Tesseroids 1.0: User manual and API documentation

Generated by Doxygen 1.6.3

Wed Feb 16 12:22:26 2011

Contents

1 Welcolme

1.1 About

Tesseroids is a software package for the direct modeling of gravitational fields in spherical coordinates.

It can model the gravitational potential, acceleration and gradient tensor. The geometric element used in the modelling processes is a **spherical prism**, also called a **tesseroid**.

Tesseroids is coded in the C programming language, making it portable to GNU/Linux and Windows systems.

This software is developed by Leonardo Uieda in cooperation with Carla Braitenberg.

For more information on the theoretical aspects of the computations, see the Theoretical background.

1.2 Download

Tesseroids can be downloaded from our `Google Code` project site as both precompiled binaries and source code.

1.3 License

1.4 Contact

In case of doubts, suggestions, help, etc., contact Leonardo Uieda: `leouieda@gmail.com`

2 API Documentation

The Application Programming Interface (API) of is divided into separate .c and .h files containing functions and data types. The .h files contain the function and data type declarations, while corresponding

.c files contain the implementations. The files destined to be executables only have .c files, usually only implementing a "main" function.

3 User manual

3.1 About

This documentation explains how to install and run Tesseroids. It also describes the theoretical aspects of the computations and describes the Application Programming Interface (API).

4 Installation

4.1 Binary distribution

If you downloaded a pre-compiled binary distribution, simply unpack in the desired directory. The executables will be in the "bin" folder and the HTML and PDF documentation in the "doc" folder. To view the HTML docs open "index.html" from the "html" folder in a web browser.

4.2 From source

Tesseroids uses the build tool SCons. A SConstruct file (Makefile equivalent) is used to define the compilation rules. You will have to download and install SCons in order to easily compile Tesseroids. SCons is available for both GNU/Linux and Windows and building should work the same on both platforms. SCons requires that you have Python installed. Check the SCons website for more information. Python is usually installed by default on most GNU/Linux systems. Under Windows you will have to put SCons on your PATH environment variable in order to use it from the command line. It is usually located in the Scripts directory of your Python installation.

On GNU/Linux SCons will use the GCC compiler to compile sources. On Windows it will search for an existing compiler. We recomment that you install GCC on Windows using MinGW.

To compile, type in a terminal (cmd.exe on Windows):

```
scons
```

The executables are placed on a "bin" folder.

4.3 Testing

The source code for the unit tests for Tesseroids are in the "test" folder. Tesseroids uses a modified version of the MinUnit unit testing framework. When compiling the source code with SCons, the unit tests will be automatically compiled into a test program called "tesstest", placed in the "bin" folder. To run the tests, executed "tesstest".

4.4 Compiling the documentation

Tesseroids uses Doxygen to generate the documentation. You will need Doxygen installed as well as make. If you want to compile the PDF documentation you will also need Latex installed. make comes pre-installed on most GNU/Linux systems.

On GNU/Linux run the command "make" from the "doc" folder to generate the HTML and Latex code for the documentation. On Windows, run "make win". To compile the Latex code, go to "doc/build/latex" and run "make". To view the HTML documentation open "doc/build/html/index.html" in an internet browser. The Latex documentation is compiled to "doc/build/latex/refman.pdf".

5 Usage

5.1 A note about heights and units

In order to have a single convention, the word "height" means "height above the Earths surface" and are interpreted as positive up and negative down, ie. oriented with the z axis of the Local coordinate system.

Also, all input units are in SI and decimal degrees. Output of tessg$*$ programs are in mGal and Eotvos. All other output is also in SI and decimal degrees.

5.2 Getting help information

All programs accept the -h and --version flags. -h will print a help message describing the usage, input and output formats and options accepted. --verbose prints version and license information about the program.

5.3 Computing the gravitational effect of a tesseroid

The tessgx, tessgy, tessgz, tessgxx, etc. programs calculate the combined effect of a list of tesseroids on given computation points. The computation points are passed via standard input and do NOT have to be in a regular grid. This allows, for example, computation on points where data was measured. The values calulated are put in the last column of the input points and printed to standard output.

For example, if calculating g_z on these points:

```
lon1 lat1 height1 value1 othervalue1
lon2 lat2 height2 value2 othervalue2
...
lonN latN heightN valueN othervalueN
```

the output would look something like:

```
lon1 lat1 height1 value1 othervalue1 gz1
lon2 lat2 height2 value2 othervalue2 gz2
...
lonN latN heightN valueN othervalueN gzN
```

The input model file should contain one tesseroid per line and have columns formated as:

```
W E S N HEIGHT_OF_TOP HEIGHT_OF_BOTTOM DENSITY
```

HEIGHT_OF_TOP and HEIGHT_OF_BOTTOM are positive if the above the Earth's surface and negavitive if bellow. Remember that HEIGHT_OF_TOP $>$ HEIGHT_OF_BOTTOM!

Use the command line option -h to view a list of all commands available.

5.3.1 Example:

Calculate the g_z field of a tesseroid model having verbose printed and logged to file "gz.log" and GLQ order 3/3/3.

```
tessgz modelfile.txt -v -lgz.log -o3/3/3 < points.txt > gz_data.txt
```

5.3.2 The -a flag

The -a flag on tessg$*$ programs enables the automatic re-sizing of tesseroids when it is needed to maintain the GLQ precision desired. As a general rule, the tesseroid should be no bigger than it's distance from the computation point. Using this flag breaks the tesseroids automatically when this criterion is breached. This means that the computations can be performed with order 2/2/2 (default) which is much faster and still maintain correctness. Some preliminary tests show that using the -a flag with order 2/2/2 is up to 5 times faster than increasing the GLQ order. **It is strongly recommended using this flag and 2/2/2 order always**.

5.4 Verbose and logging to files

The -v flag enables printing of information messages to stderr. If ommited, only error messages will appear.

The -l flag enables logging of information and error messages to a file.

5.5 Comments and provenance information

Comments can be inserted into input files by placing a "#" character at the start of a line. All comment lines are ignored. tessg$*$ programs print the comment lines of the input to standard output.

All programs insert comments about the provenance of their results (where they came from) to their output. These include names of input files, version of program used, date, etc.

5.6 Generating a regular grid

Included in the package is program "tessgrd" which creates a regular grid of points and prints them to standard output.

5.6.1 Example

```
tessgrd -r-10/10/-10/10 -b100/100 -z250e03 -v > points.txt
```

5.7 Automated model generation

Tesseroids 1.0 includes a new program called "tessmodgen" for automatically generating a tesseroid model from a map of an interface. The interface can be any surface deviating from a reference level. For example, topography (a DEM) deviates from 0, a Moho map deviates from a mean crustal thickness, etc.

This program takes as input a REGULAR grid with longitude, latitude and height values of the interface. Each tesseroid is generated with a grid point at the center of it's top face. The top and bottom faces of the tesseroid are defined as:

- Top = Interface and Bottom = Reference: if the interface is above the reference
- Top = Reference and Bottom = Interface: if the interface is bellow the reference

The density ρ of the tesseroids can be passed using the -d option. This will asign a density value of ρ when the interface is above the reference and a value of $-\rho$ if the interface is bellow the reference.

Alternatively, the density of each tesseroid can be passed as a forth column on the input grid. **As with the -d option, if the interface is bellow the reference, the density value will be multiplied by -1!** Also, an error will occur if both a forth column and the -d option are passed!

5.7.1 Example:

To generate a tesseroid model from a Digital Elevation Model (DEM) with 1° resolution using a density $\rho = 2.67\ g.cm^{-3}$:

```
tessmodgen -s1/1 -d2670 -z0 -v < dem_file.txt > dem_tess_model.txt
```

5.8 Calculating the total mass of a model

The tessmass program can be used to compute the total mass of a given tesseroid model. If desired, a density range can be given and only tesseroids that fall within the given range will be used in the calculation.

5.8.1 Example:

To calculate the total mass of all tesseroids in "model.txt" with density between 0 and 1 g.cm^-3 :

```
tessmass -r0/1000 < model.txt
```

5.9 Computing the gravitational effect of a rectangular prism

Tesseroids 1.0 also introduces programs to calculate the gravitational effect of rectangular prisms in Cartesian coordinates. This is done using the formulas of Nagy *et al.* (2000). The programs are name prismgx, prismgy, prismgz, prismgxx, etc.

Input and output for these programs is very similar to that of the tessg$*$ programs. Computation points are read from standard input and the prism model is read from a file. The model file should have the column format:

```
X1 X2 Y1 Y2 Z1 Z2 DENSITY
```

5.9.1 A note on the coordinate system

As in Nagy *et al.* (2000), the coordinate system for the rectangular prism calculations has X axis pointing North, Y axis pointing East and Z axis pointing **Down**. This is important to note because it differs from the convention adopted for the tesseroids. In practice, this means that the g_{xz} and g_{yz} components of the prism and tesseroid will have different signs. This will not be such for the g_z component, though, because the convention for tesseroids is to have Z axis Down for this component only. See the Theoretical background section for more details on this.

5.10 Piping

Tesseroids was designed with the Unix filosophy in mind:

```
Write programs that do one thing and do it well.
Write programs to work together.
Write programs to handle text streams, because that is a universal interface.
```

Therefore, all tessg$*$ and tessgrd programs can be piped together to calculate many components on a regular grid.

5.10.1 Example

Given a tesseroids file "model.txt" as follows:

```
-5 5 -5 5 0 -10e03 -500
```

Running the following would calculate g_z and gradient tensor of tesseroids in "model.txt" of a regular grid from -10W to 10E and -10S to 10N on 100x100 points at 250 km height. And the best of all is that it is done in parallel! If your system has multiple cores this would mean a great increase in the computation time.

```
tessgrd -r-10/10/-10/10 -b100/100 -z250e03 | tessgz model.txt -a | \
tessgxx model.txt -a | tessgxy model.txt -a | tessgxz model.txt -a | \
tessgyy model.txt -a | tessgyz model.txt -a | tessgzz model.txt -a > output.txt
```

The result of this should look something like:

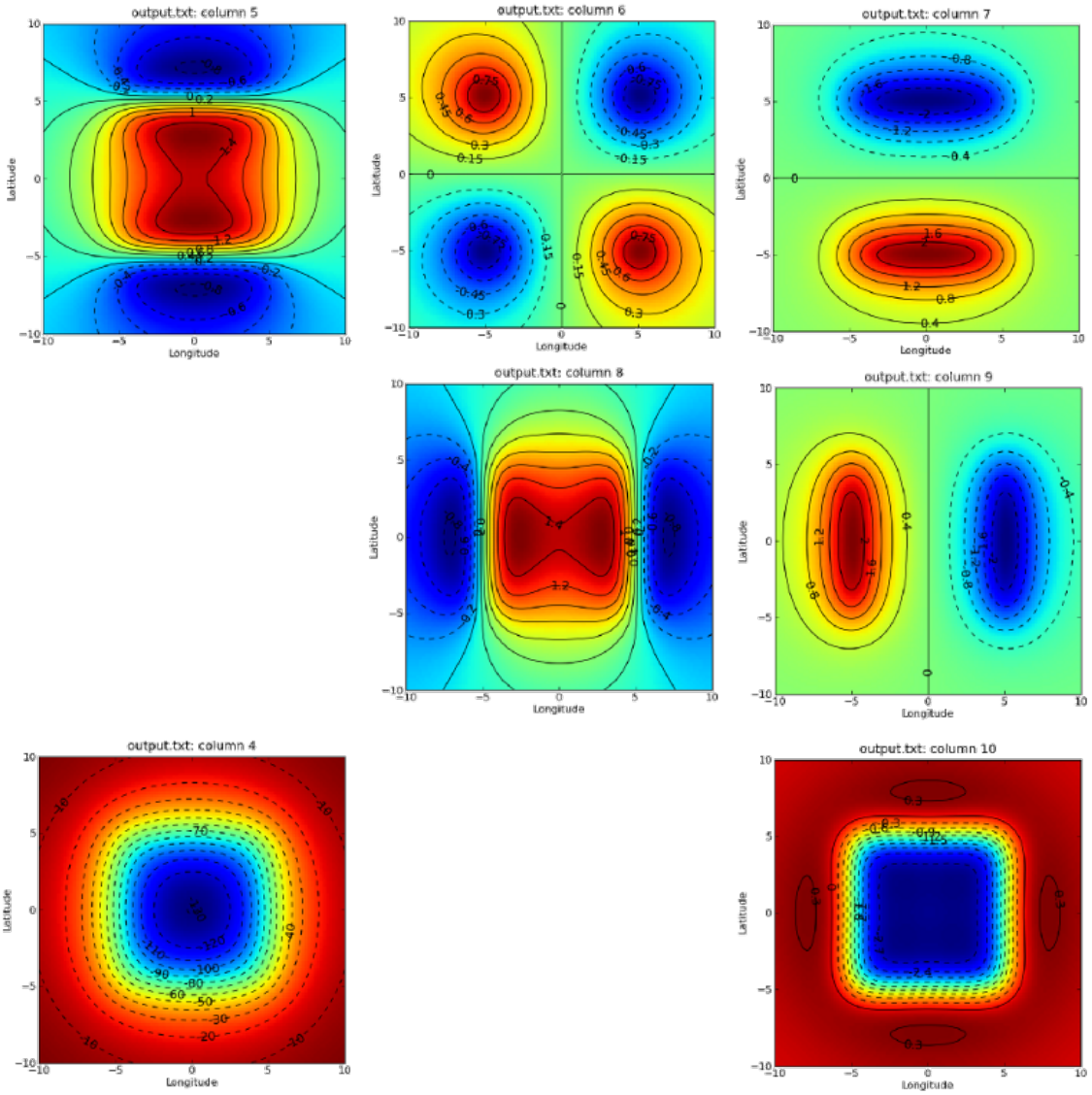

Figure 1: Plot of columns in result file output.txt (values in mGal and Eotvos).

5.11 References

- Nagy, D., Papp, G., Benedek, J. [2000]. The gravitational potential and its derivatives for the prism. Journal of Geodesy, 74, 552–560.

5.12 Theoretical background

5.12.1 About Coordinate Systems

The two coordinate systems involved in the computations are the Global and Local coordinate systems.

The Global system has origin on the center of the Earth and Z axis aligned with the Earth's mean rotation axis. The X and Y axis are contained on the equatorial parallel with X intercepting the mean Greenwich

meridian and Y completing a right-handed system.

The Local system has origin on the computation point. It's z is oriented along the radial direction and points away from the center of the Earth. The x and y axis are contained on a plane normal to the z axis and x points North and y East.

The tesseroids are defined using the Global Coordinate system with spherical coordinates, while the gravitational fields are calculated on the Local Coordinate system of the computation point.

WARNING: The g_z component is an exception to this. In order to conform with the regular convention of z-axis pointing toward the center of the Earth, this component ONLY is calculated with an inverted z axis.

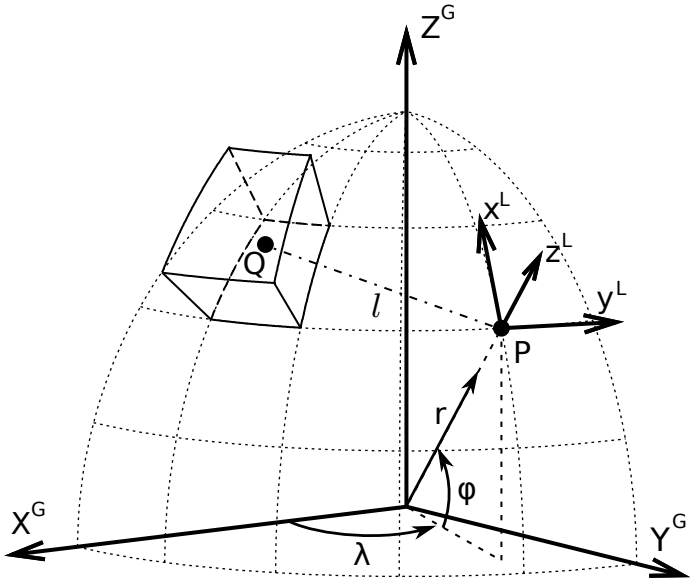

Figure 2: View of a tesseroid, the integration point Q, the global coordinate system, the computation P and it's local coordinate system.

