# rsvs3D

0.0.0

Generated by Doxygen 1.8.15

# Chapter 1

# README

Last updated 12/03/2019

Full documentation

## What is this repository for?

This repository is the C++ implementation of the 3D R-Snake Volume of Solid (RSVS) parameterisation. It includes C++ main utility and Matlab support codes. The C++ code is here to do the heavy lifting, the MATLAB code is here as it was used to prototype and test ideas.

Relevant publications for the 2D RSVS are at the end of this readme.

The compiled binary is available for download for Windows 64bits and Linux 64bits.

## Pre-requisites

For this code to work necessary programs:

- MATLAB installed (2015a or later) - including parallel toolbox
- c++ compiler compatible with MATLAB for the compilation of mex files
- Standalone c++11 compiler for the compilation of console programs (GCC/G++ v7.1 used for development)
- `make` to build the `RSVS3D` executable.
- fortran (90+) compiler for compilation of flow solvers

Required 3rd party open source libraries for compilation:

- Eigen: Library for linear algebra (templated, header only).
- boost/filesytem: Use some filesystem command for interface (needs to be compiled).
- cxxopts: Handling of command line arguments (header only).
- JSON for Modern C++: JSON handling for c++. Used for the parameter handling of the RSVS3D framework (single include header).

Optional 3rd party open source library:

- Tetgen : A Quality Tetrahedral Mesh Generator and a 3D Delaunay Triangulator. Download my modified version for this project payoto/tetgen.

## License

Any code using the Tetgen interface and functionalities is under the GNU Affero GPL license which is more restrictive than the LGPL of this project:

- LGPL : If you write an interface to this project and use it as a library, you can distribute closed source version.
- AGPL : Regardless how you use it and distribute a program you have to open the source.

Refer to the full License terms for more information.

## How do I get set up?

### Matlab

Before being able to call the Matlab codes (these are not necessary for the geometry tool execution).

```
>> Init3DMatlab
>> Include_3DCheckGrid
```

### C++

To Compile and test the C++ code:

```
cd ./SRCC
make testall
testall_RSVS3D
```

If the tests run (they should), to see some example usages;

```
make
RSVS3D.exe -h
```

### Note on using git to update a private copy of the code.

Begginning to use git? Follow these `5mn ELI5 explainer` which will help understand what the lingo means, `git begginer guide` which should get you up and running, or the full git documentation if you're trying to do something `git documentation`.

Very minimal guide I wrote a while back specific to the repository: Updating your files to be up to date with the master branch can be done using git very efficiently. With a few steps.

- Add all your local changes `git add -u` then `git add *.m` then `git add Active_Build*png`
- Commit all your local changes `git commit -m "Add comment about what was done"`
- Switch to the master branch `git checkout master`
- Pull the latest version from the remote repository: `git pull`
- If there are any merge issues resolve them using a text editor (if there are you will need to run `git add -u` and `git commit` before the next step)
- Switch to your local branch `git checkout <your branch name>`
- Merge the new master with your local branch `git merge master`
- If there are any merge issues resolve them using a text editor (if there are you will need to run `git add -u` and `git commit`)

### Getting help

Use the issues board.

## Using the 3D-RSVS

Generating a geoemtry using the 3D-RSVS method only requires the executable `RSVS3D.exe`. For basic usage information from the command line use:

```
RSVS3D --help
```

Warning:

Running `RSVS3D` with no command line arguments does nothing.

### Command line options

Below are all the possible commad line options for the RSVS3D program. These can be assembled in arbitrary ways to run a specific config. The long name is shown (called with prefixed `--` on the command line) followed by

- help (`-h`): Display command line help;
- noexec (`-n`): Do not run the RSVS process and output the configuration file;
- exec (`-e`): Execute the RSVS3D for the default case;
- use-config (`-u`): Use system configuration `STRING` (none specified yet);
- load-config (`-l`): Load a configuration file from `FILE`;
- param (`-p`): Overwrite a specific parameter specified by `KEY:VAL`. "key" is the name of that paramaeter as it appears in the flattened JSON parameter files, "val" is the value of that patameter;
- default-config: Outputs a configuration file with all the default value assigned to the parameters.

### Parameter control

Internally parameters are controlled by a single structure defined in `parameters.h`. Externally parameters are handled using *JSON* files. These provide a good balance of human and machine readable format. And support intricate tree structures and nesting. The *JSON* interaction is handled by an external library `JSON for Modern C++`. This library allows two types of *JSON* files: normal and flat. Default parameter configuration files showing all the parameters and there default options in `default_config` and `default_configflat`. Below are two *JSON* examples.

Example normal *JSON*:

```
{
  "files": {
    "appcasename2outdir": true,
    "ioin": {
      "casename": "",
      "snakemeshname": "",
      "targetfill": "",
      "volumeshname": ""
    },
  },
  "grid": {
    "voxel": {
        "gridsizebackground": [1,1,1],
    },
  },
}
```

Equivalent flat *JSON*:

```
{
  "/files/appcasename2outdir": true,
  "/files/ioin/casename": "",
  "/files/ioin/snakemeshname": "",
  "/files/ioin/targetfill": "",
  "/files/ioin/volumeshname": "",
  "/grid/voxel/gridsizebackground/0": 1,
  "/grid/voxel/gridsizebackground/1": 1,
  "/grid/voxel/gridsizebackground/2": 1,
}
```

Three command line options are currently available for parameter control: the *use-config*, *load-config* and *param* give control over the execution of the RSVS. It permits the control of execution flow, output level and output location as well as specific mesh and volume information. The *use-config*, *load-config* and *param* options can be combined to get the desired set of parameters, multiple ones of each can be called. The program throws an error if a parameter is not recognised or not correctly read from a file. These 3 types of options are parsed in order of their appearance in the help and this (readme): *use-config* then *load-config* and finally *param*. Inputs of the same type are then parsed in their order of appearance.

*Load-config* can load an incomplete set of parameters overwriting only parameters that are specified.

## Non exhaustive parameter list

For up to date parameter list check the default `configuration files`.

### files

Controls the file interaction of the program including the naming of output folders.

```
"/files/appcasename2outdir": true,
"/files/ioin/casename": "",
"/files/ioin/snakemeshname": "",
"/files/ioin/targetfill": "",
"/files/ioin/volumeshname": "",
"/files/ioout/basenameoutdir": "rsvs3d_",
"/files/ioout/basenamepattern": "%y%m%dT%H%M%S_",
"/files/ioout/logginglvl": 2,
"/files/ioout/outdir": "",
"/files/ioout/outputlvl": 2,
"/files/ioout/pathoutdir": "../out",
"/files/ioout/pathpattern": "Archive_%Y_%m/Day_%y-%m-%d",
"/files/ioout/pattern": "",
"/files/ioout/redirectcerr": false,
"/files/ioout/redirectcout": false,
```

#### ioin

- `appcasename2outdir`: append casename to output dir path?
- `casename`: Name of the case.
- `snakemeshname`: Mesh file to load.
- `targetfill`: Unused (see `rsvs`)

### ioout

- `basenameoutdir`: Name of the output directory.
- `basenamepattern`: time format string added to the basenameoutdir.
- `logginglvl`: Depth of data logging 0-minimal, 1-Logs only, 2-Snake history, 3-All data.
- `outdir`: Leave empty to use the automatic archive directory trees, otherwise the output directory.
- `outputlvl`: Depth of final data output.
- `pathoutdir`: Root directory (relative or absolute) for the archiving tree.
- `pathpattern`: Directory stub to use as a time format which will be assembled to generate an archiving output folder pattern.
- `pattern`: Used internally to store the pattern generated by `basenamepattern`.
- `redirectcerr`: redirection of standard error to a file.
- `redirectcout`: redirection of standard output to a file.

### grid

Control the underlying grid if it is generated. It can also be loaded if `"/files/ioin/snakemeshname"` is specified.

- `activegrid` : The type of grid to build (`"voxel"`, `"voronoi"` or `"load"`).
- `domain`: Domain dimensions, each of x, y and z are represented by a lower and upprt bound.
- `gridsizebackground`: Design grid size on which the volume fractions are specified.
- `gridsizesnake`: Snaking mesh as a refinement of the background mesh.
- `distancebox` : for a voronoi VOS mesh the distance outside `domain` at which the bounding points will be placed.
- `inputpoints` : A vector of data containing coordinates used for the Voronoi process.
- `pointfile` : The file from which these are loaded.

Examples: `gridsizebackground=[2, 3, 4]` and `gridsizesnake=[4, 4, 4]` will leed to an actual snaking mesh of `[8, 12, 16]`.

