_tag
ezxml.c, 490
ezxml_parse_fd
ezxml.c, 490
ezxml.h, 499
ezxml_parse_file
ezxml.c, 490
ezxml.h, 500
ezxml_parse_fp
ezxml.c, 491
ezxml.h, 500
ezxml_parse_str
ezxml.c, 491
ezxml.h, 500
ezxml_pi
ezxml.c, 491
ezxml.h, 500
ezxml_proc_inst
ezxml.c, 491
ezxml_remove
ezxml.c, 491
ezxml.h, 500
ezxml_root, 99
attr, 99
cur, 99
e, 99
ent, 99
err, 99
len, 99
m, 100
pi, 100
s, 100
standalone, 100
u, 100
xml, 100
ezxml_root_t
ezxml.c, 487
ezxml_set_attr
ezxml.c, 491
ezxml.h, 501
ezxml_set_attr_d
ezxml.h, 497
ezxml_set_flag
ezxml.c, 492
ezxml.h, 501
ezxml_set_txt
ezxml.c, 492
ezxml.h, 501
ezxml_set_txt_d
ezxml.h, 497
ezxml_str2utf8
ezxml.c, 492
ezxml_t
ezxml.h, 498
ezxml_toxml
ezxml.c, 492
ezxml.h, 501
ezxml_toxml_r
ezxml.c, 492
ezxml_txt
ezxml.h, 497
EZXML_TXTM
ezxml.h, 497
ezxml_vget
ezxml.c, 492

EZXML_WS
ezxml.c, 487

f1
GWNewton2, 277
f10
GWNewton2, 277
f3
GWNewton2, 277
f5
GWNewton2, 277
f6
GWNewton2, 277
f7
GWNewton2, 277
f8
GWNewton2, 277
f9
GWNewton2, 278
Fac_Hc
GWSto, 297
Fac_Hp
GWSto, 297
Fac_norm
GWSto, 297
Fact
TDI, 453
FactF0
NoiseFShape, 375
FactFm
NoiseFShape, 376
FactFp
NoiseFShape, 376
FactMult
NoiseFile, 353
fe
GWNewton2, 263
FEncoding
MemoryReadDisk, 325
MemoryWriteDisk, 334
FichMem
MemoryReadDisk, 325
MemoryWriteDisk, 334
FichXML
ConfigSim, 84
FileEncoding
GWFile, 231
FileEncodingDelays
ConfigSim, 84
FileEncodingPos
ConfigSim, 84
FileEncodingSC1
ConfigSim, 84
FileEncodingSC2
ConfigSim, 84
FileEncodingSC3
ConfigSim, 85
FileEncodingTDI
ConfigSim, 85
FileName
GWFile, 231
NoiseFile, 353
FileNameDelays
ConfigSim, 85
FileNamePositions
ConfigSim, 85
FileNameSigSC1
ConfigSim, 85
FileNameSigSC2
ConfigSim, 85
FileNameSigSC3
ConfigSim, 85
FileNameTDI
ConfigSim, 86
Filter, 31, 101
~Filter, 103
alpha, 105
App, 104
beta, 105
Filter, 102, 103
getAlpha, 104
getBeta, 104
getDepth, 104
getNbDataStab, 105
init, 105
NbDataStab, 105
TmpData, 106
FilterFm
NoiseFShape, 376
FilterFp
NoiseFShape, 376
FilterON
PhoDetPhaMet, 425
FilterParam
PhoDetPhaMet, 425
FilterPhyData
PhoDetPhaMet, 425
FilterQuadCell
ellipFilter, 28
FilterQuadCellChain
ellipFilter, 28
FindTDIName
ConfigSim, 69
Fknee
GWSto, 298
flags
ezxml, 97
fmax

NoiseOof, 388
FmData
NoiseFShape, 376
Fmin
GWSto, 298
fmin
NoiseOof, 388
forb
GWBinary, 182
FpData
NoiseFShape, 376
Freq
GWCusp, 192
GWFastSpinBBH, 203
GWMono, 242
GWNew, 252
GWPeriGate, 289
fsign
randlib.c, 605
ftnstop
randlib.c, 605

g1
GWNewton2, 278
G_SI
LISACODE-PhysicConstants.h, 584
gamma0
GWFastSpinBBH, 215
gamma_u
LISACODE-PhysicConstants.h, 584
gArmLength
LISA, 304
gData
Memory, 315
MemoryReadDisk, 322
MemoryWriteDisk, 331
Serie, 434
TDI_InterData, 459, 460
gDelayT
LISA, 305
genbet
randlib.c, 605
randlib.h, 610
genchi
randlib.c, 606
randlib.h, 610
GenerationDate
ConfigSim, 86
GenerationType
ConfigSim, 86
generNoise
Noise, 339
NoiseFile, 348
NoiseFilter, 361
NoiseFShape, 371
NoiseOof, 383
NoiseTwoFilter, 396
NoiseWhite, 407
genexp
randlib.c, 606
randlib.h, 611
genf
randlib.c, 606
randlib.h, 611
gengam
randlib.c, 606
randlib.h, 611
genmn
randlib.c, 606
randlib.h, 611
genmul
randlib.c, 606
randlib.h, 611
gennch
randlib.c, 606
randlib.h, 611
gennf
randlib.c, 606
randlib.h, 611
gennor
randlib.c, 607
randlib.h, 612
genprm
randlib.c, 607
randlib.h, 612
genunf
randlib.c, 607
randlib.h, 612
Geometry, 107
~Geometry, 110
alpha, 115
ArmVelocity, 110
arot, 115
cmu, 115
DelayStore, 115
e, 116
Geometry, 109
getL0, 110
gett0, 111
gettRangeStoreDelay, 111
gettRangeStorePos, 111
gposition, 111
gtdelay, 111
initGlobal, 112
L0m, 116
move, 116
nu, 116
order_default, 116

position, 112
rot0, 117
SCposStore, 117
sett0, 113
settRangeStoreDelay, 113
settRangeStorePos, 113
smu, 117
sqrtee, 117
t0, 117
tdelay, 113
tdelayOrderContribution, 114
tmu, 117
tRangeStoreDelay, 117
tRangeStorePos, 117
tStoreDelay, 118
tStorePos, 118
VectNormal, 114
velocity, 115
Geometry (directory Orbitographie), 44
GeometryAnalytic, 119
GeometryAnalytic, 122, 123
GeometryAnalytic
~GeometryAnalytic, 124
alpha, 131
ArmVelocity, 124
arot, 131
cmu, 131
crot, 131
DelayStore, 131
e, 131
exanom, 124
GeometryAnalytic, 122, 123
getL0, 125
gett0, 125
gettRangeStoreDelay, 125
gettRangeStorePos, 125
gposition, 125
gtdelay, 126
init, 126
initGlobal, 127
L0m, 132
move, 132
nu, 132
order_default, 132
position, 128
rot, 132
rot0, 132
SCposStore, 132
sett0, 128
settRangeStoreDelay, 128
settRangeStorePos, 128
smu, 133
sqrtee, 133
srot, 133
t0, 133
tdelay, 128
tdelayOrderContribution, 129
tmu, 133
tRangeStoreDelay, 133
tRangeStorePos, 133
tStoreDelay, 134
tStorePos, 134
VectNormal, 130
velocity, 130
GeometryFile, 135
GeometryFile, 138, 139
GeometryFile
~GeometryFile, 140
alpha, 145
ArmVelocity, 140
arot, 145
cmu, 146
DelayStore, 146
e, 146
exanom, 140
GeometryFile, 138, 139
getL0, 141
gett0, 141
gettRangeStoreDelay, 141
gettRangeStorePos, 141
gposition, 141
gtdelay, 141
init, 142
initGlobal, 142
InputFileName, 146
L0m, 146
move, 146
NbData, 147
nu, 147
order_default, 147
position, 143
rot0, 147
SCposStore, 147
sett0, 143
settRangeStoreDelay, 143
settRangeStorePos, 143
smu, 147
sqrtee, 147
t0, 147
tdelay, 143
tdelayOrderContribution, 144
Time_sec, 148
tmu, 148
tRangeStoreDelay, 148
tRangeStorePos, 148
tStoreDelay, 148
tStorePos, 148
VectNormal, 145

velocity, 145
XYZ, 148
GeometryMLDC, 150
GeometryMLDC, 153, 154
GeometryMLDC
~GeometryMLDC, 155
alpha, 161
ArmVelocity, 155
arot, 161
cmu, 161
crot, 161
DelayStore, 161
e, 161
e_mldc, 162
exanom, 155
GeometryMLDC, 153, 154
getL0, 155
gett0, 155
gettRangeStoreDelay, 156
gettRangeStorePos, 156
gposition, 156
gtdelay, 156
init, 157
initGlobal, 157
L0m, 162
move, 162
nu, 162
order_default, 162
position, 158
rot, 162
rot0, 162
SCposStore, 163
sett0, 158
settRangeStoreDelay, 158
settRangeStorePos, 158
smu, 163
sqrt_3, 163
sqrtee, 163
srot, 163
t0, 163
tdelay, 158
tdelayOrderContribution, 159
tmu, 163
tRangeStoreDelay, 163
tRangeStorePos, 164
tStoreDelay, 164
tStorePos, 164
VectNormal, 160
velocity, 160
getAlpha
Filter, 104
getAmplhc
GWCusp, 187
GWMono, 237
GWNew, 247
GWPeriGate, 285
getAmplhp
GWCusp, 187
GWMono, 237
GWNew, 247
GWPeriGate, 285
getAnglPol
GW, 168
GWBinary, 177
GWCusp, 188
GWFastSpinBBH, 203
GWFile, 226
GWMono, 237, 238
GWNew, 248
GWNewton2, 263
GWPeriGate, 285
GWSto, 294
getArmlength
ConfigSim, 69
getAuthor
ConfigSim, 69
getBeta
Filter, 104
GW, 168
GWBinary, 177
GWCusp, 188
GWFastSpinBBH, 203
GWFile, 227
GWMono, 238
GWNew, 248
GWNewton2, 263
GWPeriGate, 285
GWSto, 294
getBinValue
Serie, 435
getBinValueC
SerieC, 443
getCountInterDelay
TDI, 451
getCountInterEta
TDI, 451
getDelay
TDITools, 464
getDepth
Filter, 104
getDeriv
USOClock, 472
getDirProp
GW, 168
GWBinary, 177
GWCusp, 188
GWFastSpinBBH, 203
GWFile, 227