5.12.2 Gravitational Fields of a Tesseroid

The gravitational attraction of a tesseroid can be calculated using the formula:

$$g_\alpha(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{\Delta x_\alpha}{\ell^3} \kappa \, dr' d\phi' d\lambda' \quad \alpha \in \{1, 2, 3\}$$

The gravity gradients can be calculated using the general formula:

$$g_{\alpha\beta}(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} I_{\alpha\beta} \, dr' d\phi' d\lambda' \quad \alpha, \beta \in \{1, 2, 3\}$$

$$I_{\alpha\beta} = \left(\frac{3\Delta x_\alpha \Delta x_\beta}{\ell^5} - \frac{\delta_{\alpha\beta}}{\ell^3} \right) \kappa \quad \alpha, \beta \in \{1, 2, 3\}$$

where ρ is density, the subscripts 1, 2, and 3 should be interpreted as the x, y, and z axis, $\delta_{\alpha\beta}$ is the Kronecker delta function, and

$$
\begin{aligned}
\Delta x_1 &= r' K_\phi \\
\Delta x_2 &= r' \cos\phi' \sin(\lambda' - \lambda_p) \\
\Delta x_3 &= r' \cos\psi - r_p \\
\ell &= \sqrt{r'^2 + r_p^2 - 2 r' r_p \cos\psi} \\
\cos\psi &= \sin\phi_p \sin\phi' + \cos\phi_p \cos\phi' \cos(\lambda' - \lambda_p) \\
K_\phi &= \cos\phi_p \sin\phi' - \sin\phi_p \cos\phi' \cos(\lambda' - \lambda_p) \\
\kappa &= {r'}^2 \cos\phi'
\end{aligned}
$$

ϕ is latitude, λ is longitude, r is radius. The subscript p is for the computation point.

5.12.3 Numerical Integration

The above integrals are solved using the Gauss-Legendre Quadrature rule:

$$g_{\alpha\beta}(r_p, \phi_p, \lambda_p) \approx G\rho \frac{(\lambda_2 - \lambda_1)(\phi_2 - \phi_1)(r_2 - r_1)}{8} \sum_{k=0}^{N^\lambda - 1} \sum_{j=0}^{N^\phi - 1} \sum_{i=0}^{N^r - 1} W_i^r W_j^\phi W_k^\lambda I_{\alpha\beta}({r'}_i, {\phi'}_j, {\lambda'}_k)\kappa \;\; \alpha, \beta \in \{1, 2, 3\}$$

where W^r,W^ϕ, and W^λ are weighting coefficients and N^r,N^ϕ, and N^λ are the number of quadrature nodes, ie the order of the quadrature.

6 Todo List

Global gets_prism(const char *str, PRISM *prism) Catch wrong order of model inputs, ie. x1 > x2 etc

Read the position of the prism from the string

Global gets_tess(const char *str, TESSEROID *tess) Catch wrong order of model inputs, ie. w > e or s > n or top < bottom

File glq.h Put reference for formulas

File grav_prism.h Include formulas in function descriptions

Put reference for formulas

Unroll loops in gx and gy

File grav_sphere.h Possible speedup: Replace sphere.rc with a local copy

Put reference for formulas

File grav_tess.h Possible speed up: use pointers for weights and nodes

Put reference for formulas

Allow for tesseroids with depth varying density

File prismg_main.h Option for calculating on spherical coordinates

Catch errors in input points that occur at the end of the line

Global tess2prism(TESSEROID tess, PRISM *prism) Put reference for formulas

File tessg_main.h Catch errors in input points that occur at the end of the line

File tessgrd.c Catch wrong order of -r arguments ie. w > e or s > n

File TODO.h Check error in not rotating prism

Check error os using tesseroid in poles

Programs to calculate the effect of a sphere model in spherical coordinates

Make minunit into functions and put variable arguments for messages like printf

Make doxygen groups to separate programs from api

Generate VTK file to plot tesseroids in Mayavi2 or Paraview

7 Data Structure Documentation

7.1 BASIC_ARGS Struct Reference

Store basic input arguments and option flags.

`#include <cmd.h>`

Data Fields

- char * inputfname

  name of the input file

- int verbose

  flag to indicate if verbose printing is enabled

- int logtofile

  flag to indicate if logging to a file is enabled

- char * logfname

  name of the log file

7.1.1 Detailed Description

Store basic input arguments and option flags.

7.2 GLQ Struct Reference

Store the nodes and weights needed for a GLQ integration.

`#include <glq.h>`

Data Fields

- int order
  order of the quadrature, ie number of nodes
- double * nodes
  abscissas or discretization points of the quadrature
- double * weights
  weighting coefficients of the quadrature
- double * nodes_unscaled
  nodes in [-1,1] interval

7.2.1 Detailed Description

Store the nodes and weights needed for a GLQ integration.

7.3 LOGGER Struct Reference

Keep the information on the global logger.

`#include <logger.h>`

Data Fields

- int level
  level of logging
- int filelogging
  flag to know wether loggint to a file is enabled
- int file_level
  logging level for the file
- FILE * logfile
  file to log to

7.3.1 Detailed Description

Keep the information on the global logger.

7.4 PRISM Struct Reference

Store information on a rectangular prism.

`#include <utils.h>`

Data Fields

- double density
  in SI units
- double x1
  in SI units
- double x2
  in SI units
- double y1
  in SI units
- double y2
  in SI units
- double z1
  in SI units
- double z2
  in SI units

7.4.1 Detailed Description

Store information on a rectangular prism.

7.5 SPHERE Struct Reference

Store information on a sphere.

`#include <utils.h>`

Data Fields

- double density
  in SI units

- double r
  radius of the sphere in SI units
- double lonc
  longitude of the center of the sphere in degrees
- double latc
  latitude of the center of the sphere in degrees
- double rc
  radial coordinate of the center of the sphere in SI units

7.5.1 Detailed Description

Store information on a sphere.

7.6 TESSEROID Struct Reference

Store information on a tesseroid.

`#include <utils.h>`

Data Fields

- double density
  in SI units
- double w
  western longitude border in degrees
- double e
  eastern longitude border in degrees
- double s
  southern latitude border in degrees
- double n
  northern latitude border in degrees
- double r1
  smallest radius border in SI units
- double r2
  largest radius border in SI units

7.6.1 Detailed Description

Store information on a tesseroid.

7.7 TESSG_ARGS Struct Reference

Store input arguments and option flags for tessg∗ programs.

`#include <cmd.h>`

Data Fields

- int lon_order
  glq order in longitude integration
- int lat_order
  glq order in latitude integration
- int r_order
  glq order in radial integration
- char ∗ modelfname
  name of the file with the tesseroid model
- int verbose
  flag to indicate if verbose printing is enabled
- int logtofile
  flag to indicate if logging to a file is enabled
- char ∗ logfname
  name of the log file
- int adaptative
  flat to indicate wether to use the adaptative size of tesseroid algorithm

7.7.1 Detailed Description

Store input arguments and option flags for tessg∗ programs.

7.8 TESSGRD_ARGS Struct Reference

Store input arguments and option flags for tessgrd program.

`#include <cmd.h>`

Data Fields

- double w
  western border of the grid
- double e
  eastern border of the grid

- double s
  southern border of the grid
- double n
  northern border of the grid
- int nlon
  number of grid points in the longitudinal direction
- int nlat
  number of grid points in the latitudinal direction
- double height
  height above geoid of the grid
- int verbose
  flag to indicate if verbose printing is enabled
- int logtofile
  flag to indicate if logging to a file is enabled
- char * logfname
  name of the log file

7.8.1 Detailed Description

Store input arguments and option flags for tessgrd program.

7.9 TESSMASS_ARGS Struct Reference

Store input arguments and option flags for tessmass program.

`#include <cmd.h>`

Data Fields

- char * inputfname
  name of the input file
- int verbose
  flag to indicate if verbose printing is enabled
- int logtofile
  flag to indicate if logging to a file is enabled
- char * logfname
  name of the log file

- int use_range
  flag to indicate wether to use a density range or not
- double low_dens
  lower bound for density range
- double high_dens
  upper bound for density range

7.9.1 Detailed Description

Store input arguments and option flags for tessmass program.

7.10 TESSMODGEN_ARGS Struct Reference

Store input arguments and option flags for tessmodgen program.

`#include <cmd.h>`

Data Fields

- int verbose
  flag to indicate if verbose printing is enabled
- int logtofile
  flag to indicate if logging to a file is enabled
- char * logfname
  name of the log file
- double dlon
  grid spacing in longitude
- double dlat
  grid spacing in latitude
- double ref
  depth of the reference level
- double dens
  density of the tesseroids
- int fix_density
  flag to tell wether using value passed by -d

7.10.1 Detailed Description

Store input arguments and option flags for tessmodgen program.

8 File Documentation

8.1 doc/apidocs.h File Reference

API documentation summary.

8.1.1 Detailed Description

API documentation summary.

8.2 doc/mainpage.h File Reference

Main page of the documentation.

8.2.1 Detailed Description

Main page of the documentation.

8.3 doc/userman.h File Reference

User manual sumary.

8.3.1 Detailed Description

User manual sumary.

8.4 doc/userman_instal.h File Reference

User manual: Installation.

8.4.1 Detailed Description

User manual: Installation.

8.5 doc/userman_theory.h File Reference

User manual: Theoretical background.

8.5.1 Detailed Description

User manual: Theoretical background.

8.6 doc/userman_usage.h File Reference

User manual: Usage.

8.6.1 Detailed Description

User manual: Usage.

8.7 src/c/cmd.c File Reference

Command line parsing tools.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "logger.h"
```

Functions

- int parse_basic_args (int argc, char **argv, const char *progname, BASIC_ARGS *args, void(*print_help)(void))

  Parse basic command line arguments for programs.

- int parse_tessmass_args (int argc, char **argv, const char *progname, TESSMASS_ARGS *args, void(*print_help)(void))

  Parse command line arguments for tessmass program.

- int parse_tessmodgen_args (int argc, char **argv, const char *progname, TESSMODGEN_ARGS *args, void(*print_help)(void))

  Parse command line arguments for tessmodgen program.

- int parse_tessg_args (int argc, char **argv, const char *progname, TESSG_ARGS *args)

  Parse command line arguments for tessg programs.*

- int parse_tessgrd_args (int argc, char **argv, TESSGRD_ARGS *args)

  Parse command line arguments for tessgrd program.

- void print_tessg_help (const char *progname)

  Print the help message for tessg programs.*

- void print_tessgrd_help ()

  Print the help message for tessmkgrd program.

8.7.1 Detailed Description

Command line parsing tools.

Author

Leonardo Uieda

Date

02 Feb 2011

8.7.2 Function Documentation

8.7.2.1 int parse_basic_args (int *argc*, char ** *argv*, const char * *progname*, BASIC_ARGS * *args*, void(*)(void) *print_help*)

Parse basic command line arguments for programs.

Basic arguments are: -h (for help msg), -v (for verbose), -l (for log file), --version and an input file.

Parameters

argc number of command line arguments

argv command line arguments

progname name of the specific program

args to return the parsed arguments

print_help pointer to a function that prints the help message for the program

Returns

Return code:

- 0: if all went well
- 1: if there were bad arguments and program should exit
- 2: if printed help or version info and program should exit
- 3: if input file was missing (doesn't log an error)

8.7.2.2 int parse_tessg_args (int *argc*, char ** *argv*, const char * *progname*, TESSG_ARGS * *args*)

Parse command line arguments for tessg* programs.

logs the bad argument warnings using logger.h

Parameters

argc number of command line arguments

argv command line arguments

progname name of the specific program

args to return the parsed arguments

Returns

Return code:

- 0: if all went well
- 1: if there were bad arguments and program should exit
- 2: if printed help or version info and program should exit

8.7.2.3 int parse_tessgrd_args (int *argc*, char ** *argv*, TESSGRD_ARGS * *args*)

Parse command line arguments for tessgrd program.

logs the bad argument warnings using logger.h

Parameters

argc number of command line arguments

argv command line arguments

args to return the parsed arguments

Returns

Return code:

- 0: if all went well
- 1: if there were bad arguments and program should exit
- 2: if printed help or version info and program should exit

8.7.2.4 int parse_tessmass_args (int *argc*, char ** *argv*, const char * *progname*, TESSMASS_ARGS * *args*, void(*)(void) *print_help*)

Parse command line arguments for tessmass program.

Parameters

argc number of command line arguments

argv command line arguments

progname name of the program

args to return the parsed arguments

print_help pointer to a function that prints the help message for the program

Returns

Return code:

- 0: if all went well
- 1: if there were bad arguments and program should exit
- 2: if printed help or version info and program should exit
- 3: if input file was missing (doesn't log an error)

8.7.2.5 int parse_tessmodgen_args (int *argc*, char ** *argv*, const char * *progname*, TESSMODGEN_ARGS * *args*, void(*)(void) *print_help*)

Parse command line arguments for tessmodgen program.

Parameters

argc number of command line arguments

argv command line arguments

progname name of the program

args to return the parsed arguments

print_help pointer to a function that prints the help message for the program

Returns

Return code:

- 0: if all went well
- 1: if there were bad arguments and program should exit
- 2: if printed help or version info and program should exit

8.7.2.6 void print_tessg_help (const char * *progname*)

Print the help message for tessg* programs.

Parameters

progname name of the specific tessg* program

8.7.2.7 void print_tessgrd_help ()

Print the help message for tessmkgrd program.

Prints to stdout.

8.8 src/c/cmd.h File Reference

Command line parsing tools.

Data Structures

- struct BASIC_ARGS

  Store basic input arguments and option flags.

- struct TESSMASS_ARGS

Store input arguments and option flags for tessmass program.

- struct TESSMODGEN_ARGS

  Store input arguments and option flags for tessmodgen program.

- struct TESSG_ARGS

  Store input arguments and option flags for tessg∗ programs.

- struct TESSGRD_ARGS

  Store input arguments and option flags for tessgrd program.

Functions

- int parse_basic_args (int argc, char ∗∗argv, const char ∗progname, BASIC_ARGS ∗args, void(∗print_help)(void))

  Parse basic command line arguments for programs.

- int parse_tessmass_args (int argc, char ∗∗argv, const char ∗progname, TESSMASS_ARGS ∗args, void(∗print_help)(void))

  Parse command line arguments for tessmass program.

- int parse_tessmodgen_args (int argc, char ∗∗argv, const char ∗progname, TESSMODGEN_ARGS ∗args, void(∗print_help)(void))

  Parse command line arguments for tessmodgen program.

- int parse_tessg_args (int argc, char ∗∗argv, const char ∗progname, TESSG_ARGS ∗args)

  Parse command line arguments for tessg∗ programs.

- int parse_tessgrd_args (int argc, char ∗∗argv, TESSGRD_ARGS ∗args)

  Parse command line arguments for tessgrd program.

- void print_tessg_help (const char ∗progname)

  Print the help message for tessg∗ programs.

- void print_tessgrd_help ()

  Print the help message for tessmkgrd program.

8.8.1 Detailed Description

Command line parsing tools.

Author

Leonardo Uieda

Date

02 Feb 2011

8.8.2 Function Documentation

8.8.2.1 int parse_basic_args (int *argc*, char ** *argv*, const char * *progname*, BASIC_ARGS * *args*, void(*)(void) *print_help*)

Parse basic command line arguments for programs.

Basic arguments are: -h (for help msg), -v (for verbose), -l (for log file), --version and an input file.

Parameters

argc number of command line arguments

argv command line arguments

progname name of the specific program

args to return the parsed arguments

print_help pointer to a function that prints the help message for the program

Returns

Return code:

- 0: if all went well
- 1: if there were bad arguments and program should exit
- 2: if printed help or version info and program should exit
- 3: if input file was missing (doesn't log an error)

8.8.2.2 int parse_tessg_args (int *argc*, char ** *argv*, const char * *progname*, TESSG_ARGS * *args*)

Parse command line arguments for tessg* programs.

logs the bad argument warnings using logger.h

Parameters

argc number of command line arguments

argv command line arguments

progname name of the specific program

args to return the parsed arguments

Returns

Return code:

- 0: if all went well
- 1: if there were bad arguments and program should exit
- 2: if printed help or version info and program should exit

8.8.2.3 int parse_tessgrd_args (int *argc*, char ** *argv*, TESSGRD_ARGS * *args*)

Parse command line arguments for tessgrd program.

logs the bad argument warnings using logger.h

Parameters

argc number of command line arguments

argv command line arguments

args to return the parsed arguments

Returns

Return code:

- 0: if all went well
- 1: if there were bad arguments and program should exit
- 2: if printed help or version info and program should exit

8.8.2.4 int parse_tessmass_args (int *argc*, char ** *argv*, const char * *progname*, TESSMASS_ARGS * *args*, void(*)(void) *print_help*)

Parse command line arguments for tessmass program.

Parameters

argc number of command line arguments

argv command line arguments

progname name of the program

args to return the parsed arguments

print_help pointer to a function that prints the help message for the program

Returns

Return code:

- 0: if all went well
- 1: if there were bad arguments and program should exit
- 2: if printed help or version info and program should exit
- 3: if input file was missing (doesn't log an error)

8.8.2.5 int parse_tessmodgen_args (int *argc*, char ** *argv*, const char * *progname*, TESSMODGEN_ARGS * *args*, void(*)(void) *print_help*)

Parse command line arguments for tessmodgen program.

Parameters

argc number of command line arguments

argv command line arguments

progname name of the program

args to return the parsed arguments

print_help pointer to a function that prints the help message for the program

Returns

Return code:

- 0: if all went well
- 1: if there were bad arguments and program should exit
- 2: if printed help or version info and program should exit

8.8.2.6 void print_tessg_help (const char * *progname*)

Print the help message for tessg* programs.

Parameters

progname name of the specific tessg* program

8.8.2.7 void print_tessgrd_help ()

Print the help message for tessmkgrd program.

Prints to stdout.

8.9 src/c/constants.c File Reference

Define constants used, like the gravitational constant and unit conversions.