Parameters:

```
"/grid/activegrid": "voxel",
"/grid/domain/0/0": 0.0,
"/grid/domain/0/1": 1.0,
"/grid/domain/1/0": 0.0,
"/grid/domain/1/1": 1.0,
"/grid/domain/2/0": 0.0,
"/grid/domain/2/1": 1.0,
"/grid/stretch/0": 1.0,
"/grid/stretch/1": 1.0,
"/grid/stretch/2": 1.0,
"/grid/voronoi/distancebox": 0.1,
"/grid/voronoi/inputpoints/0": 0.0,
"/grid/voronoi/pointfile": "",
"/grid/voxel/gridsizebackground/0": 1,
"/grid/voxel/gridsizebackground/1": 1,
"/grid/voxel/gridsizebackground/2": 1,
"/grid/voxel/gridsizesnake/0": 6,
"/grid/voxel/gridsizesnake/1": 6,
"/grid/voxel/gridsizesnake/2": 6,
```

### rsvs

RSVS process control. Includes the selection of which volume fraction the 3D-RSVS needs to match.

- `cstfill`: constant fill in all the volume cells.
- `filefill`: Fill is specified in a file (space delimited data).
- `makefill`: Programmaticaly defined fill information.
- `solveralgorithm`: Chooses the solution process for the Quadratic Problem of the RSVS.

Only one of `filefill`, `makefill` or `cstfill` is taken into account if they are all set to *active*. The order of precendence is:

1. `filefill`
2. `makefill`
3. `cstfill`

Parameters:
```
"/rsvs/cstfill/active": false,
"/rsvs/cstfill/fill": 0.5,
"/rsvs/filefill/active": false,
"/rsvs/filefill/fill": "",
"/rsvs/makefill/active": true,
"/rsvs/makefill/fill": "",
"/rsvs/solveralgorithm": 0,
```

### snak

Control the restricted snaking process, can have a large impact on the speed and quality of the convergence of the RSVS process.

- `arrivaltolerance`: Distance from a vertex at which a snaxel is considered "arrived".
- `initboundary`: Initialisation boundary (1 or 0). Which volume fraction boundary should the surface be started at.
- `maxsteps`: Maximum number of snake steps.
- `multiarrivaltolerance`: When two snaxels converge on a vertex what is the radius at which the arrival procedure is triggered.
- `snaxdiststep`: Maximum non-dimensional distance which can be covered by a snaxel in 1 step.
- `snaxtimestep`: Maximum time step (used for damping of the SQP).

Parameters
```
"/snak/arrivaltolerance": 1e-07,
"/snak/initboundary": 1,
"/snak/maxsteps": 50,
"/snak/multiarrivaltolerance": 0.01,
"/snak/snaxdiststep": 0.9,
"/snak/snaxtimestep": 0.9
```

## I don't get it what does this ACTUALLY do and who do I talk to?

For more information about what the code does (i.e. the science of it)

Restricted Snakes: a Flexible Topology Parameterisation Method for Aerodynamic Optimisation

Mixing and Refinement of Design Variables for Geometry and Topology Optimization in Aerodynamics

(Also available on research gate)

Alexandre Payot - a.payot@bristol.ac.uk

ResearchGate profile

Google Scholar profile

personal GitHub/payoto

Research group GitHub/farg-bristol

# Chapter 2

# Namespace Index

## 2.1 Namespace List

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

# Chapter 3

# Hierarchical Index

## 3.1 Class Hierarchy

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

# Chapter 4

# Class Index

## 4.1 Class List

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

# Chapter 5

# File Index

## 5.1 File List

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

# Chapter 6

# Namespace Documentation

## 6.1 param Namespace Reference

Namespace containing the parameter classes used to control execution of the 3D-RSVS program.

### Namespaces

- io

  *Provide functions for reading and writing of the parameter structure.*
- test

  *Tests for the parameter implementation.*

### Classes

- class files

  *Class containing all parameter settings for file operations.*
- struct filltype

  *The input type of fill information.*
- class grid

  *Class for parameters of the grid generation.*
- class ioin

  *Class containing the input configuration these are files to load.*
- class ioout

  *Class containing the output configuration these are files to store and where to store them.*
- class parameters

  *Root class for all the parameters.*
- class rsvs

  *Parameters related to the Velocity calculation and VOS steps.*
- class snaking

  *Parameters controlling tuning parameters for the stepping of the restricted surface.*
- class voronoi

  *Class for handling of voronoi VOS meshing parameters.*
- class voxel

  *Parameters controlling cartesian grid properties.*

## Typedefs

- typedef std::array< double, 2 > realbounds

  *Collects a lower and an upper bound.*
- typedef std::vector< std::pair< std::string, std::string > > exports

  *Collects the export settings which is a vector of pairs of strings.*

## Functions

- template<class T >
  void **to_json** (rsvsjson::json &j, const filltype< T > &p)
- template<class T >
  void **from_json** (const rsvsjson::json &j, filltype< T > &p)
- void **to_json** (rsvsjson::json &j, const rsvs &p)
- void **from_json** (const rsvsjson::json &j, rsvs &p)
- void **to_json** (rsvsjson::json &j, const snaking &p)
- void **from_json** (const rsvsjson::json &j, snaking &p)
- void **to_json** (rsvsjson::json &j, const voxel &p)
- void **from_json** (const rsvsjson::json &j, voxel &p)
- void **to_json** (rsvsjson::json &j, const voronoi &p)
- void **from_json** (const rsvsjson::json &j, voronoi &p)
- void **to_json** (rsvsjson::json &j, const grid &p)
- void **from_json** (const rsvsjson::json &j, grid &p)
- void **to_json** (rsvsjson::json &j, const parameters &p)
- void **from_json** (const rsvsjson::json &j, parameters &p)
- void **to_json** (rsvsjson::json &j, const ioin &p)
- void **from_json** (const rsvsjson::json &j, ioin &p)
- void **to_json** (rsvsjson::json &j, const ioout &p)
- void **from_json** (const rsvsjson::json &j, ioout &p)
- void **to_json** (rsvsjson::json &j, const files &p)
- void **from_json** (const rsvsjson::json &j, files &p)

### 6.1.1 Detailed Description

Namespace containing the parameter classes used to control execution of the 3D-RSVS program.

### 6.1.2 Typedef Documentation

#### 6.1.2.1 exports

```
typedef std::vector<std::pair<std::string,std::string> > param::exports
```

Collects the export settings which is a vector of pairs of strings.

Each pair is: ["valid export type", "export config string"]

Definition at line 39 of file parameters.hpp.

## 6.2 param::io Namespace Reference

Provide functions for reading and writing of the parameter structure.

### Functions

- void **read** (const std::string &fileName, parameters &p)
- void **readflat** (const std::string &fileName, parameters &p)
- void **write** (const std::string &fileName, const parameters &p)
- void **writeflat** (const std::string &fileName, const parameters &p)
- int **updatefromstring** (const std::vector< std::string > &flatjsonKeyVal, parameters &p, const std::string &&sep=std::string(":"))
- void **defaultconf** ()

### 6.2.1 Detailed Description

Provide functions for reading and writing of the parameter structure.

## 6.3 param::test Namespace Reference

Tests for the parameter implementation.

### Functions

- int **base** ()
- int **io** ()
- int **ioflat** ()
- int **ipartialread** ()
- int **prepareforuse** ()
- int **autoflat** ()
- int **symmetry** ()

### 6.3.1 Detailed Description

Tests for the parameter implementation.

## 6.4 rsvs3d Namespace Reference

Namespace for general purpose tools of the RSVS project.

### Classes

- class rsvs_exception

  *Exception for signaling rsvs errors.*

## Functions

- template<class E = rsvs_exception>
  void error (const char ∗message="", const char ∗caller="", const char ∗file="", int line=0, bool throwError=true)

  *Custom error function.*

## 6.4.1 Detailed Description

Namespace for general purpose tools of the RSVS project.

## 6.4.2 Function Documentation

### 6.4.2.1 error()

```
template<class E = rsvs_exception>
void rsvs3d::error (
             const char ∗ message = "",
             const char ∗ caller = "",
             const char ∗ file = "",
             int line = 0,
             bool throwError = true )
```

Custom error function.

Displays the name of the caller function and throw an exception type object with the message specified. can be turned off by setting the last parameter to false.

**Parameters**

| | | |
|---|---|---|
| in | *message* | Error message |
| in | *caller* | Caller function |
| in | *file* | The file in which the caller is. |
| in | *line* | The line at which the caller is. |
| in | *throwError* | should the error be thrown (True) or a warning (False)? |

**Template Parameters**

| | |
|---|---|
| *E* | Exception type to throw |

Convenience macros are also provided to use this function without typing all the file, line and caller function macro names:

- RSVS3D_ERROR(M) : throws the default exception type (std::exception);
- RSVS3D_ERROR_LOGIC(M) : throws std::logic_error;
- RSVS3D_ERROR_ARGUMENT(M) : throws std::invalid_argument;

- RSVS3D_ERROR_TYPE(M, T) : throws T(M);

Definition at line 159 of file warning.hpp.

## 6.5 rsvstest Namespace Reference

Namespace for rsvs tests.