GWMono, 238
GWNew, 248
GWNewton2, 263
GWPeriGate, 285
GWSto, 294
getDistance
GWBinary, 177
GWFastSpinBBH, 203
GWNewton2, 263
getFileEncodingDelays
ConfigSim, 69
getFileEncodingPositions
ConfigSim, 69
getFileEncodingSig
ConfigSim, 70
getFileEncodingTDI
ConfigSim, 70
getFileName
GWFile, 227
NoiseFile, 349
getFileNameDelays
ConfigSim, 70
getFileNamePositions
ConfigSim, 70
getFileNameSig
ConfigSim, 70
getFileNameSigSC1
ConfigSim, 71
getFileNameSigSC2
ConfigSim, 71
getFileNameSigSC3
ConfigSim, 71
getFileNameTDI
ConfigSim, 71
getFilterAlpha
NoiseFilter, 361
NoiseOof, 384
NoiseTwoFilter, 396
getFilterBeta
NoiseFilter, 361
NoiseOof, 384
NoiseTwoFilter, 397
getForb
GWBinary, 177
getFreq
GWCusp, 188
GWMono, 238
GWNew, 248
GWPeriGate, 286
getGenerationDate
ConfigSim, 71
getGenerationType
ConfigSim, 71
getGenTDIPacks
ConfigSim, 71
getGenTDIPacksFact
ConfigSim, 72
getGeometry
ConfigSim, 72
getGlobalRandomSeed
ConfigSim, 72
getGW
ConfigSim, 72
getGWB
ConfigSim, 72
getGWs
ConfigSim, 73
getInc
GWBinary, 177
GWNewton2, 263
getIndirectDir
PhoDetPhaMet, 421
getiSC
PhoDetPhaMet, 421
getL0
Geometry, 110
GeometryAnalytic, 125
GeometryFile, 141
GeometryMLDC, 155
getLambda
GW, 168
GWBinary, 177
GWCusp, 188
GWFastSpinBBH, 203
GWFile, 227
GWMono, 238
GWNew, 248
GWNewton2, 264
GWPeriGate, 286
GWSto, 294
getLaserPower
ConfigSim, 73
getM1
GWBinary, 177
GWNewton2, 264
getM2
GWBinary, 178
GWNewton2, 264
getMass1
GWFastSpinBBH, 203
getMass2
GWFastSpinBBH, 203
getNameGenTDI
ConfigSim, 73
getNbBinAdd
Noise, 340
NoiseFile, 349
NoiseFilter, 362

NoiseFShape, 372
NoiseOof, 384
NoiseTwoFilter, 397
NoiseWhite, 408
getNbDataStab
Filter, 105
getNbDataStored
NoiseFile, 349
getNbMaxDelays
ConfigSim, 73
getNbSerie
Memory, 316
MemoryReadDisk, 322
MemoryWriteDisk, 331
getNbStored
GWFile, 227
getNbVal
Serie, 435
getNbValC
SerieC, 443
getNoise
Noise, 340
NoiseFile, 349, 350
NoiseFilter, 362
NoiseFShape, 372
NoiseOof, 384, 385
NoiseTwoFilter, 397
NoiseWhite, 408
USOClock, 472
getNoises
ConfigSim, 73
getNoNoise
ConfigSim, 73
PhoDetPhaMet, 421
getNParam
GW, 168
GWBinary, 178
GWCusp, 188
GWFastSpinBBH, 204
GWFile, 227
GWMono, 238
GWNew, 248
GWNewton2, 264
GWPeriGate, 286
GWSto, 294
getOffset
USOClock, 472
getOrbInitRot
ConfigSim, 74
getOrbMove
ConfigSim, 74
getOrbOrder
ConfigSim, 74
getOrbStartTime
ConfigSim, 74
getParam
GW, 169
GWBinary, 178
GWCusp, 188
GWFastSpinBBH, 204
GWFile, 227
GWMono, 238
GWNew, 248
GWNewton2, 264
GWPeriGate, 286
GWSto, 294
getPhaMetFilter
ConfigSim, 74
getPhaMetFilterParam
ConfigSim, 74
getPhCoal
GWNewton2, 264
getPhi0
GWBinary, 178
getPhi0hc
GWCusp, 189
GWMono, 238
GWNew, 249
getPhi0hp
GWCusp, 189
GWMono, 239
GWNew, 249
getPSD
NoiseWhite, 409
getRapidOption
TDITools, 464
getRef
Serie, 435
getRefC
SerieC, 443
getRefStep
Serie, 435
getRefStepC
SerieC, 444
getSC1ParamName
ConfigSim, 74
getSC2ParamName
ConfigSim, 75
getSC3ParamName
ConfigSim, 75
getsd
com.c, 480
randlib.h, 612
getSimulator
ConfigSim, 75
getSqPSD
NoiseWhite, 409
getSystemEncoding

ConfigSim, 75
gett0
Geometry, 111
GeometryAnalytic, 125
GeometryFile, 141
GeometryMLDC, 155
getTcoal
GWNewton2, 264
gettDelayCompute
TDI_InterData, 460
gettDeltaTDIDelay
ConfigSim, 75
getTDIDelayApprox
ConfigSim, 75
getTDIInterp
ConfigSim, 75
getTDIInterpUtilVal
ConfigSim, 76
getTDIParamName
ConfigSim, 76
getTDIParamNameType
ConfigSim, 76
gettDisplay
ConfigSim, 76
gettDurAdd
Noise, 341
NoiseFile, 350
NoiseFilter, 363
NoiseFShape, 373
NoiseOof, 385
NoiseTwoFilter, 398
NoiseWhite, 409
gettFirst
Noise, 341
NoiseFile, 350
NoiseFilter, 363
NoiseFShape, 373
NoiseOof, 385
NoiseTwoFilter, 398
NoiseWhite, 409
getTimeOffset
ConfigSim, 76
getTimeStore
TDI_InterData, 460
gettLast
Noise, 341
NoiseFile, 350
NoiseFilter, 363
NoiseFShape, 373
NoiseOof, 386
NoiseTwoFilter, 398
NoiseWhite, 410
gettMax
ConfigSim, 76
Memory, 316
MemoryReadDisk, 322
MemoryWriteDisk, 332
gettMemNoiseFirst
ConfigSim, 77
gettMemNoiseLast
ConfigSim, 77
gettMemSig
ConfigSim, 77
gettRangeStoreDelay
Geometry, 111
GeometryAnalytic, 125
GeometryFile, 141
GeometryMLDC, 156
gettRangeStorePos
Geometry, 111
GeometryAnalytic, 125
GeometryFile, 141
GeometryMLDC, 156
gettStab
PhoDetPhaMet, 421
gettStep
Noise, 341
NoiseFile, 351
NoiseFilter, 363
NoiseFShape, 373
NoiseOof, 386
NoiseTwoFilter, 398
NoiseWhite, 410
TDI_InterData, 460
gettStepMes
ConfigSim, 77
gettStepPhy
ConfigSim, 77
gettStepRecord
Memory, 316
MemoryReadDisk, 323
MemoryWriteDisk, 332
gettStoreData
Memory, 316
MemoryReadDisk, 323
MemoryWriteDisk, 332
getUsable
TDI_InterData, 461
getUSOs
ConfigSim, 77
getXmlOutputFile
ConfigSim, 77
gGap
USOClock, 472
gGWB
PhoDetPhaMet, 421
GlobalRandomSeed
ConfigSim, 86

gN
PhoDetPhaMet, 422
gposition
Geometry, 111
GeometryAnalytic, 125
GeometryFile, 141
GeometryMLDC, 156
gPosSC
LISA, 305
Gravitational waves (directory Ondes_Gravit), 37
gscgn
randlib.c, 607
randlib.h, 612
gsrgs
randlib.c, 607
gssst
randlib.c, 607
gtdelay
Geometry, 111
GeometryAnalytic, 126
GeometryFile, 141
GeometryMLDC, 156
gTime
USOClock, 473
GW, 165
~GW, 167
AnglPol, 171
Beta, 171
CalculDirProp, 168
DirProp, 171
DispTempVal, 168
getAnglPol, 168
getBeta, 168
getDirProp, 168
getLambda, 168
getNParam, 168
getParam, 169
GW, 167
hc, 169
hp, 169
init, 169
Lambda, 171
NParam, 171
setAnglPol, 169
setBeta, 170
setDirProp, 170
setLambda, 170
setParam, 170
gw
GWNewton2, 278
GWB
ConfigSim, 86
LISA, 307
PhoDetPhaMet, 425
GWBinary, 173
~GWBinary, 176
Ac, 181
AnglPol, 181
Ap, 182
Beta, 182
CalculDirProp, 176
DirProp, 182
DispTempVal, 176
forb, 182
getAnglPol, 177
getBeta, 177
getDirProp, 177
getDistance, 177
getForb, 177
getInc, 177
getLambda, 177
getM1, 177
getM2, 178
getNParam, 178
getParam, 178
getPhi0, 178
GWBinary, 175
hc, 178
hp, 178
inc, 182
init, 179
Lambda, 182
M1, 183
M2, 183
NParam, 183
phi0, 183
r, 183
setAnglPol, 179
setBeta, 179
setDirProp, 179
setDistance, 180
setForb, 180
setInc, 180
setLambda, 180
setM1, 180
setM2, 180
setParam, 181
setPhi0, 181
GWCusp, 184
~GWCusp, 187
Amplhc, 191
Amplhp, 191
AnglPol, 192
Beta, 192
CalculDirProp, 187
DirProp, 192
DispTempVal, 187
Freq, 192