```
#include "constants.h"
```

Variables

- const double MEAN_EARTH_RADIUS = 6378137.0

  Mean Earth radius [m].

- const double G = 0.00000000006673

  The gravitational constant [$m^3 * kg^{-1} * s^{-1}$].

- const double SI2EOTVOS = 1000000000.0

  Conversion factor from SI units to Eotvos [$\frac{1}{s^2} = 10^9\ Eotvos$].

- const double SI2MGAL = 100000.0

  Conversion factor from SI units to mGal $[1\frac{m}{s^2} = 10^5\ mGal]$.

- const double PI = 3.14159265358979323846264338327 95

  Pi.

8.9.1 Detailed Description

Define constants used, like the gravitational constant and unit conversions. **All values are in SI units!**

Author

Leonardo Uieda

Date

24 Jan 2011

8.10 src/c/constants.h File Reference

Define constants used, like the gravitational constant and unit conversions.

Variables

- const double MEAN_EARTH_RADIUS

  Mean Earth radius $[m]$.

- const double G

  The gravitational constant $[m^3 * kg^{-1} * s^{-1}]$.

- const double SI2EOTVOS

  Conversion factor from SI units to Eotvos $[\frac{1}{s^2} = 10^9\ Eotvos]$.

- const double SI2MGAL

  Conversion factor from SI units to mGal $[1\frac{m}{s^2} = 10^5\ mGal]$.

- const double PI

  Pi.

8.10.1 Detailed Description

Define constants used, like the gravitational constant and unit conversions. Values are assigned in file constants.c

All values are in SI units!

Author

Leonardo Uieda

Date

24 Jan 2011

8.11 src/c/glq.c File Reference

Functions for implementing a Gauss-Legendre Quadrature numerical integration.

```
#include <stdlib.h>
#include <math.h>
#include "constants.h"
#include "utils.h"
#include <stdio.h>
#include "logger.h"
```

Defines

- #define **GLQ_ABS**(x) ((x) < 0 ? -1*(x) : (x))

Functions

- GLQ * glq_new (int order, double lower, double upper)
  Make a new GLQ structure and set all the parameters needed.
- void glq_free (GLQ *glq)
  Free the memory allocated to make a GLQ structure.
- int glq_nodes (int order, double *nodes)
  Calculates the GLQ nodes using glq_next_root.
- int glq_set_limits (double lower, double upper, GLQ *glq)
  *Put the GLQ nodes to the integration limits **IN PLACE**.*
- int glq_next_root (double initial, int root_index, int order, double *roots)
  Calculate the next Legendre polynomial root given the previous root found.
- int glq_weights (int order, double *nodes, double *weights)
  Calculates the weighting coefficients for the GLQ integration.

Variables

- const int GLQ_MAXIT = 1000
  Max iterations of the root-finder algorithm.
- const double GLQ_MAXERROR = 0.000000000000001
  Max error allowed for the root-finder algorithm.

8.11.1 Detailed Description

Functions for implementing a Gauss-Legendre Quadrature numerical integration.

Author

Leonardo Uieda

Date

24 Jan 2011

8.11.2 Function Documentation

8.11.2.1 void glq_free (GLQ * *glq*)

Free the memory allocated to make a GLQ structure.

Parameters

glq pointer to the allocated memory

8.11.2.2 GLQ* glq_new (int *order*, double *lower*, double *upper*)

Make a new GLQ structure and set all the parameters needed.

WARNING: Don't forget to free the memory malloced by this function using glq_free()!

Prints error and warning messages using the logging.h module.

Parameters

order order of the quadrature, ie number of nodes

lower lower integration limit

upper upper integration limit

Returns

GLQ data structure with the nodes and weights calculated. NULL if there was an error with allocation.

8.11.2.3 int glq_next_root (double *initial*, int *root_index*, int *order*, double * *roots*)

Calculate the next Legendre polynomial root given the previous root found.

Uses the root-finder algorithm of:

Barrera-Figueroa, V., Sosa-Pedroza, J. and López-Bonilla, J., 2006, "Multiple root finder algorithm for Legendre and Chebyshev polynomials via Newton's method", 2006, Annales mathematicae et Informaticae, 33, pp 3-13

Parameters

initial initial estimate of the next root. I recommend the use of $\cos\left(\pi\frac{(N-i-0.25)}{N+0.5}\right)$, where i is the index of the desired root

root_index index of the desired root, starting from 0

order order of the Legendre polynomial, ie number of roots.

roots array with the roots found so far. Will return the next root in roots[root_index], so make sure to malloc enough space.

Returns

Return code:

- 0: if everything went OK
- 1: if order is not valid
- 2: if root_index is not valid (negative)
- 3: if number of maximum iterations was reached when calculating the root. This usually means that the desired accuracy was not achieved. Default desired accuracy is GLQ_MAXERROR. Default maximum iterations is GLQ_MAXIT.

Compute the absolute value of x

8.11.2.4 int glq_nodes (int *order*, double * *nodes*)

Calculates the GLQ nodes using glq_next_root.

Nodes will be in the [-1,1] interval. To convert them to the integration limits use glq_scale_nodes

Parameters

order order of the quadrature, ie how many nodes. Must be >= 2.

nodes pre-allocated array to return the nodes.

Returns

Return code:

- 0: if everything went OK
- 1: if invalid order
- 2: if NULL pointer for nodes
- 3: if number of maximum iterations was reached when calculating the root. This usually means that the desired accuracy was not achieved. Default desired accuracy is GLQ_MAXERROR. Default maximum iterations is GLQ_MAXIT.

8.11.2.5 int glq_set_limits (double *lower*, double *upper*, GLQ * *glq*)

Put the GLQ nodes to the integration limits **IN PLACE**.

Will replace the values of glq.nodes with ones in the specified integration limits.

In case the GLQ structure was created with glq_new(), the integration limits can be reset using this function.

Parameters

lower lower integration limit

upper upper integration limit

glq pointer to a GLQ structure created with glq_new() and with all necessary memory allocated

Returns

Return code:

- 0: if everything went OK
- 1: if invalid order
- 2: if NULL pointer for nodes or nodes_unscaled

8.11.2.6 int glq_weights (int *order*, double * *nodes*, double * *weights*)

Calculates the weighting coefficients for the GLQ integration.

Parameters

order order of the quadrature, ie number of nodes and weights.

nodes array containing the GLQ nodes calculated by glq_nodes. **IMPORTANT**: needs the nodes in [-1,1] interval! Scaled nodes will result in wrong weights!

weights pre-allocated array to return the weights

Returns

Return code:

- 0: if everything went OK
- 1: if order is not valid
- 2: if nodes is a NULL pointer
- 3: if weights is a NULL pointer

8.12 src/c/glq.h File Reference

Functions for implementing a Gauss-Legendre Quadrature numerical integration.

Data Structures

- struct GLQ

  Store the nodes and weights needed for a GLQ integration.

Functions

- GLQ * glq_new (int order, double lower, double upper)

  Make a new GLQ structure and set all the parameters needed.

- void glq_free (GLQ *glq)

  Free the memory allocated to make a GLQ structure.

- int glq_set_limits (double lower, double upper, GLQ *glq)

  *Put the GLQ nodes to the integration limits **IN PLACE**.*

- int glq_nodes (int order, double *nodes)

  Calculates the GLQ nodes using glq_next_root.

- int glq_next_root (double initial, int root_index, int order, double *roots)

  Calculate the next Legendre polynomial root given the previous root found.

- int glq_weights (int order, double *nodes, double *weights)

  Calculates the weighting coefficients for the GLQ integration.

Variables

- const int GLQ_MAXIT

  Max iterations of the root-finder algorithm.

- const double GLQ_MAXERROR

  Max error allowed for the root-finder algorithm.

8.12.1 Detailed Description

Functions for implementing a Gauss-Legendre Quadrature numerical integration.

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\sum_{i=0}^{N-1} w_i f(x_i)$$

N is the order of the quadrature.

Usage example:

To integrate the cossine function from 0 to 90 degrees

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "src/c/glq.h"

int main(){
    // Create a new glq structure
    GLQ *glq;
    double result = 0, a = 0, b = 0.5*3.14;
    int i;

    glq = glq_new(5, a, b);

    if(glq == NULL){
        printf("malloc error");
        return 1;
    }
```

```
    // Calculate the integral
    for(i = 0; i < glq->order; i++)
        result += glq->weights[i]*cos(glq->nodes[i]);

    // Need to multiply by a scale factor of the integration limits
    result *= 0.5*(b - a);

    printf("Integral of cossine from 0 to 90 degrees = %lf\n", result);
    return 0;
}
```

Todo

Put reference for formulas

Author

Leonardo Uieda

Date

24 Jan 2011

8.12.2 Function Documentation

8.12.2.1 void glq_free (GLQ * *glq*)

Free the memory allocated to make a GLQ structure.

Parameters

glq pointer to the allocated memory

8.12.2.2 GLQ* glq_new (int *order*, double *lower*, double *upper*)

Make a new GLQ structure and set all the parameters needed.

WARNING: Don't forget to free the memory malloced by this function using glq_free()!

Prints error and warning messages using the logging.h module.

Parameters

order order of the quadrature, ie number of nodes

lower lower integration limit

upper upper integration limit

Returns

GLQ data structure with the nodes and weights calculated. NULL if there was an error with allocation.

8.12.2.3 int glq_next_root (double *initial*, int *root_index*, int *order*, double * *roots*)

Calculate the next Legendre polynomial root given the previous root found.

Uses the root-finder algorithm of:

Barrera-Figueroa, V., Sosa-Pedroza, J. and López-Bonilla, J., 2006, "Multiple root finder algorithm for Legendre and Chebyshev polynomials via Newton's method", 2006, Annales mathematicae et Informaticae, 33, pp 3-13

Parameters

initial initial estimate of the next root. I recommend the use of $\cos\left(\pi\frac{(N-i-0.25)}{N+0.5}\right)$, where i is the index of the desired root

root_index index of the desired root, starting from 0

order order of the Legendre polynomial, ie number of roots.

roots array with the roots found so far. Will return the next root in roots[root_index], so make sure to malloc enough space.

Returns

Return code:

- 0: if everything went OK
- 1: if order is not valid
- 2: if root_index is not valid (negative)
- 3: if number of maximum iterations was reached when calculating the root. This usually means that the desired accuracy was not achieved. Default desired accuracy is GLQ_MAXERROR. Default maximum iterations is GLQ_MAXIT.

Compute the absolute value of x

8.12.2.4 int glq_nodes (int *order*, double * *nodes*)

Calculates the GLQ nodes using glq_next_root.

Nodes will be in the [-1,1] interval. To convert them to the integration limits use glq_scale_nodes

Parameters

order order of the quadrature, ie how many nodes. Must be >= 2.

nodes pre-allocated array to return the nodes.

Returns

Return code:

- 0: if everything went OK
- 1: if invalid order
- 2: if NULL pointer for nodes
- 3: if number of maximum iterations was reached when calculating the root. This usually means that the desired accuracy was not achieved. Default desired accuracy is GLQ_MAXERROR. Default maximum iterations is GLQ_MAXIT.

8.12.2.5 int glq_set_limits (double *lower*, double *upper*, GLQ * *glq*)

Put the GLQ nodes to the integration limits **IN PLACE**.

Will replace the values of glq.nodes with ones in the specified integration limits.

In case the GLQ structure was created with glq_new(), the integration limits can be reset using this function.

Parameters

lower lower integration limit

upper upper integration limit

glq pointer to a GLQ structure created with glq_new() and with all necessary memory allocated

Returns

Return code:

- 0: if everything went OK
- 1: if invalid order
- 2: if NULL pointer for nodes or nodes_unscaled

8.12.2.6 int glq_weights (int *order*, double * *nodes*, double * *weights*)

Calculates the weighting coefficients for the GLQ integration.

Parameters

order order of the quadrature, ie number of nodes and weights.

nodes array containing the GLQ nodes calculated by glq_nodes. **IMPORTANT**: needs the nodes in [-1,1] interval! Scaled nodes will result in wrong weights!

weights pre-allocated array to return the weights

Returns

Return code:

- 0: if everything went OK
- 1: if order is not valid
- 2: if nodes is a NULL pointer
- 3: if weights is a NULL pointer

8.13 src/c/grav_prism.c File Reference

Functions that calculate the gravitational potential and its first and second derivatives for the rectangular prism.

```
#include <math.h>
#include "utils.h"
#include "constants.h"
#include "grav_prism.h"
```

Functions

- double prism_gx (PRISM prism, double xp, double yp, double zp)
  Calculates gx component caused by a right rectangular prism.
- double prism_gy (PRISM prism, double xp, double yp, double zp)
  Calculates gy component caused by a right rectangular prism.
- double prism_gz (PRISM prism, double xp, double yp, double zp)
  Calculates gz component caused by a right rectangular prism.
- double prism_gxx (PRISM prism, double xp, double yp, double zp)
  Calculates gxx component caused by a right rectangular prism.
- double prism_gxy (PRISM prism, double xp, double yp, double zp)
  Calculates gxy component caused by a right rectangular prism.
- double prism_gxz (PRISM prism, double xp, double yp, double zp)
  Calculates gxz component caused by a right rectangular prism.
- double prism_gyy (PRISM prism, double xp, double yp, double zp)
  Calculates gyy component caused by a right rectangular prism.
- double prism_gyz (PRISM prism, double xp, double yp, double zp)
  Calculates gyz component caused by a right rectangular prism.
- double prism_gzz (PRISM prism, double xp, double yp, double zp)
  Calculates gzz component caused by a right rectangular prism.

8.13.1 Detailed Description

Functions that calculate the gravitational potential and its first and second derivatives for the rectangular prism. Using the formulas in Nagy et al. (2000).

The coordinate system used is that of the article, ie:

x -> North y -> East z -> Down

Author

Leonardo Uieda

Date

01 March 2010

8.13.2 Function Documentation

8.13.2.1 double prism_gx (PRISM *prism*, double *xp*, double *yp*, double *zp*)

Calculates gx component caused by a right rectangular prism.

Input values in SI units and returns values in mGal!

Parameters

prism data structure describing the prism

xp x coordinate of the computation point

yp y coordinate of the computation point

zp z coordinate of the computation point

Returns

field calculated at P

8.13.2.2 double prism_gxx (PRISM *prism*, double *xp*, double *yp*, double *zp*)

Calculates gxx component caused by a right rectangular prism.

Input values in SI units and returns values in Eotvos!

Parameters

prism data structure describing the prism

xp x coordinate of the computation point

yp y coordinate of the computation point

zp z coordinate of the computation point

Returns

field calculated at P

8.13.2.3 double prism_gxy (PRISM *prism*, double *xp*, double *yp*, double *zp*)

Calculates gxy component caused by a right rectangular prism.

Input values in SI units and returns values in Eotvos!

Parameters

prism data structure describing the prism

xp x coordinate of the computation point

yp y coordinate of the computation point

zp z coordinate of the computation point

Returns

field calculated at P

8.13.2.4 double prism_gxz (PRISM *prism*, double *xp*, double *yp*, double *zp*)

Calculates gxz component caused by a right rectangular prism.

Input values in SI units and returns values in Eotvos!

Parameters

prism data structure describing the prism
xp x coordinate of the computation point
yp y coordinate of the computation point
zp z coordinate of the computation point

Returns

field calculated at P

8.13.2.5 double prism_gy (PRISM *prism*, double *xp*, double *yp*, double *zp*)

Calculates gy component caused by a right rectangular prism.

Input values in SI units and returns values in mGal!