### Classes

- class customtest

  *Class for customtest.*

### Functions

- int **maintest** ()
- int **newtest** ()

### 6.5.1 Detailed Description

Namespace for rsvs tests.

# Chapter 7

# Class Documentation

## 7.1 tetgen::apiparam Class Reference

### Public Member Functions

- void **ReadJsonString** (const std::string &jsonStr)
- apiparam (const std::string &jsonStr)

  *Constructs the object from a json string.*

### Public Attributes

- std::array< double, 3 > lowerB

  *Lower domain bound.*
- std::array< double, 3 > upperB

  *Upper domain bound.*
- std::array< double, 2 > surfedgelengths

  *Controls the surface edgelengths in CFD in the order: {point of lowest curvature, point of highest curvature}.*
- int **curvatureSmoothing**
- std::vector< double > edgelengths

  *Controls the edgelengths at regular intervals.*
- double distanceTol

  *Distance tolerance.*
- bool **generateMeshInside**
- std::string **command**

### 7.1.1 Detailed Description

Definition at line 133 of file tetgenrsvs.hpp.

### 7.1.2 Constructor & Destructor Documentation

#### 7.1.2.1 apiparam()

```
tetgen::apiparam::apiparam (
            const std::string & jsonStr )  [inline]
```

Constructs the object from a json string.

**Parameters**

| | | |
|---|---|---|
| `in` | *jsonStr* | The json string |

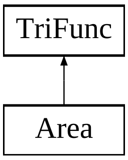
Definition at line 169 of file tetgenrsvs.hpp.

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

- incl/tetgenrsvs.hpp
- src/interfaces/tetgenrsvs.cpp

## 7.2 Area Class Reference

Inheritance diagram for Area:



### Public Member Functions

- void **Calc** () override

### Private Member Functions

- **TriFunc** ()
- **TriFunc** (int a)
- void **PreCalc** ()

### Private Attributes

- vector< double > const ∗ **p0**
- vector< double > const ∗ **p1**
- vector< double > const ∗ **p2**
- double **fun**
- ArrayVec< double > **jac**
- ArrayVec< double > **hes**

### Additional Inherited Members

### 7.2.1 Detailed Description

Definition at line 159 of file RSVSmath.hpp.

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

- incl/RSVSmath.hpp
- src/rsvs/RSVSmath.cpp

## 7.3 tetgenmesh::arraypool Class Reference

### Public Member Functions

- void **restart** ()
- void **poolinit** (int sizeofobject, int log2objperblk)
- char ∗ **getblock** (int objectindex)
- void ∗ **lookup** (int objectindex)
- int **newindex** (void ∗∗newptr)
- **arraypool** (int sizeofobject, int log2objperblk)

### Public Attributes

- int **objectbytes**
- int **objectsperblock**
- int **log2objectsperblock**
- int **objectsperblockmark**
- int **toparraylen**
- char ∗∗ **toparray**
- long **objects**
- unsigned long **totalmemory**

### 7.3.1 Detailed Description

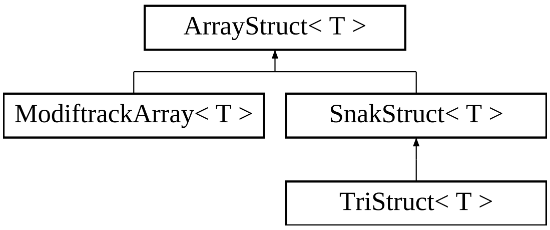
Definition at line 1014 of file tetgen.h.

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

- modules/tetgen/tetgen.h
- modules/tetgen/tetgen.cpp
- modules/tetgen/tetgen.cxx

## 7.4 ArrayStruct< T > Class Template Reference

Inheritance diagram for ArrayStruct< T >:



## Public Member Functions

- void **disp** () const
- void **disp** (const vector< int > &subs) const
- void **disp** (int iStart, int iEnd) const
- int **find** (int key, bool noWarn=false) const
- vector< int > **find_list** (const vector< int > &key, bool noWarn=false) const
- int **GetMaxIndex** () const
- void **Init** (int n)
- bool **isready** () const
- bool **checkready** ()
- void **Concatenate** (const ArrayStruct< T > &other)
- void **PopulateIndices** ()
- void **SetMaxIndex** ()
- void **HashArray** ()
- void **PrepareForUse** ()
- void **ChangeIndices** (int nVert, int nEdge, int nSurf, int nVolu)
- void **write** (FILE ∗fid) const
- void **read** (FILE ∗fid)
- void **remove** (vector< int > delInd)
- void **TightenConnectivity** ()
- int **size** () const
- int **capacity** () const
- void **assign** (int n, T &newelem)
- void **push_back** (T &newelem)
- void **reserve** (int n)
- void **clear** ()
- void **issafeaccess** (const int a)
- const T ∗ **operator()** (const int a) const
- const T ∗ **isearch** (const int b) const
- T & **operator[ ]** (const int a)

## Protected Member Functions

- void **ForceArrayReady** ()
- void **SetLastIndex** ()

## Protected Attributes

- int **maxIndex**
- int **isHash** =0
- int **isSetMI** =0
- bool **readyforuse** =false
- bool **isInMesh** =false
- vector< T > **elems**
- unordered_multimap< int, int > **hashTable**

## Friends

- class **mesh**
- class **snake**
- class **surf**
- int **TestTemplate_ArrayStruct** ()

### 7.4.1 Detailed Description

**template<class T>**
**class ArrayStruct< T >**

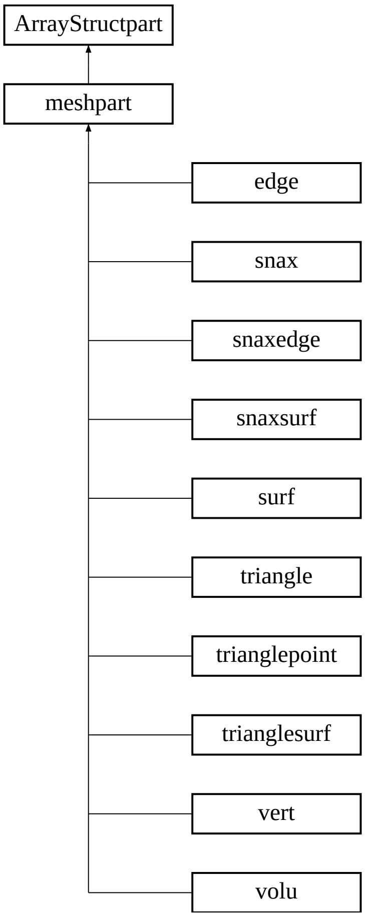
Definition at line 69 of file arraystructures.hpp.

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

- incl/arraystructures.hpp
- incl/arraystructures_incl.cpp

## 7.5 ArrayStructpart Class Reference

Inheritance diagram for ArrayStructpart:



### Public Member Functions

- virtual void **disp** () const =0
- virtual int **Key** () const =0
- virtual void **ChangeIndices** (int nVert, int nEdge, int nSurf, int nVolu)=0
- virtual void **PrepareForUse** ()=0
- virtual bool **isready** (bool isInMesh) const =0
- virtual void **read** (FILE ∗fid)=0
- virtual void **write** (FILE ∗fid) const =0
- virtual void **TightenConnectivity** ()=0

### Public Attributes

- int **index** =0
- bool **isBorder** =false

### 7.5.1 Detailed Description

Definition at line 367 of file arraystructures.hpp.

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

- incl/arraystructures.hpp

## 7.6 ArrayVec< T > Class Template Reference

Template class for vector of vectors (matrix).

```
#include <vectorarray.hpp>
```

### Public Member Functions

- void **assign** (int nR, int nC, T newelem)
- void **size** (int &nR, int &nC) const
- void **clear** ()
- vector< T > & **operator[ ]** (const int a)
- const vector< T > & **operator[ ]** (const int a) const

### Protected Attributes

- vector< vector< T > > **elems**
- vector< int > **dim**

### 7.6.1 Detailed Description

**template<class T>**
**class ArrayVec< T >**

Template class for vector of vectors (matrix).

This is designed to be rectangular.

**Template Parameters**

| | |
|---|---|
| *T* | Type of the vector elements. |

Definition at line 51 of file vectorarray.hpp.

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

- incl/vectorarray.hpp
- incl/vectorarray_incl.cpp

## 7.7 tetgenmesh::badface Class Reference

### Public Attributes

- triface **tt**
- face **ss**
- REAL **key**
- REAL **cent** [6]
- point **forg**
- point **fdest**
- point **fapex**
- point **foppo**
- point **noppo**
- badface ∗ **nextitem**

### 7.7.1 Detailed Description

Definition at line 1108 of file tetgen.h.