getAmplhc, 187
getAmplhp, 187
getAnglPol, 188
getBeta, 188
getDirProp, 188
getFreq, 188
getLambda, 188
getNParam, 188
getParam, 188
getPhi0hc, 189
getPhi0hp, 189
GWCusp, 186
hc, 189
hp, 189
init, 189
Lambda, 192
NParam, 193
Phi0hc, 193
Phi0hp, 193
setAmplhc, 189
setAmplhp, 189
setAnglPol, 190
setBeta, 190
setDirProp, 190
setFreq, 190
setLambda, 190
setParam, 191
setPhi0hc, 191
setPhi0hp, 191
GWFastSpinBBH, 194
GWFastSpinBBH, 200
GWFastSpinBBH
~GWFastSpinBBH, 201
A2, 209
A3, 209
A4, 209
A5, 209
A6, 209
Amp, 209
AmplPNorder, 209
AmpShared, 201
AnglPol, 209
Ax, 210
AzimuthalAngleOfSpin1, 210
AzimuthalAngleOfSpin2, 210
B21, 210
B31, 210
B32, 210
B41, 210
B42, 210
B43, 211
B51, 211
B52, 211
B53, 211
B54, 211
B61, 211
B62, 211
B63, 211
B64, 211
B65, 212
Beta, 212
betavec, 212
betay2, 212
C1, 212
C2, 212
c2psi, 212
C3, 213
C4, 213
C5, 213
C6, 213
calcderivvals, 201
calcLcrossN, 201
calcLdotN, 202
calcLdotS1, 202
calcLdotS2, 202
calcSdotS, 202
CalculDirProp, 202
chi1, 213
chi2, 213
ci, 213
costheta, 213
cphilvec, 213
cphily2, 214
D1, 214
D2, 214
D3, 214
D4, 214
D5, 214
D6, 214
DeltaPhase2L, 214
DirProp, 214
DispTempVal, 202
DL, 215
dm, 215
eta, 215
etain, 215
Freq, 203
gamma0, 215
getAnglPol, 203
getBeta, 203
getDirProp, 203
getDistance, 203
getLambda, 203
getMass1, 203
getMass2, 203
getNParam, 204
getParam, 204
GWFastSpinBBH, 200

hc, 204
hp, 204
idxcur, 215
idxm, 215
init, 204
InitialAzimuthalAngleL, 216
InitialPolarAngleL, 216
initRK, 205
Lambda, 216
m1, 216
m2, 216
Mchirp, 216
Mtot, 217
mu, 217
mulvec, 217
muly2, 217
NParam, 217
p15, 217
p150, 217
p20, 217
p200, 218
phi, 218
Phi0, 218
Phi10, 218
phic, 218
PolarAngleOfSpin1, 218
PolarAngleOfSpin2, 218
psi, 218
RK_EPS, 219
RK_H, 219
RK_VECLENGTH, 219
rkckstep, 205
Rmin, 219
S1xdot, 205
S1ydot, 205
S1zdot, 205
s2psi, 219
S2xdot, 205
S2ydot, 206
S2zdot, 206
setAmplPNorder, 206
setAnglPol, 206
setBeta, 206
setDirProp, 207
setDistance, 207
setLambda, 207
setMass1, 207
setMass2, 207
setParam, 207
sf, 219
shc, 219
shp, 219
sigmavec, 220
sigmay2, 220
sintheta, 220
sphilvec, 220
sphily2, 220
spline, 208
sx, 220
TaperApplied, 220
TaperSteepness, 220
tc, 221
thomasvec, 221
thomasy2, 221
timeCur, 221
timevec, 221
tmax, 221
Tobs, 221
Toffset, 221
update, 208
xmax, 222
xold, 222
GWFile, 223
~GWFile, 226
AnglPol, 231
Beta, 231
CalculDirProp, 226
DirProp, 231
DispTempVal, 226
FileEncoding, 231
FileName, 231
getAnglPol, 226
getBeta, 227
getDirProp, 227
getFileName, 227
getLambda, 227
getNbStored, 227
getNParam, 227
getParam, 227
GWFile, 225, 226
hc, 227
hList, 231
hp, 228
init, 228
Interpol, 228
Lambda, 232
LastUsedBin, 232
Length, 232
NbDat, 232
NParam, 232
ReadASCIIFile, 229
ReadBinaryFile, 229
Records, 232
setAnglPol, 229
setBeta, 229
setDirProp, 229
setFileName, 230
setLambda, 230

setParam, 230
TimeList, 232
TimeOffset, 233
TimeStep, 233
GWMono, 234
~GWMono, 237
Amplhc, 241
Amplhp, 241
AnglPol, 242
Beta, 242
CalculDirProp, 237
DirProp, 242
DispTempVal, 237
Freq, 242
getAmplhc, 237
getAmplhp, 237
getAnglPol, 237, 238
getBeta, 238
getDirProp, 238
getFreq, 238
getLambda, 238
getNParam, 238
getParam, 238
getPhi0hc, 238
getPhi0hp, 239
GWMono, 236
hc, 239
hp, 239
init, 239
Lambda, 242
NParam, 243
Phi0hc, 243
Phi0hp, 243
setAmplhc, 239
setAmplhp, 239
setAnglPol, 240
setBeta, 240
setDirProp, 240
setFreq, 240
setLambda, 240
setParam, 241
setPhi0hc, 241
setPhi0hp, 241
GWNew, 244
~GWNew, 247
Amplhc, 251
Amplhp, 251
AnglPol, 252
Beta, 252
CalculDirProp, 247
DirProp, 252
DispTempVal, 247
Freq, 252
getAmplhc, 247
getAmplhp, 247
getAnglPol, 248
getBeta, 248
getDirProp, 248
getFreq, 248
getLambda, 248
getNParam, 248
getParam, 248
getPhi0hc, 249
getPhi0hp, 249
GWNew, 246
hc, 249
hp, 249
init, 249
Lambda, 252
NParam, 253
Phi0hc, 253
Phi0hp, 253
setAmplhc, 249
setAmplhp, 249
setAnglPol, 250
setBeta, 250
setDirProp, 250
setFreq, 250
setLambda, 250
setParam, 251
setPhi0hc, 251
setPhi0hp, 251
GWNewton2, 254
~GWNewton2, 262
a1, 271
a11, 271
a1x, 271
a2, 271
a22, 272
a3, 272
a4, 272
a5, 272
a6, 272
a7, 272
AnglPol, 272
b1, 272
b11, 273
b1x, 273
b2, 273
b3, 273
b4, 273
Beta, 273
c1, 273
c1x, 273
c2, 274
c2x, 274
c3, 274
CalculDirProp, 262

ci, 274
cmass, 274
commun, 262
d1, 274
d1x, 274
d2, 274
d2x, 275
d3x, 275
d4x, 275
d5x, 275
deltam, 275
DirProp, 275
DispTempVal, 263
e1, 275
e1x, 275
e2, 276
e2x, 276
e3, 276
e3x, 276
e4, 276
e4x, 276
e5, 276
e5x, 276
e6x, 276
e7x, 277
e8x, 277
f1, 277
f10, 277
f3, 277
f5, 277
f6, 277
f7, 277
f8, 277
f9, 278
fe, 263
g1, 278
getAnglPol, 263
getBeta, 263
getDirProp, 263
getDistance, 263
getInc, 263
getLambda, 264
getM1, 264
getM2, 264
getNParam, 264
getParam, 264
getPhCoal, 264
getTcoal, 264
gw, 278
GWNewton2, 261
hc, 264
hint, 278
hp, 265
inc, 278
init, 266
Lambda, 278
lambda, 278
m1, 279
m2, 279
mtot, 279
mu, 279
NParam, 279
nu, 279
omega, 279
omega0, 280
phase, 268
phcoal, 280
phi, 280
psi, 280
rdist, 280
setAnglPol, 268
setBeta, 269
setDirProp, 269
setDistance, 269
setInc, 269
setLambda, 270
setM1, 270
setM2, 270
setParam, 270
setPhCoal, 271
setTcoal, 271
si, 280
taud, 280
taud0, 281
tcoal, 281
teta, 281
time_encour, 281
type, 281
GWPeriGate, 282
GWPeriGate, 284
GWPeriGate
~GWPeriGate, 284
Amplhc, 289
Amplhp, 289
AnglPol, 289
Beta, 289
CalculDirProp, 285
DirProp, 289
DispTempVal, 285
Freq, 289
getAmplhc, 285
getAmplhp, 285
getAnglPol, 285
getBeta, 285
getDirProp, 285
getFreq, 286
getLambda, 286
getNParam, 286

