GRAND Library

Generated by Doxygen 1.8.20

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 AIRESEvent Struct Reference

Data Fields

- Runinfo runinfo
  run information
- Eventinfo eventinfo
  event information
- Showersiminfo showersiminfo
  shower simulation information
- Signalsiminfo signalsiminfo
  radio signal simulation information
- int n_ant
  Number of antennas.
- char * antennabuffer
  buffer holding the antenna information of each antenna
- Antennainfo * antennainfo
  Pointer to all antenna information structures.
- char * tracebuffer
  buffer holding all traces
- AntennaTrace * antennatrace
  Pointer to all antenna trace structures.
- char * antennaP2Pbuffer
  buffer holding all peak-to-peak data
- AntennaP2P * **antennap2p**
- ShowerTable showertable
  the shower table information

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

- AIRESevent.h

3.2 Antennainfo Struct Reference

Data Fields

- char ∗ ID

  identifier of antenna
- float ∗ pos

  (x,y,z) position of the antenna (m)
- float ∗ T0

  time offset (ns)
- double ∗ **SlopeA**
- double ∗ SlopeB

  I don't know what these slopes are (m)

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

- AIRESevent.h

3.3 AntennaP2P Struct Reference

Data Fields

- char ∗ ID

  identifier of the antenna
- float ∗ P2P_efield

  The P2P of E_tot, E_x, E_y, E_z (uV/m)
- float ∗ P2P_voltage

  The P2P of V_tot, V_x, V_y, V_z (uV)
- float ∗ P2P_filtered

  The P2P of the filtered V_tot, V_x, V_y, V_z (uV)
- float ∗ HilbertPeakE

  The peak of the Hilbert envelope of the E-field (uV/m)
- float ∗ HilbertPeakTimeE

  The time at which the Hilbert envelope of the E-field peaks (ns)
- float ∗ HilbertPeakV

  The peak of the Hilbert envelope of V_tot (uV)
- float ∗ HilbertPeakTimeV

  The time at which the Hilbert envelope of V_tot peaks (ns)
- float ∗ HilbertPeakVf

  The peak of the Hilbert envelope of the filtered V_tot (uV)
- float ∗ HilbertPeakTimeVf

  The time at which the Hilbert envelope of the filtered V_tot peaks (ns)

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

- AIRESevent.h

3.4 AntennaTrace Struct Reference

Data Fields

- int n_point

  number of points in the simulation
- float ∗ **Ex**
- float ∗ **Ey**
- float ∗ Ez

  time series E-field (uV/m)
- float ∗ **Vx**
- float ∗ **Vy**
- float ∗ Vz

  time series Voltage values (uV)
- float ∗ **Vfx**
- float ∗ **Vfy**
- float ∗ Vfz

  time series filtered voltages (uV)
- float ∗ time

  the array of time values (ns)

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

- AIRESevent.h

3.5 Eventinfo Struct Reference

Data Fields

- int size

  size of the Eventinfo table
- char ∗ buffer

  buffer in which the data is stored
- char ∗ EventName

  Name of the Aires event (same as in RunInfo)
- char ∗ EventID

  identifier of the Aires event (same as in Runinfo)
- char ∗ Primary

  Primary particle used in simulation (same as in RunInfo)
- double ∗ Energy

  Energy of the primary particle (EeV) (same as in RunInfo)
- double ∗ Zenith

  Zenith angle (degrees) in AIRES convention (in the direction the particle is moving) (same as in RunInfo)
- double ∗ Azimuth

  Azimuth angle (degrees) in AIRES convention (in the direction the particle is moving) (same as in RunInfo)
- double ∗ XmaxDistance

  Distance between the core position and Xmax (m) (same as in RunInfo)
- double ∗ XmaxPosition

Coordinates of the Xmax position (x,y,z)

- double * XmaxAltitude

  Altitude of Xmax above sea level (m)
- double * SlantXmax

  amout of atmosphere traversed by incoming particle and shower until Xmax (g/cm^2) (same as in RunInfo)
- double * InjectionAltitude

  Altitude above sea level of the injection of the primary particle (same as in RunInfo)
- double * GroundAltitude