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

- modules/tetgen/tetgen.h

## 7.8 ConnecRemv Class Reference

Class containing the information needed to trim objects from a mesh.

```
#include <mesh.hpp>
```

### Public Member Functions

- void **disp** ()

### Public Attributes

- int **keepind**
- int **typeobj**
- vector< int > **rmvind**
- vector< int > **scopeind**

### 7.8.1 Detailed Description

Class containing the information needed to trim objects from a mesh.

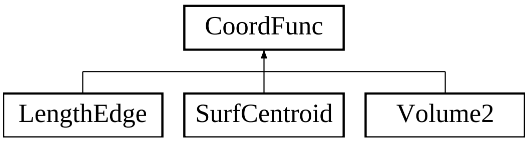
Definition at line 422 of file mesh.hpp.

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

- incl/mesh.hpp
- src/snake/snakeengine.cpp

## 7.9 CoordFunc Class Reference

Inheritance diagram for CoordFunc:



### Public Member Functions

- bool **CheckValid** ()
- bool **MakeValid** ()
- void **PreCalc** ()
- void **assign** (vector< vector< double > const ∗ > &coords)
- void **assign** (int pRepl, const vector< double > &pRep)
- void **ReturnDat** (double &a, ArrayVec< double > &b, ArrayVec< double > &c)
- void **ReturnDat** (ArrayVec< double > &a, ArrayVec< double > &b, ArrayVec< double > &c)
- void **ReturnDatPoint** (double ∗∗a, ArrayVec< double > ∗∗b, ArrayVec< double > ∗∗c)
- void **ReturnDatPoint** (ArrayVec< double > ∗∗a, ArrayVec< double > ∗∗b, ArrayVec< double > ∗∗c)
- virtual void **Calc** ()=0
- void **ResetDim** (int n)
- void **ResetNCoord** (int n)
- void **ResetNFun** (int n)
- **CoordFunc** (int n1)
- **CoordFunc** (int n1, int n2)
- **CoordFunc** (int n1, int n2, int n3)

### Protected Member Functions

- bool MakeValidField (vector< double > const ∗mp)

  *CoordFunc supports the same stuff as tri func but can have any number of points.*
- void **InitialiseArrays** ()

### Protected Attributes

- vector< vector< double > const ∗ > **coords**
- double **fun**
- ArrayVec< double > **funA**
- ArrayVec< double > **jac**
- ArrayVec< double > **hes**
- bool **isReady**
- bool **isCalc**
- int **nDim**
- int **nCoord**
- int **nFun**

### 7.9.1 Detailed Description

Definition at line 78 of file RSVSmath.hpp.

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

- incl/RSVSmath.hpp
- src/rsvs/RSVSmath.cpp

## 7.10 coordvec Class Reference

Handles the use and norm of a vector for which the norm and the unit value might be needed.

```
#include <mesh.hpp>
```

### Public Member Functions

- double **CalcNorm** ()
- double **GetNorm** ()
- double **GetNorm** () const
- void **PrepareForUse** ()
- coordvec **Unit** () const
- double **Unit** (const int a) const
- double **Normalize** ()
- void **assign** (double a, double b, double c)
- double & **operator[ ]** (int a)
- double **operator()** (int a) const
- void **disp** () const
- bool **isready** () const
- const vector< double > & **usedata** () const
- const vector< double > ∗ **retPtr** () const
- void **max** (const vector< double > &vecin)
- void **min** (const vector< double > &vecin)
- void **add** (const vector< double > &vecin)
- void **substract** (const vector< double > &vecin)
- void **substractfrom** (const vector< double > &vecin)
- void **div** (const vector< double > &vecin)
- void **div** (double scalin)
- void **mult** (const vector< double > &vecin)
- void **mult** (double scalin)
- vector< double > **cross** (const vector< double > &vecin) const
- double **dot** (const vector< double > &vecin) const
- double **angle** (const coordvec &coordin) const
- void **operator=** (const vector< double > &a)

### Protected Attributes

- vector< double > **elems**
- double **norm**
- int **isuptodate**

### 7.10.1 Detailed Description

Handles the use and norm of a vector for which the norm and the unit value might be needed.

Implements some simple mathematical operations for coordinate (3-D) vectors.

Definition at line 96 of file mesh.hpp.

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

- incl/mesh.hpp
- src/grid/mesh.cpp

## 7.11 rsvstest::customtest Class Reference

Class for customtest.

```
#include <test.hpp>
```

### Public Member Functions

- **customtest** (const char ∗testNameIn="")
- int **Run** (function< int()> test, const char ∗funcName)
- int RunSilent (function< int()> test, const char ∗funcName)

  *Runs a test function silently except if it returns an error.*
- void **PrintSummary** ()

### Private Attributes

- int **testCount**
- int **errFlag**
- int **errCount**
- int **unhandledError**
- int **prevTime**
- int **runTotal**
- int **lastRunTime**
- std::string **testName**

### 7.11.1 Detailed Description

Class for customtest.

Definition at line 51 of file test.hpp.

### 7.11.2 Member Function Documentation

#### 7.11.2.1 RunSilent()

```
int customtest::RunSilent (
            function< int()> test,
            const char * funcName )
```

Runs a test function silently except if it returns an error.

**Parameters**

| | | |
|---|---|---|
| in | *test* | The test function |
| in | *funcName* | string descriptor for the test. |

**Returns**

int number of errors captured.

Definition at line 145 of file test.cpp.

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

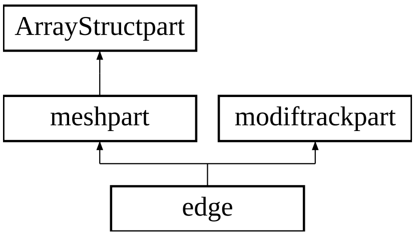
- incl/test.hpp
- src/test/test.cpp

## 7.12 edge Class Reference

Class for an edge object in a mesh.

```
#include <mesh.hpp>
```

Inheritance diagram for edge:



## Public Member Functions

- void **ChangeIndices** (int nVert, int nEdge, int nSurf, int nVolu)
- void **disp** () const
- void **disptree** (const mesh &meshin, int n) const
- void **PrepareForUse** ()
- bool **isready** (bool isInMesh) const
- void **read** (FILE ∗fid)
- void **write** (FILE ∗fid) const
- void **TightenConnectivity** ()
- void GeometricProperties (const mesh ∗meshin, coordvec &centre, double &length) const

  *MAth operations in mesh.*
- double Length (const mesh &meshin) const

  *Calculate the edge length.*
- double LengthSquared (const mesh &meshin) const

  *Calculate squared edge length.*
- bool IsLength0 (const mesh &meshin, double eps=__DBL_EPSILON__) const

  *Returns.*
- bool **vertconneq** (const edge &other) const
- **edge** (const edge &oldEdge)
- void **operator=** (const edge ∗other)
- int **Key** () const

## Public Attributes

- friend **edgearray**
- vector< int > **vertind**
- vector< int > **surfind**

## Friends

- class **mesh**

## Additional Inherited Members

### 7.12.1 Detailed Description

Class for an edge object in a mesh.

Definition at line 307 of file mesh.hpp.

### 7.12.2 Member Function Documentation

#### 7.12.2.1 IsLength0()

```
bool edge::IsLength0 (
             const mesh & meshin,
             double eps = __DBL_EPSILON__ ) const
```

Returns.

**Parameters**

| | | |
|---|---|---|
| in | *meshin* | the mesh in which the edge existes |
| in | *eps* | Tolerance, number under which the length must be to be considered 0. Defaults to **DBL_EPSILON**. |

**Returns**

Wether Length squared is below eps squared.

Definition at line 1003 of file mesh.cpp.

#### 7.12.2.2 Length()

```
double edge::Length (
             const mesh & meshin ) const
```

Calculate the edge length.

**Parameters**

| | | |
|---|---|---|
| in | *meshin* | the mesh in which the edge existes |

**Returns**

the length of the edge

Definition at line 991 of file mesh.cpp.

#### 7.12.2.3 LengthSquared()

```
double edge::LengthSquared (
             const mesh & meshin ) const
```

Calculate squared edge length.

**Parameters**

| | | |
|---|---|---|
| in | *meshin* | the mesh in which the edge existes |

**Returns**

the squared length of the edge

Definition at line 971 of file mesh.cpp.

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

- incl/mesh.hpp
- src/grid/mesh.cpp

## 7.13 tetgenmesh::face Class Reference

### Public Member Functions

- face & **operator=** (const face &s)

### Public Attributes

- shellface ∗ **sh**
- int **shver**

### 7.13.1 Detailed Description

Definition at line 985 of file tetgen.h.

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

- modules/tetgen/tetgen.h

## 7.14 tetgenio::facet Struct Reference

### Public Attributes

- polygon ∗ **polygonlist**
- int **numberofpolygons**
- REAL ∗ **holelist**
- int **numberofholes**

### 7.14.1 Detailed Description

Definition at line 128 of file tetgen.h.