getParam, 286
GWPeriGate, 284
hc, 286
hp, 286
init, 287
Lambda, 289
NParam, 290
setAmplhc, 287
setAmplhp, 287
setAnglPol, 287
setBeta, 287
setDirProp, 287
setFreq, 288
setLambda, 288
setParam, 288
GWs
ConfigSim, 86
GWSources
TrFctGW, 468
GWSto, 291
AnglPol, 297
Beta, 297
CalculDirProp, 294
DirProp, 297
DispTempVal, 294
Fac_Hc, 297
Fac_Hp, 297
Fac_norm, 297
Fknee, 298
Fmin, 298
getAnglPol, 294
getBeta, 294
getDirProp, 294
getLambda, 294
getNParam, 294
getParam, 294
GWSto, 293
hc, 295
hp, 295
init, 295
Lambda, 298
Nb_Ageing, 298
NFhc, 298
NFhp, 298
NParam, 298
OrderLa, 299
setAnglPol, 295
setBeta, 295
setDirProp, 296
setLambda, 296
setParam, 296
Slope, 299
tder_hc, 299
tder_hp, 299
tDurAdd, 299
tFirst, 299
tLast, 299
Tsample, 299
tStep, 299
gXMLAngle
ConfigSim, 78
gXMLAstroDistance
ConfigSim, 78
gXMLAstroMass
ConfigSim, 78
gXMLdouble
ConfigSim, 78
gXMLFrequency
ConfigSim, 78
gXMLint
ConfigSim, 79
gXMLstring
ConfigSim, 79
gXMLTime
ConfigSim, 79
gXMLTimeSeries
ConfigSim, 79
gXMLUnit
ConfigSim, 80
gXMLWord
ConfigSim, 80

h
randlib.c, 605
H0_cgs
LISACODE-PhysicConstants.h, 584
h_SI
LISACODE-PhysicConstants.h, 584
hb_SI
LISACODE-PhysicConstants.h, 584
hc
GW, 169
GWBinary, 178
GWCusp, 189
GWFastSpinBBH, 204
GWFile, 227
GWMono, 239
GWNew, 249
GWNewton2, 264
GWPeriGate, 286
GWSto, 295
hint
GWNewton2, 278
hList
GWFile, 231
HmQuadCell
ellipFilter, 28
hp

GW, 169
GWBinary, 178
GWCusp, 189
GWFastSpinBBH, 204
GWFile, 228
GWMono, 239
GWNew, 249
GWNewton2, 265
GWPeriGate, 286
GWSto, 295

I
ellipFilter, 28
idxcur
GWFastSpinBBH, 215
idxm
GWFastSpinBBH, 215
ignbin
randlib.c, 607
randlib.h, 612
ignlgi
com.c, 480
randlib.h, 612
ignnbn
randlib.c, 608
randlib.h, 613
ignpoi
randlib.c, 608
randlib.h, 613
ignuin
randlib.c, 608
randlib.h, 613
IM
detect, 18
inc
GWBinary, 182
GWNewton2, 278
IndexDelay
TDI, 453
IndexEta
TDI, 453
IndexInReadData
MemoryReadDisk, 325
IndirectDir
PhoDetPhaMet, 426
infnty
randlib.c, 605
init
Filter, 105
GeometryAnalytic, 126
GeometryFile, 142
GeometryMLDC, 157
GW, 169
GWBinary, 179
GWCusp, 189
GWFastSpinBBH, 204
GWFile, 228
GWMono, 239
GWNew, 249
GWNewton2, 266
GWPeriGate, 287
GWSto, 295
PhoDetPhaMet, 422
TrFctGW, 468
USOClock, 473
initGlobal
Geometry, 112
GeometryAnalytic, 127
GeometryFile, 142
GeometryMLDC, 157
initgn
com.c, 481
randlib.h, 613
InitialAzimuthalAngleL
GWFastSpinBBH, 216
InitialPolarAngleL
GWFastSpinBBH, 216
initRK
GWFastSpinBBH, 205
Input data (directory Input_data), 19
InputFileName
GeometryFile, 146
inrgcm
com.c, 481
IntegrateSignal
PhoDetPhaMet, 423
InterCubic
Serie, 435
InterfPhyData
PhoDetPhaMet, 426
InterfType
PhoDetPhaMet, 426
InterHermite
Serie, 436
InterLagrange
Serie, 436
InterLinear
Serie, 437
INTERP
serie, 35
Interpol
GWFile, 228
InterpType
TDI_InterData, 461
InterpUtilValue
TDI_InterData, 461
iRead
BackgroundGalactic, 58

iSC
PhoDetPhaMet, 426
iSerie
TDI, 453

k
TrFctGW, 468
k_SI
LISACODE-PhysicConstants.h, 584
KmQuadCell
ellipFilter, 28
kpc_m
LISACODE-PhysicConstants.h, 584

L0_m_default
LISACODE-LISAConstants.h, 553
L0m
Geometry, 116
GeometryAnalytic, 132
GeometryFile, 146
GeometryMLDC, 162
LA
detect, 18
la0Laser_m
LISACODE-LISAConstants.h, 553
LAG
serie, 35
Lambda
GW, 171
GWBinary, 182
GWCusp, 192
GWFastSpinBBH, 216
GWFile, 232
GWMono, 242
GWNew, 252
GWNewton2, 278
GWPeriGate, 289
GWSto, 298
lambda
GWNewton2, 278
LaserPower
ConfigSim, 86
LaserPower_W_default
LISACODE-LISAConstants.h, 554
LastUsedBin
GWFile, 232
LCVersion
LISACODE-LISAConstants.h, 554
len
ezxml_root, 99
Length
GWFile, 232
lennob
randlib.c, 608
LIN
serie, 35
linpack.c, 502
sdot, 502
spofa, 502
LISA, 301
∼LISA, 304
gArmLength, 304
gDelayT, 305
gPosSC, 305
GWB, 307
LISA, 302, 303
MakeOneStepOfTime, 305
NoisePointers, 307
PhotoDetects, 307
PresentMeanNoise, 306
RecordPDPM, 307
SCPos, 307
sGW, 307
Stabilization, 306
tMemRAM, 307
tStepMes, 308
tStepPhy, 308
USOs, 308
LISA_L0_m
LISACODE-PhysicConstants.h, 585
LISA_L0_s
LISACODE-PhysicConstants.h, 585
LISACODE-Background.cpp, 503
LISACODE-Background.h, 504
LISACODE-BackgroundGalactic.cpp, 505
LISACODE-BackgroundGalactic.h, 506
LISACODE-ConfigSim.cpp, 507
LISACODE-ConfigSim.h, 508
LISACODE-Couple.cpp, 510
operator ∗, 510
operator+, 510
operator-, 511
operator/, 511
LISACODE-Couple.h, 512
LISACODE-DnonGW.cpp, 513
LISACODE-EllipticFilter.cpp, 514
LISACODE-EllipticFilter.h, 516
LISACODE-Filter.cpp, 518
LISACODE-Filter.h, 519
LISACODE-Geometry.cpp, 520
LISACODE-Geometry.h, 521
LISACODE-GeometryAnalytic.cpp, 522
LISACODE-GeometryAnalytic.h, 523
LISACODE-GeometryFile.cpp, 524
LISACODE-GeometryFile.h, 525
LISACODE-GeometryMLDC.cpp, 526
LISACODE-GeometryMLDC.h, 527
LISACODE-GW.cpp, 528

LISACODE-GW.h, 529
LISACODE-GWBinary.cpp, 530
LISACODE-GWBinary.h, 531
LISACODE-GWCusp.cpp, 532
LISACODE-GWCusp.h, 533
LISACODE-GWFastSpinBBH.cpp, 534
LISACODE-GWFastSpinBBH.h, 535
LISACODE-GWFile.cpp, 536
LISACODE-GWFile.h, 537
LISACODE-GWMono.cpp, 538
LISACODE-GWMono.h, 539
LISACODE-GWNew.cpp, 540
LISACODE-GWNew.h, 541
LISACODE-GWNewton2.cpp, 542
LISACODE-GWNewton2.h, 543
LISACODE-GWPeriGate.cpp, 544
LISACODE-GWPeriGate.h, 545
LISACODE-GWSto.cpp, 546
LISACODE-GWSto.h, 547
LISACODE-LISA.cpp, 548
LISACODE-LISA.h, 549
LISACODE-LISACode.cpp, 550
    main, 551
LISACODE-LISACode.ep.cpp, 552
LISACODE-LISAConstants.h, 553
    L0_m_default, 553
    la0Laser_m, 553
    LaserPower_W_default, 554
    LCVersion, 554
    nu0Laser_Hz, 554
    omega, 554
    Rgc, 554
    tRangeStoreDelay_default, 554
    tRangeStorePos_default, 554
LISACODE-Mat.cpp, 556
    operator ∗, 556
    operator+, 556
    operator-, 556
LISACODE-Mat.h, 557
LISACODE-MathUtils.h, 558
LISACODE-Memory.cpp, 559
LISACODE-Memory.h, 560
LISACODE-MemoryReadDisk.cpp, 561
LISACODE-MemoryReadDisk.h, 562
LISACODE-MemoryWriteDisk.cpp, 563
LISACODE-MemoryWriteDisk.h, 564
LISACODE-Noise.cpp, 565
LISACODE-Noise.h, 566
LISACODE-NoiseFile.cpp, 567
LISACODE-NoiseFile.h, 568
LISACODE-NoiseFilter.cpp, 569
LISACODE-NoiseFilter.h, 570
LISACODE-NoiseFShape.cpp, 571
LISACODE-NoiseFShape.h, 572
LISACODE-NoiseOof.cpp, 573
LISACODE-NoiseOof.h, 574
LISACODE-NoiseTwoFilter.cpp, 575
LISACODE-NoiseTwoFilter.h, 576
LISACODE-NoiseWhite.cpp, 577
LISACODE-NoiseWhite.h, 578
LISACODE-PhoDetPhaMet.cpp, 579
LISACODE-PhoDetPhaMet.h, 580
LISACODE-PhysicConstants.h, 581
LISACODE-PhysicConstants.h
    au_m, 583
    au_s, 583
    c_SI, 583
    CE_RG, 583
    Dy_SI, 583
    eps0_SI, 584
    G_SI, 584
    gamma_u, 584
    H0_cgs, 584
    h_SI, 584
    hb_SI, 584
    k_SI, 584
    kpc_m, 584
    LISA_L0_m, 585
    LISA_L0_s, 585
    LS_SI, 585
    ly_au, 585
    ly_m, 585
    me_SI, 585
    mn_SI, 585
    mp_SI, 585
    MS_SI, 586
    mu0_SI, 586
    Na_SI, 586
    omegaYr, 586
    pc_au, 586
    pc_ly, 586
    pc_m, 586
    PRECISION, 586
    RS_SI, 587
    RSchw, 587
    S_SI, 587
    TSUN, 587
    Yr_SI, 587
LISACODE-Serie.cpp, 588
LISACODE-Serie.h, 589
LISACODE-TDI.cpp, 590
LISACODE-TDI.h, 591
LISACODE-TDI_InterData.cpp, 592
LISACODE-TDI_InterData.h, 593
LISACODE-TDIApply.cpp, 594
    main, 594
LISACODE-TDITools.cpp, 595
LISACODE-TDITools.h, 596