Parameters

prism data structure describing the prism
xp x coordinate of the computation point
yp y coordinate of the computation point
zp z coordinate of the computation point

Returns

field calculated at P

8.13.2.6 double prism_gyy (PRISM *prism*, double *xp*, double *yp*, double *zp*)

Calculates gyy component caused by a right rectangular prism.

Input values in SI units and returns values in Eotvos!

Parameters

prism data structure describing the prism
xp x coordinate of the computation point
yp y coordinate of the computation point
zp z coordinate of the computation point

Returns

field calculated at P

8.13.2.7 double prism_gyz (PRISM *prism*, double *xp*, double *yp*, double *zp*)

Calculates gyz component caused by a right rectangular prism.

Input values in SI units and returns values in Eotvos!

Parameters

prism data structure describing the prism
xp x coordinate of the computation point
yp y coordinate of the computation point
zp z coordinate of the computation point

Returns

field calculated at P

8.13.2.8 double prism_gz (PRISM *prism*, double *xp*, double *yp*, double *zp*)

Calculates gz component caused by a right rectangular prism.

Input values in SI units and returns values in mGal!

Parameters

prism data structure describing the prism
xp x coordinate of the computation point
yp y coordinate of the computation point
zp z coordinate of the computation point

Returns

field calculated at P

8.13.2.9 double prism_gzz (PRISM *prism*, double *xp*, double *yp*, double *zp*)

Calculates gzz component caused by a right rectangular prism.

Input values in SI units and returns values in Eotvos!

Parameters

prism data structure describing the prism
xp x coordinate of the computation point
yp y coordinate of the computation point
zp z coordinate of the computation point

Returns

field calculated at P

8.14 src/c/grav_prism.h File Reference

Functions that calculate the gravitational potential and its first and second derivatives for the rectangular prism.

```
#include "utils.h"
```

Functions

- double prism_gx (PRISM prism, double xp, double yp, double zp)
  Calculates gx component caused by a right rectangular prism.
- double prism_gy (PRISM prism, double xp, double yp, double zp)
  Calculates gy component caused by a right rectangular prism.
- double prism_gz (PRISM prism, double xp, double yp, double zp)
  Calculates gz component caused by a right rectangular prism.
- double prism_gxx (PRISM prism, double xp, double yp, double zp)
  Calculates gxx component caused by a right rectangular prism.
- double prism_gxy (PRISM prism, double xp, double yp, double zp)
  Calculates gxy component caused by a right rectangular prism.
- double prism_gxz (PRISM prism, double xp, double yp, double zp)
  Calculates gxz component caused by a right rectangular prism.
- double prism_gyy (PRISM prism, double xp, double yp, double zp)
  Calculates gyy component caused by a right rectangular prism.
- double prism_gyz (PRISM prism, double xp, double yp, double zp)
  Calculates gyz component caused by a right rectangular prism.
- double prism_gzz (PRISM prism, double xp, double yp, double zp)
  Calculates gzz component caused by a right rectangular prism.

8.14.1 Detailed Description

Functions that calculate the gravitational potential and its first and second derivatives for the rectangular prism. Using the formulas in Nagy et al. (2000).

The coordinate system used is that of the article, ie:

x -> North y -> East z -> Down

Todo

Include formulas in function descriptions
Put reference for formulas
Unroll loops in gx and gy

Author

Leonardo Uieda

Date

01 March 2010

8.14.2 Function Documentation

8.14.2.1 double prism_gx (PRISM *prism*, double *xp*, double *yp*, double *zp*)

Calculates gx component caused by a right rectangular prism.

Input values in SI units and returns values in mGal!

Parameters

prism data structure describing the prism

xp x coordinate of the computation point

yp y coordinate of the computation point

zp z coordinate of the computation point

Returns

field calculated at P

8.14.2.2 double prism_gxx (PRISM *prism*, double *xp*, double *yp*, double *zp*)

Calculates gxx component caused by a right rectangular prism.

Input values in SI units and returns values in Eotvos!

Parameters

prism data structure describing the prism

xp x coordinate of the computation point

yp y coordinate of the computation point

zp z coordinate of the computation point

Returns

field calculated at P

8.14.2.3 double prism_gxy (PRISM *prism*, double *xp*, double *yp*, double *zp*)

Calculates gxy component caused by a right rectangular prism.

Input values in SI units and returns values in Eotvos!

Parameters

prism data structure describing the prism

xp x coordinate of the computation point

yp y coordinate of the computation point

zp z coordinate of the computation point

Returns

field calculated at P

8.14.2.4 double prism_gxz (PRISM *prism*, double *xp*, double *yp*, double *zp*)

Calculates gxz component caused by a right rectangular prism.

Input values in SI units and returns values in Eotvos!

Parameters

prism data structure describing the prism

xp x coordinate of the computation point

yp y coordinate of the computation point

zp z coordinate of the computation point

Returns

field calculated at P

8.14.2.5 double prism_gy (PRISM *prism*, double *xp*, double *yp*, double *zp*)

Calculates gy component caused by a right rectangular prism.

Input values in SI units and returns values in mGal!

Parameters

prism data structure describing the prism

xp x coordinate of the computation point

yp y coordinate of the computation point

zp z coordinate of the computation point

Returns

field calculated at P

8.14.2.6 double prism_gyy (PRISM *prism*, double *xp*, double *yp*, double *zp*)

Calculates gyy component caused by a right rectangular prism.

Input values in SI units and returns values in Eotvos!

Parameters

prism data structure describing the prism
xp x coordinate of the computation point
yp y coordinate of the computation point
zp z coordinate of the computation point

Returns

field calculated at P

8.14.2.7 double prism_gyz (PRISM *prism*, double *xp*, double *yp*, double *zp*)

Calculates gyz component caused by a right rectangular prism.

Input values in SI units and returns values in Eotvos!

Parameters

prism data structure describing the prism
xp x coordinate of the computation point
yp y coordinate of the computation point
zp z coordinate of the computation point

Returns

field calculated at P

8.14.2.8 double prism_gz (PRISM *prism*, double *xp*, double *yp*, double *zp*)

Calculates gz component caused by a right rectangular prism.

Input values in SI units and returns values in mGal!

Parameters

prism data structure describing the prism
xp x coordinate of the computation point
yp y coordinate of the computation point
zp z coordinate of the computation point

Returns

field calculated at P

8.14.2.9 double prism_gzz (PRISM *prism*, double *xp*, double *yp*, double *zp*)

Calculates gzz component caused by a right rectangular prism.

Input values in SI units and returns values in Eotvos!

Parameters

prism data structure describing the prism

xp x coordinate of the computation point

yp y coordinate of the computation point

zp z coordinate of the computation point

Returns

field calculated at P

8.15 src/c/grav_sphere.c File Reference

This module contains a set of functions that calculate the gravitational potential and its first and second derivatives for the sphere in spherical coordinates.

```
#include <math.h>
#include "utils.h"
#include "constants.h"
#include "grav_sphere.h"
```

Functions

- double sphere_gx (SPHERE sphere, double lonp, double latp, double rp)
  Calculates gx caused by a sphere.
- double sphere_gy (SPHERE sphere, double lonp, double latp, double rp)
  Calculates gy caused by a sphere.
- double sphere_gz (SPHERE sphere, double lonp, double latp, double rp)
  Calculates gz caused by a sphere.
- double sphere_gxx (SPHERE sphere, double lonp, double latp, double rp)
  Calculates gxx caused by a sphere.
- double sphere_gxy (SPHERE sphere, double lonp, double latp, double rp)
  Calculates gxy caused by a sphere.
- double sphere_gxz (SPHERE sphere, double lonp, double latp, double rp)
  Calculates gxz caused by a sphere.
- double sphere_gyy (SPHERE sphere, double lonp, double latp, double rp)
  Calculates gyy caused by a sphere.

- double sphere_gyz (SPHERE sphere, double lonp, double latp, double rp)
  Calculates gyz caused by a sphere.
- double sphere_gzz (SPHERE sphere, double lonp, double latp, double rp)
  Calculates gzz caused by a sphere.

8.15.1 Detailed Description

This module contains a set of functions that calculate the gravitational potential and its first and second derivatives for the sphere in spherical coordinates. The position of the sphere and computation point are in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system x->North, y->East, z->out. So it would be normal for a sphere of positive density to have negative gz

Author

Leonardo Uieda

Date

25 Jan 2011

8.15.2 Function Documentation

8.15.2.1 double sphere_gx (SPHERE *sphere*, double *lonp*, double *latp*, double *rp*)

Calculates gx caused by a sphere.

$$g_x(r_p, \phi_p, \lambda_p) = GM\frac{r_c K_\phi}{\ell^3}$$

The position of the sphere and computation point should be in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in mGal!

Parameters

sphere data structure describing the sphere

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

Returns

field calculated at P

8.15.2.2 double sphere_gxx (SPHERE *sphere*, double *lonp*, double *latp*, double *rp*)

Calculates gxx caused by a sphere.

$$g_{xx}(r_p, \phi_p, \lambda_p) = GM \frac{3(r_c K_\phi)^2 - \ell^2}{\ell^5}$$

The position of the sphere and computation point are in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Parameters

sphere data structure describing the sphere

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

Returns

field calculated at P

8.15.2.3 double sphere_gxy (SPHERE *sphere*, double *lonp*, double *latp*, double *rp*)

Calculates gxy caused by a sphere.

$$g_{xy}(r_p, \phi_p, \lambda_p) = GM \frac{3r_c^2 K_\phi \cos\phi_c \sin(\lambda_c - \lambda_p)}{\ell^5}$$

The position of the sphere and computation point are in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Parameters

sphere data structure describing the sphere

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

Returns

field calculated at P

8.15.2.4 double sphere_gxz (SPHERE *sphere*, double *lonp*, double *latp*, double *rp*)

Calculates gxz caused by a sphere.

$$g_{xz}(r_p, \phi_p, \lambda_p) = GM \frac{3 r_c K_\phi (r_c \cos\psi - r_p)}{\ell^5}$$

The position of the sphere and computation point are in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Parameters

sphere data structure describing the sphere

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

Returns

field calculated at P

8.15.2.5 double sphere_gy (SPHERE *sphere*, double *lonp*, double *latp*, double *rp*)

Calculates gy caused by a sphere.

$$g_y(r_p, \phi_p, \lambda_p) = GM \frac{r_c \cos\phi_c \sin(\phi_c - \phi_p)}{\ell^3}$$

The position of the sphere and computation point should be in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in mGal!

Parameters

sphere data structure describing the sphere

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

Returns

field calculated at P

8.15.2.6 double sphere_gyy (SPHERE *sphere*, double *lonp*, double *latp*, double *rp*)

Calculates gyy caused by a sphere.

$$g_{yy}(r_p, \phi_p, \lambda_p) = GM \frac{3(r_c \cos\phi_c \sin(\lambda_c - \lambda_p))^2 - \ell^2}{\ell^5}$$

The position of the sphere and computation point are in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Parameters

sphere data structure describing the sphere

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

Returns

field calculated at P

8.15.2.7 double sphere_gyz (SPHERE *sphere*, double *lonp*, double *latp*, double *rp*)

Calculates gyz caused by a sphere.

$$g_{yz}(r_p, \phi_p, \lambda_p) = GM \frac{3 r_c \cos\phi_c \sin(\lambda_c - \lambda_p)(r_c \cos\psi - r_p)}{\ell^5}$$

The position of the sphere and computation point are in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Parameters

sphere data structure describing the sphere

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

Returns

field calculated at P

8.15.2.8 double sphere_gz (SPHERE *sphere*, double *lonp*, double *latp*, double *rp*)

Calculates gz caused by a sphere.

$$g_z(r_p, \phi_p, \lambda_p) = GM \frac{r_c \cos\psi - r_p}{\ell^3}$$

The position of the sphere and computation point should be in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in mGal!

Parameters

sphere data structure describing the sphere

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

Returns

field calculated at P

8.15.2.9 double sphere_gzz (SPHERE *sphere*, double *lonp*, double *latp*, double *rp*)

Calculates gzz caused by a sphere.

$$g_{zz}(r_p, \phi_p, \lambda_p) = GM \frac{3(r_c \cos\psi - r_p)^2 - \ell^2}{\ell^5}$$

The position of the sphere and computation point are in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Parameters

sphere data structure describing the sphere

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

Returns

field calculated at P

8.16 src/c/grav_sphere.h File Reference

Functions that calculate the gravitational potential and its first and second derivatives for the sphere in spherical coordinates.

```
#include "utils.h"
```

Functions

- double sphere_gx (SPHERE sphere, double lonp, double latp, double rp)
  Calculates gx caused by a sphere.
- double sphere_gy (SPHERE sphere, double lonp, double latp, double rp)
  Calculates gy caused by a sphere.
- double sphere_gz (SPHERE sphere, double lonp, double latp, double rp)
  Calculates gz caused by a sphere.
- double sphere_gxx (SPHERE sphere, double lonp, double latp, double rp)
  Calculates gxx caused by a sphere.
- double sphere_gxy (SPHERE sphere, double lonp, double latp, double rp)
  Calculates gxy caused by a sphere.
- double sphere_gxz (SPHERE sphere, double lonp, double latp, double rp)
  Calculates gxz caused by a sphere.
- double sphere_gyy (SPHERE sphere, double lonp, double latp, double rp)
  Calculates gyy caused by a sphere.
- double sphere_gyz (SPHERE sphere, double lonp, double latp, double rp)
  Calculates gyz caused by a sphere.
- double sphere_gzz (SPHERE sphere, double lonp, double latp, double rp)
  Calculates gzz caused by a sphere.

8.16.1 Detailed Description

Functions that calculate the gravitational potential and its first and second derivatives for the sphere in spherical coordinates. The position of the sphere and computation point are in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**. So it would be normal for a sphere of positive density to have negative gz.

Used the generic formula for gravity gradient computation:

$$g_{ij}(r_p, \phi_p, \lambda_p) = GM \left(\frac{3\Delta x_i \Delta x_j}{\ell^5} - \frac{\delta_{ij}}{\ell^3} \right) \quad i, j \in \{1, 2, 3\}$$

where M is the mass of the sphere, the subscripts 1, 2, and 3 should be interpreted as the x, y, and z axis and

$$
\begin{aligned}
\Delta x_1 &= r_c K_\phi \\
\Delta x_2 &= r_c \cos\phi_c \sin(\lambda_c - \lambda_p) \\
\Delta x_3 &= r_c \cos\psi - r_p \\
\ell &= \sqrt{r_c^2 + r_p^2 - 2 r_c r_p \cos\psi} \\
\cos\psi &= \sin\phi_p \sin\phi_c + \cos\phi_p \cos\phi_c \cos(\lambda_c - \lambda_p) \\
K_\phi &= \cos\phi_p \sin\phi_c - \sin\phi_p \cos\phi_c \cos(\lambda_c - \lambda_p)
\end{aligned}
$$

ϕ is latitude, λ is longitude, r is radius. The subscript c is for the center of the sphere and p for the computation point.

Todo

Possible speedup: Replace sphere.rc with a local copy
Put reference for formulas

Author

Leonardo Uieda

Date

25 Jan 2011

8.16.2 Function Documentation

8.16.2.1 double sphere_gx (SPHERE *sphere*, double *lonp*, double *latp*, double *rp*)

Calculates gx caused by a sphere.

$$g_x(r_p, \phi_p, \lambda_p) = GM \frac{r_c K_\phi}{\ell^3}$$

The position of the sphere and computation point should be in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in mGal!

Parameters

sphere data structure describing the sphere

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

Returns

field calculated at P

8.16.2.2 double sphere_gxx (SPHERE *sphere*, double *lonp*, double *latp*, double *rp*)

Calculates gxx caused by a sphere.

$$g_{xx}(r_p, \phi_p, \lambda_p) = GM \frac{3(r_c K_\phi)^2 - \ell^2}{\ell^5}$$

The position of the sphere and computation point are in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Parameters

sphere data structure describing the sphere

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

Returns

field calculated at P

8.16.2.3 double sphere_gxy (SPHERE *sphere*, double *lonp*, double *latp*, double *rp*)

Calculates gxy caused by a sphere.

$$g_{xy}(r_p, \phi_p, \lambda_p) = GM \frac{3r_c^2 K_\phi \cos\phi_c \sin(\lambda_c - \lambda_p)}{\ell^5}$$

The position of the sphere and computation point are in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Parameters

sphere data structure describing the sphere

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

Returns

field calculated at P

8.16.2.4 double sphere_gxz (SPHERE *sphere*, double *lonp*, double *latp*, double *rp*)

Calculates gxz caused by a sphere.

$$g_{xz}(r_p, \phi_p, \lambda_p) = GM \frac{3 r_c K_\phi (r_c \cos\psi - r_p)}{\ell^5}$$

The position of the sphere and computation point are in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Parameters

sphere data structure describing the sphere

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

Returns

field calculated at P

8.16.2.5 double sphere_gy (SPHERE *sphere*, double *lonp*, double *latp*, double *rp*)

Calculates gy caused by a sphere.

$$g_y(r_p, \phi_p, \lambda_p) = GM \frac{r_c \cos\phi_c \sin(\phi_c - \phi_p)}{\ell^3}$$

The position of the sphere and computation point should be in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in mGal!