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

- modules/tetgen/tetgen.h

## 7.15 param::files Class Reference

Class containing all parameter settings for file operations.

```
#include <parameters.hpp>
```

### Public Member Functions

- void **PrepareForUse** ()

### Public Attributes

- bool **appcasename2outdir**
- ioin **ioin**
- ioout **ioout**
- exports **exportconfig**

### 7.15.1 Detailed Description

Class containing all parameter settings for file operations.

Definition at line 198 of file parameters.hpp.

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

- incl/parameters.hpp
- src/parameters.cpp

## 7.16 param::filltype< T > Struct Template Reference

The input type of fill information.

```
#include <parameters.hpp>
```

### Public Attributes

- bool **active** =false
- T **fill**

### 7.16.1 Detailed Description

**template<class T>**
**struct param::filltype< T >**

The input type of fill information.

Definition at line 42 of file parameters.hpp.

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

- incl/parameters.hpp

## 7.17 tetgenmesh::flipconstraints Class Reference

### Public Attributes

- int **enqflag**
- int **chkencflag**
- int **unflip**
- int **collectnewtets**
- int **collectencsegflag**
- int **remove_ndelaunay_edge**
- REAL **bak_tetprism_vol**
- REAL **tetprism_vol_sum**
- int **remove_large_angle**
- REAL **cosdihed_in**
- REAL **cosdihed_out**
- int **checkflipeligibility**
- point **seg** [2]
- point **fac** [3]
- point **remvert**

### 7.17.1 Detailed Description

Definition at line 1170 of file tetgen.h.

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

- modules/tetgen/tetgen.h

## 7.18 param::grid Class Reference

Class for parameters of the grid generation.

```
#include <parameters.hpp>
```

### Public Member Functions

- void **PrepareForUse** ()

### Public Attributes

- voxel **voxel**
- voronoi **voronoi**
- std::array< realbounds, 3 > domain

    *Domain size in internal coordinates.*
- std::array< realbounds, 3 > physdomain

    *Physical domain size for export.*
- std::string activegrid

    *The type of grid to use either "voxel" or "voronoi" .*

### 7.18.1 Detailed Description

Class for parameters of the grid generation.

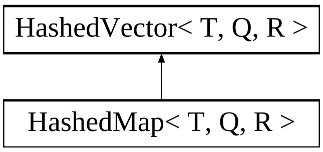
Definition at line 137 of file parameters.hpp.

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

- incl/parameters.hpp
- src/parameters.cpp

## 7.19 HashedMap< T, Q, R > Class Template Reference

Inheritance diagram for HashedMap< T, Q, R >:



### Public Member Functions

- void **GenerateHash** ()

### Public Attributes

- vector< R > **targ**

### 7.19.1 Detailed Description

**template< class T, class Q, class R>**
**class HashedMap< T, Q, R >**

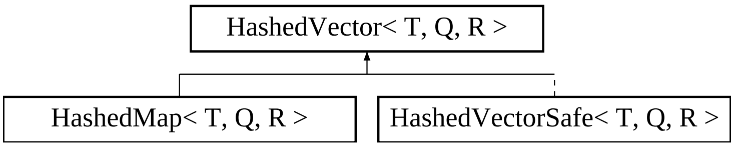
Definition at line 293 of file arraystructures.hpp.

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

- incl/arraystructures.hpp
- incl/arraystructures_incl.cpp

## 7.20 HashedVector< T, Q, R > Class Template Reference

Inheritance diagram for HashedVector< T, Q, R >:



### Public Member Functions

- void **GenerateHash** ()
- int **find** (const T key) const
- vector< int > **findall** (const T key) const
- int **count** (const T key) const
- vector< int > **count** (const vector< T > &key) const
- vector< int > **find_list** (const vector< T > &key) const
- bool **operator()** (const Q &key) const
- bool **IsInVec** (const Q &key) const

### Public Attributes

- vector< T > **vec**
- unordered_multimap< T, R > **hashTable**
- bool **isHash** =false

### 7.20.1 Detailed Description

**template<class T, class Q, class R>**
**class HashedVector< T, Q, R >**

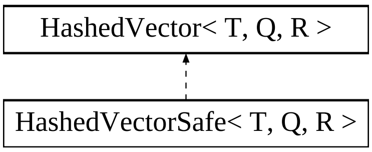
Definition at line 70 of file arraystructures.hpp.

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

- incl/arraystructures.hpp
- incl/arraystructures_incl.cpp

## 7.21 HashedVectorSafe< T, Q, R > Class Template Reference

Inheritance diagram for HashedVectorSafe< T, Q, R >:



### Public Member Functions

- void **operator=** (const vector< T > &a)
- void **operator=** (const HashedVector< T, Q > &a)
- T & **operator[ ]** (const int a)
- const T & **operator[ ]** (const int a) const
- const T & **isearch** (const int b) const

### Additional Inherited Members

### 7.21.1 Detailed Description

**template<class T, class Q, class R = int>**
**class HashedVectorSafe< T, Q, R >**

Definition at line 305 of file arraystructures.hpp.

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

- incl/arraystructures.hpp

## 7.22 tetgenmesh::insertvertexflags Class Reference

### Public Attributes

- int **iloc**
- int **bowywat**
- int **lawson**
- int **splitbdflag**
- int **validflag**
- int **respectbdflag**
- int **rejflag**
- int **chkencflag**
- int **cdtflag**
- int **assignmeshsize**
- int **sloc**
- int **sbowywat**
- int **refineflag**
- triface **refinetet**
- face **refinesh**
- int **smlenflag**
- REAL **smlen**
- point **parentpt**

### 7.22.1 Detailed Description

Definition at line 1127 of file tetgen.h.

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

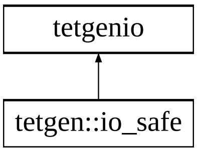
- modules/tetgen/tetgen.h

## 7.23 tetgen::io_safe Class Reference

Class for memory safe interface with tetgen.h.

```
#include <tetgenrsvs.hpp>
```

Inheritance diagram for tetgen::io_safe:



### Public Member Functions

- void **allocate** ()
- void **allocatefacet** (int fIndex)
- void **allocatefacet** (int fIndex, int numPoly)
- void **allocatefacetpolygon** (int fIndex, int pIndex)
- void **allocatefacetpolygon** (int fIndex, int pIndex, int numVerts)
- void **SpecifyTetPointMetric** (int startPnt, int numPnt, const std::vector< double > &mtrs)
- void **SpecifyIndividualTetPointMetric** (int startPnt, int numPnt, const std::vector< double > &mtrs)
- void **SpecifyTetFacetMetric** (int startPnt, int numPnt, int marker)

### Public Attributes

- REAL ∗ **pointlist**
- REAL ∗ **pointattributelist**
- REAL ∗ **pointmtrlist**
- int ∗ **pointmarkerlist**
- int **numberofpointmtrs**
- int ∗ **tetrahedronlist**
- REAL ∗ **tetrahedronattributelist**
- REAL ∗ **tetrahedronvolumelist**
- int ∗ **neighborlist**
- facet ∗ **facetlist**
- int ∗ **facetmarkerlist**
- REAL ∗ **facetconstraintlist**
- int **numberoffacetconstraints**
- REAL ∗ **holelist**
- REAL ∗ **regionlist**
- REAL ∗ **segmentconstraintlist**
- int ∗ **edgelist**
- int ∗ **edgemarkerlist**
- int ∗ **o2edgelist**
- int ∗ **edge2tetlist**
- int ∗ **face2edgelist**
- int ∗ **face2tetlist**
- REAL ∗ **vpointlist**
- voroedge ∗ **vedgelist**
- vorofacet ∗ **vfacetlist**
- int ∗∗ **vcelllist**

### Additional Inherited Members

### 7.23.1 Detailed Description

Class for memory safe interface with tetgen.h.

This class provides a method called `allocate` which allocates the memory for the io arrays using the new command. Command `deallocate` can be used to free the memory before destruction, or otherwise it is called uppon when object goes out of scope.

Definition at line 51 of file tetgenrsvs.hpp.

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

- incl/tetgenrsvs.hpp
- src/interfaces/tetgenrsvs.cpp

## 7.24 param::ioin Class Reference

Class containing the input configuration these are files to load.

```
#include <parameters.hpp>
```

### Public Member Functions

- void **PrepareForUse** ()

### Public Attributes

- std::string **snakemeshname**
- std::string **volumeshname**
- std::string **targetfill**
- std::string **casename**

### 7.24.1 Detailed Description

Class containing the input configuration these are files to load.

Definition at line 156 of file parameters.hpp.

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

- incl/parameters.hpp
- src/parameters.cpp

## 7.25 param::ioout Class Reference

Class containing the output configuration these are files to store and where to store them.

```
#include <parameters.hpp>
```

### Public Member Functions

- void **PrepareForUse** ()

### Public Attributes

- std::string **pathoutdir**
- std::string **pathpattern**
- std::string **basenamepattern**
- std::string **basenameoutdir**
- std::string **outdir**
- std::string **pattern**
- bool **redirectcout**
- bool **redirectcerr**
- int **logginglvl**
- int **outputlvl**

### 7.25.1 Detailed Description

Class containing the output configuration these are files to store and where to store them.