LISACODE-TrFctGW.cpp, 597
LISACODE-TrFctGW.h, 598
LISACODE-USOClock.cpp, 599
LISACODE-USOClock.h, 600
LISACODE-Vect.cpp, 601
operator ∗, 601
operator+, 602
operator-, 602
operator/, 602
LISACODE-Vect.h, 603
LISAGeo
Background, 53
BackgroundGalactic, 58
TrFctGW, 469
ListTmpData
Memory, 318
MemoryReadDisk, 325
MemoryWriteDisk, 334
loadNoise
Noise, 341
NoiseFile, 351
NoiseFilter, 363
NoiseFShape, 374
NoiseOof, 386
NoiseTwoFilter, 398
NoiseWhite, 410
LS_SI
LISACODE-PhysicConstants.h, 585
ly_au
LISACODE-PhysicConstants.h, 585
ly_m
LISACODE-PhysicConstants.h, 585

m
ezxml_root, 100
M1
GWBinary, 183
m1
GWFastSpinBBH, 216
GWNewton2, 279
M2
GWBinary, 183
m2
GWFastSpinBBH, 216
GWNewton2, 279
main
LISACODE-LISACode.cpp, 551
LISACODE-TDIApply.cpp, 594
main, 45
Main (directory Main), 45
majuscule
ConfigSim, 80
MakeOneStepOfTime
LISA, 305
MakeTitles
Memory, 316
MemoryReadDisk, 323
MemoryWriteDisk, 332
Mat, 309
∼Mat, 310
display, 310
Mat, 310
operator ∗, 310
operator+, 310
operator-, 311
p, 311
Mathematical Tools (directory Outils_Math), 33
MathUtils, 312
MathUtils
deg2rad, 312
rad2deg, 312
mathUtils
MAX, 34
MIN, 34
SWAP, 34
Matrix, 32
MAX
mathUtils, 34
max
randlib.c, 605
maxnum
randlib.c, 605
Mchirp
GWFastSpinBBH, 216
me_SI
LISACODE-PhysicConstants.h, 585
Memory, 313
∼Memory, 315
AddSerieData, 315
AlreadyRecDat, 318
gData, 315
getNbSerie, 316
gettMax, 316
gettStepRecord, 316
gettStoreData, 316
ListTmpData, 318
MakeTitles, 316
Memory, 314
ReceiveData, 316
RecordAccData, 317
settStepRecord, 317
settStoreData, 317
tStepRecord, 318
tStoreData, 318
unusable, 317
Memory (directory Memoire), 47
MemoryReadDisk, 319
MemoryReadDisk, 321

MemoryReadDisk
~MemoryReadDisk, 321
AddSerieData, 322
AlreadyRecDat, 325
FEncoding, 325
FichMem, 325
gData, 322
getNbSerie, 322
gettMax, 322
gettStepRecord, 323
gettStoreData, 323
IndexInReadData, 325
ListTmpData, 325
MakeTitles, 323
MemoryReadDisk, 321
NomFichMem, 325
ReadASCIIFile, 323
ReadBinaryFile, 323
ReadData, 325
ReceiveData, 323
RecordAccData, 324
settStepRecord, 324
settStoreData, 324
TitlesReadData, 326
tStepRecord, 326
tStoreData, 326
unusable, 324
MemoryWriteDisk, 327
MemoryWriteDisk, 329, 330
MemoryWriteDisk
~MemoryWriteDisk, 330
AddSerieData, 331
AlreadyRecDat, 334
BinHeader, 334
CloseFile, 331
FEncoding, 334
FichMem, 334
gData, 331
getNbSerie, 331
gettMax, 332
gettStepRecord, 332
gettStoreData, 332
ListTmpData, 334
MakeTitles, 332
MemoryWriteDisk, 329, 330
NomFichMem, 334
ReceiveData, 332
RecordAccData, 333
SCSerie, 334
settStepRecord, 333
settStoreData, 333
TimeEnd, 335
TimeOffset, 335
TitleSerie, 335
tStepRecord, 335
tStoreData, 335
unusable, 333
MIN
mathUtils, 34
min
randlib.c, 605
minlog
randlib.c, 605
mltmod
randlib.c, 608
randlib.h, 613
mn_SI
LISACODE-PhysicConstants.h, 585
move
Geometry, 116
GeometryAnalytic, 132
GeometryFile, 146
GeometryMLDC, 162
mp_SI
LISACODE-PhysicConstants.h, 585
MS_SI
LISACODE-PhysicConstants.h, 586
Mtot
GWFastSpinBBH, 217
mtot
GWNewton2, 279
mu
GWFastSpinBBH, 217
GWNewton2, 279
mu0_SI
LISACODE-PhysicConstants.h, 586
mulvec
GWFastSpinBBH, 217
muly2
GWFastSpinBBH, 217

Na_SI
LISACODE-PhysicConstants.h, 586
name
ezxml, 97
Nb_Ageing
GWSto, 298
NoiseOof, 388
NbBinAdd
Noise, 343
NoiseFile, 353
NoiseFilter, 365
NoiseFShape, 376
NoiseOof, 388
NoiseTwoFilter, 400
NoiseWhite, 412
NbDat
GWFile, 232

NbData
BackgroundGalactic, 58
GeometryFile, 147
Noise, 343
NoiseFile, 353
NoiseFilter, 365
NoiseFShape, 376
NoiseOof, 388
NoiseTwoFilter, 400
NoiseWhite, 412
NbDataAdd
PhoDetPhaMet, 426
NbDataStab
Filter, 105
NbDataStored
NoiseFile, 353
PhoDetPhaMet, 426
NbDelayMax
TDI, 452
NbGenTDI
ConfigSim, 80
NbMaxDelays
ConfigSim, 87
next
ezxml, 97
NFhc
GWSto, 298
NFhp
GWSto, 298
NFilter
NoiseFilter, 366
NoiseOof, 389
NoiseTwoFilter, 401
NFilter_2
NoiseTwoFilter, 401
NFm
NoiseFShape, 377
NFp
NoiseFShape, 377
Noise, 336
~Noise, 339
addNoise, 339
generNoise, 339
getNbBinAdd, 340
getNoise, 340
gettDurAdd, 341
gettFirst, 341
gettLast, 341
gettStep, 341
loadNoise, 341
NbBinAdd, 343
NbData, 343
Noise, 338
NoiseData, 343
NoiseType, 343
settDurAdd, 341
settFirst, 342
settLast, 342
settStep, 342
tDurAdd, 344
TestType, 343
tFirst, 344
tLast, 344
tStep, 344
Noise (directory Bruits), 17
NoiseData
Noise, 343
NoiseFile, 353
NoiseFilter, 366
NoiseFShape, 377
NoiseOof, 389
NoiseTwoFilter, 401
NoiseWhite, 412
NoiseData_tmp1
NoiseTwoFilter, 401
NoiseData_tmp2
NoiseTwoFilter, 401
NoiseFile, 345
NoiseFile, 347, 348
NoiseFile
addNoise, 348
FactMult, 353
FileName, 353
generNoise, 348
getFileName, 349
getNbBinAdd, 349
getNbDataStored, 349
getNoise, 349, 350
gettDurAdd, 350
gettFirst, 350
gettLast, 350
gettStep, 351
loadNoise, 351
NbBinAdd, 353
NbData, 353
NbDataStored, 353
NoiseData, 353
NoiseFile, 347, 348
NoiseType, 354
ReadBin, 354
setFileName, 351
settDurAdd, 351
settFirst, 351
settLast, 352
settStep, 352
StoredData, 354
tDurAdd, 354
TestType, 352

