GremlinEq

Generated by Doxygen 1.8.16

1 Class Index 1

1.1 Class List . . . 1

2 Class Documentation 3

2.1 CEDR Class Reference . . . 3

2.1.1 * . . . 3

2.1.2 * . . . 3

2.1.3 Detailed Description . . . 4

2.2 CEID Class Reference . . . 4

2.2.1 * . . . 4

2.2.2 * . . . 4

2.2.3 Detailed Description . . . 5

2.3 CEKG Class Reference . . . 5

2.3.1 * . . . 5

2.3.2 * . . . 5

2.3.3 Detailed Description . . . 5

2.3.4 Constructor & Destructor Documentation . . . 5

2.3.4.1 CEKG() . . . 5

2.3.5 Member Data Documentation . . . 6

2.3.5.1 a . . . 6

2.3.5.2 E . . . 6

2.3.5.3 Lz . . . 6

2.3.5.4 Om_phi . . . 6

2.3.5.5 r . . . 6

2.4 CEKR Class Reference . . . 6

2.4.1 * . . . 6

2.4.2 * . . . 7

2.4.3 Detailed Description . . . 7

2.5 CETD Class Reference . . . 7

2.5.1 * . . . 7

2.5.2 * . . . 7

2.5.3 Detailed Description . . . 8

2.6 Clebsch Class Reference . . . 8

2.6.1 * . . . 8

2.6.2 Detailed Description . . . 9

2.6.3 Member Function Documentation . . . 9

2.6.3.1 sinthetabrac() . . . 9

2.7 DataHolder Struct Reference . . . 9

2.7.1 * . . . 9

2.7.2 Detailed Description . . . 10

2.8 FT Class Reference . . . 10

2.8.1 * . . . 10

2.8.2 * . . . 10
2.8.3 * . . . 12
2.8.4 Detailed Description . . . 13
2.8.5 Member Function Documentation . . . 13
2.8.5.1 Accuracy_in() . . . 13
2.8.5.2 Accuracy_up() . . . 13
2.8.5.3 Bin() . . . 13
2.8.5.4 Btrans() . . . 13
2.8.5.5 CalcRFields() . . . 13
2.8.5.6 choose_solvers() . . . 13
2.8.5.7 Ctrans() . . . 14
2.8.5.8 ddr_K() . . . 14
2.8.5.9 ddr_TeukRin() . . . 14
2.8.5.10 ddr_TeukRup() . . . 14
2.8.5.11 dr_K() . . . 14
2.8.5.12 dr_TeukRin() . . . 14
2.8.5.13 dr_TeukRup() . . . 14
2.8.5.14 get_ddrteuk() . . . 14
2.8.5.15 K() . . . 15
2.8.5.16 request_precision_in() [1/2] . . . 15
2.8.5.17 request_precision_in() [2/2] . . . 15
2.8.5.18 request_precision_up() [1/2] . . . 15
2.8.5.19 request_precision_up() [2/2] . . . 15
2.8.5.20 teuk_potential() . . . 15
2.8.5.21 TeukRin() . . . 15
2.8.5.22 TeukRup() . . . 15
2.9 GKG Class Reference . . . 16
2.9.1 * . . . 16
2.9.2 Detailed Description . . . 16
2.10 Kerr Class Reference . . . 16
2.10.1 * . . . 16
2.10.2 Detailed Description . . . 17
2.10.3 Member Function Documentation . . . 17
2.10.3.1 ddr_Delta() . . . 17
2.10.3.2 ddr_Sigma() . . . 17
2.10.3.3 Delta() . . . 17
2.10.3.4 dr_Delta() . . . 17
2.10.3.5 dr_Sigma() . . . 18
2.10.3.6 rminus() . . . 18
2.10.3.7 rplus() . . . 18
2.10.3.8 rstar() . . . 18
2.10.3.9 Sigma() . . . 18

2.11 RRGW Class Reference . . . 19
2.11.1 * . . . 19
2.11.2 Detailed Description . . . 19
2.11.3 Member Function Documentation . . . 19
2.11.3.1 alpha_func() . . . 19
2.11.3.2 Psi4() . . . 20
2.11.3.3 Wave() [1/2] . . . 20
2.11.3.4 Wave() [2/2] . . . 20
2.12 SWSH Class Reference . . . 21
2.12.1 * . . . 21
2.12.2 * . . . 21
2.12.3 Detailed Description . . . 21
2.12.4 Constructor & Destructor Documentation . . . 21
2.12.4.1 SWSH() . . . 21
2.13 Tensor< TypeHere > Class Template Reference . . . 22
2.13.1 * . . . 22
2.13.2 Detailed Description . . . 22
2.14 TidalH Class Reference . . . 23
2.14.1 * . . . 23
2.14.2 * . . . 23
2.14.3 * . . . 23
2.14.4 Detailed Description . . . 24

Index **25**

Chapter 1

Class Index

1.1 Class List

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

CEDR
: Circular Equatorial Data Reader Class . . . 3

CEID
: Circular Equatorial Inspiral Data Class . . . 4

CEKG
: Circular Equatorial Kerr Geodesic Class . . . 5

CEKR
: Circular Equatorial Kerr Radiation Class . . . 6

CETD
: Circular Equatorial Teukolsky Driver Class . . . 7

Clebsch
: Clebsch-Gordan Coefficients Class . . . 8

DataHolder
: Data Holder Struct (deprecated March 2019) . . . 9

FT
: Fujita Tagoshi Class . . . 10

GKG
: Generic Kerr Geodesics Class . . . 16

Kerr
: Kerr Quantities Class . . . 16

RRGW
: Radiation Reaction Gravitational Waves Class . . . 19

SWSH
: Spin-Weighted Spheroidal Harmonics Class . . . 21

Tensor< TypeHere >
: Tensors Class . . . 22

TidalH
: Tidal Horizon Class . . . 23

Chapter 2

Class Documentation

2.1 CEDR Class Reference

Circular Equatorial Data Reader Class.