This automatically parses output directory patterns to produce archival folders with time stamps and logical numbering.

Definition at line 173 of file parameters.hpp.

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

- incl/parameters.hpp
- src/parameters.cpp

## 7.26 integrate::iteratereturns Class Reference

### Public Member Functions

- **iteratereturns** (int n, int s, double t)

### Public Attributes

- int **nVoluZone** =0
- int **stepNum** =0
- double **timeT** =0.0

### 7.26.1 Detailed Description

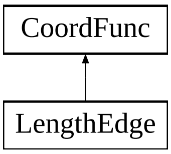
Definition at line 55 of file RSVSintegration.hpp.

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

- incl/RSVSintegration.hpp

## 7.27 LengthEdge Class Reference

Inheritance diagram for LengthEdge:



### Public Member Functions

- void **Calc** () override

### Private Member Functions

- void **PreCalc** ()

### Private Attributes

- vector$<$ vector$<$ double $>$ const $*$ $>$ **coords**
- double **fun**
- ArrayVec$<$ double $>$ **jac**
- ArrayVec$<$ double $>$ **hes**

### Additional Inherited Members

### 7.27.1 Detailed Description

Definition at line 174 of file RSVSmath.hpp.

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

- incl/RSVSmath.hpp
- src/rsvs/RSVSmath.cpp

## 7.28 tetgenmesh::memorypool Class Reference

### Public Member Functions

- **memorypool** (int, int, int, int)
- void **poolinit** (int, int, int, int)
- void **restart** ()
- void ∗ **alloc** ()
- void **dealloc** (void ∗)
- void **traversalinit** ()
- void ∗ **traverse** ()

### Public Attributes

- void ∗∗ **firstblock**
- void ∗∗ **nowblock**
- void ∗ **nextitem**
- void ∗ **deaditemstack**
- void ∗∗ **pathblock**
- void ∗ **pathitem**
- int **alignbytes**
- int **itembytes**
- int **itemwords**
- int **itemsperblock**
- long **items**
- long **maxitems**
- int **unallocateditems**
- int **pathitemsleft**

### 7.28.1 Detailed Description

Definition at line 1066 of file tetgen.h.

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

- modules/tetgen/tetgen.h
- modules/tetgen/tetgen.cpp
- modules/tetgen/tetgen.cxx

## 7.29 mesh Class Reference

Class for mesh handling.

```
#include <mesh.hpp>
```

### Public Member Functions

- void **RemoveFromFamily** ()
- void **AddChild** (mesh ∗meshin)
- void **AddParent** (mesh ∗meshin)
- void **AddParent** (mesh ∗meshin, vector< int > &parentind)
- void **AddChild** (mesh ∗meshin, vector< int > &parentind)
- void **SetMeshDepElm** ()
- void **MaintainLineage** ()
- int **CountParents** () const
- int **SurfInParent** (int surfind) const
- void **SurfInParent** (vector< int > &listInParent) const
- void **ElmOnParentBound** (vector< int > &listInParent, vector< int > &voluInd, bool isBorderBound=true, bool outerVolume=true) const
- void **SurfOnParentBound** (vector< int > &listInParent, vector< int > &voluInd, bool isBorderBound, bool outerVolume) const
- void **EdgeOnParentBound** (vector< int > &listInParent, vector< int > &voluInd, bool isBorderBound, bool outerVolume) const
- int **CountVoluParent** () const
- void **ReturnParentMap** (vector< int > &currind, vector< int > &parentpos, vector< pair< int, int >> &parentcases, vector< double > &voluVals) const
- void **MapVolu2Parent** (const vector< double > &fillIn, const vector< pair< int, int >> &parentcases, double volu::∗mp=&volu::fill)
- void **MapVolu2Self** (const vector< double > &fillIn, const vector< int > &elms, double volu::∗mp=&volu::fill)
- void **VoluValuesofParents** (int elmInd, vector< double > &vals, int volType=0) const
- void **VoluValuesofParents** (int elmInd, vector< double > &vals, double volu::∗mp) const
- void **SurfValuesofParents** (int elmInd, vector< double > &vals, int volType=0) const
- void **SurfValuesofParents** (int elmInd, vector< double > &vals, double surf::∗mp) const
- int **ParentElementIndex** (int childElmInd, int parentInd=0) const
- int **WhatDim** () const
- void **HashArray** ()
- void **SetMaxIndex** ()
- void **GetMaxIndex** (int ∗nVert, int ∗nEdge, int ∗nSurf, int ∗nVolu) const
- void **Init** (int nVe, int nE, int nS, int nVo)
- void **size** (int &nVe, int &nE, int &nS, int &nVo) const
- void **reserve** (int nVe, int nE, int nS, int nVo)
- void **PrepareForUse** (bool needOrder=true)
- void **disp** () const
- void **displight** () const
- void **Concatenate** (const mesh &other)
- bool **isready** () const
- void **PopulateIndices** ()
- void **TightenConnectivity** ()
- int **TestConnectivity** (const char ∗strRoot="") const
- int **TestConnectivityBiDir** (const char ∗strRoot="", bool emptyIsErr=true) const
- void **write** (FILE ∗fid) const
- void **read** (FILE ∗fid)
- int **write** (const char ∗str) const
- int **read** (const char ∗str)
- void **MakeCompatible_inplace** (mesh &other) const
- mesh **MakeCompatible** (mesh other) const
- void **ChangeIndices** (int nVert, int nEdge, int nSurf, int nVolu)
- void **SwitchIndex** (int typeInd, int oldInd, int newInd, const vector< int > &scopeInd={0})
- void **RemoveIndex** (int typeInd, int oldInd)
- int ConnectedVertex (vector< int > &vertBlock) const

*Return in a vector for each vertex a block number which it is part of.*

- int **ConnectedVolumes** (vector< int > &volBlock, const vector< bool > &boundaryFaces={}) const
- void **ForceCloseContainers** ()
- void **RemoveSingularConnectors** (const std::vector< int > &rmvVertInds={}, bool voidError=true)
- std::vector< int > **MergeGroupedVertices** (HashedVector< int, int > &closeVert, bool delVerts=true)
- vector< int > **OrderEdges** ()
- void **SetBorders** ()
- void **OrientFaces** ()
- void **GetOffBorderVert** (vector< int > &vertList, vector< int > &voluInd, int outerVolume=-1)
- void **GetOffBorderVert** (vector< int > &vertList, vector< int > &voluInd, int outerVolume=-1) const
- void **GetOffBorderVert3D** (vector< int > &vertList, vector< int > &voluInd, int outerVolume=-1) const
- void **GetOffBorderVert2D** (vector< int > &vertInd, vector< int > &surfind, int outerVolume=-1) const
- coordvec **CalcCentreVolu** (int ind) const
- coordvec **CalcPseudoNormalSurf** (int ind) const
- vector< int > VertexInVolume (const vector< double > testVertices, int sizeVert=3) const

  *Finds for each vertex, the volume object containing it.*
- grid::transformation **Scale** ()
- grid::transformation **Scale** (const grid::limits &domain)
- void LinearTransform (const grid::transformation &transform)

  *Applies a linear transformation to the points on a grid.*
- void LinearTransformFamily (const grid::transformation &transform)

  *Applies a linear transform to child and parent meshes.*
- void **LoadTargetFill** (const std::string &fileName)
- grid::limits **BoundingBox** () const
- void **ReturnBoundingBox** (std::array< double, 3 > &lowerB, std::array< double, 3 > &upperB) const
- void **Crop** (vector< int > indList, int indType=1)
- vector< int > AddBoundary (const vector< double > &lb, const vector< double > &ub)

  *Adds boundaries alond max and min xyz planes.*
- void **CropAtBoundary** (const vector< double > &lb, const vector< double > &ub)

## Public Attributes

- vertarray **verts**
- edgearray **edges**
- surfarray **surfs**
- voluarray **volus**
- meshdependence **meshtree**

## Private Member Functions

- void **SetLastIndex** ()
- void **OrientSurfaceVolume** ()
- void **OrientEdgeSurface** ()
- int **OrientRelativeSurfaceVolume** (vector< int > &surfOrient)
- void **ArraysAreHashed** ()
- void _LinearTransformGeneration (const grid::transformation &transform, vector< mesh ∗ > meshdependence::∗mp)

  *Applies reccursively linear transforms to a tree of meshes.*

## Private Attributes

- bool **borderIsSet** =false
- bool **meshDepIsSet** =false
- bool **facesAreOriented** =false
- int **meshDim** =0

## Friends

- class **snake**

### 7.29.1 Detailed Description

Class for mesh handling.