Parameters

sphere data structure describing the sphere

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

Returns

field calculated at P

8.16.2.6 double sphere_gyy (SPHERE *sphere*, double *lonp*, double *latp*, double *rp*)

Calculates gyy caused by a sphere.

$$g_{yy}(r_p, \phi_p, \lambda_p) = GM \frac{3(r_c \cos\phi_c \sin(\lambda_c - \lambda_p))^2 - \ell^2}{\ell^5}$$

The position of the sphere and computation point are in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Parameters

sphere data structure describing the sphere

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

Returns

field calculated at P

8.16.2.7 double sphere_gyz (SPHERE *sphere*, double *lonp*, double *latp*, double *rp*)

Calculates gyz caused by a sphere.

$$g_{yz}(r_p, \phi_p, \lambda_p) = GM \frac{3 r_c \cos\phi_c \sin(\lambda_c - \lambda_p)(r_c \cos\psi - r_p)}{\ell^5}$$

The position of the sphere and computation point are in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Parameters

sphere data structure describing the sphere

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

Returns

field calculated at P

8.16.2.8 double sphere_gz (SPHERE *sphere*, double *lonp*, double *latp*, double *rp*)

Calculates gz caused by a sphere.

$$g_z(r_p, \phi_p, \lambda_p) = GM \frac{r_c \cos\psi - r_p}{\ell^3}$$

The position of the sphere and computation point should be in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in mGal!

Parameters

sphere data structure describing the sphere

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

Returns

field calculated at P

8.16.2.9 double sphere_gzz (SPHERE *sphere*, double *lonp*, double *latp*, double *rp*)

Calculates gzz caused by a sphere.

$$g_{zz}(r_p, \phi_p, \lambda_p) = GM \frac{3(r_c \cos\psi - r_p)^2 - \ell^2}{\ell^5}$$

The position of the sphere and computation point are in spherical coordinates.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Parameters

sphere data structure describing the sphere

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

Returns

field calculated at P

8.17 src/c/grav_tess.c File Reference

Functions that calculate the gravitational potential and its first and second derivatives for the tesseroid.

```
#include <math.h>
#include "utils.h"
#include "glq.h"
#include "constants.h"
#include "grav_tess.h"
```

Functions

- double calc_tess_model (TESSEROID *model, int size, double lonp, double latp, double rp, GLQ *glq_lon, GLQ *glq_lat, GLQ *glq_r, double(*field)(TESSEROID, double, double, double, GLQ, GLQ, GLQ))

  Calculates the field of a tesseroid model at a given point.

- double calc_tess_model_adapt (TESSEROID *model, int size, double lonp, double latp, double rp, GLQ *glq_lon, GLQ *glq_lat, GLQ *glq_r, double(*field)(TESSEROID, double, double, double, GLQ, GLQ, GLQ))

  Adaptatively calculate the field of a tesseroid model at a given point by splitting the tesseroids if necessary to maintain GLQ stability.

- double tess_gx (TESSEROID tess, double lonp, double latp, double rp, GLQ glq_lon, GLQ glq_lat, GLQ glq_r)

  Calculates gx caused by a tesseroid.

- double tess_gy (TESSEROID tess, double lonp, double latp, double rp, GLQ glq_lon, GLQ glq_lat, GLQ glq_r)

  Calculates gy caused by a tesseroid.

- double tess_gz (TESSEROID tess, double lonp, double latp, double rp, GLQ glq_lon, GLQ glq_lat, GLQ glq_r)

  Calculates gz caused by a tesseroid.

- double tess_gxx (TESSEROID tess, double lonp, double latp, double rp, GLQ glq_lon, GLQ glq_lat, GLQ glq_r)

  Calculates gxx caused by a tesseroid.

- double tess_gxy (TESSEROID tess, double lonp, double latp, double rp, GLQ glq_lon, GLQ glq_lat, GLQ glq_r)

  Calculates gxy caused by a tesseroid.

- double tess_gxz (TESSEROID tess, double lonp, double latp, double rp, GLQ glq_lon, GLQ glq_lat, GLQ glq_r)

  Calculates gxz caused by a tesseroid.

- double tess_gyy (TESSEROID tess, double lonp, double latp, double rp, GLQ glq_lon, GLQ glq_lat, GLQ glq_r)

  Calculates gyy caused by a tesseroid.

- double tess_gyz (TESSEROID tess, double lonp, double latp, double rp, GLQ glq_lon, GLQ glq_lat, GLQ glq_r)

  Calculates gyz caused by a tesseroid.

- double tess_gzz (TESSEROID tess, double lonp, double latp, double rp, GLQ glq_lon, GLQ glq_lat, GLQ glq_r)

  Calculates gzz caused by a tesseroid.

8.17.1 Detailed Description

Functions that calculate the gravitational potential and its first and second derivatives for the tesseroid.

Author

Leonardo Uieda

Date

27 Jan 2011

8.17.2 Function Documentation

8.17.2.1 double calc_tess_model (TESSEROID * *model*, int *size*, double *lonp*, double *latp*, double *rp*, GLQ * *glq_lon*, GLQ * *glq_lat*, GLQ * *glq_r*, double(*)(TESSEROID, double, double, double, GLQ, GLQ, GLQ) *field*)

Calculates the field of a tesseroid model at a given point.

Uses a function pointer to call one of the apropriate field calculating functions:

- tess_gx()
- tess_gy()
- tess_gz()
- tess_gxx()
- tess_gxy()
- tess_gxz()
- tess_gyy()
- tess_gyz()
- tess_gzz()

To pass a function pointer to a function use something like:

```
calc_tess_model(my_model, 10, 0, 10, 1, glqlon, glqlat, glqr, &tess_gx);
```

This would calculate the gx effect of the model my_model with 10 tesseroids at lon=0 lat=10 r=1.

Will re-use the same GLQ structures, and therefore the **same order, for all the tesseroids**.

Parameters

model TESSEROID array defining the model

size number of tesseroids in the model

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon pointer to GLQ structure used for the longitudinal integration

glq_lat pointer to GLQ structure used for the latitudinal integration

glq_r pointer to GLQ structure used for the radial integration

field pointer to one of the field calculating functions

Returns

the sum of the fields of all the tesseroids in the model

8.17.2.2 double calc_tess_model_adapt (TESSEROID * *model*, int *size*, double *lonp*, double *latp*, double *rp*, GLQ * *glq_lon*, GLQ * *glq_lat*, GLQ * *glq_r*, double(*)(TESSEROID, double, double, double, GLQ, GLQ, GLQ) *field*)

Adaptatively calculate the field of a tesseroid model at a given point by splitting the tesseroids if necessary to maintain GLQ stability.

See calc_tess_model() for more details.

Will re-use the same GLQ structures, and therefore the **same order, for all the tesseroids**.

Parameters

model TESSEROID array defining the model

size number of tesseroids in the model

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon pointer to GLQ structure used for the longitudinal integration

glq_lat pointer to GLQ structure used for the latitudinal integration

glq_r pointer to GLQ structure used for the radial integration

field pointer to one of the field calculating functions

Returns

the sum of the fields of all the tesseroids in the model

8.17.2.3 double tess_gx (TESSEROID *tess*, double *lonp*, double *latp*, double *rp*, GLQ *glq_lon*, GLQ *glq_lat*, GLQ *glq_r*)

Calculates gx caused by a tesseroid.

$$g_x(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{r' K_\phi}{\ell^3} \kappa \, dr' d\phi' d\lambda'$$

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in mGal!

Use function glq_new() to create the GLQ parameters required. The integration limits should be set to:

- glq_lon: lower = tess.w and upper = tess.e (in degrees)
- glq_lat: lower = tess.s and upper = tess.n (in degrees)
- glq_r: lower = tess.r1 and upper = tess.r2

Parameters

tess data structure describing the tesseroid

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon GLQ structure with the nodes, weights and integration limits set for the longitudinal integration

glq_lat GLQ structure with the nodes, weights and integration limits set for the latitudinal integration

glq_r GLQ structure with the nodes, weights and integration limits set for the radial integration

Returns

field calculated at P

8.17.2.4 double tess_gxx (TESSEROID *tess*, double *lonp*, double *latp*, double *rp*, GLQ *glq_lon*, GLQ *glq_lat*, GLQ *glq_r*)

Calculates gxx caused by a tesseroid.

$$g_{xx}(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{3(r' K_\phi)^2 - \ell^2}{\ell^5} \kappa \, dr' d\phi' d\lambda'$$

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Use function glq_new() to create the GLQ parameters required. The integration limits should be set to:

- glq_lon: lower = tess.w and upper = tess.e (in degrees)
- glq_lat: lower = tess.s and upper = tess.n (in degrees)
- glq_r: lower = tess.r1 and upper = tess.r2

Parameters

tess data structure describing the tesseroid

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon GLQ structure with the nodes, weights and integration limits set for the longitudinal integration

glq_lat GLQ structure with the nodes, weights and integration limits set for the latitudinal integration

glq_r GLQ structure with the nodes, weights and integration limits set for the radial integration

Returns

field calculated at P

8.17.2.5 double tess_gxy (TESSEROID *tess*, double *lonp*, double *latp*, double *rp*, GLQ *glq_lon*, GLQ *glq_lat*, GLQ *glq_r*)

Calculates gxy caused by a tesseroid.

$$g_{xy}(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{3r'^2 K_\phi \cos\phi' \sin(\lambda' - \lambda_p)}{\ell^5} \kappa \, dr' d\phi' d\lambda'$$

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Use function glq_new() to create the GLQ parameters required. The integration limits should be set to:

- glq_lon: lower = tess.w and upper = tess.e (in degrees)
- glq_lat: lower = tess.s and upper = tess.n (in degrees)
- glq_r: lower = tess.r1 and upper = tess.r2

Parameters

tess data structure describing the tesseroid

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon GLQ structure with the nodes, weights and integration limits set for the longitudinal integration

glq_lat GLQ structure with the nodes, weights and integration limits set for the latitudinal integration

glq_r GLQ structure with the nodes, weights and integration limits set for the radial integration

Returns

field calculated at P

8.17.2.6 double tess_gxz (TESSEROID *tess*, double *lonp*, double *latp*, double *rp*, GLQ *glq_lon*, GLQ *glq_lat*, GLQ *glq_r*)

Calculates gxz caused by a tesseroid.

$$g_{xz}(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{3r' K_\phi (r' \cos\psi - r_p)}{\ell^5} \kappa \, dr' d\phi' d\lambda'$$

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Use function glq_new() to create the GLQ parameters required. The integration limits should be set to:

- glq_lon: lower = tess.w and upper = tess.e (in degrees)
- glq_lat: lower = tess.s and upper = tess.n (in degrees)
- glq_r: lower = tess.r1 and upper = tess.r2

Parameters

tess data structure describing the tesseroid

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon GLQ structure with the nodes, weights and integration limits set for the longitudinal integration

glq_lat GLQ structure with the nodes, weights and integration limits set for the latitudinal integration

glq_r GLQ structure with the nodes, weights and integration limits set for the radial integration

Returns

field calculated at P

8.17.2.7 double tess_gy (TESSEROID *tess*, double *lonp*, double *latp*, double *rp*, GLQ *glq_lon*, GLQ *glq_lat*, GLQ *glq_r*)

Calculates gy caused by a tesseroid.

$$g_y(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{r' \cos\phi' \sin(\lambda' - \lambda)}{\ell^3} \kappa \, dr' d\phi' d\lambda'$$

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in mGal!

Use function glq_new() to create the GLQ parameters required. The integration limits should be set to:

- glq_lon: lower = tess.w and upper = tess.e (in degrees)
- glq_lat: lower = tess.s and upper = tess.n (in degrees)
- glq_r: lower = tess.r1 and upper = tess.r2

Parameters

tess data structure describing the tesseroid

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon GLQ structure with the nodes, weights and integration limits set for the longitudinal integration

glq_lat GLQ structure with the nodes, weights and integration limits set for the latitudinal integration

glq_r GLQ structure with the nodes, weights and integration limits set for the radial integration

Returns

field calculated at P

8.17.2.8 double tess_gyy (TESSEROID *tess*, double *lonp*, double *latp*, double *rp*, GLQ *glq_lon*, GLQ *glq_lat*, GLQ *glq_r*)

Calculates gyy caused by a tesseroid.

$$g_{yy}(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{3(r' \cos\phi' \sin(\lambda' - \lambda_p))^2 - \ell^2}{\ell^5} \kappa \, dr' d\phi' d\lambda'$$

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Use function glq_new() to create the GLQ parameters required. The integration limits should be set to:

- glq_lon: lower = tess.w and upper = tess.e (in degrees)
- glq_lat: lower = tess.s and upper = tess.n (in degrees)

- glq_r: lower = tess.r1 and upper = tess.r2

Parameters

tess data structure describing the tesseroid

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon GLQ structure with the nodes, weights and integration limits set for the longitudinal integration

glq_lat GLQ structure with the nodes, weights and integration limits set for the latitudinal integration

glq_r GLQ structure with the nodes, weights and integration limits set for the radial integration

Returns

field calculated at P

8.17.2.9 double tess_gyz (TESSEROID *tess*, double *lonp*, double *latp*, double *rp*, GLQ *glq_lon*, GLQ *glq_lat*, GLQ *glq_r*)

Calculates gyz caused by a tesseroid.

$$g_{yz}(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{3r' \cos\phi' \sin(\lambda' - \lambda_p)(r' \cos\psi - r_p)}{\ell^5} \kappa \, dr' d\phi' d\lambda'$$

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Use function glq_new() to create the GLQ parameters required. The integration limits should be set to:

- glq_lon: lower = tess.w and upper = tess.e (in degrees)
- glq_lat: lower = tess.s and upper = tess.n (in degrees)
- glq_r: lower = tess.r1 and upper = tess.r2

Parameters

tess data structure describing the tesseroid

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon GLQ structure with the nodes, weights and integration limits set for the longitudinal integration

glq_lat GLQ structure with the nodes, weights and integration limits set for the latitudinal integration

glq_r GLQ structure with the nodes, weights and integration limits set for the radial integration

Returns

field calculated at P

8.17.2.10 double tess_gz (TESSEROID *tess*, double *lonp*, double *latp*, double *rp*, GLQ *glq_lon*, GLQ *glq_lat*, GLQ *glq_r*)

Calculates gz caused by a tesseroid.

$$g_z(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{r' \cos\psi - r_p}{\ell^3} \kappa \, dr' d\phi' d\lambda'$$

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in mGal!

Use function glq_new() to create the GLQ parameters required. The integration limits should be set to:

- glq_lon: lower = tess.w and upper = tess.e (in degrees)
- glq_lat: lower = tess.s and upper = tess.n (in degrees)
- glq_r: lower = tess.r1 and upper = tess.r2

Parameters

tess data structure describing the tesseroid

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon GLQ structure with the nodes, weights and integration limits set for the longitudinal integration

glq_lat GLQ structure with the nodes, weights and integration limits set for the latitudinal integration

glq_r GLQ structure with the nodes, weights and integration limits set for the radial integration

Returns

field calculated at P

8.17.2.11 double tess_gzz (TESSEROID *tess*, double *lonp*, double *latp*, double *rp*, GLQ *glq_lon*, GLQ *glq_lat*, GLQ *glq_r*)

Calculates gzz caused by a tesseroid.

$$g_{zz}(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{3(r' \cos\psi - r_p)^2 - \ell^2}{\ell^5} \kappa \, dr' d\phi' d\lambda'$$

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Use function glq_new() to create the GLQ parameters required. The integration limits should be set to:

- glq_lon: lower = tess.w and upper = tess.e (in degrees)
- glq_lat: lower = tess.s and upper = tess.n (in degrees)
- glq_r: lower = tess.r1 and upper = tess.r2

Parameters

tess data structure describing the tesseroid

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon GLQ structure with the nodes, weights and integration limits set for the longitudinal integration

glq_lat GLQ structure with the nodes, weights and integration limits set for the latitudinal integration

glq_r GLQ structure with the nodes, weights and integration limits set for the radial integration

Returns

field calculated at P

8.18 src/c/grav_tess.h File Reference

Functions that calculate the gravitational potential and its first and second derivatives for the tesseroid.

```
#include "utils.h"
#include "glq.h"
```

Functions

- double calc_tess_model (TESSEROID *model, int size, double lonp, double latp, double rp, GLQ *glq_lon, GLQ *glq_lat, GLQ *glq_r, double(*field)(TESSEROID, double, double, double, GLQ, GLQ, GLQ))

  Calculates the field of a tesseroid model at a given point.