  Altitude of the Earth surface above sea level (m)
- char * Site

  name of the site
- char * Date

  date of the simulated event (day/month/year, eg 13/May/2021)
- char * Latitude

  Latitude of the site in degrees.
- char * Longitude

  Longitude of the site in degrees.
- double * BField

  Magnitude of the B-field in uT.
- double * BFieldIncl

  Inclination B-field (degrees)
- double * BFieldDecl

  Declination B-field (degrees)
- char * AtmosphericModel

  Name of the atmospheric model used.
- char * AtmosphericModelParameters

  parameters set in the atmospheric model
- double * EnergyInNeutrinos

  amount of energy in shower-neutrinos (EeV)

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

- AIRESevent.h

3.6 GPS Struct Reference

Data Fields

- long Second

  GPS seconds (hardware or simulation date)
- unsigned int NanoSec

  GPS nanoseconds (hardware or simulation)

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

- GRANDevent.h

3.7 GRANDDetectorData Struct Reference

Data Fields

- int SimPoints

  trace length in simulation
- int RawPoints

  trace length in raw data
- GPS SimGPS

  simulated time of the start of the detector data
- GPS RawGPS

  raw time of the start of the detector data
- float TPulse

  offset between start and the peak of the signal
- double DetTime

  time of the peak of the signal relative to the time that the shower hit the surface
- double e_DetTime

  uncertainty on this peak time
- bool IsTriggered

  flag if this unit was triggered
- float SimMSPS

  simulated sampling speed (Mega Samples per Second)
- float RawMSPS

  raw sampling speed (Mega Samples per Second)
- float ∗ SimEfield [3]

  simulated time traces of the electric field (x,y,z) in uV/m
- float ∗ SimE_fftMag [3]

  magnitude of the FFT of the simulated E-field
- float ∗ SimE_fftPhase [3]

  phase of the FFT of the simulated E-field (radians)
- float ∗ SimVoltage [3]

  Voltage after applying the antenna model/electronics to the E-field (uV)
- float ∗ RawADC [3]

  time traces (x,y,z) raw ADC values (from electronics or digitizing and resampling the simulation)
- float ∗ RawVoltage [3]

  Voltage time traces after applying electronics description to ADC (uV)
- float ∗ RecEfield [3]

  time traces after applying inverting antenna description to voltage (x,y,z) (uV/m)
- float ∗ RecE_fftMag [3]

  magnitude of the FFT of the reconstructed E field (x,y,z)
- float ∗ RecE_fftPhase [3]

  phase of the FFT of the reconstructed E field (x,y,z) in radians

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

- GRANDevent.h

3.8 GRANDDetectorInfo Struct Reference

Data Fields

- char ID [30]

    identifier of the detector unit
- char DetectorModel [80]

    description of the hardware (eg antenna type, particle detector type, ..)
- char ElectronicsModel [80]

    description of the electronics used
- double Position [3]

    (x,y,z) of the detector unit in the field/simulation

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

- GRANDevent.h

3.9 GRANDEvent Struct Reference

Data Fields

- int n_det

    number of detectors in the event
- GRANDGenEventInfo GenEventInfo

    general information (site, atmosphere)
- GRANDSimEventInfo SimEventInfo

    simulation specific info
- GRANDRecEventInfo RecEventInfo

    reconstruction specific info
- GRANDDetectorInfo $*$ DetectorInfo

    detector information (type, location)
- float $*$ TraceBuffer

    buffer holding the data (length not a-priori known)
- GRANDDetectorData $*$ DetectorData

    all traces and related detector readout parameters

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

- GRANDevent.h

3.10 GRANDGenEventInfo Struct Reference

Data Fields

- char EventName [80]

  event name (simulation)
- char EventID [80]

  event ID simulation/raw data
- char Site [80]

  site for which the simulation was done/ at which data was taken
- char Date [80]

  date for which the simulation was done/at which data was taken
- double Latitude

  latitude of site (degrees)
- double Longitude

  longitude of site (degrees)
- double GroundAltitude

  altitude of site (m)
- double BField

  magnitude of B-field (uT)
- double BFieldIncl

  inclination B field (degrees)
- double BFieldDecl

  declination B-field (degrees)
- char AtmosphericModel [80]

  atmospheric model used in simulation/reconstruction
- char AtmosphericModelParameters [100]

  optional: parameters of the atmospheric model

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

- GRANDevent.h

3.11 GRANDRecEventInfo Struct Reference

Data Fields

- double **Energy**
- double e_Energy

  Shower energy and uncertainty (EeV)
- double **Zenith**
- double e_Zenith

  Shower zenith angle and uncertainty (degrees, 0=from above)
- double **Azimuth**
- double e_Azimuth

  Shower azimuth and uncertainty (degrees, 0=from North)
- double **Core** [3]
- double e_Core [3]