Class implementing the functionality of this file. The mesh class allow, the robust evolution of a grid. Element connectivity is stored bi-directionnaly. This allows no connectivity to need to be infered and allows very fast and robust traversing of the mesh by using hashed lists of the indices of the mesh components.

Definition at line 477 of file mesh.hpp.

### 7.29.2 Member Function Documentation

#### 7.29.2.1 _LinearTransformGeneration()

```
void mesh::_LinearTransformGeneration (
             const grid::transformation & transform,
             vector< mesh * > meshdependence::* mp )  [private]
```

Applies reccursively linear transforms to a tree of meshes.

**Parameters**

| | | |
|---|---|---|
| in | *transform* | The transform |
| in | *meshdependence* | A member pointer to either the parent meshes or the child meshes of the meshtree. |

Definition at line 3986 of file mesh.cpp.

#### 7.29.2.2 AddBoundary()

```
std::vector< int > mesh::AddBoundary (
             const vector< double > & lb,
             const vector< double > & ub )
```

Adds boundaries alond max and min xyz planes.

Arguments

**Parameters**

| | | |
|---|---|---|
| in | *lb* | lower boundary vector of 3 doubles. |
| in | *ub* | upper boundary vector of 3 doubles. |

**Returns**

List of vertex indices in the mesh which lie outside.

Raises:

- logic_error,
-
-

Process: THis method could be readily refactored to allow treatment of more complex boundaries

Steps: 1 - Identify vertices lying outside 2 - Indentify connectors lying on boundary a - edges b - surfs c - volus 3 - Introduce boundary vertices (BV) 4 - Connect those BV to form new boundary edges (BE) 5 - Assemble BEs inside a volu into a boundary surf (BS) (This process is similar to the voronoisation)

Definition at line 4092 of file mesh.cpp.

#### 7.29.2.3 ConnectedVertex()

```
int mesh::ConnectedVertex (
            vector< int > & vertBlock ) const
```

Return in a vector for each vertex a block number which it is part of.

Fills a vector with a number for each vertex corresponding to a group of connected edges it is part of , can be used close surfaces in 2D or volumes in 3D. Uses a flood fill with queue method.

**Parameters**

| | |
|---|---|
| *[in/out]* | vertBlock Either a vector of the same size contaigning 0 for vertices which need to be labelled and some other integers in other positions. OR an empty vector. |

**Returns**

The total number of blocks of vertices identified.

Definition at line 3414 of file mesh.cpp.

#### 7.29.2.4 LinearTransform()

```
void mesh::LinearTransform (
             const grid::transformation & transform )
```

Applies a linear transformation to the points on a grid.

**Parameters**

| | | |
|---|---|---|
| in | *transform* | The transform to apply. |

Definition at line 3956 of file mesh.cpp.

#### 7.29.2.5 LinearTransformFamily()

```
void mesh::LinearTransformFamily (
             const grid::transformation & transform )
```

Applies a linear transform to child and parent meshes.

**Parameters**

| | | |
|---|---|---|
| in | *transform* | The transform |

Definition at line 3972 of file mesh.cpp.

#### 7.29.2.6 VertexInVolume()

```
vector< int > mesh::VertexInVolume (
             const vector< double > testVertices,
             int sizeVert = 3 ) const
```

Finds for each vertex, the volume object containing it.

This only works robustly for outside points for convex meshes.

**Parameters**

| | | |
|---|---|---|
| in | *testVertices* | The test vertices |
| in | *sizeVert* | The size of each vertex data |

**Returns**

returns a list of indices containing the same number of values as there are input vertices (testVertices/sizeVert)

Definition at line 352 of file mesh.cpp.

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

- incl/mesh.hpp
- src/grid/mesh.cpp

## 7.30 meshdependence Class Reference

Class for connecting meshes.

```
#include <mesh.hpp>
```

### Protected Member Functions

- int **AddParent** (mesh ∗meshin)
- int **AddChild** (mesh ∗meshin)
- void **AddParent** (mesh ∗meshin, vector< int > &parentind)
- void **RemoveChild** (mesh ∗meshin)
- void **RemoveParent** (mesh ∗meshin)

### Protected Attributes

- int nParents = 0

  *Number of parent meshes.*
- vector< int > elemind

  *Indices of the active elements of the owning mesh.*
- vector< mesh ∗ > parentmesh

  *Vector of pointers to the mesh which are coarser (parents).*
- vector< mesh ∗ > childmesh

  *Vector of pointers to the mesh which are finer (children).*
- vector< HashedVectorSafe< int, int > > parentconn

  *parent/to self connectivity, 1 vector element per parent.*

### Friends

- class **mesh**

### 7.30.1 Detailed Description

Class for connecting meshes.

Stores a vector of mesh references for parent and children. Needs to support partial meshes for constraint handling.

Definition at line 439 of file mesh.hpp.

### 7.30.2 Member Data Documentation

### 7.30.2.1 parentconn

```
vector<HashedVectorSafe<int,int> > meshdependence::parentconn  [protected]
```

parent/to self connectivity, 1 vector element per parent.

This is an vector with the index of each parent element stored at the location of each self element.

Definition at line 453 of file mesh.hpp.

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

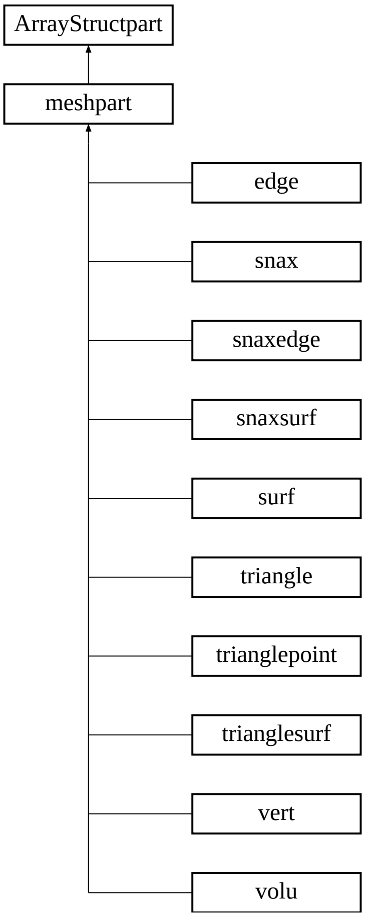
- incl/mesh.hpp
- src/grid/mesh.cpp

## 7.31 meshpart Class Reference

/Abstract class to ensure mesh interfaces are correct.

```
#include <mesh.hpp>
```

Inheritance diagram for meshpart:



## Public Member Functions

- virtual void **disptree** (const mesh &meshin, int n) const =0

## Additional Inherited Members

### 7.31.1 Detailed Description

/Abstract class to ensure mesh interfaces are correct.

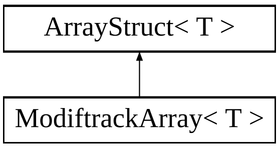
Definition at line 155 of file mesh.hpp.

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

- incl/mesh.hpp

## 7.32 ModiftrackArray< T > Class Template Reference

Inheritance diagram for ModiftrackArray< T >:



## Public Member Functions

- void **SetNoModif** ()
- void **ReturnModifInd** (vector< int > &vecind)
- void **ReturnModifLog** (vector< bool > &modiflog)
- T & **operator[ ]** (const int a)

## Friends

- class **mesh**
- class **snake**

## Additional Inherited Members

### 7.32.1 Detailed Description

**template<class T>**
**class ModiftrackArray< T >**

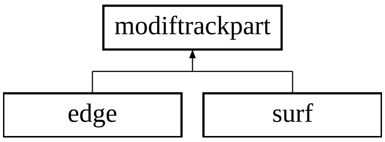
Definition at line 255 of file arraystructures.hpp.

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

- incl/arraystructures.hpp
- incl/snakstruct_incl.cpp

## 7.33 modiftrackpart Class Reference

Inheritance diagram for modiftrackpart:



### Public Member Functions

- bool **returnIsModif** () const

### Protected Attributes

- bool **isModif** =true

### 7.33.1 Detailed Description

Definition at line 389 of file arraystructures.hpp.

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

- incl/arraystructures.hpp

## 7.34 tetgenmesh::optparameters Class Reference

### Public Attributes

- int **max_min_volume**
- int **min_max_aspectratio**
- int **min_max_dihedangle**
- REAL **initval**
- REAL **imprval**
- int **numofsearchdirs**
- REAL **searchstep**
- int **maxiter**
- int **smthiter**

### 7.34.1 Detailed Description

Definition at line 1228 of file tetgen.h.

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

- modules/tetgen/tetgen.h

## 7.35 param::parameters Class Reference

Root class for all the parameters.

```
#include <parameters.hpp>
```

### Public Member Functions

- void **PrepareForUse** ()

### Public Attributes

- rsvs **rsvs**
- snaking **snak**
- grid **grid**
- files **files**

### 7.35.1 Detailed Description

Root class for all the parameters.

Definition at line 213 of file parameters.hpp.