`#include <CEDR.h>`

2.1.1 *

Public Member Functions

- **CEDR** (char inname[ ])
- int **ReadData** (const int l, const int m)

2.1.2 *

Public Attributes

- hid_t **infile**
- Real **r**
- Real **a**
- Real **E**
- Real **Lz**
- Real **Om_phi**
- int **lmin**
- int **lmax**
- int **mmin**
- int **mmax**
- Real **lambda**
- Complex **ZI**
- Complex **ZH**
- Complex **Rin**
- Complex **dRin**
- Complex **Rup**
- Complex **dRup**
- Real **EdotI**
- Real **EdotH**
- Real **LzdotI**
- Real **LzdotH**
- Real **rdotI**
- Real **rdotH**
- bool **exists**

2.1.3 Detailed Description

Circular Equatorial Data Reader Class.

CEDR stands for circular, equatorial data reader. This class collects methods which read the output of a data run (i.e., the HDF5 data produced by the code) and present it in human readable format.

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

- include/CEDR.h
- src/circeq/CEDR.cc

2.2 CEID Class Reference

Circular Equatorial Inspiral Data Class.

`#include <CEID.h>`

2.2.1 *

Public Member Functions

- **CEID** (char *basename, const Real arrmin, const Real rmax, const Real dr, const int ellmax)
- **CEID** (char *basename, const int Nhi, const int ellmax)
- void **Spheroids** (const Real costheta_view)
- void **Get_rdot_omega** (const Real r)
- void **Get_fluxes** (const Real r)
- void **Get_wave** (const Real r, const Real phi, const Real phi_view)
- void **Get_flux_Smode** (const Real r, const int l, const int m, Real &EdotHlm, Real &EdotIlm)
- void **Get_flux_mmode** (const Real r, const int m, Real &EdotHm, Real &EdotIm)
- void **Get_flux_lmode** (const Real r, const int l, Real &EdotHl, Real &EdotIl)
- void **Get_flux_Ymode** (const Real r, const int l, const int m, Real &EdotHlm, Real &EdotIlm)
- void **Get_ZIlm** (const Real r, const int l, const int m, Complex &ZIlm)
- void **Get_CIlm** (const Real r, const Real phi, const int l, const int m, Complex &CIlm)
- void **Spline_CIlm** (const int l, const int m)

2.2.2 *

Public Attributes

- Real **rdot**
- Real **rdot_noH**
- Real **Omega**
- Real **Omega_max**
- Real **rmin**
- Real **EdotI**
- Real **LzdotI**
- Real **EdotH**
- Real **LzdotH**
- Real **hp**
- Real **hc**
- Real **r_isco**
- Real **a**
- int **jmax**
- Real * **r_arr**
- int * **max_l_computed**

2.2.3 Detailed Description

Circular Equatorial Inspiral Data Class.

This class uses CEDR, and provides methods for smoothly interpolating through a large set of output from many orbits to study waveforms and inspirals.

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

- include/CEID.h
- src/circeq/CEID.cc

2.3 CEKG Class Reference

Circular Equatorial Kerr Geodesic Class.

`#include <CEKG.h>`

2.3.1 *

Public Member Functions

- CEKG (const int orbitsense, const Real rad, const Real spin)
  A constructor for the CEKG Class.

2.3.2 *

Public Attributes

- Real r
- Real a
- Real E
- Real Lz
- Real Om_phi

2.3.3 Detailed Description

Circular Equatorial Kerr Geodesic Class.

Defines methods for computing various quantities related to circular, equatorial Kerr geodesics.

2.3.4 Constructor & Destructor Documentation

2.3.4.1 CEKG()

```
CEKG::CEKG (
            const int orbitsense,
```

```
            const Real rad,
            const Real spin )
```

A constructor for the CEKG Class.

Initialized with orientation (prograde/retrograde), orbital radius, and spin. Circular equatorial Kerr geodesics only.

2.3.5 Member Data Documentation

2.3.5.1 a

```
Real CEKG::a
```

spin paramter

2.3.5.2 E

```
Real CEKG::E
```

energy

2.3.5.3 Lz

```
Real CEKG::Lz
```

z component of angular momentum

2.3.5.4 Om_phi

```
Real CEKG::Om_phi
```

Axial Frequency

2.3.5.5 r

```
Real CEKG::r
```

orbital radius

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

- include/CEKG.h
- src/circeq/CEKG.cc

2.4 CEKR Class Reference

Circular Equatorial Kerr Radiation Class.

```
#include <CEKR.h>
```

2.4.1 *

Public Member Functions

- **CEKR** (SWSH *swsh_in, FT *ft_in, CEKG *cekg_in)

2.4.2 *

Public Attributes

- Complex **ZI**
- Complex **ZH**

2.4.3 Detailed Description

Circular Equatorial Kerr Radiation Class.