- double calc_tess_model_adapt (TESSEROID *model, int size, double lonp, double latp, double rp, GLQ *glq_lon, GLQ *glq_lat, GLQ *glq_r, double(*field)(TESSEROID, double, double, double, GLQ, GLQ, GLQ))

  Adaptatively calculate the field of a tesseroid model at a given point by splitting the tesseroids if necessary to maintain GLQ stability.

- double tess_gx (TESSEROID tess, double lonp, double latp, double rp, GLQ glq_lon, GLQ glq_lat, GLQ glq_r)

  Calculates gx caused by a tesseroid.

- double tess_gy (TESSEROID tess, double lonp, double latp, double rp, GLQ glq_lon, GLQ glq_lat, GLQ glq_r)

  Calculates gy caused by a tesseroid.

- double tess_gz (TESSEROID tess, double lonp, double latp, double rp, GLQ glq_lon, GLQ glq_lat, GLQ glq_r)

Calculates gz caused by a tesseroid.

- double tess_gxx (TESSEROID tess, double lonp, double latp, double rp, GLQ glq_lon, GLQ glq_lat, GLQ glq_r)

  Calculates gxx caused by a tesseroid.

- double tess_gxy (TESSEROID tess, double lonp, double latp, double rp, GLQ glq_lon, GLQ glq_lat, GLQ glq_r)

  Calculates gxy caused by a tesseroid.

- double tess_gxz (TESSEROID tess, double lonp, double latp, double rp, GLQ glq_lon, GLQ glq_lat, GLQ glq_r)

  Calculates gxz caused by a tesseroid.

- double tess_gyy (TESSEROID tess, double lonp, double latp, double rp, GLQ glq_lon, GLQ glq_lat, GLQ glq_r)

  Calculates gyy caused by a tesseroid.

- double tess_gyz (TESSEROID tess, double lonp, double latp, double rp, GLQ glq_lon, GLQ glq_lat, GLQ glq_r)

  Calculates gyz caused by a tesseroid.

- double tess_gzz (TESSEROID tess, double lonp, double latp, double rp, GLQ glq_lon, GLQ glq_lat, GLQ glq_r)

  Calculates gzz caused by a tesseroid.

8.18.1 Detailed Description

Functions that calculate the gravitational potential and its first and second derivatives for the tesseroid. The gravity gradients can be calculated using the general formula:

$$g_{\alpha\beta}(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} I_{\alpha\beta}\, dr' d\phi' d\lambda' \quad \alpha, \beta \in \{1, 2, 3\}$$

$$I_{\alpha\beta} = \left(\frac{3\Delta x_i \Delta x_j}{\ell^5} - \frac{\delta_{ij}}{\ell^3} \right) \kappa$$

and solved using the Gauss-Legendre Quadrature rule:

$$g_{\alpha\beta}(r_p, \phi_p, \lambda_p) \approx G\rho \frac{(\lambda_2 - \lambda_1)(\phi_2 - \phi_1)(r_2 - r_1)}{8} \sum_{k=0}^{N^\lambda - 1} \sum_{j=0}^{N^\phi - 1} \sum_{i=0}^{N^r - 1} W_i^r W_j^\phi W_k^\lambda I_{\alpha\beta}(r'_i, \phi'_j, \lambda'_k) \kappa \quad \alpha, \beta \in \{1, 2, 3\}$$

where ρ is density, the subscripts 1, 2, and 3 should be interpreted as the x, y, and z axis and

$$\begin{aligned} \Delta x_1 &= r' K_\phi \\ \Delta x_2 &= r' \cos\phi' \sin(\lambda' - \lambda_p) \end{aligned}$$

$$
\begin{aligned}
\Delta x_3 &= r' \cos\psi - r_p \\
\ell &= \sqrt{r'^2 + r_p^2 - 2r' r_p \cos\psi} \\
\cos\psi &= \sin\phi_p \sin\phi' + \cos\phi_p \cos\phi' \cos(\lambda' - \lambda_p) \\
K_\phi &= \cos\phi_p \sin\phi' - \sin\phi_p \cos\phi' \cos(\lambda' - \lambda_p) \\
\kappa &= r'^2 \cos\phi'
\end{aligned}
$$

ϕ is latitude, λ is longitude, r is radius. The subscript p is for the computation point.

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->Up** (away from center of the Earth).

To maintain the standard convention, only for component gz the z axis is inverted, so a positive density results in positive gz.

Example:

To calculate the gzz component due to a tesseroid on a regular grid.

```
#include <stdio.h>
#include "glq.h"r
#include "constants.h"
#include "grav_tess.h"

int main()
{
    TESSEROID tess = {1000, 44, 46, -1, 1, MEAN_EARTH_RADIUS - 100000,
                      MEAN_EARTH_RADIUS};
    GLQ *glqlon, *glqlat, *glqr;
    double lon, lat, r = MEAN_EARTH_RADIUS + 1500000, res;
    int order = 8;

    glqlon = glq_new(order, tess.w, tess.e);
    glqlat = glq_new(order, tess.s, tess.n);
    glqr = glq_new(order, tess.r1, tess.r2);

    for(lat = 20; lat <= 70; lat += 0.5)
    {
        for(lon = -25; lon <= 25; lon += 0.5)
        {
            res = tess_gzz(tess, lon, lat, r, *glqlon, *glqlat, *glqr);
            printf("%g %g %g\n", lon, lat, res);
        }
    }

    glq_free(glqlon);
    glq_free(glqlat);
    glq_free(glqr);

    return 0;
}
```

Todo

Possible speed up: use pointers for weights and nodes
Put reference for formulas
Allow for tesseroids with depth varying density

Author

Leonardo Uieda

Date

27 Jan 2011

8.18.2 Function Documentation

8.18.2.1 double calc_tess_model (TESSEROID * *model*, int *size*, double *lonp*, double *latp*, double *rp*, GLQ * *glq_lon*, GLQ * *glq_lat*, GLQ * *glq_r*, double(*)(TESSEROID, double, double, double, GLQ, GLQ, GLQ) *field*)

Calculates the field of a tesseroid model at a given point.

Uses a function pointer to call one of the apropriate field calculating functions:

- tess_gx()
- tess_gy()
- tess_gz()
- tess_gxx()
- tess_gxy()
- tess_gxz()
- tess_gyy()
- tess_gyz()
- tess_gzz()

To pass a function pointer to a function use something like:

```
calc_tess_model(my_model, 10, 0, 10, 1, glqlon, glqlat, glqr, &tess_gx);
```

This would calculate the gx effect of the model my_model with 10 tesseroids at lon=0 lat=10 r=1.

Will re-use the same GLQ structures, and therefore the **same order, for all the tesseroids**.

Parameters

model TESSEROID array defining the model
size number of tesseroids in the model
lonp longitude of the computation point P
latp latitude of the computation point P
rp radial coordinate of the computation point P
glq_lon pointer to GLQ structure used for the longitudinal integration
glq_lat pointer to GLQ structure used for the latitudinal integration
glq_r pointer to GLQ structure used for the radial integration
field pointer to one of the field calculating functions

Returns

the sum of the fields of all the tesseroids in the model

8.18.2.2 double calc_tess_model_adapt (TESSEROID * *model*, int *size*, double *lonp*, double *latp*, double *rp*, GLQ * *glq_lon*, GLQ * *glq_lat*, GLQ * *glq_r*, double(*)(TESSEROID, double, double, double, GLQ, GLQ, GLQ) *field*)

Adaptatively calculate the field of a tesseroid model at a given point by splitting the tesseroids if necessary to maintain GLQ stability.

See calc_tess_model() for more details.

Will re-use the same GLQ structures, and therefore the **same order, for all the tesseroids**.

Parameters

model TESSEROID array defining the model

size number of tesseroids in the model

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon pointer to GLQ structure used for the longitudinal integration

glq_lat pointer to GLQ structure used for the latitudinal integration

glq_r pointer to GLQ structure used for the radial integration

field pointer to one of the field calculating functions

Returns

the sum of the fields of all the tesseroids in the model

8.18.2.3 double tess_gx (TESSEROID *tess*, double *lonp*, double *latp*, double *rp*, GLQ *glq_lon*, GLQ *glq_lat*, GLQ *glq_r*)

Calculates gx caused by a tesseroid.

$$g_x(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{r' K_\phi}{\ell^3} \kappa \, dr' d\phi' d\lambda'$$

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in mGal!

Use function glq_new() to create the GLQ parameters required. The integration limits should be set to:

- glq_lon: lower = tess.w and upper = tess.e (in degrees)
- glq_lat: lower = tess.s and upper = tess.n (in degrees)
- glq_r: lower = tess.r1 and upper = tess.r2

Parameters

tess data structure describing the tesseroid

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon GLQ structure with the nodes, weights and integration limits set for the longitudinal integration

glq_lat GLQ structure with the nodes, weights and integration limits set for the latitudinal integration

glq_r GLQ structure with the nodes, weights and integration limits set for the radial integration

Returns

field calculated at P

8.18.2.4 double tess_gxx (TESSEROID *tess*, double *lonp*, double *latp*, double *rp*, GLQ *glq_lon*, GLQ *glq_lat*, GLQ *glq_r*)

Calculates gxx caused by a tesseroid.

$$g_{xx}(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{3(r' K_\phi)^2 - \ell^2}{\ell^5} \kappa \, dr' d\phi' d\lambda'$$

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Use function glq_new() to create the GLQ parameters required. The integration limits should be set to:

- glq_lon: lower = tess.w and upper = tess.e (in degrees)
- glq_lat: lower = tess.s and upper = tess.n (in degrees)
- glq_r: lower = tess.r1 and upper = tess.r2

Parameters

tess data structure describing the tesseroid

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon GLQ structure with the nodes, weights and integration limits set for the longitudinal integration

glq_lat GLQ structure with the nodes, weights and integration limits set for the latitudinal integration

glq_r GLQ structure with the nodes, weights and integration limits set for the radial integration

Returns

field calculated at P

8.18.2.5 double tess_gxy (TESSEROID *tess*, double *lonp*, double *latp*, double *rp*, GLQ *glq_lon*, GLQ *glq_lat*, GLQ *glq_r*)

Calculates gxy caused by a tesseroid.

$$g_{xy}(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{3r'^2 K_\phi \cos\phi' \sin(\lambda' - \lambda_p)}{\ell^5} \kappa \, dr' d\phi' d\lambda'$$

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Use function glq_new() to create the GLQ parameters required. The integration limits should be set to:

- glq_lon: lower = tess.w and upper = tess.e (in degrees)
- glq_lat: lower = tess.s and upper = tess.n (in degrees)
- glq_r: lower = tess.r1 and upper = tess.r2

Parameters

tess data structure describing the tesseroid

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon GLQ structure with the nodes, weights and integration limits set for the longitudinal integration

glq_lat GLQ structure with the nodes, weights and integration limits set for the latitudinal integration

glq_r GLQ structure with the nodes, weights and integration limits set for the radial integration

Returns

field calculated at P

8.18.2.6 double tess_gxz (TESSEROID *tess*, double *lonp*, double *latp*, double *rp*, GLQ *glq_lon*, GLQ *glq_lat*, GLQ *glq_r*)

Calculates gxz caused by a tesseroid.

$$g_{xz}(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{3r' K_\phi (r' \cos\psi - r_p)}{\ell^5} \kappa \, dr' d\phi' d\lambda'$$

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Use function glq_new() to create the GLQ parameters required. The integration limits should be set to:

- glq_lon: lower = tess.w and upper = tess.e (in degrees)
- glq_lat: lower = tess.s and upper = tess.n (in degrees)
- glq_r: lower = tess.r1 and upper = tess.r2

Parameters

tess data structure describing the tesseroid

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon GLQ structure with the nodes, weights and integration limits set for the longitudinal integration

glq_lat GLQ structure with the nodes, weights and integration limits set for the latitudinal integration

glq_r GLQ structure with the nodes, weights and integration limits set for the radial integration

Returns

field calculated at P

8.18.2.7 double tess_gy (TESSEROID *tess*, double *lonp*, double *latp*, double *rp*, GLQ *glq_lon*, GLQ *glq_lat*, GLQ *glq_r*)

Calculates gy caused by a tesseroid.

$$g_y(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{r' \cos\phi' \sin(\lambda' - \lambda)}{\ell^3} \kappa \, dr' d\phi' d\lambda'$$

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in mGal!

Use function glq_new() to create the GLQ parameters required. The integration limits should be set to:

- glq_lon: lower = tess.w and upper = tess.e (in degrees)
- glq_lat: lower = tess.s and upper = tess.n (in degrees)
- glq_r: lower = tess.r1 and upper = tess.r2

Parameters

tess data structure describing the tesseroid

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon GLQ structure with the nodes, weights and integration limits set for the longitudinal integration

glq_lat GLQ structure with the nodes, weights and integration limits set for the latitudinal integration

glq_r GLQ structure with the nodes, weights and integration limits set for the radial integration

Returns

field calculated at P

8.18.2.8 double tess_gyy (TESSEROID *tess*, double *lonp*, double *latp*, double *rp*, GLQ *glq_lon*, GLQ *glq_lat*, GLQ *glq_r*)

Calculates gyy caused by a tesseroid.

$$g_{yy}(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{3(r' \cos\phi' \sin(\lambda' - \lambda_p))^2 - \ell^2}{\ell^5} \kappa \, dr' d\phi' d\lambda'$$

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Use function glq_new() to create the GLQ parameters required. The integration limits should be set to:

- glq_lon: lower = tess.w and upper = tess.e (in degrees)
- glq_lat: lower = tess.s and upper = tess.n (in degrees)
- glq_r: lower = tess.r1 and upper = tess.r2

Parameters

tess data structure describing the tesseroid

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon GLQ structure with the nodes, weights and integration limits set for the longitudinal integration

glq_lat GLQ structure with the nodes, weights and integration limits set for the latitudinal integration

glq_r GLQ structure with the nodes, weights and integration limits set for the radial integration

Returns

field calculated at P

8.18.2.9 double tess_gyz (TESSEROID *tess*, double *lonp*, double *latp*, double *rp*, GLQ *glq_lon*, GLQ *glq_lat*, GLQ *glq_r*)

Calculates gyz caused by a tesseroid.

$$g_{yz}(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{3r' \cos\phi' \sin(\lambda' - \lambda_p)(r' \cos\psi - r_p)}{\ell^5} \kappa \, dr' d\phi' d\lambda'$$

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Use function glq_new() to create the GLQ parameters required. The integration limits should be set to:

- glq_lon: lower = tess.w and upper = tess.e (in degrees)
- glq_lat: lower = tess.s and upper = tess.n (in degrees)
- glq_r: lower = tess.r1 and upper = tess.r2

Parameters

tess data structure describing the tesseroid

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon GLQ structure with the nodes, weights and integration limits set for the longitudinal integration

glq_lat GLQ structure with the nodes, weights and integration limits set for the latitudinal integration

glq_r GLQ structure with the nodes, weights and integration limits set for the radial integration

Returns

field calculated at P

8.18.2.10 double tess_gz (TESSEROID *tess*, double *lonp*, double *latp*, double *rp*, GLQ *glq_lon*, GLQ *glq_lat*, GLQ *glq_r*)

Calculates gz caused by a tesseroid.

$$g_z(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{r' \cos\psi - r_p}{\ell^3} \kappa \, dr' d\phi' d\lambda'$$

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in mGal!

Use function glq_new() to create the GLQ parameters required. The integration limits should be set to:

- glq_lon: lower = tess.w and upper = tess.e (in degrees)
- glq_lat: lower = tess.s and upper = tess.n (in degrees)

- glq_r: lower = tess.r1 and upper = tess.r2

Parameters

tess data structure describing the tesseroid

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon GLQ structure with the nodes, weights and integration limits set for the longitudinal integration

glq_lat GLQ structure with the nodes, weights and integration limits set for the latitudinal integration

glq_r GLQ structure with the nodes, weights and integration limits set for the radial integration

Returns

field calculated at P

8.18.2.11 double tess_gzz (TESSEROID *tess*, double *lonp*, double *latp*, double *rp*, GLQ *glq_lon*, GLQ *glq_lat*, GLQ *glq_r*)

Calculates gzz caused by a tesseroid.

$$g_{zz}(r_p, \phi_p, \lambda_p) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{3(r' \cos\psi - r_p)^2 - \ell^2}{\ell^5} \kappa \, dr' d\phi' d\lambda'$$

The derivatives of the potential are made with respect to the local coordinate system **x->North, y->East, z->out**

Input values in SI units and degrees and returns values in Eotvos!