Shower core (x,y,z) at surface; defined by Z-coordinate (m)

- double **XmaxDistance**
- double e_XmaxDistance

  distance between core and Xmax (m)

- double **SlantXmax**
- double e_SlantXmax

  amount of atmosphere encountered by the shower/primary particle until Xmax (g/cm^2)

- GPS CoreTime

  time when the shower was at the core position

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

- GRANDevent.h

3.12 GRANDSimEventInfo Struct Reference

Data Fields

- char Primary [80]

  primary particle in simulation

- double Energy

  energy of primary particle (EeV)

- double Zenith

  zenith angle incoming particle (degrees, 0 = coming from above)

- double Azimuth

  azimuth angle primary particle (degrees, 0=coming from North)

- double XmaxDistance

  distance between Xmax and the core position

- double XmaxPosition [3]

  The Xmax coordinates (m) (x,y,z)

- double XmaxAltitude

  the altitude of Xmax (m) above the Earth

- double SlantXmax

  amount of atmosphere encountered by the shower/primary particle until Xmax (g/cm^2)

- double InjectionAltitude

  altitude above Earth (m) at which primary particle was injected

- double EnergyInNeutrinos

  amount of energy in Neutrinos (EeV)

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

- GRANDevent.h

3.13 LateralProfile Struct Reference

Data Fields

- float * Distance

  Distance from shower axis (m)
- float * Ngamma

  Number of photons.
- float * Ne_plus_minus

  Number of electrons+positrons.
- float * Ne_plus

  Number of positrons.
- float * Nmu_plus_minus

  Number of positively and negatively charged muons.
- float * Nmu_plus

  Number of positively charged muons.
- float * Nall_charged

  Number of charged particles.

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

- AIRESevent.h

3.14 LongProfile Struct Reference

Data Fields

- float * SlantDepth

  Amount of atmosphere traversed along the path of shower/primary (g/cm^2)
- float * VerticalDepth

  vetrical depth of Xmax from top of atmosphere (g/cm^2)
- float * Ngamma

  Number of photons.
- float * Ne_plus_minus

  Number of electrons+positrons.
- float * Ne_plus

  Number of positrons.
- float * Nmu_plus_minus

  Number of positively and negatively charged muons.
- float * Nmu_plus

  Number of positively charged muons.
- float * Npi_plus_minus

  Number of positively and negatively charged pions.
- float * Npi_plus

  Number of positively charged pions.
- float * Nall_charged

  Number of charged particles.

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

- AIRESevent.h

3.15 Runinfo Struct Reference

Data Fields

- int size

  size of the RunInfo Table
- char ∗ buffer

  Buffer in which the data is stored.
- char ∗ EventName

  Name of the Aires event (same as in EventInfo)
- char ∗ EventID

  Identifier of the Aires event (same as in EventInfo)
- char ∗ Primary

  Primary particle used in simulation (same as in EventInfo)
- double ∗ Energy

  Energy of the primary particle (EeV) (same as in EventInfo)
- double ∗ Zenith

  Zenith angle (degrees) in AIRES convention (in the direction the particle is moving) (same as in EventInfo)
- double ∗ Azimuth

  Azimuth angle (degrees) in AIRES convention (in the direction the particle is moving) (same as in EventInfo)
- double ∗ XmaxDistance

  Distance between the core position and Xmax (m) (same as in EventInfo)
- double ∗ SlantXmax

  amout of atmosphere traversed by incoming particle and shower until Xmax (g/cm^2) (same as in EventInfo)
- char ∗ HadronicModel

  hadronic interaction model used (same as in ShowersimInfo)
- double ∗ InjectionAltitude

  Altitude above sea level of the injection of the primary particle (same as in EventInfo)

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

- AIRESevent.h

3.16 Showersiminfo Struct Reference

Data Fields

- int size

  size of the Showersiminfo table
- char ∗ buffer

  buffer in which the data is stored
- char ∗ ShowerSimulator

  name and version of Aires
- char ∗ HadronicModel

  hadronic interaction model used (same as in RunInfo)
- char ∗ RandomSeed

  Seed used in simulation.
- char ∗ RelativeThinning

Thinning factor.

- double ∗ WeightFactor

  Wight factor.
- char ∗ GammaEnergyCut

  Minimal gamma energy (MeV)
- char ∗ ElectronEnergyCut

  Minimal electron energy (MeV)
- char ∗ MuonEnergyCut

  Minimal muon energy (MeV)
- char ∗ MesonEnergyCut

  Minimal meson energy (MeV)
- char ∗ NucleonEnergyCut

  Minimal nucleonenergy (MeV)
- double ∗ CPUTime

  CPU time used for simulation (s)
- char ∗ OtherParameters

  Potential other parameters.