CEKR describes methods for computing various quantities to radiation from circular, equatorial Kerr geodesics. It relies on input from instances of the classes SWSH, FT, and CEKG.

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

- include/CEKR.h
- src/circeq/CEKR.cc

2.5 CETD Class Reference

Circular Equatorial Teukolsky Driver Class.

`#include <CETD.h>`

2.5.1 *

Public Member Functions

- **CETD** (const int orbitsense, const Real rad, const Real spin, char outbase[ ])
- void **Driver** (const int lmax)
- void **Driver** (const Real EPS_L, const int lmax_min)
- void **Driver** (const Real EPS_L, const int lmax, const int lmax_min)
- void **DoHarmonic** (const int l, const int m)

2.5.2 *

Public Attributes

- char **outname** [256]
- hid_t **hdffile**
- RRGW * **rrgw**
- CEKG * **cekg**
- int **proret**
- int **l**
- int **m**
- Real **r**
- Real **a**
- int **lmin**
- int **lmax**
- int **mmin**
- int **mmax**
- Real **EdotH**
- Real **EdotI**
- Complex **ZH**
- Complex **ZI**

2.5.3 Detailed Description

Circular Equatorial Teukolsky Driver Class.

A class which is used to "drive" studies that look at a many circular equatorial orbits. It is essentially a holder for methods that solve CEKR repeatedly. This class is also used to write the HDF5 data files (the "d" stands for data as well as driver). The synopsis is that the HDF5 file contains 2 groups: the 2 modes (i.e., the indices l and m) and the parameters (orbital radius r, spin parameter a, energy E, axial angular momentum Lz, and axial frequency Om_phi). Into the modes group goes 19 different bits of data,

- The spin-weighted spheroidal harmonic eigenvalue lambda
- The real and the imaginary parts of Z_Inf (the amplitude of the "to infinity" Teukolsky amplitude)
- The real and the imaginary parts of Z_H (the amplitude of the "down horizon" Teukolsky amplitude)
- The real and the imaginary parts of Rin (the ingoing separated radial part of the Teukolsky function, at the orbit)
- The real and the imaginary parts of d/dr(Rin)
- The real and the imaginary parts of Rup (the outgoing separated radial part of the Teukolsky function, at the orbit)
- The real and the imaginary parts of d/dr(Rup)
- 4 fluxes carried by the radiation, Edot_Inf, Edot_H, Lzdot_Inf, Lzdot_H
- The rate of change of orbital radius arising from radiation flux to infinity, rdot_Inf
- The rate of change of orbital radius arising from the down-horizon radiation flux, rdot_H

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

- include/CETD.h
- src/circeq/CETD.cc

2.6 Clebsch Class Reference

Clebsch-Gordan Coefficients Class.

```
#include <SWSH.h>
```

2.6.1 *

Public Member Functions

- Real **xbrac** (const int s, const int q, const int p, const int m)
- Real **xsqrbrac** (const int s, const int q, const int p, const int m)
- Real sinthetabrac (const int s, const int p, const int q, const int m, const int mp)

2.6.2 Detailed Description

Clebsch-Gordan Coefficients Class.

Defines methods for computing and manipulating Clebsch-Gordan coefficients

2.6.3 Member Function Documentation

2.6.3.1 sinthetabrac()

```
Real Clebsch::sinthetabrac (
            const int s,
            const int p,
            const int q,
            const int m,
            const int mp )
```

Computes <s,p,m|sin(theta)|s,q,mp>

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

- include/SWSH.h
- src/swsh/SWSHCGUtil.cc

2.7 DataHolder Struct Reference

Data Holder Struct (deprecated March 2019)

```
#include <Globals.h>
```

2.7.1 *

Public Attributes

- int **l**
- int **m**
- int **k**
- Real **r**
- Real **a**
- Real **Lz**
- Real **E**
- Real **Q**
- Real **w**
- Real **p**
- Real **lambda**
- Complex **ZI**
- Complex **ZH**
- Real **EdotInf**
- Real **LzdotInf**
- Real **QdotInf**
- Real **rdotInf**
- Real **EdotH**
- Real **LzdotH**
- Real **QdotH**
- Real **rdotH**

2.7.2 Detailed Description

Data Holder Struct (deprecated March 2019)

Holder for all the data we need to write out. (deprecated March 2019)

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

- include/Globals.h

2.8 FT Class Reference

Fujita Tagoshi Class.