Use function glq_new() to create the GLQ parameters required. The integration limits should be set to:

- glq_lon: lower = tess.w and upper = tess.e (in degrees)
- glq_lat: lower = tess.s and upper = tess.n (in degrees)
- glq_r: lower = tess.r1 and upper = tess.r2

Parameters

tess data structure describing the tesseroid

lonp longitude of the computation point P

latp latitude of the computation point P

rp radial coordinate of the computation point P

glq_lon GLQ structure with the nodes, weights and integration limits set for the longitudinal integration

glq_lat GLQ structure with the nodes, weights and integration limits set for the latitudinal integration

glq_r GLQ structure with the nodes, weights and integration limits set for the radial integration

Returns

field calculated at P

8.19 src/c/logger.c File Reference

Functions to set up logging.

```
#include <stdio.h>
#include <stdarg.h>
#include <time.h>
#include "logger.h"
```

Functions

- void log_init (int level)
  Setup logging to stderr.
- void log_tofile (FILE *logfile, int level)
  Set logging to a file.
- void log_debug (const char *fmt,...)
  Log a message at debug level.
- void log_info (const char *fmt,...)
  Log a message at info level.
- void log_warning (const char *fmt,...)
  Log a message at warning level.
- void log_error (const char *fmt,...)
  Log a message at error level.

Variables

- LOGGER logger
  Global logger struct.

8.19.1 Detailed Description

Functions to set up logging.

Author

Leonardo Uieda

Date

31 Jan 2011

8.19.2 Function Documentation

8.19.2.1 void log_debug (const char * *fmt*, ...)

Log a message at debug level.

Pass parameters in the same format as printf()

Prints a newline at the end.

8.19.2.2 void log_error (const char * *fmt*, ...)

Log a message at error level.

Pass parameters in the same format as printf()

Prints a newline at the end.

8.19.2.3 void log_info (const char * *fmt*, ...)

Log a message at info level.

Pass parameters in the same format as printf()

Does not print "INFO: " in front of the message when logging

Prints a newline at the end.

8.19.2.4 void log_init (int *level*)

Setup logging to stderr.

Parameters

level level of logging to be made. Can be one of:

- LOG_DEBUG
- LOG_INFO
- LOG_WARNING
- LOG_ERROR

8.19.2.5 void log_tofile (FILE * *logfile*, int *level*)

Set logging to a file.

Parameters

logfile FILE pointer to the already open file to log to.

level level of logging to be made to the file. Can be one of:

- LOG_DEBUG
- LOG_INFO
- LOG_WARNING
- LOG_ERROR

8.19.2.6 void log_warning (const char * *fmt*, ...)

Log a message at warning level.

Pass parameters in the same format as printf()

Prints a newline at the end.

8.19.3 Variable Documentation

8.19.3.1 LOGGER logger

Initial value:

```
{.level = 100,
                .file_level = 100,
                .filelogging = 0,
                .logfile = NULL}
```

Global logger struct.

Only declare in the main program!

8.20 src/c/logger.h File Reference

Functions to set up logging.

```
#include <stdio.h>
```

Data Structures

- struct LOGGER

  Keep the information on the global logger.

Defines

- #define LOG_DEBUG 0

  Logging level for debug messages.

- #define LOG_INFO 1

  Logging level for general information.

- #define LOG_WARNING 2

  Logging level for warning messages.

- #define LOG_ERROR 3

  Logging level for error messages.

Functions

- void log_init (int level)

  Setup logging to stderr.

- void log_tofile (FILE *logfile, int level)

  Set logging to a file.

- void log_debug (const char *fmt,...)

  Log a message at debug level.

- void log_info (const char *fmt,...)

  Log a message at info level.

- void log_warning (const char *fmt,...)

  Log a message at warning level.

- void log_error (const char *fmt,...)

  Log a message at error level.

Variables

- LOGGER logger

  Global logger struct.

8.20.1 Detailed Description

Functions to set up logging. Example:

```
#include "logger.h"

void my_func(){
    log_info("From my_func!\n");
}

int main(){
    log_init(LOG_DEBUG);
    log_debug("debug line. The code is %d", LOG_DEBUG);
    log_info("info line. The code is %d", LOG_INFO);
    log_warning("warning line. The code is %d", LOG_WARNING);
    log_error("error line. The code is %d", LOG_ERROR);
    return 0;
}
```

Will print:

```
DEBUG: debug line. The code is 0
info line. The code is 1
WARNING: warning line. The code is 2
ERROR: error line. The code is 3
```

If function log_init() is not called than logging is disabled and no messages will be printed to stderr.

Author

Leonardo Uieda

Date

31 Jan 2011

8.20.2 Function Documentation

8.20.2.1 void log_debug (const char * *fmt*, ...)

Log a message at debug level.

Pass parameters in the same format as printf()

Prints a newline at the end.

8.20.2.2 void log_error (const char * *fmt*, ...)

Log a message at error level.

Pass parameters in the same format as printf()

Prints a newline at the end.

8.20.2.3 void log_info (const char * *fmt*, ...)

Log a message at info level.

Pass parameters in the same format as printf()

Does not print "INFO: " in front of the message when logging

Prints a newline at the end.

8.20.2.4 void log_init (int *level*)

Setup logging to stderr.

Parameters

level level of logging to be made. Can be one of:

- LOG_DEBUG
- LOG_INFO
- LOG_WARNING
- LOG_ERROR

8.20.2.5 void log_tofile (FILE * *logfile*, int *level*)

Set logging to a file.

Parameters

logfile FILE pointer to the already open file to log to.

level level of logging to be made to the file. Can be one of:

- LOG_DEBUG
- LOG_INFO
- LOG_WARNING
- LOG_ERROR

8.20.2.6 void log_warning (const char * *fmt*, ...)

Log a message at warning level.

Pass parameters in the same format as printf()

Prints a newline at the end.

8.20.3 Variable Documentation

8.20.3.1 LOGGER logger

Global logger struct.

Only declare in the main program!

8.21 src/c/prismg_main.c File Reference

Generic main function for the prismg* programs.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
```

```
#include "logger.h"
#include "version.h"
#include "grav_prism.h"
#include "utils.h"
#include "cmd.h"
#include "prismg_main.h"
```

Functions

- void print_help ()
  Print the help message.
- int run_prismg_main (int argc, char **argv, const char *progname, double(*field)(PRISM, double, double, double))
  Run the main for a generic prismg program.*

Variables

- char **global_progname** [100]

8.21.1 Detailed Description

Generic main function for the prismg* programs.

Author

Leonardo Uieda

Date

08 Feb 2011

8.21.2 Function Documentation

8.21.2.1 int run_prismg_main (int *argc*, char ** *argv*, const char * *progname*, double(*)(PRISM, double, double, double) *field*)

Run the main for a generic prismg* program.

Parameters

argc number of command line arguments
argv command line arguments
progname name of the specific program
field pointer to function that calculates the field of a single prism

Returns

0 is all went well. 1 if failed.

8.22 src/c/prismg_main.h File Reference

Generic main function for the prismg∗ programs.

```
#include "utils.h"
```

Functions

- void print_help ()

  Print the help message.

- int run_prismg_main (int argc, char ∗∗argv, const char ∗progname, double(∗field)(PRISM, double, double, double))

  Run the main for a generic prismg∗ program.

8.22.1 Detailed Description

Generic main function for the prismg∗ programs.

Todo

Option for calculating on spherical coordinates
Catch errors in input points that occur at the end of the line

Author

Leonardo Uieda

Date

08 Feb 2011

8.22.2 Function Documentation

8.22.2.1 int run_prismg_main (int *argc*, char ∗∗ *argv*, const char ∗ *progname*, double(∗)(PRISM, double, double, double) *field*)

Run the main for a generic prismg∗ program.

Parameters

argc number of command line arguments

argv command line arguments

progname name of the specific program

field pointer to function that calculates the field of a single prism

Returns

0 is all went well. 1 if failed.

8.23 src/c/prismgx.c File Reference

Program to calculate gx of a rectangular prism model on a set of points.

```
#include "grav_prism.h"
#include "prismg_main.h"
```

Functions

- int main (int argc, char **argv)
  Main.

8.23.1 Detailed Description

Program to calculate gx of a rectangular prism model on a set of points.

Author

Leonardo Uieda

Date

08 Feb 2011

8.24 src/c/prismgxx.c File Reference

Program to calculate gxx of a rectangular prism model on a set of points.

```
#include "grav_prism.h"
#include "prismg_main.h"
```

Functions

- int main (int argc, char **argv)
  Main.

8.24.1 Detailed Description

Program to calculate gxx of a rectangular prism model on a set of points.

Author

Leonardo Uieda

Date

08 Feb 2011

8.25 src/c/prismgxy.c File Reference

Program to calculate gxy of a rectangular prism model on a set of points.

```
#include "grav_prism.h"
#include "prismg_main.h"
```

Functions

- int main (int argc, char **argv)
  Main.

8.25.1 Detailed Description

Program to calculate gxy of a rectangular prism model on a set of points.

Author

Leonardo Uieda

Date

08 Feb 2011

8.26 src/c/prismgxz.c File Reference

Program to calculate gxz of a rectangular prism model on a set of points.

```
#include "grav_prism.h"
#include "prismg_main.h"
```

Functions

- int main (int argc, char **argv)
  Main.

8.26.1 Detailed Description

Program to calculate gxz of a rectangular prism model on a set of points.

Author

Leonardo Uieda

Date

08 Feb 2011

8.27 src/c/prismgy.c File Reference

Program to calculate gy of a rectangular prism model on a set of points.

```
#include "grav_prism.h"
#include "prismg_main.h"
```

Functions

- int main (int argc, char **argv)
  Main.

8.27.1 Detailed Description

Program to calculate gy of a rectangular prism model on a set of points.

Author

Leonardo Uieda

Date

08 Feb 2011

8.28 src/c/prismgyy.c File Reference

Program to calculate gyy of a rectangular prism model on a set of points.

```
#include "grav_prism.h"
#include "prismg_main.h"
```

Functions

- int main (int argc, char **argv)
  Main.

8.28.1 Detailed Description

Program to calculate gyy of a rectangular prism model on a set of points.

Author

Leonardo Uieda

Date

08 Feb 2011

8.29 src/c/prismgyz.c File Reference

Program to calculate gyz of a rectangular prism model on a set of points.

```
#include "grav_prism.h"
#include "prismg_main.h"
```

Functions

- int main (int argc, char **argv)
  Main.

8.29.1 Detailed Description

Program to calculate gyz of a rectangular prism model on a set of points.

Author

Leonardo Uieda

Date

08 Feb 2011

8.30 src/c/prismgz.c File Reference

Program to calculate gz of a rectangular prism model on a set of points.

```
#include "grav_prism.h"
#include "prismg_main.h"
```

Functions

- int main (int argc, char **argv)
  Main.

8.30.1 Detailed Description

Program to calculate gz of a rectangular prism model on a set of points.

Author

Leonardo Uieda

Date

08 Feb 2011

8.31 src/c/prismgzz.c File Reference

Program to calculate gzz of a rectangular prism model on a set of points.

```
#include "grav_prism.h"
#include "prismg_main.h"
```

Functions

- int main (int argc, char **argv)
  Main.

8.31.1 Detailed Description

Program to calculate gzz of a rectangular prism model on a set of points.

Author

Leonardo Uieda

Date

08 Feb 2011

8.32 src/c/tess2prism.c File Reference

Convert a tesseroid model into a prism model in spherical coordinates.

```
#include <stdio.h>
#include <time.h>
#include "version.h"
#include "cmd.h"
#include "logger.h"
#include "utils.h"
```

Functions

- void print_help ()
  Print the help message.
- int main (int argc, char **argv)
  Main.

8.32.1 Detailed Description

Convert a tesseroid model into a prism model in spherical coordinates.

Author

Leonardo Uieda

Date

04 Feb 2011

8.33 src/c/tessdefaults.c File Reference

Print the default values of the constants used in the calculations.

```
#include <stdio.h>
#include <time.h>
#include "version.h"
#include "logger.h"
#include "constants.h"
#include "glq.h"
```

Functions

- void print_help ()
  Print the help message.
- int main (int argc, char **argv)
  Main.

8.33.1 Detailed Description

Print the default values of the constants used in the calculations.

Author

Leonardo Uieda

Date

09 Feb 2011

8.34 src/c/tessg_main.c File Reference

Generic main function for the tessg* programs.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "logger.h"
```

```
#include "version.h"
#include "grav_tess.h"
#include "glq.h"
#include "constants.h"
#include "utils.h"
#include "cmd.h"
#include "tessg_main.h"
```

Functions

- int run_tessg_main (int argc, char **argv, const char *progname, double(*field)(TESSEROID, double, double, double, GLQ, GLQ, GLQ))

  Run the main for a generic tessg program.*

8.34.1 Detailed Description

Generic main function for the tessg* programs.

Author

Leonardo Uieda

Date

03 Feb 2011

8.34.2 Function Documentation

8.34.2.1 int run_tessg_main (int *argc*, char ** *argv*, const char * *progname*, double(*)(TESSEROID, double, double, double, GLQ, GLQ, GLQ) *field*)

Run the main for a generic tessg* program.

Parameters

argc number of command line arguments

argv command line arguments

progname name of the specific program

field pointer to function that calculates the field of a single tesseroid

Returns

0 is all went well. 1 if failed.

8.35 src/c/tessg_main.h File Reference

Generic main function for the tessg∗ programs.

```
#include "glq.h"
#include "utils.h"
```

Functions

- int run_tessg_main (int argc, char ∗∗argv, const char ∗progname, double(∗field)(TESSEROID, double, double, double, GLQ, GLQ, GLQ))

  Run the main for a generic tessg∗ program.

8.35.1 Detailed Description

Generic main function for the tessg∗ programs.

Todo

Catch errors in input points that occur at the end of the line

Author

Leonardo Uieda

Date

03 Feb 2011

8.35.2 Function Documentation

8.35.2.1 int run_tessg_main (int *argc*, char ∗∗ *argv*, const char ∗ *progname*, double(∗)(TESSEROID, double, double, double, GLQ, GLQ, GLQ) *field*)

Run the main for a generic tessg∗ program.

Parameters

argc number of command line arguments

argv command line arguments

progname name of the specific program

field pointer to function that calculates the field of a single tesseroid

Returns

0 is all went well. 1 if failed.

8.36 src/c/tessgrd.c File Reference

Program to generate a regular grid of points.

```
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "logger.h"
#include "version.h"
#include "cmd.h"
```

Functions

- int main (int argc, char **argv)
  Main.

8.36.1 Detailed Description

Program to generate a regular grid of points.

Todo

Catch wrong order of -r arguments ie. w > e or s > n

Author

Leonardo Uieda

Date

01 Feb 2011

8.37 src/c/tessgx.c File Reference

Program to calculate gx of a tesseroid model on a set of points.

```
#include "grav_tess.h"
#include "tessg_main.h"
```

Functions

- int main (int argc, char **argv)
  Main.

8.37.1 Detailed Description

Program to calculate gx of a tesseroid model on a set of points.

Author

Leonardo Uieda

Date

02 Feb 2011

8.38 src/c/tessgxx.c File Reference

Program to calculate gxx of a tesseroid model on a set of points.

```
#include "grav_tess.h"
#include "tessg_main.h"
```

Functions

- int main (int argc, char **argv)
  Main.

8.38.1 Detailed Description

Program to calculate gxx of a tesseroid model on a set of points.

Author

Leonardo Uieda

Date

02 Feb 2011

8.39 src/c/tessgxy.c File Reference

Program to calculate gxy of a tesseroid model on a set of points.

```
#include "grav_tess.h"
#include "tessg_main.h"
```

Functions

- int main (int argc, char **argv)
  Main.

8.39.1 Detailed Description

Program to calculate gxy of a tesseroid model on a set of points.

Author

Leonardo Uieda

Date

02 Feb 2011

8.40 src/c/tessgxz.c File Reference

Program to calculate gxz of a tesseroid model on a set of points.

```
#include "grav_tess.h"
#include "tessg_main.h"
```

Functions

- int main (int argc, char **argv)
  Main.

8.40.1 Detailed Description

Program to calculate gxz of a tesseroid model on a set of points.