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

- AIRESevent.h

3.17 ShowerTable Struct Reference

Data Fields

- int n_lateral

  number of lateral samples
- char ∗ lateralbuffer

  buffer containing the lateral profiles
- LateralProfile ∗ lateralprofile

  Pointer to the n_lateral profiles.
- int n_long

  number of longitudinal samples
- char ∗ longbuffer

  buffer containing the longitudinal profiles
- LongProfile ∗ longprofile

  Pointer to the n_long profiles.

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

- AIRESevent.h

3.18 Signalsiminfo Struct Reference

Data Fields

- int size

    Size of Signalsiminfo table.
- char ∗ buffer

    buffer in which the data is stored
- char ∗ FieldSimulator

    program used to simulate the electric field
- char ∗ RefractionIndexModel

    model simulating the refractive index in air
- char ∗ RefractionIndexModelParameters

    parameters of the model
- double ∗ TimeBinSize

    size of the time steps (ns)
- double ∗ TimeWindowMin

    minimal time of the window (ns)
- double ∗ TimeWindowMax

    maximal time of window (ns)
- char ∗ OtherParameters

    placeholder

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

- AIRESevent.h

Chapter 4

File Documentation

4.1 aires_util.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "hdf5.h"
#include "AIRESevent.h"
#include "GRANDevent.h"
```

Macros

- #define **GRANDMSPS** 500

Functions

- void convert_aires_GRAND (AIRESEvent ∗Aires, GRANDEvent ∗GRAND)
- void clear_aires_event (AIRESEvent ∗event)
- int read_runinfo (hid_t file, AIRESEvent ∗event)
- int read_antennainfo (hid_t file, AIRESEvent ∗event)
- int read_antennatraces (hid_t file, AIRESEvent ∗event)
- int read_p2p (hid_t file, AIRESEvent ∗event)
- int read_showertables (hid_t file, AIRESEvent ∗event)
- int read_signalsiminfo (hid_t file, AIRESEvent ∗event)
- int read_eventinfo (hid_t file, AIRESEvent ∗event)
- int read_showersiminfo (hid_t file, AIRESEvent ∗event)
- int read_aires_event (char ∗fname, AIRESEvent ∗event)

4.1.1 Detailed Description

This file contains the input of AIRES events and the conversion to GRAND. It requires hdf5. For use outside the library are the routines read_aires_event, clear_aires_event and convert_aires_GRAND

4.1.2 Function Documentation

4.1.2.1 clear_aires_event()

```
void clear_aires_event (
            AIRESEvent * event )
```

release all the allocated memories connected to an Aires event and set all elements to zero

Parameters

in	*event*	the AIRES event to be cleared

4.1.2.2 convert_aires_GRAND()

```
void convert_aires_GRAND (
            AIRESEvent * Aires,
            GRANDEvent * GRAND )
```

Convert an ZHAires event (as given in the files from Matias) to a GRAND event

Parameters

in	*Aires*	the ZHaires event
in	*GRAND*	the GRAND event in which the AIRES information is copied

4.1.2.3 read_aires_event()

```
int read_aires_event (
            char * fname,
            AIRESEvent * event )
```

read in the complete Aires event. This is the only read-in routine you need as it calls the othes

Parameters

in	*file*	the hdf5 file
in	*event*	the AIRES event in which the info is stored

4.1.2.4 read_antennainfo()

```
int read_antennainfo (
            hid_t file,
            AIRESEvent * event )
```

read in the antennainfo table from an Aires event file

Parameters

in	*file*	the hdf5 file
in	*event*	the AIRES event in which the info is stored

4.1.2.5 read_antennatraces()

```
int read_antennatraces (
            hid_t file,
            AIRESEvent * event )
```

read in the antenna traces group from an Aires event file. This contains the e-field, voltages and filtered voltages for all simulated antennas

Parameters

in	*file*	the hdf5 file
in	*event*	the AIRES event in which the info is stored

4.1.2.6 read_eventinfo()

```
int read_eventinfo (
            hid_t file,
            AIRESEvent * event )
```

read in the eventinfo table from an Aires event file. General event information (angles, particle type, xmax, etc)

Parameters

in	*file*	the hdf5 file
in	*event*	the AIRES event in which the info is stored

4.1.2.7 read_p2p()