`#include <FT.h>`

2.8.1 *

Public Types

- typedef double **Real**
- typedef long double **LongReal**
- typedef std::complex< Real > **Complex**
- typedef std::complex< LongReal > **LongComplex**

2.8.2 *

Public Member Functions

- **FT** (const int l, const int m, const Real r, const Real a, const Real omega, const Real lambda, const Real tolerance)
- **FT** (const int l, const int m, const Real p, const Real ecc, const Real a, const Real omega, const Real lambda, const Real tolerance)
- Complex Bin ()
- Complex Btrans ()
- Complex Ctrans ()
- void CalcRFields (const Real rad, const int DoH)
- Complex TeukRin ()
- Complex TeukRup ()
- Complex dr_TeukRin ()
- Complex dr_TeukRup ()
- Complex ddr_TeukRin ()
- Complex ddr_TeukRup ()
- Real Accuracy_in ()
- Real Accuracy_up ()
- Real request_precision_in (const Real p)
- Real request_precision_up (const Real p)
- Real request_precision_in ()
- Real request_precision_up ()
- int **terms_evaluated** ()

- void **set_gmp** (int flag)
- int **get_gmp** ()
- Real K (const Real rad)
- Real dr_K (const Real rad)
- Real ddr_K ()
- Complex teuk_potential (const Real rad)
- Complex get_ddrteuk (const Real rad, const Complex rteuk, const Complex drteuk)
- int choose_solvers (const Real low, const Real high, const Real ∗lowerrs, const Real ∗higherrs, const int isin, const int oldchoice, Real ∗∗locptr, int ∗∗choiceptr, const int ∗maxchoiceptr)
- void **geterrs** (const Real r, Real ∗errs, int isin)
- Complex **callsolver** (const int isin, const int which, const Real r, const Real epsilon, const Complex nu, const Real precision, Complex ∗deriv)
- Complex **callin** (const Real r, Complex ∗deriv)
- Complex **callup** (const Real r, Complex ∗deriv)
- void **getlocs** (const int isin)
- Real **plusfrac** (Real nu, Real epsilon, Real q, int m, Real lambda, Real ∗term)
- Complex **cplusfrac** (Complex nu, Real epsilon, Real q, int m, Real lambda, Real ∗term)
- Real **minusfrac** (Real nu, Real epsilon, Real q, int m, Real lambda, Real ∗term)
- Complex **cminusfrac** (Complex nu, Real epsilon, Real q, int m, Real lambda, Real ∗term)
- Real **radialfrac** (Real nu, Real epsilon, Real q, int m, Real lambda, Real ∗term)
- Complex **cradialfrac** (Complex nu, Real epsilon, Real q, int m, Real lambda, Real ∗term)
- Real **radialfrac_half** (Real imnu, Real epsilon, Real q, int m, Real lambda, Real ∗term)
- int **renangmom** (Real epsilon, Real q, int l, int m, Real lambda, Complex ∗nu)
- int **get_deciders** (Real epsilon, Real q, int m, Real lambda, Real ∗halfval, Real ∗oneslope)
- int **renangmom_real_search_procedure** (Real epsilon, Real q, int l, int m, Real lambda, Complex ∗nu)
- int **singularity_fit** (Real epsilon, Real q, int l, int m, Real lambda, int guessint, int guessint2, Real spacing, Complex ∗nu)
- int **renangmom_real** (Real epsilon, Real q, int l, int m, Real lambda, Real detect_limit, Real ∗nu)
- int **renangmom_real_divide_search** (Real epsilon, Real q, int l, int m, Real lambda, Real ∗nu)
- int **renangmom_half** (Real epsilon, Real q, int l, int m, Real lambda, Real ∗imnu)
- int **renangmom_iint** (Real epsilon, Real q, int l, int m, Real lambda, Real ∗imnu)
- int **renangmom_far** (Real epsilon, Real q, int l, int m, Real lambda, Complex ∗nu)
- Real **nu_predictor** (Real epsilon, Real q, int m, Real lambda)
- int **renangmom_real_guess** (Real epsilon, Real q, int m, Real lambda, Real guess, Real ∗nu)
- Real **fractions_poly_approx** (Real epsilon, Real q, int m, Real lambda)
- Real **radial_half_value** (Real epsilon, Real q, int m, Real lambda, Real ∗term)
- Real **radial_int_slope** (Real epsilon, Real q, int m, Real lambda, Real ∗term)
- void **asympt_amps** (Complex nu, Real epsilon, Real q, int m, Real lambda, Complex ∗b_trans, Complex ∗b_inc, Complex ∗b_ref, Complex ∗c_trans)
- Complex **fsum** (Complex nu, Real epsilon, Real q, int m, Real lambda)
- Complex **kfactor** (Complex nu, Real epsilon, Real q, int m, Real lambda)
- Complex **aminus** (Complex nu, Real epsilon, Real q, int m, Real lambda)
- Complex **rzero** (Complex nu, Real epsilon, Real q, int m, Real lambda, Real x, Complex ∗deriv)
- Complex **rin_hyper** (Complex nu, Real epsilon, Real q, int m, Real lambda, Real x, Real precision, Complex ∗deriv)
- Complex **rcoulomb** (Complex nu, Real epsilon, Real q, int m, Real lambda, Real z, Complex ∗deriv)
- Complex **rin_coulomb** (Complex nu, Real epsilon, Real q, int m, Real lambda, Real z, Real precision, Complex ∗deriv)
- Complex **rup_tricomi** (Complex nu, Real epsilon, Real q, int m, Real lambda, Real z, Real precision, Complex ∗deriv)
- Complex **rin_small** (Complex nu, Real epsilon, Real q, int m, Real lambda, Real x, Real precision, Complex ∗deriv)
- Complex **rup_hyper** (Complex nu, Real epsilon, Real q, int m, Real lambda, Real x, Real precision, Complex ∗deriv)
- std::complex< double > **gammln** (const std::complex< double > x)