tFirst, 354
tLast, 354
tStep, 355
NoiseFilter, 356
NoiseFilter, 358–360
NoiseFilter
addNoise, 361
generNoise, 361
getFilterAlpha, 361
getFilterBeta, 361
getNbBinAdd, 362
getNoise, 362
gettDurAdd, 363
gettFirst, 363
gettLast, 363
gettStep, 363
loadNoise, 363
NbBinAdd, 365
NbData, 365
NFilter, 366
NoiseData, 366
NoiseFilter, 358–360
NoiseType, 366
settDurAdd, 364
settFirst, 364
settLast, 364
settStep, 365
tDurAdd, 366
TestType, 365
tFirst, 366
tLast, 366
tStep, 367
WhiteData, 367
NoiseFShape, 368
NoiseFShape, 370, 371
NoiseFShape
addNoise, 371
FactF0, 375
FactFm, 376
FactFp, 376
FilterFm, 376
FilterFp, 376
FmData, 376
FpData, 376
generNoise, 371
getNbBinAdd, 372
getNoise, 372
gettDurAdd, 373
gettFirst, 373
gettLast, 373
gettStep, 373
loadNoise, 374
NbBinAdd, 376
NbData, 376
NFm, 377
NFp, 377
NoiseData, 377
NoiseFShape, 370, 371
NoiseType, 377
settDurAdd, 374
settFirst, 374
settLast, 375
settStep, 375
tDurAdd, 377
TestType, 375
tFirst, 378
tLast, 378
tmpData_m, 378
tmpData_p, 378
tStep, 378
WhiteData, 378
NoiseOof, 380
NoiseOof, 382, 383
NoiseOof
addNoise, 383
alpha, 388
fmax, 388
fmin, 388
generNoise, 383
getFilterAlpha, 384
getFilterBeta, 384
getNbBinAdd, 384
getNoise, 384, 385
gettDurAdd, 385
gettFirst, 385
gettLast, 386
gettStep, 386
loadNoise, 386
Nb_Ageing, 388
NbBinAdd, 388
NbData, 388
NFilter, 389
NoiseData, 389
NoiseOof, 382, 383
NoiseType, 389
settDurAdd, 386
settFirst, 387
settLast, 387
settStep, 387
tDurAdd, 389
TestType, 387
tFirst, 389
tLast, 390
tStep, 390
WhiteData, 390
NOISEORIG
detect, 18
NoisePlace

ConfigSim, 80
NoisePointers
LISA, 307
Noises
ConfigSim, 87
NoisesCreation
ConfigSim, 81
NoisesData
ConfigSim, 87
NoiseSpec, 391
NoiseSpec
NStr, 391
NType, 391
NVal0, 392
NVal01, 392
NVal1, 392
NVal2, 392
NoiseTwoFilter, 393
NoiseTwoFilter, 395
NoiseTwoFilter
addNoise, 396
generNoise, 396
getFilterAlpha, 396
getFilterBeta, 397
getNbBinAdd, 397
getNoise, 397
gettDurAdd, 398
gettFirst, 398
gettLast, 398
gettStep, 398
loadNoise, 398
NbBinAdd, 400
NbData, 400
NFilter, 401
NFilter_2, 401
NoiseData, 401
NoiseData_tmp1, 401
NoiseData_tmp2, 401
NoiseTwoFilter, 395
NoiseType, 401
settDurAdd, 399
settFirst, 399
settLast, 400
settStep, 400
tDurAdd, 402
TestType, 400
tFirst, 402
tLast, 402
tStep, 402
WhiteData, 402
WhiteData2, 402
NoiseType
Noise, 343
NoiseFile, 354
NoiseFilter, 366
NoiseFShape, 377
NoiseOof, 389
NoiseTwoFilter, 401
NoiseWhite, 412
NoiseWhite, 404
NoiseWhite, 406, 407
NoiseWhite
addNoise, 407
generNoise, 407
getNbBinAdd, 408
getNoise, 408
getPSD, 409
getSqPSD, 409
gettDurAdd, 409
gettFirst, 409
gettLast, 410
gettStep, 410
loadNoise, 410
NbBinAdd, 412
NbData, 412
NoiseData, 412
NoiseType, 412
NoiseWhite, 406, 407
setSqPSD, 410
settDurAdd, 410
settFirst, 411
settLast, 411
settStep, 411
Sigma, 413
tDurAdd, 413
TestType, 412
tFirst, 413
tLast, 413
tStep, 413
NomFichMem
MemoryReadDisk, 325
MemoryWriteDisk, 334
NoNoise
PhoDetPhaMet, 426
TDI_InterData, 461
norme
Vect, 476
NParam
GW, 171
GWBinary, 183
GWCusp, 193
GWFastSpinBBH, 217
GWFile, 232
GWMono, 243
GWNew, 253
GWNewton2, 279
GWPeriGate, 290
GWSto, 298

NPs
PhoDetPhaMet, 427
NStr
NoiseSpec, 391
NType
NoiseSpec, 391
nu
Geometry, 116
GeometryAnalytic, 132
GeometryFile, 147
GeometryMLDC, 162
GWNewton2, 279
nu0Laser_Hz
LISACODE-LISAConstants.h, 554
numg
com.c, 480
randlib.c, 605
NVal0
NoiseSpec, 392
NVal01
NoiseSpec, 392
NVal1
NoiseSpec, 392
NVal2
NoiseSpec, 392

OB
detect, 18
off
ezxml, 97
Offset
USOClock, 473
omega
GWNewton2, 279
LISACODE-LISAConstants.h, 554
omega0
GWNewton2, 280
omegaYr
LISACODE-PhysicConstants.h, 586
OP
detect, 18
operator ∗
Couple, 94
LISACODE-Couple.cpp, 510
LISACODE-Mat.cpp, 556
LISACODE-Vect.cpp, 601
Mat, 310
Vect, 477
operator+
Couple, 94
LISACODE-Couple.cpp, 510
LISACODE-Mat.cpp, 556
LISACODE-Vect.cpp, 602
Mat, 310
Vect, 477
operator-
Couple, 95
LISACODE-Couple.cpp, 511
LISACODE-Mat.cpp, 556
LISACODE-Vect.cpp, 602
Mat, 311
Vect, 478
operator/
Couple, 95
LISACODE-Couple.cpp, 511
LISACODE-Vect.cpp, 602
Vect, 478
OrbInitRot
ConfigSim, 87
OrbMove
ConfigSim, 87
OrbOrder
ConfigSim, 87
OrbStartTime
ConfigSim, 88
order_default
Geometry, 116
GeometryAnalytic, 132
GeometryFile, 147
GeometryMLDC, 162
OrderCellMaxNorm
ellipFilter, 29
ordered
ezxml, 97
OrderLa
GWSto, 299
OutFile
TDI, 453
OutFileEncoding
TDI, 454

p
Mat, 311
Vect, 478
p15
GWFastSpinBBH, 217
p150
GWFastSpinBBH, 217
p20
GWFastSpinBBH, 217
p200
GWFastSpinBBH, 218
parent
ezxml, 97
PBFilter
PhoDetPhaMet, 427
pc_au
LISACODE-PhysicConstants.h, 586

pc_ly
LISACODE-PhysicConstants.h, 586
pc_m
LISACODE-PhysicConstants.h, 586
PDPMINTERF
detect, 18
PDPMMem
TDI_InterData, 461
PhaMetFilterON
ConfigSim, 88
PhaMetFilterParam
ConfigSim, 88
phase
GWNewton2, 268
phcoal
GWNewton2, 280
phi
GWFastSpinBBH, 218
GWNewton2, 280
Phi0
GWFastSpinBBH, 218
phi0
GWBinary, 183
Phi0hc
GWCusp, 193
GWMono, 243
GWNew, 253
Phi0hp
GWCusp, 193
GWMono, 243
GWNew, 253
Phi10
GWFastSpinBBH, 218
phic
GWFastSpinBBH, 218
PhoDetPhaMet, 415
PhoDetPhaMet, 417–419
PhoDetPhaMet
~PhoDetPhaMet, 420
DisplayStoredData, 420
FilterON, 425
FilterParam, 425
FilterPhyData, 425
getIndirectDir, 421
getiSC, 421
getNoNoise, 421
gettStab, 421
gGWB, 421
gN, 422
GWB, 425
IndirectDir, 426
init, 422
IntegrateSignal, 423
InterfPhyData, 426
InterfType, 426
iSC, 426
NbDataAdd, 426
NbDataStored, 426
NoNoise, 426
NPs, 427
PBFilter, 427
PhoDetPhaMet, 417–419
ReceiveSignal, 424
RecordData, 427
SCPos, 427
sGW, 427
tStepMes, 427
tStepPhy, 427
USO, 428
PhotoDetects
LISA, 307
phrtsd
randlib.c, 608
randlib.h, 613
pi
ezxml_root, 100
PolarAngleOfSpin1
GWFastSpinBBH, 218
PolarAngleOfSpin2
GWFastSpinBBH, 218
pole
QuadCell, 430
PoleMatching
ellipFilter, 29
position
Geometry, 112
GeometryAnalytic, 128
GeometryFile, 143
GeometryMLDC, 158
PRECISION
LISACODE-PhysicConstants.h, 586
PresentMeanNoise
LISA, 306
psi
GWFastSpinBBH, 218
GWNewton2, 280

QuadCell, 429
QuadCell
a0, 429
a1, 429
b1, 429
b2, 430
pole, 430
u, 430
zero, 430

r

GWBinary, 183
rad2deg
MathUtils, 312
randlib.c, 604
ABS, 605
expmax, 605
fsign, 605
ftnstop, 605
genbet, 605
genchi, 606
genexp, 606
genf, 606
gengam, 606
genmn, 606
genmul, 606
gennch, 606
gennf, 606
gennor, 607
genprm, 607
genunf, 607
gscgn, 607
gsrgs, 607
gssst, 607
h, 605
ignbin, 607
ignnbn, 608
ignpoi, 608
ignuin, 608
infnty, 605
lennob, 608
max, 605
maxnum, 605
min, 605
minlog, 605
mltmod, 608
numg, 605
phrtsd, 608
ranf, 608
setgmn, 608
sexpo, 609
sgamma, 609
snorm, 609
randlib.h, 610
advnst, 610
genbet, 610
genchi, 610
genexp, 611
genf, 611
gengam, 611
genmn, 611
genmul, 611
gennch, 611
gennf, 611
gennor, 612
genprm, 612
genunf, 612
getsd, 612
gscgn, 612
ignbin, 612
ignlgi, 612
ignnbn, 613
ignpoi, 613
ignuin, 613
initgn, 613
mltmod, 613
phrtsd, 613
ranf, 613
setall, 614
setant, 614
setgmn, 614
setsd, 614
sexpo, 614
sgamma, 614
snorm, 614
ranf
randlib.c, 608
randlib.h, 613
RapidOption
TDITools, 465
rdist
GWNewton2, 280
ReadASCIIFile
ConfigSim, 81
GWFile, 229
MemoryReadDisk, 323
ReadBin
NoiseFile, 354
ReadBinaryFile
GWFile, 229
MemoryReadDisk, 323
ReadData
MemoryReadDisk, 325
ReadFile
BackgroundGalactic, 57
ConfigSim, 82
ReadSignEtaDelays
TDI, 452
ReadXMLFile
ConfigSim, 82
ReceiveData
Memory, 316
MemoryReadDisk, 323
MemoryWriteDisk, 332
ReceiveSignal
PhoDetPhaMet, 424
RecordAccData
Memory, 317
MemoryReadDisk, 324