```
int read_p2p (
```

```
            hid_t file,
            AIRESEvent * event )
```

read in the p2p (peak-to-peak) info table from an Aires event file

Parameters

in	*file*	the hdf5 file
in	*event*	the AIRES event in which the info is stored

4.1.2.8 read_runinfo()

```
int read_runinfo (
            hid_t file,
            AIRESEvent * event )
```

read in the runinfo table from an Aires event file

Parameters

in	*file*	the hdf5 file
in	*event*	the AIRES event in which the runinfo is stored

4.1.2.9 read_showersiminfo()

```
int read_showersiminfo (
            hid_t file,
            AIRESEvent * event )
```

read in the showersiminfo table from an Aires event file. Parameters dictating the shower development

Parameters

in	*file*	the hdf5 file
in	*event*	the AIRES event in which the info is stored

4.1.2.10 read_showertables()

```
int read_showertables (
            hid_t file,
            AIRESEvent * event )
```

readthe showertables group from an Aires event file. This contains the lateral and longitudinal profile of a simulated shower

Parameters

in	*file*	the hdf5 file
in	*event*	the AIRES event in which the info is stored

4.1.2.11 read_signalsiminfo()

```
int read_signalsiminfo (
            hid_t file,
            AIRESEvent * event )
```

read in the signalsiminfo table from an Aires event file. This provides parameters used in the simulation

Parameters

in	*file*	the hdf5 file
in	*event*	the AIRES event in which the info is stored

4.2 AIRESevent.h File Reference

Data Structures

- struct Runinfo
- struct Eventinfo
- struct Showersiminfo
- struct Signalsiminfo
- struct Antennainfo
- struct AntennaTrace
- struct AntennaP2P
- struct LateralProfile
- struct LongProfile
- struct ShowerTable
- struct AIRESEvent

4.3 antenna_util.c File Reference

```
#include <string.h>
#include "fftw3.h"
#include "fftdata.h"
#include "antenna.h"
#include "GRANDevent.h"
#include "matrix_util.h"
#include "complex_util.h"
```

Macros

- #define **RADDEG** 57.2957795131

Functions

- void apply_antenna (GRANDEvent *evnt, int iant, struct grand_antenna *antenna)
- void inverse_antenna (GRANDEvent *evnt, int iant, struct grand_antenna *antenna)

Variables

- int **fft_len**
- fftw_complex * **fftin**
- fftw_complex * **fftout**
- fftw_plan **fftpf**
- fftw_plan **fftpb**

4.3.1 Detailed Description

In this file we have routines to convert an electric field to voltage "apply antenna" and to go from voltage to electric field "invert_antenna" it requires the event structure from GRAND and the antenna definition External package fftw is required In addition, some calculations with complex numbers require the complex_util part of the GRAND library

4.3.2 Function Documentation

4.3.2.1 apply_antenna()

```
void apply_antenna (
            GRANDEvent * evnt,
            int iant,
            struct grand_antenna * antenna )
```

Applying an antenna model to a GRAND detector in an event. The result of this routine will be that:

In the detectorinfo the antennamodel and electronics model are set (now fixed to GP300 and GRANDProto_V 2)
For all 3 antenna arms an FFT of the simulated E-field is stored in SimE_fftmag and SimE_fftPhase
This is multiplied with the effective length of the antenna, using the simulated direction of the incoming primary particle
Next an inverse FFT is performed and the result is stored in the SimVoltage arrays for each antenna arm

Parameters

in	*evnt*	The GRAND event
in	*iant*	The index of the detectordata on which the antenna model will be applied
in	*antenna*	The antenna model

4.3.2.2 inverse_antenna()

```
void inverse_antenna (
            GRANDEvent * evnt,
            int iant,
            struct grand_antenna * antenna )
```

inverting an antenna model using voltage traces in a GRAND detector in an event. The result of this routine will be that:

In the detectorinfo the antennamodel and electronics model are set (now fixed to GP300 and GRANDProto_V 2)
For all 3 antenna arms an FFT of the raw voltages is created
For each frequency the effective length of all antenna arms is obtained, using the reconstructed zenith and azimuth angles.
For each frequency Matrix inversion is attempted to obtain the full 3D E-field from the 3 voltage values
If the matrix inversion fails, the 3D e-field is obtained from 2 voltage values combined with the demand that the E-field is perpendicular to the incoming shower
The result is stored in the RecE_fftPhase and RecE_fftMag arrays
Next an inverse FFT is performed and the result is stored in the RecEfield arrays for each antenna arm

Parameters

in	*evnt*	The GRAND event
in	*iant*	The index of the detectordata on which the antenna model will be applied
in	*antenna*	The antenna model

4.4 complex_util.c File Reference