- std::complex< double > **sinIn** (const std::complex< double > x)
- std::complex< HYPERGEOM_REAL_TYPE > **hypergeom2F1** (std::complex< double > n1, std::complex< double > n2, std::complex< double > d1, std::complex< double > x)
- std::complex< HYPERGEOM_REAL_TYPE > **hypergeom1F1** (std::complex< double > n1, std::complex< double > d1, std::complex< double > x)
- std::complex< HYPERGEOM_REAL_TYPE > **hypergeomU** (std::complex< double > a, std::complex< double > b, std::complex< double > x)
- std::complex< HYPERGEOM_REAL_TYPE > **hypergeomU_reduced** (std::complex< double > a, std::complex< double > b, std::complex< double > x)
- std::complex< HYPERGEOM_REAL_TYPE > **hypergeom2F1_gmp** (std::complex< double > n1, std::complex< double > n2, std::complex< double > d1, std::complex< double > x, int precision)
- std::complex< HYPERGEOM_REAL_TYPE > **hypergeom1F1_gmp** (std::complex< double > n1, std::complex< double > d1, std::complex< double > x, int precision)

2.8.3 *

Public Attributes

- int **l**
- int **m**
- Real **a**
- Real **p**
- Real **e**
- Real **omega**
- Real **lambda**
- Real **tolerance**
- Real **accuracy_in**
- Real **accuracy_up**
- Real **rin_request_precision**
- Real **rup_request_precision**
- Complex **nu**
- Complex **b_trans**
- Complex **b_inc**
- Complex **c_trans**
- Complex **rteuk_in**
- Complex **rteuk_up**
- Complex **drteuk_in**
- Complex **drteuk_up**
- Complex **ddrteuk_in**
- Complex **ddrteuk_up**
- int **up_choices** [FT_MAXDIVS]
- Real **up_choice_locs** [FT_MAXDIVS]
- int **in_choices** [FT_MAXDIVS]
- Real **in_choice_locs** [FT_MAXDIVS]
- Complex **test_renangmoms** [TEST_LENGTH]
- int **gmp_on**
- int **hypergeom_terms**

2.8.4 Detailed Description

Fujita Tagoshi Class.

FT defines methods for computing solutions to the homogeneous Teukolsky equation as laid out in papers by Ryuichi Fujita and Hideyushi Tagoshi

2.8.5 Member Function Documentation

2.8.5.1 Accuracy_in()

```
Real FT::Accuracy_in ( )  [inline]
```

the actual error bound

2.8.5.2 Accuracy_up()

```
Real FT::Accuracy_up ( )  [inline]
```

the actual error bound

2.8.5.3 Bin()

```
Complex FT::Bin ( )  [inline]
```

asymptotic amplitude

2.8.5.4 Btrans()

```
Complex FT::Btrans ( )  [inline]
```

asymptotic amplitudes

2.8.5.5 CalcRFields()

```
void FT::CalcRFields (
             const Real rad,
             const int DoH )
```

set the point at which the fields are calculated

2.8.5.6 choose_solvers()

```
int FT::choose_solvers (
             const Real low,
             const Real high,
             const Real * lowerrs,
             const Real * higherrs,
             const int isin,
             const int oldchoice,
             Real ** locptr,
```

```
            int ** choiceptr,
            const int * maxchoiceptr )
```

some quantities in the Teukolsky equation

2.8.5.7 Ctrans()

```
Complex FT::Ctrans ( )  [inline]
```

asymptotic amplitudes

2.8.5.8 ddr_K()

```
Real FT::ddr_K ( )  [inline]
```

some quantities in the Teukolsky equation

2.8.5.9 ddr_TeukRin()

```
Complex FT::ddr_TeukRin ( )  [inline]
```

value of the homogeneous solutions

2.8.5.10 ddr_TeukRup()

```
Complex FT::ddr_TeukRup ( )  [inline]
```

value of the homogeneous solutions

2.8.5.11 dr_K()

```
Real FT::dr_K (
            const Real rad )  [inline]
```

some quantities in the Teukolsky equation

2.8.5.12 dr_TeukRin()

```
Complex FT::dr_TeukRin ( )  [inline]
```

value of the homogeneous solutions

2.8.5.13 dr_TeukRup()

```
Complex FT::dr_TeukRup ( )  [inline]
```

value of the homogeneous solutions

2.8.5.14 get_ddrteuk()

```
Complex FT::get_ddrteuk (
            const Real rad,
            const Complex rteuk,
            const Complex drteuk )  [inline]
```

some quantities in the Teukolsky equation

2.8.5.15 K()

```
Real FT::K (
            const Real rad )  [inline]
```

some quantities in the Teukolsky equation

2.8.5.16 request_precision_in() [1/2]

```
Real FT::request_precision_in ( )  [inline]
```

requesting precision

2.8.5.17 request_precision_in() [2/2]

```
Real FT::request_precision_in (
            const Real p )  [inline]
```

requesting precision

2.8.5.18 request_precision_up() [1/2]

```
Real FT::request_precision_up ( )  [inline]
```

requesting precision

2.8.5.19 request_precision_up() [2/2]

```
Real FT::request_precision_up (
            const Real p )  [inline]
```

requesting precision

2.8.5.20 teuk_potential()

```
Complex FT::teuk_potential (
            const Real rad )  [inline]
```

some quantities in the Teukolsky equation

2.8.5.21 TeukRin()

```
Complex FT::TeukRin ( )  [inline]
```

value of the homogeneous solutions

2.8.5.22 TeukRup()

```
Complex FT::TeukRup ( )  [inline]
```

value of the homogeneous solutions

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

- include/FT.h
- src/fujtag/FT.cc

2.9 GKG Class Reference

Generic Kerr Geodesics Class.