Author

Leonardo Uieda

Date

02 Feb 2011

8.41 src/c/tessgy.c File Reference

Program to calculate gy of a tesseroid model on a set of points.

```
#include "grav_tess.h"
#include "tessg_main.h"
```

Functions

- int main (int argc, char **argv)
  Main.

8.41.1 Detailed Description

Program to calculate gy of a tesseroid model on a set of points.

Author

Leonardo Uieda

Date

02 Feb 2011

8.42 src/c/tessgyy.c File Reference

Program to calculate gyy of a tesseroid model on a set of points.

```
#include "grav_tess.h"
#include "tessg_main.h"
```

Functions

- int main (int argc, char **argv)
  Main.

8.42.1 Detailed Description

Program to calculate gyy of a tesseroid model on a set of points.

Author

Leonardo Uieda

Date

02 Feb 2011

8.43 src/c/tessgyz.c File Reference

Program to calculate gyz of a tesseroid model on a set of points.

```
#include "grav_tess.h"
#include "tessg_main.h"
```

Functions

- int main (int argc, char **argv)
  Main.

8.43.1 Detailed Description

Program to calculate gyz of a tesseroid model on a set of points.

Author

Leonardo Uieda

Date

02 Feb 2011

8.44 src/c/tessgz.c File Reference

Program to calculate gz of a tesseroid model on a set of points.

```
#include "grav_tess.h"
#include "tessg_main.h"
```

Functions

- int main (int argc, char **argv)
  Main.

8.44.1 Detailed Description

Program to calculate gz of a tesseroid model on a set of points.

Author

Leonardo Uieda

Date

02 Feb 2011

8.45 src/c/tessgzz.c File Reference

Program to calculate gzz of a tesseroid model on a set of points.

```
#include "grav_tess.h"
#include "tessg_main.h"
```

Functions

- int main (int argc, char **argv)
  Main.

8.45.1 Detailed Description

Program to calculate gzz of a tesseroid model on a set of points.

Author

Leonardo Uieda

Date

02 Feb 2011

8.46 src/c/tessmass.c File Reference

Calculate the mass of a tesseroid model.

```
#include <stdio.h>
#include <time.h>
#include "version.h"
#include "cmd.h"
#include "logger.h"
#include "utils.h"
```

Functions

- void print_help ()
  Print the help message.
- int main (int argc, char **argv)
  Main.

8.46.1 Detailed Description

Calculate the mass of a tesseroid model.

Author

Leonardo Uieda

Date

09 Feb 2011

8.47 src/c/tessmodgen.c File Reference

Generate tesseroid model from a regular grid.

```
#include <stdio.h>
#include <time.h>
#include "version.h"
#include "cmd.h"
#include "logger.h"
#include "utils.h"
```

Functions

- void print_help ()
  Print the help message.

- int main (int argc, char **argv)
  Main.

8.47.1 Detailed Description

Generate tesseroid model from a regular grid.

Author

Leonardo Uieda

Date

09 Feb 2011

8.48 src/c/utils.c File Reference

Set of misc utilities and data structures.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include "constants.h"
#include "logger.h"
#include "utils.h"
```

Functions

- void split_tess (TESSEROID tess, TESSEROID *split)
  Split a tesseroid into 8.
- double tess_total_mass (TESSEROID *model, int size)
  Calculate the total mass of a tesseroid model.
- double tess_range_mass (TESSEROID *model, int size, double low_dens, double high_dens)
  Calculate the mass of a tesseroid model within a density range.
- void tess2prism (TESSEROID tess, PRISM *prism)
  Convert a tesseroid into a rectangular prism of equal volume.
- void tess2sphere (TESSEROID tess, SPHERE *sphere)
  Convert a tesseroid into a sphere of equal volume.
- void prism2sphere (PRISM prism, double lonc, double latc, double rc, SPHERE *sphere)
  Convert a rectangular prism into a sphere of equal volume.
- double tess_volume (TESSEROID tess)

Calculate the volume of a tesseroid.

- double sphere_volume (SPHERE sphere)
  Calculate the volume of a sphere.
- double prism_volume (PRISM prism)
  Calculate the volume of a prism.
- void strstrip (char *str)
  Strip trailing spaces and newlines from the end of a string.
- int gets_tess (const char *str, TESSEROID *tess)
  Read a single tesseroid from a string.
- TESSEROID * read_tess_model (FILE *modelfile, int *size)
  Read tesseroids from an open file and store them in an array.
- int gets_prism (const char *str, PRISM *prism)
  Read a single rectangular prism from a string.
- PRISM * read_prism_model (FILE *modelfile, int *size)
  Read rectangular prisms from an open file and store them in an array.

8.48.1 Detailed Description

Set of misc utilities and data structures. Defines the TESSEROID, SPHERE and PRISM structures.

Author

Leonardo Uieda

Date

25 Jan 2011

8.48.2 Function Documentation

8.48.2.1 int gets_prism (const char * *str*, PRISM * *prism*)

Read a single rectangular prism from a string.

Parameters

str string with the tesseroid parameters

prism used to return the read prism

Returns

0 if all went well, 1 if failed to read.

Todo

Catch wrong order of model inputs, ie. x1 > x2 etc

Todo

Read the position of the prism from the string

8.48.2.2 int gets_tess (const char * *str*, TESSEROID * *tess*)

Read a single tesseroid from a string.

Parameters

str string with the tesseroid parameters
tess used to return the read tesseroid

Returns

0 if all went well, 1 if failed to read.

Todo

Catch wrong order of model inputs, ie. w > e or s > n or top < bottom

8.48.2.3 void prism2sphere (PRISM *prism*, double *lonc*, double *latc*, double *rc*, SPHERE * *sphere*)

Convert a rectangular prism into a sphere of equal volume.

Parameters:

Parameters

prism prism to convert
lonc longitude of the desired center of the sphere, in degrees
latc latitude of the desired center of the sphere, in degrees
rc radial coordinate of the desired center of the sphere, in SI units
sphere sphere with equal volume of the prism (used to return)

8.48.2.4 double prism_volume (PRISM *prism*)

Calculate the volume of a prism.

Parameters

prism the prism whose volume will be calculated

Returns

the volume in the respective units

8.48.2.5 PRISM* read_prism_model (FILE * *modelfile*, int * *size*)

Read rectangular prisms from an open file and store them in an array.

Allocates memory. Don't forget to free 'model'!

Parameters

modelfile open FILE for reading with the model
size used to return the size of the model read

Returns

pointer to array with the model. NULL if there was an error

8.48.2.6 TESSEROID* read_tess_model (FILE * *modelfile*, int * *size*)

Read tesseroids from an open file and store them in an array.

Allocates memory. Don't forget to free 'model'!

Parameters

modelfile open FILE for reading with the tesseroid model
size used to return the size of the model read

Returns

pointer to array with the model. NULL if there was an error

8.48.2.7 double sphere_volume (SPHERE *sphere*)

Calculate the volume of a sphere.

Parameters

sphere the sphere whose volume will be calculated

Returns

the volume in the respective units

8.48.2.8 void split_tess (TESSEROID *tess*, TESSEROID * *split*)

Split a tesseroid into 8.

Parameters

tess tesseroid that will be split
split array of 8 tesseroids with memory allocated. Used to return.

8.48.2.9 void strstrip (char * *str*)

Strip trailing spaces and newlines from the end of a string.

Done IN PLACE!

Parameters

str string to strip

8.48.2.10 void tess2prism (TESSEROID *tess*, PRISM * *prism*)

Convert a tesseroid into a rectangular prism of equal volume.

Parameters

tess tesseroid to convert

prism prism with equal volume of the tesseroid (used to return)

Todo

Put reference for formulas

8.48.2.11 void tess2sphere (TESSEROID *tess*, SPHERE * *sphere*)

Convert a tesseroid into a sphere of equal volume.

Parameters:

Parameters

tess tesseroid to convert

sphere sphere with equal volume of the tesseroid (used to return)

8.48.2.12 double tess_range_mass (TESSEROID * *model*, int *size*, double *low_dens*, double *high_dens*)

Calculate the mass of a tesseroid model within a density range.

Give all in SI units and degrees!

Parameters

model array of tesseroids

size size of the model

low_dens lower bound of the density range

high_dens upper bound of the density range

Returns

The calculated mass

8.48.2.13 double tess_total_mass (TESSEROID * *model*, int *size*)

Calculate the total mass of a tesseroid model.

Give all in SI units and degrees!

Parameters

model array of tesseroids

size size of the model

Returns

The calculated mass

8.48.2.14 double tess_volume (TESSEROID *tess*)

Calculate the volume of a tesseroid.

Parameters

tess the tesseroid whose volume will be calculated

Returns

the volume in the respective units

8.49 src/c/utils.h File Reference

Set of misc utilities and data structures.

```
#include <stdio.h>
```

Data Structures

- struct TESSEROID
  Store information on a tesseroid.
- struct PRISM
  Store information on a rectangular prism.
- struct SPHERE
  Store information on a sphere.

Functions

- void split_tess (TESSEROID tess, TESSEROID *split)
  Split a tesseroid into 8.
- double tess_total_mass (TESSEROID *model, int size)
  Calculate the total mass of a tesseroid model.
- double tess_range_mass (TESSEROID *model, int size, double low_dens, double high_dens)
  Calculate the mass of a tesseroid model within a density range.
- void tess2prism (TESSEROID tess, PRISM *prism)
  Convert a tesseroid into a rectangular prism of equal volume.
- void tess2sphere (TESSEROID tess, SPHERE *sphere)
  Convert a tesseroid into a sphere of equal volume.
- void prism2sphere (PRISM prism, double lonc, double latc, double rc, SPHERE *sphere)
  Convert a rectangular prism into a sphere of equal volume.
- double tess_volume (TESSEROID tess)
  Calculate the volume of a tesseroid.
- double sphere_volume (SPHERE sphere)
  Calculate the volume of a sphere.
- double prism_volume (PRISM prism)
  Calculate the volume of a prism.
- void strstrip (char *str)
  Strip trailing spaces and newlines from the end of a string.
- int gets_tess (const char *str, TESSEROID *tess)
  Read a single tesseroid from a string.
- TESSEROID * read_tess_model (FILE *modelfile, int *size)
  Read tesseroids from an open file and store them in an array.
- int gets_prism (const char *str, PRISM *prism)
  Read a single rectangular prism from a string.
- PRISM * read_prism_model (FILE *modelfile, int *size)
  Read rectangular prisms from an open file and store them in an array.

8.49.1 Detailed Description

Set of misc utilities and data structures. Defines the TESSEROID, SPHERE and PRISM structures.

Author

Leonardo Uieda

Date

25 Jan 2011

8.49.2 Function Documentation

8.49.2.1 int gets_prism (const char * *str*, PRISM * *prism*)

Read a single rectangular prism from a string.

Parameters

str string with the tesseroid parameters

prism used to return the read prism

Returns

0 if all went well, 1 if failed to read.

Todo

Catch wrong order of model inputs, ie. x1 > x2 etc

Todo

Read the position of the prism from the string

8.49.2.2 int gets_tess (const char * *str*, TESSEROID * *tess*)

Read a single tesseroid from a string.

Parameters

str string with the tesseroid parameters

tess used to return the read tesseroid

Returns

0 if all went well, 1 if failed to read.

Todo

Catch wrong order of model inputs, ie. w > e or s > n or top < bottom

8.49.2.3 void prism2sphere (PRISM *prism*, double *lonc*, double *latc*, double *rc*, SPHERE * *sphere*)

Convert a rectangular prism into a sphere of equal volume.

Parameters:

Parameters

prism prism to convert

lonc longitude of the desired center of the sphere, in degrees

latc latitude of the desired center of the sphere, in degrees

rc radial coordinate of the desired center of the sphere, in SI units

sphere sphere with equal volume of the prism (used to return)

8.49.2.4 double prism_volume (PRISM *prism*)

Calculate the volume of a prism.

Parameters

prism the prism whose volume will be calculated

Returns

the volume in the respective units

8.49.2.5 PRISM* read_prism_model (FILE * *modelfile*, int * *size*)

Read rectangular prisms from an open file and store them in an array.

Allocates memory. Don't forget to free 'model'!

Parameters

modelfile open FILE for reading with the model

size used to return the size of the model read

Returns

pointer to array with the model. NULL if there was an error

8.49.2.6 TESSEROID* read_tess_model (FILE * *modelfile*, int * *size*)

Read tesseroids from an open file and store them in an array.

Allocates memory. Don't forget to free 'model'!

Parameters

modelfile open FILE for reading with the tesseroid model

size used to return the size of the model read

Returns

pointer to array with the model. NULL if there was an error

8.49.2.7 double sphere_volume (SPHERE *sphere*)

Calculate the volume of a sphere.

Parameters

sphere the sphere whose volume will be calculated

Returns

the volume in the respective units

8.49.2.8 void split_tess (TESSEROID *tess*, TESSEROID * *split*)

Split a tesseroid into 8.

Parameters

tess tesseroid that will be split

split array of 8 tesseroids with memory allocated. Used to return.

8.49.2.9 void strstrip (char * *str*)

Strip trailing spaces and newlines from the end of a string.

Done IN PLACE!

Parameters

str string to strip

8.49.2.10 void tess2prism (TESSEROID *tess*, PRISM * *prism*)

Convert a tesseroid into a rectangular prism of equal volume.

Parameters

tess tesseroid to convert
prism prism with equal volume of the tesseroid (used to return)

Todo

Put reference for formulas

8.49.2.11 void tess2sphere (TESSEROID *tess*, SPHERE * *sphere*)

Convert a tesseroid into a sphere of equal volume.

Parameters:

Parameters

tess tesseroid to convert
sphere sphere with equal volume of the tesseroid (used to return)

8.49.2.12 double tess_range_mass (TESSEROID * *model*, int *size*, double *low_dens*, double *high_dens*)

Calculate the mass of a tesseroid model within a density range.

Give all in SI units and degrees!

Parameters

model array of tesseroids
size size of the model
low_dens lower bound of the density range
high_dens upper bound of the density range

Returns

The calculated mass

8.49.2.13 double tess_total_mass (TESSEROID * *model*, int *size*)

Calculate the total mass of a tesseroid model.

Give all in SI units and degrees!

Parameters

model array of tesseroids
size size of the model

Returns

The calculated mass

8.49.2.14 double tess_volume (TESSEROID *tess*)

Calculate the volume of a tesseroid.

Parameters

tess the tesseroid whose volume will be calculated

Returns

the volume in the respective units

8.50 src/c/version.c File Reference

Hold the version number of the project.

```
#include <stdio.h>
#include "version.h"
```

Functions

- void print_version (const char *progname)
  Print version and license information.

Variables

- const char tesseroids_version [ ] = "1.0"
  Current project version number.

8.50.1 Detailed Description

Hold the version number of the project.

Author

Leonardo Uieda

Date

02 Feb 2011

8.50.2 Function Documentation

8.50.2.1 void print_version (const char * *progname*)

Print version and license information.

Parameters

progname name of the program

8.51 src/c/version.h File Reference

Hold the version number of the project.

Functions

- void print_version (const char *progname)
  Print version and license information.

Variables

- const char tesseroids_version [ ]
  Current project version number.

8.51.1 Detailed Description

Hold the version number of the project.

Author

Leonardo Uieda

Date

01 Feb 2011

8.51.2 Function Documentation

8.51.2.1 void print_version (const char * *progname*)

Print version and license information.

Parameters

progname name of the program

8.52 TODO.h File Reference

To do list.

8.52.1 Detailed Description

To do list. TESTS:

Todo

Check error in not rotating prism
Check error os using tesseroid in poles

PROGRAMS:

Todo

Programs to calculate the effect of a sphere model in spherical coordinates

API:

Todo

Make minunit into functions and put variable arguments for messages like printf

DOC:

Todo

Make doxygen groups to separate programs from api

MAYBE:

Todo

Generate VTK file to plot tesseroids in Mayavi2 or Paraview

DONE:

- (Done 27/Jan/2011) Include automatic documentation with doxygen
- (Done 28/Jan/2011) Make integration test to compare tess to sphere and prism to sphere
- (Done 02/Feb/2011) Move grid generation to a separate program
- (Done 08/Feb/2011) Programs to calculate the effect of a prism model in cartesian coordinates
- (Done 09/Feb/2011) Program to calculate the total mass of a model (with options for cut-off density values)
- (Done 09/Feb/2011) Program to generate tesseroid model from an regular grid
- (Done 09/Feb/2011) Program to print default values for constants
- (Done 10/Feb/2011) Fix local z axis positive upward. Change all depths when defining tesseroids to negative heights.
- (Done 11/Feb/2011) Adaptatively resize tesseroid when too close to computation point (with fixed GLQ order)

Index