```
#include "complex_util.h"
#include "math.h"
```

Functions

- fftw_complex ∗ c_multiply (fftw_complex a, fftw_complex b, fftw_complex ∗res)
- fftw_complex ∗ c_divide (fftw_complex a, fftw_complex b, fftw_complex ∗res)
- fftw_complex ∗ c_add (fftw_complex a, fftw_complex b, fftw_complex ∗res)
- fftw_complex ∗ c_sub (fftw_complex a, fftw_complex b, fftw_complex ∗res)
- double c_phase (fftw_complex a)
- double c_mag (fftw_complex a)
- fftw_complex ∗ c_pow (fftw_complex a, float pw, fftw_complex ∗res)

4.4.1 Detailed Description

some simple calculations using fftw_complex representation of complex numbers

4.4.2 Function Documentation

4.4.2.1 c_add()

```
fftw_complex* c_add (
            fftw_complex a,
            fftw_complex b,
            fftw_complex * res )
```

Adding complex numbers

Parameters

in	*a*	
in	*b*	
out	*res=a+b*	

4.4.2.2 c_divide()

```
fftw_complex* c_divide (
            fftw_complex a,
            fftw_complex b,
            fftw_complex * res )
```

Dividing complex numbers

Parameters

in	*a*	
in	*b*	
out	*res=a/b*	

4.4.2.3 c_mag()

```
double c_mag (
            fftw_complex a )
```

Getting the magnitude of a complex number

Parameters

in	*a*	

Returns

|a|

4.4.2.4 c_multiply()

```
fftw_complex* c_multiply (
            fftw_complex a,
            fftw_complex b,
            fftw_complex * res )
```

Multiplying complex numbers

Parameters

in	*a*	
in	*b*	
out	*res=a*b*	

4.4.2.5 c_phase()

```
double c_phase (
            fftw_complex a )
```

Getting the angle of a complex number in the complex plane

Parameters

in	*a*	

Returns

angle in radians

4.4.2.6 c_pow()

```
fftw_complex* c_pow (
            fftw_complex a,
            float pw,
            fftw_complex * res )
```

raising a complex number to an arbitrary power

Parameters

in	*a*	the complex number
in	*pw*	the power
out	*res=a^pw*	

4.4.2.7 c_sub()

```
fftw_complex* c_sub (
            fftw_complex a,
            fftw_complex b,
            fftw_complex * res )
```

Subtracting complex numbers

Parameters

in	*a*	
in	*b*	
out	*res=a-b*	

4.5 fft_util.c File Reference

```
#include <string.h>
#include "math.h"
#include "fftw3.h"
```

Functions

- void fft_init (int len)
- void fft_forward (float *input, float *output)
- void fft_backward (float *input, float *output)
- void mag_and_phase (float *in, float *out_mag, float *out_phase)
- void trace_from_mag_phase (float *in_mag, float *in_phase, float *out_trace)
- void envelope (float *in, float *out)
- void filter_data (float *in, float *out, float freqsample, float freqmin, float freqmax)

Variables

- fftw_plan **fftpf**
- fftw_plan **fftpb**
- fftw_complex * **fftin** =NULL
- fftw_complex * **fftout** =NULL
- int **fft_len** =0
- int **iswap** =0
- short * **datbuf** =NULL

4.5.1 Detailed Description

This file contains routines that aid implementation of fftw functionality in GRAND

4.5.2 Function Documentation

4.5.2.1 envelope()

```
void envelope (
            float * in,
            float * out )
```

create a Hilbert Envelope of a time series

Parameters

in	*in*	input time series (float)
out	*out*	Hilbert Envelope (float)

4.5.2.2 fft_backward()

```
void fft_backward (
            float * input,
            float * output )
```

perform a backward FFT (frequency to time)

Parameters

in	*input*	the frequency series (complex number as 2 floats)
out	*output*	the time series (complex number as 2 floats)

4.5.2.3 fft_forward()

```
void fft_forward (
            float * input,
            float * output )
```

perform a forward FFT (time to frequency)

Parameters

in	*input*	the time series (complex number as 2 floats)
out	*output*	the frequency series (complex number as 2 floats)

4.5.2.4 fft_init()