`#include <GKG.h>`

2.9.1 *

Public Member Functions

- Real **RFunc** (const Real r, const Real a, const Real z, const Real E, const Real Lz, const Real Q)
- Real **dr_RFunc** (const Real r, const Real a, const Real z, const Real E, const Real Lz, const Real Q)
- Real **ddr_RFunc** (const Real r, const Real a, const Real z, const Real E, const Real Lz, const Real Q)
- Real **ThetaFunc** (const Real r, const Real a, const Real z, const Real E, const Real Lz, const Real Q)
- Real **PhiFunc** (const Real r, const Real a, const Real z, const Real E, const Real Lz, const Real Q)
- Real **TFunc** (const Real r, const Real a, const Real z, const Real E, const Real Lz, const Real Q)

2.9.2 Detailed Description

Generic Kerr Geodesics Class.

This class defines functions that are useful for generic Kerr geodesics. It is not used very much; its content may be subsumed into a different piece of the code in a future release.

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

- include/GKG.h
- src/utility/GKG.cc

2.10 Kerr Class Reference

Kerr Quantities Class.

`#include <Globals.h>`

2.10.1 *

Static Public Member Functions

- static Real rplus (const Real a)
- static Real rminus (const Real a)
- static Real rstar (const Real r, const Real a)
- static Real Delta (const Real r, const Real a)
- static Real dr_Delta (const Real r)
- static Real ddr_Delta ()
- static Real Sigma (const Real r, const Real a, const Real z)
- static Real dr_Sigma (const Real r)
- static Real ddr_Sigma ()
- static Real **Eeqpro** (const Real r, const Real a)
- static Real **Eeqret** (const Real r, const Real a)
- static Real **Lzeqpro** (const Real r, const Real a)
- static Real **Lzeqret** (const Real r, const Real a)
- static Real **Omega_phi_eqpro** (const Real r, const Real a)
- static Real **Omega_phi_eqret** (const Real r, const Real a)
- static Real **isco_pro** (const Real a)
- static Real **isco_ret** (const Real a)

2.10.2 Detailed Description

Kerr Quantities Class.

This is a simple container class for a collection of useful simply static functions that pop up over and over again when studying Kerr black hole orbits.

2.10.3 Member Function Documentation

2.10.3.1 ddr_Delta()

```
static Real Kerr::ddr_Delta ( )  [inline], [static]
```

All right, so this one's kind of silly ...d^2\Delta/dr^2 = 2

2.10.3.2 ddr_Sigma()

```
static Real Kerr::ddr_Sigma ( )  [inline], [static]
```

All right, so this one's kind of silly too ...d^2\Sigma/dr^2 = 2

2.10.3.3 Delta()

```
static Real Kerr::Delta (
             const Real r,
             const Real a )  [inline], [static]
```

\Delta = r^2 - 2 M r + a^2

2.10.3.4 dr_Delta()

```
static Real Kerr::dr_Delta (
             const Real r )  [inline], [static]
```

d\Delta/dr = 2 r - 2 M

2.10.3.5 dr_Sigma()

```
static Real Kerr::dr_Sigma (
            const Real r )  [inline], [static]
```

d\Sigma/dr = 2 r

2.10.3.6 rminus()

```
static Real Kerr::rminus (
            const Real a )  [inline], [static]
```

r_- = M - \sqrt{M^2 - a^2}

2.10.3.7 rplus()

```
static Real Kerr::rplus (
            const Real a )  [inline], [static]
```

r_+ = M + \sqrt{M^2 + a^2}; M \equiv 1

2.10.3.8 rstar()

```
static Real Kerr::rstar (
            const Real r,
            const Real a )  [inline], [static]
```

r^* = r + 2 r_+/(r_+ - r_-) \ln{(r - r_+)/2M} - 2 r_-/(r_+ - r_-) \ln{(r - r_-)/2M}

2.10.3.9 Sigma()

```
static Real Kerr::Sigma (
            const Real r,
            const Real a,
            const Real z )  [inline], [static]
```

\Sigma = r^2 + a^2 \cos^2\theta

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

- include/Globals.h

2.11 RRGW Class Reference

Radiation Reaction Gravitational Waves Class.

`#include <RRGW.h>`

2.11.1 *

Public Member Functions

- void **Flux_Infinity** (const Real a, const int m, const Real lamb, const Real w, const Real p, const Complex ZI, Real &Edot, Real &Lzdot)
- void **Flux_Horizon** (const Real a, const int m, const Real lamb, const Real w, const Real p, const Complex ZH, Real &Edot, Real &Lzdot)
- void **Qdotrdot** (const Real r, const Real a, const Real Q, const Real E, const Real Lz, const Real Edot, const Real Lzdot, Real &Qdot, Real &rdot)
- void Wave (const int m, const Real t_ret, const Real phi, const Real S, const Real w, const Complex ZI, Real &hplus, Real &hcross)
- void Wave (const int m, const int k, const Real N_m, const Real N_k, const Real phi, const Real S, const Real w, const Complex ZH, Real &hplus, Real &hcross)
- void Psi4 (const int m, const Real t_ret, const Real phi, const Real S, const Real w, const Complex ZI, Complex &psi4)
- Real alpha_func (const Real a, const int m, const Real lamb, const Real w, const Real p)

2.11.2 Detailed Description

Radiation Reaction Gravitational Waves Class.