MemoryWriteDisk, 333
RecordAndReturnResult
TDI, 452
RecordData
PhoDetPhaMet, 427
RecordPDPM
LISA, 307
Records
GWFile, 232
RefreshDelay
TDITools, 464
rfile
Serie, 437
rfileC
SerieC, 444
Rgc
LISACODE-LISAConstants.h, 554
RK_EPS
GWFastSpinBBH, 219
RK_H
GWFastSpinBBH, 219
RK_VECLENGTH
GWFastSpinBBH, 219
rkckstep
GWFastSpinBBH, 205
Rmin
GWFastSpinBBH, 219
rot
GeometryAnalytic, 132
GeometryMLDC, 162
rot0
Geometry, 117
GeometryAnalytic, 132
GeometryFile, 147
GeometryMLDC, 162
RS_SI
LISACODE-PhysicConstants.h, 587
RSchw
LISACODE-PhysicConstants.h, 587

S
detect, 18
s
ezxml_root, 100
S1xdot
GWFastSpinBBH, 205
S1ydot
GWFastSpinBBH, 205
S1zdot
GWFastSpinBBH, 205
s2psi
GWFastSpinBBH, 219
S2xdot
GWFastSpinBBH, 205
S2ydot
GWFastSpinBBH, 206
S2zdot
GWFastSpinBBH, 206
S_SI
LISACODE-PhysicConstants.h, 587
SC1ParamName
ConfigSim, 88
SC2ParamName
ConfigSim, 89
SC3ParamName
ConfigSim, 89
SCPos
LISA, 307
PhoDetPhaMet, 427
SCposStore
Geometry, 117
GeometryAnalytic, 132
GeometryFile, 147
GeometryMLDC, 163
SCSerie
MemoryWriteDisk, 334
SCSig
TDI, 454
sdot
linpack.c, 502
Serie, 35, 431
~Serie, 434
addData, 434
delLastData, 434
dx, 439
gData, 434
getBinValue, 435
getNbVal, 435
getRef, 435
getRefStep, 435
InterCubic, 435
InterHermite, 436
InterLagrange, 436
InterLinear, 437
rfile, 437
Serie, 433
setBinValue, 437
setNbVal, 438
setRefStart, 438
setRefStep, 438
TruncVal, 438
wfile, 438
x0, 439
ys, 439
serie
CUB, 35
INTERP, 35
LAG, 35

LIN, 35
SIN, 35
TRU, 35
SerieC, 440
SerieC, 441, 442
SerieC
~SerieC, 442
addDataC, 443
delLastDataC, 443
dx, 445
getBinValueC, 443
getNbValC, 443
getRefC, 443
getRefStepC, 444
rfileC, 444
SerieC, 441, 442
setBinValueC, 444
setNbValC, 444
setRefStartC, 444
setRefStepC, 444
wfileC, 445
x0, 445
ys, 445
setall
com.c, 481
randlib.h, 614
setAmplhc
GWCusp, 189
GWMono, 239
GWNew, 249
GWPeriGate, 287
setAmplhp
GWCusp, 189
GWMono, 239
GWNew, 249
GWPeriGate, 287
setAmplPNorder
GWFastSpinBBH, 206
setAnglPol
GW, 169
GWBinary, 179
GWCusp, 190
GWFastSpinBBH, 206
GWFile, 229
GWMono, 240
GWNew, 250
GWNewton2, 268
GWPeriGate, 287
GWSto, 295
setant
com.c, 481
randlib.h, 614
setBeta
GW, 170
GWBinary, 179
GWCusp, 190
GWFastSpinBBH, 206
GWFile, 229
GWMono, 240
GWNew, 250
GWNewton2, 269
GWPeriGate, 287
GWSto, 295
setBinValue
Serie, 437
setBinValueC
SerieC, 444
setDirProp
GW, 170
GWBinary, 179
GWCusp, 190
GWFastSpinBBH, 207
GWFile, 229
GWMono, 240
GWNew, 250
GWNewton2, 269
GWPeriGate, 287
GWSto, 296
setDistance
GWBinary, 180
GWFastSpinBBH, 207
GWNewton2, 269
setFileName
GWFile, 230
NoiseFile, 351
setForb
GWBinary, 180
setFreq
GWCusp, 190
GWMono, 240
GWNew, 250
GWPeriGate, 288
setGeometry
Background, 52
BackgroundGalactic, 57
setgmn
randlib.c, 608
randlib.h, 614
setInc
GWBinary, 180
GWNewton2, 269
setLambda
GW, 170
GWBinary, 180
GWCusp, 190
GWFastSpinBBH, 207
GWFile, 230
GWMono, 240

GWNew, 250
GWNewton2, 270
GWPeriGate, 288
GWSto, 296
setM1
GWBinary, 180
GWNewton2, 270
setM2
GWBinary, 180
GWNewton2, 270
setMass1
GWFastSpinBBH, 207
setMass2
GWFastSpinBBH, 207
setNbVal
Serie, 438
setNbValC
SerieC, 444
setParam
GW, 170
GWBinary, 181
GWCusp, 191
GWFastSpinBBH, 207
GWFile, 230
GWMono, 241
GWNew, 251
GWNewton2, 270
GWPeriGate, 288
GWSto, 296
setPhCoal
GWNewton2, 271
setPhi0
GWBinary, 181
setPhi0hc
GWCusp, 191
GWMono, 241
GWNew, 251
setPhi0hp
GWCusp, 191
GWMono, 241
GWNew, 251
setRefStart
Serie, 438
setRefStartC
SerieC, 444
setRefStep
Serie, 438
setRefStepC
SerieC, 444
setsd
com.c, 481
randlib.h, 614
setSqPSD
NoiseWhite, 410
sett0
Geometry, 113
GeometryAnalytic, 128
GeometryFile, 143
GeometryMLDC, 158
setTcoal
GWNewton2, 271
settDurAdd
Noise, 341
NoiseFile, 351
NoiseFilter, 364
NoiseFShape, 374
NoiseOof, 386
NoiseTwoFilter, 399
NoiseWhite, 410
settFirst
Noise, 342
NoiseFile, 351
NoiseFilter, 364
NoiseFShape, 374
NoiseOof, 387
NoiseTwoFilter, 399
NoiseWhite, 411
setTimeStore
TDI_InterData, 461
settLast
Noise, 342
NoiseFile, 352
NoiseFilter, 364
NoiseFShape, 375
NoiseOof, 387
NoiseTwoFilter, 400
NoiseWhite, 411
settRangeStoreDelay
Geometry, 113
GeometryAnalytic, 128
GeometryFile, 143
GeometryMLDC, 158
settRangeStorePos
Geometry, 113
GeometryAnalytic, 128
GeometryFile, 143
GeometryMLDC, 158
settStep
Noise, 342
NoiseFile, 352
NoiseFilter, 365
NoiseFShape, 375
NoiseOof, 387
NoiseTwoFilter, 400
NoiseWhite, 411
settStepRecord
Memory, 317
MemoryReadDisk, 324

MemoryWriteDisk, 333
settStoreData
Memory, 317
MemoryReadDisk, 324
MemoryWriteDisk, 333
sexpo
randlib.c, 609
randlib.h, 614
sf
GWFastSpinBBH, 219
sgamma
randlib.c, 609
randlib.h, 614
sGW
LISA, 307
PhoDetPhaMet, 427
shc
GWFastSpinBBH, 219
shp
GWFastSpinBBH, 219
si
GWNewton2, 280
sibling
ezxml, 98
Sigma
NoiseWhite, 413
SigmaNoise
USOClock, 473
sigmavec
GWFastSpinBBH, 220
sigmay2
GWFastSpinBBH, 220
Sign
TDI, 454
SignalList
BackgroundGalactic, 58
Simulator
ConfigSim, 89
SIN
serie, 35
sintheta
GWFastSpinBBH, 220
Slope
GWSto, 299
smu
Geometry, 117
GeometryAnalytic, 133
GeometryFile, 147
GeometryMLDC, 163
sn
ellipFilter, 29
snorm
randlib.c, 609
randlib.h, 614
sphilvec
GWFastSpinBBH, 220
sphily2
GWFastSpinBBH, 220
spline
GWFastSpinBBH, 208
spofa
linpack.c, 502
sqrt_3
GeometryMLDC, 163
sqrtee
Geometry, 117
GeometryAnalytic, 133
GeometryFile, 147
GeometryMLDC, 163
srot
GeometryAnalytic, 133
GeometryMLDC, 163
Stabilization
LISA, 306
standalone
ezxml_root, 100
std, 49
StoredData
NoiseFile, 354
SWAP
mathUtils, 34
sx
GWFastSpinBBH, 220
SystemEncoding
ConfigSim, 89
SystemEncoding_int
ConfigSim, 89

t0
Geometry, 117
GeometryAnalytic, 133
GeometryFile, 147
GeometryMLDC, 163
TaperApplied
GWFastSpinBBH, 220
TaperSteepness
GWFastSpinBBH, 220
TAU
detect, 18
taud
GWNewton2, 280
taud0
GWNewton2, 281
tc
GWFastSpinBBH, 221
tcoal
GWNewton2, 281
TDelay