```
void fft_init (
            int len )
```

Initialize the fft. If fftw was already initialized, first memory is released.

Parameters

in	*len*	the length of the trace(s) to be Fourier transformed

4.5.2.5 filter_data()

```
void filter_data (
            float * in,
            float * out,
            float freqsample,
            float freqmin,
            float freqmax )
```

block filter the data

Parameters

in	*in*	input time series (float)
in	*freqsample*	sampling frequency
in	*freqmin*	all frequencies below freqmin are blocked
in	*freqmax*	all frequencies above freqmax are blocked
out	*out*	filtered time series (float)

4.5.2.6 mag_and_phase()

```
void mag_and_phase (
            float * in,
            float * out_mag,
            float * out_phase )
```

perform a forward FFT (time to frequency)

Parameters

in	*in*	the time series (only real values!!)
out	*out_mag*	magnitudes of the frequency series
out	*out_phase*	phases of the frequency series

4.5.2.7 trace_from_mag_phase()

```
void trace_from_mag_phase (
            float * in_mag,
            float * in_phase,
            float * out_trace )
```

perform a backward FFT (frequency to time)

Parameters

in	*in_mag*	magnitudes of the frequency series
in	*in_phase*	phases of the frequency series
out	*out_trace*	the time series (only real part)

4.6 GRANDevent.h File Reference

Data Structures

- struct GPS
- struct GRANDGenEventInfo
- struct GRANDSimEventInfo
- struct GRANDRecEventInfo
- struct GRANDDetectorInfo
- struct GRANDDetectorData
- struct GRANDEvent

4.7 hardware_util.c File Reference

```
#include "GRANDevent.h"
#include "TRandom2.h"
```

Macros

- #define **GRANDMSPS** 500
- #define **THRESADC** 100
- #define **GPSRES** 10
- #define **GIGA** 1000000000

Functions

- void apply_hardware (GRANDEvent *evnt, int iant)

4.7.1 Detailed Description

This file contains an implementation of the hardware (electronics except antenna) The ROOT external package is required for random number generation

4.7.2 Function Documentation

4.7.2.1 apply_hardware()

```
void apply_hardware (
            GRANDEvent * evnt,
            int iant )
```

go from simulated voltages to raw ADC and voltage values. At this moment there is no proper trigger simulation, nor does it read the hardware configuration. It is a very basic "digitization" based upon the sampling rate and bit depth of the ADC. The trigger is met when the maximal value is larger than a hardcoded threshold.

Parameters

in	*evnt*	the event
in	*iant*	the detector number

4.8 io_util.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include "GRANDevent.h"
#include "hdf5.h"
```

Functions

- int write_GRANDevent (GRANDEvent *event)

4.8.1 Detailed Description

In this file we should have input and output routines for GRAND events. Right now, only an output routine to hdf5 is written. The development of this file depends on the agreed file-format in GRAND.

4.8.2 Function Documentation

4.8.2.1 write_GRANDevent()

```
int write_GRANDevent (
            GRANDEvent * event )
```

writing a GRAND event into a hdf5 file. The name of the hdf5 file is determined from the event name.

Parameters

in	*event*	GRAND event

4.9 matrix.c File Reference

```
#include "complex_util.h"
```

Functions

- float invert_matrix (float in[3][3], float out[3][3])
- float c_invert_matrix (fftw_complex in[3][3], fftw_complex out[3][3])
- void c_matrix_times_vector (fftw_complex mat[3][3], fftw_complex vec[3], fftw_complex *result)

4.9.1 Detailed Description

This file contains routines that aid matrix manipulation

4.9.2 Function Documentation

4.9.2.1 c_invert_matrix()

```
float c_invert_matrix (
            fftw_complex in[3][3],
            fftw_complex out[3][3] )
```

Invert a complex 3x3 matrix

Parameters

in	*in*	original matrix
out	*out*	inverted matrix

4.9.2.2 c_matrix_times_vector()

```
void c_matrix_times_vector (
            fftw_complex mat[3][3],
            fftw_complex vec[3],
            fftw_complex * result )
```

Multiply a complex 3x3 matrix with a 3D complex vector

Parameters

in	*mat*	matrix
in	*vec*	input vector
out	*result*	resulting vector

4.9.2.3 invert_matrix()

```
float invert_matrix (
            float in[3][3],
            float out[3][3] )
```