This class defines functions that are useful for computing gravitational waves and radiation reaction. Aspects of this class may fit better in another module, and may be moved elsewhere in a future release.

2.11.3 Member Function Documentation

2.11.3.1 alpha_func()

```
Real RRGW::alpha_func (
            const Real a,
            const int m,
            const Real lamb,
            const Real w,
            const Real p )
```

Used to get down-horizon fluxes.

2.11.3.2 Psi4()

```
void RRGW::Psi4 (
            const int m,
            const Real t_ret,
            const Real phi,
            const Real S,
            const Real w,
            const Complex ZI,
            Complex & psi4 )
```

Quasi-general purpose, but does not work well for inspirals!

2.11.3.3 Wave() [1/2]

```
void RRGW::Wave (
            const int m,
            const int k,
            const Real N_m,
            const Real N_k,
            const Real phi,
            const Real S,
            const Real w,
            const Complex ZH,
            Real & hplus,
            Real & hcross )
```

Good for restricted inspiral, with indices m & k.

2.11.3.4 Wave() [2/2]

```
void RRGW::Wave (
            const int m,
            const Real t_ret,
            const Real phi,
            const Real S,
            const Real w,
            const Complex ZI,
            Real & hplus,
            Real & hcross )
```

Quasi-general purpose, but does not work well for inspirals!

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

- include/RRGW.h
- src/utility/RRGW.cc

2.12 SWSH Class Reference

Spin-Weighted Spheroidal Harmonics Class.

`#include <SWSH.h>`

2.12.1 *

Public Member Functions

- SWSH (const int ss, const int ll, const int mm, const Real spintimesfreq)

  A constructor for the SWSH Class.
- void **expand** (Real *E, Real *b, int *n)
- Real **error** (const Real E, const Real b[ ], const int n)
- Real **spheroid** (const Real x)
- Real **l2dagspheroid** (const Real x)
- Real **l1dagl2dagspheroid** (const Real x)
- Complex **edthbaredthbarspheroid** (const Real a, const Real x)
- Real **pos2Y** (const int l, const int m, const Real x)
- Real **pos1Y** (const int l, const int m, const Real x)
- Real **zeroY** (const int l, const int m, const Real x)
- Real **dzeroYdx** (const int l, const int m, const Real x)
- Real **neg1Y** (const int l, const int m, const Real x)
- Real **neg2Y** (const int l, const int m, const Real x)

2.12.2 *

Public Attributes

- Real **E**
- Real **lambda**
- Real **b** [MAXCOFS+1]
- int **lmin**
- int **N**
- int **Gl**

2.12.3 Detailed Description

Spin-Weighted Spheroidal Harmonics Class.

Contains all the routines that do things with spin-weighted spheroidal harmonics

2.12.4 Constructor & Destructor Documentation

2.12.4.1 SWSH()

```
SWSH::SWSH (
            const int ss,
            const int ll,
            const int mm,
            const Real spintimesfreq )
```

A constructor for the SWSH Class.

Initialized with (s,l,m,spheroidicity)

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

- include/SWSH.h
- src/swsh/SWSHSpherical.cc
- src/swsh/SWSHSpheroid.cc

2.13 Tensor< TypeHere > Class Template Reference

Tensors Class.

`#include <Tensors.h>`

2.13.1 *

Static Public Member Functions

- static TypeHere * **vector** (const long al, const long ah)
- static void **free_vector** (TypeHere *t, const long al, const long ah)
- static TypeHere ** **vectorptr** (const long al, const long ah)
- static void **free_vectorptr** (TypeHere **t, const long al, const long ah)
- static TypeHere ** **matrix** (const long al, const long ah, const long bl, const long bh)
- static void **free_matrix** (TypeHere **t, const long al, const long ah, const long bl, const long bh)
- static TypeHere *** **matrixptr** (const long al, const long ah, const long bl, const long bh)
- static void **free_matrixptr** (TypeHere ***t, const long al, const long ah, const long bl, const long bh)
- static TypeHere *** **tensor3** (const long al, const long ah, const long bl, const long bh, const long cl, const long ch)
- static void **free_tensor3** (TypeHere ***t, const long al, const long ah, const long bl, const long bh, const long cl, const long ch)
- static TypeHere **** **tensor4** (const long al, const long ah, const long bl, const long bh, const long cl, const long ch, const long dl, const long dh)
- static void **free_tensor4** (TypeHere ****t, const long al, const long ah, const long bl, const long bh, const long cl, const long ch, const long dl, const long dh)
- static TypeHere ***** **tensor5** (const long al, const long ah, const long bl, const long bh, const long cl, const long ch, const long dl, const long dh, const long el, const long eh)
- static void **free_tensor5** (TypeHere *****t, const long al, const long ah, const long bl, const long bh, const long cl, const long ch, const long dl, const long dh, const long el, const long eh)
- static TypeHere ****** **tensor6** (const long al, const long ah, const long bl, const long bh, const long cl, const long ch, const long dl, const long dh, const long el, const long eh, const long fl, const long fh)
- static void **free_tensor6** (TypeHere ******t, const long al, const long ah, const long bl, const long bh, const long cl, const long ch, const long dl, const long dh, const long el, const long eh, const long fl, const long fh)
- static TypeHere ******* **tensor7** (const long al, const long ah, const long bl, const long bh, const long cl, const long ch, const long dl, const long dh, const long el, const long eh, const long fl, const long fh, const long gl, const long gh)
- static void **free_tensor7** (TypeHere *******t, const long al, const long ah, const long bl, const long bh, const long cl, const long ch, const long dl, const long dh, const long el, const long eh, const long fl, const long fh, const long gl, const long gh)