TDI, 454
TDI_InterData, 462
TDITools, 465
tdelay
Geometry, 113
GeometryAnalytic, 128
GeometryFile, 143
GeometryMLDC, 158
tdelayOrderContribution
Geometry, 114
GeometryAnalytic, 129
GeometryFile, 144
GeometryMLDC, 159
tDeltaTDIDelay
ConfigSim, 89
tder_hc
GWSto, 299
tder_hp
GWSto, 299
TDI, 41, 446
~TDI, 450
Compute, 451
ComputeNoEta, 451
Eta, 453
Fact, 453
getCountInterDelay, 451
getCountInterEta, 451
IndexDelay, 453
IndexEta, 453
iSerie, 453
NbDelayMax, 452
OutFile, 453
OutFileEncoding, 454
ReadSignEtaDelays, 452
RecordAndReturnResult, 452
SCSig, 454
Sign, 454
TDelay, 454
TDI, 448–450
TDIQuickMod, 454
tmpCountInterDelay, 454
tmpCountInterEta, 454
TDI handling (directory TDI), 40
TDI_InterData, 42, 456
TDI_InterData, 457, 458
TDI_InterData
~TDI_InterData, 459
ComputeEta, 459
Eta, 461
gData, 459, 460
gettDelayCompute, 460
getTimeStore, 460
gettStep, 460
getUsable, 461
InterpType, 461
InterpUtilValue, 461
NoNoise, 461
PDPMMem, 461
setTimeStore, 461
TDelay, 462
TDI_InterData, 457, 458
TimeStore, 462
tShift, 462
Usable, 462
TDIDelayApprox
ConfigSim, 89
TDIInterp
ConfigSim, 89
TDIInterpUtilVal
ConfigSim, 90
TDIParamName
ConfigSim, 90
TDIParamNameType
ConfigSim, 90
TDIQuickMod
TDI, 454
TDIsName
ConfigSim, 90
TDIsPacks
ConfigSim, 90
TDIsPacksFact
ConfigSim, 90
tDisplay
ConfigSim, 90
TDITools, 43, 463
~TDITools, 464
DelayMem, 465
getDelay, 464
getRapidOption, 464
RapidOption, 465
RefreshDelay, 464
TDelay, 465
TDITools, 463, 464
tDurAdd
GWSto, 299
Noise, 344
NoiseFile, 354
NoiseFilter, 366
NoiseFShape, 377
NoiseOof, 389
NoiseTwoFilter, 402
NoiseWhite, 413
testbyteorder
ConfigSim, 82
TestType
Noise, 343
NoiseFile, 352
NoiseFilter, 365

NoiseFShape, 375
NoiseOof, 387
NoiseTwoFilter, 400
NoiseWhite, 412
teta
GWNewton2, 281
tFirst
GWSto, 299
Noise, 344
NoiseFile, 354
NoiseFilter, 366
NoiseFShape, 378
NoiseOof, 389
NoiseTwoFilter, 402
NoiseWhite, 413
thomasvec
GWFastSpinBBH, 221
thomasy2
GWFastSpinBBH, 221
time_encour
GWNewton2, 281
Time_sec
GeometryFile, 148
timeCur
GWFastSpinBBH, 221
TimeEnd
MemoryWriteDisk, 335
TimeList
BackgroundGalactic, 58
GWFile, 232
TimeOffset
ConfigSim, 91
GWFile, 233
MemoryWriteDisk, 335
TimeStep
GWFile, 233
TimeStore
TDI_InterData, 462
timevec
GWFastSpinBBH, 221
TitleSerie
MemoryWriteDisk, 335
TitlesReadData
MemoryReadDisk, 326
tLast
GWSto, 299
Noise, 344
NoiseFile, 354
NoiseFilter, 366
NoiseFShape, 378
NoiseOof, 390
NoiseTwoFilter, 402
NoiseWhite, 413
tMax
ConfigSim, 91
tmax
GWFastSpinBBH, 221
tMaxDelay
ConfigSim, 82
tMemNecInterpTDI
ConfigSim, 83
tMemNoiseFirst
ConfigSim, 91
tMemNoiseLast
ConfigSim, 91
tMemRAM
LISA, 307
tMemSig
ConfigSim, 91
tMinDelay
ConfigSim, 83
tmp_ci
BackgroundGalactic, 58
tmp_cip1
BackgroundGalactic, 59
tmp_Sig_i
BackgroundGalactic, 59
tmp_Sig_ip1
BackgroundGalactic, 59
tmp_t
BackgroundGalactic, 59
tmpCountInterDelay
TDI, 454
tmpCountInterEta
TDI, 454
TmpData
Filter, 106
tmpData_m
NoiseFShape, 378
tmpData_p
NoiseFShape, 378
tmu
Geometry, 117
GeometryAnalytic, 133
GeometryFile, 148
GeometryMLDC, 163
Tobs
GWFastSpinBBH, 221
Toffset
GWFastSpinBBH, 221
tRangeStoreDelay
Geometry, 117
GeometryAnalytic, 133
GeometryFile, 148
GeometryMLDC, 163
tRangeStoreDelay_default
LISACODE-LISAConstants.h, 554
tRangeStorePos

Geometry, 117
GeometryAnalytic, 133
GeometryFile, 148
GeometryMLDC, 164
tRangeStorePos_default
LISACODE-LISAConstants.h, 554
TransfZQuadCell
ellipFilter, 30
TransfZQuadCellChain
ellipFilter, 30
TrFctGW, 466
TrFctGW, 467
TrFctGW
~TrFctGW, 467
deltanu, 467
GWSources, 468
init, 468
k, 468
LISAGeo, 469
TrFctGW, 467
u, 469
v, 469
TRU
serie, 35
TruncVal
Serie, 438
Tsample
GWSto, 299
tShift
TDI_InterData, 462
tStep
GWSto, 299
Noise, 344
NoiseFile, 355
NoiseFilter, 367
NoiseFShape, 378
NoiseOof, 390
NoiseTwoFilter, 402
NoiseWhite, 413
tStepMes
ConfigSim, 91
LISA, 308
PhoDetPhaMet, 427
tStepPhy
ConfigSim, 91
LISA, 308
PhoDetPhaMet, 427
tStepRecord
Memory, 318
MemoryReadDisk, 326
MemoryWriteDisk, 335
tStoreData
Memory, 318
MemoryReadDisk, 326
MemoryWriteDisk, 335
tStoreDelay
Geometry, 118
GeometryAnalytic, 134
GeometryFile, 148
GeometryMLDC, 164
tStorePos
Geometry, 118
GeometryAnalytic, 134
GeometryFile, 148
GeometryMLDC, 164
TSUN
LISACODE-PhysicConstants.h, 587
txt
ezxml, 98
type
GWNewton2, 281

u
ezxml_root, 100
QuadCell, 430
TrFctGW, 469
unit
Vect, 477
unusable
Memory, 317
MemoryReadDisk, 324
MemoryWriteDisk, 333
update
GWFastSpinBBH, 208
uppercase
ConfigSim, 83
Usable
TDI_InterData, 462
USO
PhoDetPhaMet, 428
USO clock (directory USO_Temps), 39
USOClock, 470
~USOClock, 472
DerivLinearCoef, 473
getDeriv, 472
getNoise, 472
getOffset, 472
gGap, 472
gTime, 473
init, 473
Offset, 473
SigmaNoise, 473
USOClock, 471
USONoise, 474
USONoise
USOClock, 474
USOs
ConfigSim, 92

LISA, 308

v
TrFctGW, 469
Vect, 475
~Vect, 476
display, 476
norme, 476
operator *, 477
operator+, 477
operator-, 478
operator/, 478
p, 478
unit, 477
Vect, 476
VectNormal
Geometry, 114
GeometryAnalytic, 130
GeometryFile, 145
GeometryMLDC, 160
Vector, 36
velocity
Geometry, 115
GeometryAnalytic, 130
GeometryFile, 145
GeometryMLDC, 160

wfile
Serie, 438
wfileC
SerieC, 445
WhiteData
NoiseFilter, 367
NoiseFShape, 378
NoiseOof, 390
NoiseTwoFilter, 402
WhiteData2
NoiseTwoFilter, 402

x
Couple, 95
x0
Serie, 439
SerieC, 445
Xa1
com.c, 481
Xa1vw
com.c, 481
Xa1w
com.c, 482
Xa2
com.c, 482
Xa2vw
com.c, 482
Xa2w
com.c, 482
Xcg1
com.c, 482
Xcg2
com.c, 482
Xig1
com.c, 482
Xig2
com.c, 482
Xlg1
com.c, 482
Xlg2
com.c, 483
Xm1
com.c, 483
Xm2
com.c, 483
xmax
GWFastSpinBBH, 222
xml
ezxml_root, 100
XmlOutputFile
ConfigSim, 92
xold
GWFastSpinBBH, 222
Xqanti
com.c, 483
XYZ
GeometryFile, 148

y
Couple, 95
Yr_SI
LISACODE-PhysicConstants.h, 587
ys
Serie, 439
SerieC, 445

zero
QuadCell, 430