Invert a real 3x3 matrix

Parameters

in	*in*	original matrix
out	*out*	inverted matrix

4.10 reco_util.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "TROOT.h"
#include "TFile.h"
#include "TTree.h"
#include "TClass.h"
```

```
#include "TSystem.h"
#include "TH1.h"
#include "TH2.h"
#include "TMinuit.h"
#include "TCanvas.h"
#include "TStyle.h"
#include "TProfile.h"
#include "TProfile2D.h"
#include "TRandom3.h"
#include "GRANDevent.h"
```

Macros

- #define **SPLIGHT** 0.299792458
- #define **PI** 3.141592653589793
- #define **RAD_TO_DEG** 57.295779513082321
- #define **GIGA** 1000000000

Functions

- double distance (double ∗p1, double ∗p2)
- double calcopang (double phi1, double theta1, double phi2, double theta2)
- double calcdt (GPS gps1, GPS gps2)
- void calc_uv (double ∗phi, double ∗theta)
- void fplane (Int_t &npar, Double_t ∗gin, Double_t &f, Double_t ∗par, Int_t iflag)
- void fcn (Int_t &npar, Double_t ∗gin, Double_t &f, Double_t ∗par, Int_t iflag)
- int reconstruct_core ()
- int reconstruct_event (GRANDEvent ∗evt)

Variables

- GRANDEvent ∗ **event**

4.10.1 Detailed Description

This file contains a geometrical reconstruction to be used in GRAND. Most routines are not to be used by the user. Only reconstruct_event is really for external use. The ROOT external package is required

4.10.2 Function Documentation

4.10.2.1 calc_uv()

```
void calc_uv (
            double * phi,
            double * theta )
```

provides an analytical least squares solution for the zenith and azimuth angle for an event, assuming all detectors are at the same altitude. Routine intended for internal use only. It uses the positions and pulse times of the individual triggered detectors as input

Parameters

out	*phi*	azimuth (radians)
out	*theta*	zenith (radians)

4.10.2.2 calcdt()

```
double calcdt (
            GPS gps1,
            GPS gps2 )
```

calculate the time difference (in nanoseconds) between 2 GPS time stamps. Routine intended for internal use only

Parameters

in	*gps1*	
in	*gps2*	

Returns

gps1-gps2 (nanoseconds)

4.10.2.3 calcopang()

```
double calcopang (
            double phi1,
            double theta1,
            double phi2,
            double theta2 )
```

Calculate the opening angle between 2 vectors,each given as 2 angles. Routine intended for internal use only

Parameters

in	*phi1*	azimuth first vector (radians)
in	*theta1*	zenith first vector (radians)
in	*phi2*	azimuth second vector (radians)
in	*theta2*	zenith second vector (radians)

Returns

opening angle in radians

4.10.2.4 distance()

```
double distance (
            double * p1,
            double * p2 )
```

Calculate the distance between 2 point in 3D space. Routine intended for internal use only

Parameters

in	*p1*	point 1
in	*p2*	point 2

Returns

distance between the points

4.10.2.5 fcn()

```
void fcn (
            Int_t & npar,
            Double_t * gin,
            Double_t & f,
            Double_t * par,
            Int_t iflag )
```

function used in the ROOT implementation of Minuit to fit an expanding sphere using the peak times and the locations of all triggered detectors

4.10.2.6 fplane()

```
void fplane (
            Int_t & npar,
            Double_t * gin,
            Double_t & f,
            Double_t * par,
            Int_t iflag )
```

function used in the ROOT implementation of Minuit to fit a plane wave using the peak times and the locations of all triggered detectors

4.10.2.7 reconstruct_core()

```
int reconstruct_core ( )
```

calculate the core position through the weighted average of the raw voltages
Next obtain the nearest detector to the core and use its timing as the core time
Finally calculate the time difference wrt the core time for each detector

4.10.2.8 reconstruct_event()

```
int reconstruct_event (
            GRANDEvent * evt )
```

Perform a geometrical reconstruction of the event as follows:

1. reconstruct the core position; this also sets the time of each station wrt the core. This is used in:
2. analytical plane wave calculation to get initial guesses of the zenith and azimuth. Start values for
3. minuit plane wave fit, which uses the core and core timing as a reference for all. The result is a start for
4. minuit expanding sphere fit as follows:
   a. fit distance to xmax (leaving all others fixed)
   b. fit the zenith and azimuth, fixing all others
   c. fit the core position, fixing all others
   d. fit core,direction and distance
5. Record all reconstructed parameters in the event; set the proper core time and time differences

Index