2.13.2 Detailed Description

template<class TypeHere>
class Tensor< TypeHere >

Tensors Class.

This is a container class for memory allocation routines which define multi-index objects with arbitrary index range. They are inspired by Numerical Recipe's such routines [e.g., vector() and matrix()], but have been implemented in a new way and use C++ templates to make arrays of arbitrary type.

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

- include/Tensors.h

2.14 TidalH Class Reference

Tidal Horizon Class.

`#include <TidalH.h>`

2.14.1 *

Public Member Functions

- **TidalH** (const Real spin, const int ellmax)
- void **loadClm** ()
- void **loadEpslm** ()
- Complex **mR_vector** (const int q, const int m)
- Real **mD_matrix** (const int q, const int ell, const int m)
- Real **R1lm** (const int l, const int m, const Real x, const Real psi)
- Real **epsr** (const int l, const int m, const Real x, const Real psi)
- Real **C0** (const Real x)
- Real **C1** (const Real x)
- Real **D** (const Real x)
- Complex **IGRND** (const Real x)

2.14.2 *

Static Public Member Functions

- static Real **IGRNDre_wrapper** (Real x, void *params)
- static Real **IGRNDim_wrapper** (Real x, void *params)

2.14.3 *

Public Attributes

- Real **a**
- Real **rp**
- Real **eps**
- Real **Kph**
- int **lmax**
- SWSH ** **swshp**
- Complex ** **ZH**
- Complex ** **Clm**
- Complex ** **Epslm**
- Real * **w**
- Real * **p**
- Real ** **lamb**
- int **Gq**
- int **Gl**
- int **Gm**
- int **INTEGRAND**

2.14.4 Detailed Description

Tidal Horizon Class.

This class defines functions which are useful for analyzing how the on-horizon Teukolsky solution affects the geometry of a black hole. This methods were extensively used in work with former MIT student Stephen O'Sullivan, but may not be of broad interest; as such, they may be removed from general release.

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

- include/TidalH.h
- src/utility/TidalH.cc

Index

a
    CEKG, 6
Accuracy_in
    FT, 13
Accuracy_up
    FT, 13
alpha_func
    RRGW, 19

Bin
    FT, 13
Btrans
    FT, 13

CalcRFields
    FT, 13
CEDR, 3
CEID, 4
CEKG, 5
    a, 6
    CEKG, 5
    E, 6
    Lz, 6
    Om_phi, 6
    r, 6
CEKR, 6
CETD, 7
choose_solvers
    FT, 13
Clebsch, 8
    sinthetabrac, 9
Ctrans
    FT, 14

DataHolder, 9
ddr_Delta
    Kerr, 17
ddr_K
    FT, 14
ddr_Sigma
    Kerr, 17
ddr_TeukRin
    FT, 14
ddr_TeukRup
    FT, 14
Delta
    Kerr, 17
dr_Delta
    Kerr, 17
dr_K
    FT, 14
dr_Sigma
    Kerr, 17
dr_TeukRin
    FT, 14
dr_TeukRup
    FT, 14

E
    CEKG, 6

FT, 10
    Accuracy_in, 13
    Accuracy_up, 13
    Bin, 13
    Btrans, 13
    CalcRFields, 13
    choose_solvers, 13
    Ctrans, 14
    ddr_K, 14
    ddr_TeukRin, 14
    ddr_TeukRup, 14
    dr_K, 14
    dr_TeukRin, 14
    dr_TeukRup, 14
    get_ddrteuk, 14
    K, 15
    request_precision_in, 15
    request_precision_up, 15
    teuk_potential, 15
    TeukRin, 15
    TeukRup, 15

get_ddrteuk
    FT, 14
GKG, 16

K
    FT, 15
Kerr, 16
    ddr_Delta, 17
    ddr_Sigma, 17
    Delta, 17
    dr_Delta, 17
    dr_Sigma, 17
    rminus, 18
    rplus, 18
    rstar, 18
    Sigma, 18

Lz

CEKG, 6

Om_phi
CEKG, 6

Psi4
RRGW, 19

r
CEKG, 6
request_precision_in
FT, 15
request_precision_up
FT, 15
rminus
Kerr, 18
rplus
Kerr, 18
RRGW, 19
alpha_func, 19
Psi4, 19
Wave, 20
rstar
Kerr, 18

Sigma
Kerr, 18
sinthetabrac
Clebsch, 9
SWSH, 21
SWSH, 21

Tensor< TypeHere >, 22
teuk_potential
FT, 15
TeukRin
FT, 15
TeukRup
FT, 15
TidalH, 23

Wave
RRGW, 20