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

- incl/parameters.hpp
- src/parameters.cpp

## 7.36 tetgenio::pointparam Struct Reference

### Public Attributes

- REAL **uv** [2]
- int **tag**
- int **type**

### 7.36.1 Detailed Description

Definition at line 162 of file tetgen.h.

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

- modules/tetgen/tetgen.h

## 7.37 tetgenio::polygon Struct Reference

### Public Attributes

- int ∗ **vertexlist**
- int **numberofvertices**

### 7.37.1 Detailed Description

Definition at line 120 of file tetgen.h.

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

- modules/tetgen/tetgen.h

## 7.38 param::rsvs Class Reference

Parameters related to the Velocity calculation and VOS steps.

```
#include <parameters.hpp>
```

### Public Member Functions

- void **PrepareForUse** ()

### Public Attributes

- int solveralgorithm

  *Algorithm used by Eigen to solve the SQP system.*
- filltype< double > cstfill

  *Fill the VOS values with a constant value.*
- filltype< std::string > filefill

  *Fill the VOS values from file filefill.fill.*
- filltype< std::string > makefill

  *Fill the VOS values from a run time function accessible from makefill.fill.*

### 7.38.1 Detailed Description

Parameters related to the Velocity calculation and VOS steps.

Definition at line 49 of file parameters.hpp.

### 7.38.2 Member Data Documentation

#### 7.38.2.1 solveralgorithm

```
int param::rsvs::solveralgorithm
```

Algorithm used by Eigen to solve the SQP system.

See RSVScalc::SQPStep for details of the valid options.

Definition at line 57 of file parameters.hpp.

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

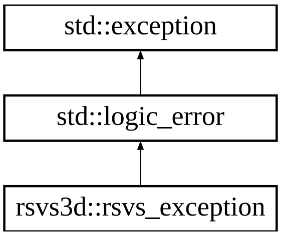
- incl/parameters.hpp
- src/parameters.cpp

## 7.39 rsvs3d::rsvs_exception Class Reference

Exception for signaling rsvs errors.

```
#include <warning.hpp>
```

Inheritance diagram for rsvs3d::rsvs_exception:



### 7.39.1 Detailed Description

Exception for signaling rsvs errors.

Definition at line 47 of file warning.hpp.

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

- incl/warning.hpp

## 7.40 RSVScalc Class Reference

Class to handle the RSVS calculation.

```
#include <RSVScalc.hpp>
```

### Public Member Functions

- void BuildMathArrays (int nDv, int nConstr)

  *Builds mathematics arrays.*
- void BuildConstrMap (const triangulation &triangleRSVS)

  *Builds the constraint mapping.*
- void BuildConstrMap (const mesh &meshin)

  *Builds the constraint mapping.*
- int BuildDVMap (const std::vector< int > &vecin)

  *Builds a Design variable map.*
- bool SnakDVcond (const triangulation &triRSVS, int ii)

  *Returns wether a snaxel is a design variable or not.*
- void PrepTriangulationCalc (const triangulation &triRSVS)

  *Groups actions needed before the calculation of triangular quantities.*
- void CalculateMesh (mesh &meshin)

  *Calculates the mesh volumes.*
- void CalculateTriangulation (const triangulation &triRSVS, int derivMethod=0)

  *Calculates the triangulation volume and area derivatives.*
- void CalcTriangle (const triangle &triIn, const triangulation &triRSVS, bool isObj=true, bool isConstr=true, bool isDeriv=true)

  *Calculates the properties of single triangle.*
- void CalcTriangleFD (const triangle &triIn, const triangulation &triRSVS, bool isObj=true, bool isConstr=true, bool isDeriv=true)

  *Calculates the properties of single triangle using Finite difference.*
- void CalcTriangleDirectVolume (const triangle &triIn, const triangulation &triRSVS, bool isObj=true, bool isConstr=true, bool isDeriv=true)

  *Calculates the properties of single triangle using direct calculation.*
- void CalcTriangleEdgeLength (const triangle &triIn, const triangulation &triRSVS, bool isObj=true, bool isConstr=true, bool isDeriv=true)

  *Calculates the properties of single triangle for 2D RSVS.*
- void ReturnConstrToMesh (triangulation &triRSVS) const

  *Returns a constraint to the triangulation::meshDep.*
- void ReturnConstrToMesh (mesh &meshin, double volu::∗mp=&volu::volume) const

  *Returns a constraint to the mesh.*
- void CheckAndCompute (int calcMethod=0)

  *Prepare the active arrays for SQP calculation and calculate the SQP step.*
- void ComputeSQPstep (int calcMethod, MatrixXd &dConstrAct, RowVectorXd &dObjAct, VectorXd &constrAct, VectorXd &lagMultAct)

  *Calculates the next SQP step.*
- bool PrepareMatricesForSQP (MatrixXd &dConstrAct, MatrixXd &HConstrAct, MatrixXd &HObjAct, RowVectorXd &dObjAct, VectorXd &constrAct, VectorXd &lagMultAct)

  *Prepares the matrices needed for the SQP step calculation.*
- void ReturnVelocities (triangulation &triRSVS)

  *Returns velocities to the snaxels.*
- int numConstr ()

  *Getter for the number of constraints.*
- void Print2Screen (int outType=0) const

  *Prints different amounts of `RSVScalc` owned data to the screen.*
- void ConvergenceLog (ofstream &out, int loglvl=3) const

  *Print convergence information to file stream.*

## Public Attributes

- MatrixXd dConstr

  *Constraint Jacobian, size: [nConstr, nDv].*
- MatrixXd HConstr

  *Constraint Hessian, size: [nDv, nDv].*
- MatrixXd HObj

  *Objective Hessian, size: [nDv, nDv].*
- MatrixXd HLag

  *Lagrangian Hessian, size: [nDv, nDv].*
- RowVectorXd dObj

  *Objective Jacobian, size: [1, nDv].*
- VectorXd constr

  *Constraint value vector, size: [nConstr, 1].*
- VectorXd lagMult

  *Lagrangian multiplier, size: [nConstr, 1].*
- VectorXd deltaDV

  *Change in design variable, assigned to snake velocity, size: [nDv, 1].*
- VectorXd constrTarg

  *Constraint target values, size: [nConstr, 1].*
- MatrixXd **dvCallConstr**
- double obj =0.0

  *Objective function value.*
- double limLag = INFINITY

  *Value at which a Lagrangian multiplier is considered problematically large.*
- std::vector< bool > isConstrAct

  *is the corresponding constraint active?*
- std::vector< bool > isDvAct

  *Is the corresponding design variable active?*
- std::vector< int > subConstrAct

  *Vector of subscripts of the active constraints.*
- std::vector< int > subDvAct

  *Vector of subscripts of the active design variables.*
- HashedVector< int, int > dvMap

  *Maps the snake indices to the position in the design variable vector.*
- HashedMap< int, int, int > constrMap

  *maps snakemesh volu onto constr*
- std::vector< pair< int, int > > constrList

  *keeps pairs with parentindex and voluindex*

## Protected Attributes

- int nDv =0

  *Number of design variables.*
- int nConstr =0

  *Number of constraints.*
- int falseaccess =0

  *Number of false access operations.*
- bool returnDeriv =true

  *Return the derivatives (obsolete/unused)*

### 7.40.1 Detailed Description

Class to handle the RSVS calculation.

This class calculates volume and area metrics in a triangulated snake to update the velocity and volumes. It uses an SQP algorithm to compute the velocities.

Definition at line 39 of file RSVScalc.hpp.

### 7.40.2 Member Function Documentation

#### 7.40.2.1 BuildConstrMap() [1/2]

```
void RSVScalc::BuildConstrMap (
            const triangulation & triangleRSVS )
```

Builds the constraint mapping.

**Parameters**

| | | |
|---|---|---|
| in | *triangleRSVS* | Triangulation containing the RSVS. |

Definition at line 295 of file RSVScalc.cpp.

#### 7.40.2.2 BuildConstrMap() [2/2]

```
void RSVScalc::BuildConstrMap (
            const mesh & meshin )
```

Builds the constraint mapping.

**Parameters**

| | | |
|---|---|---|
| in | *meshin* | mesh for constraint building. |

Definition at line 312 of file RSVScalc.cpp.

#### 7.40.2.3 BuildDVMap()

```
int RSVScalc::BuildDVMap (
            const std::vector< int > & vecin )
```

Builds a Design variable map.

**Parameters**

| | | |
|---|---|---|
| in | *vecin* | The input vector of design variable indices. |

**Returns**

The number of design variable.

Definition at line 328 of file RSVScalc.cpp.

#### 7.40.2.4 BuildMathArrays()

```
void RSVScalc::BuildMathArrays (
             int nDv,
             int nConstr )
```

Builds mathematics arrays.

**Parameters**

| | | |
|---|---|---|
| in | *nDv* | Number of design variables. |
| in | *nConstr* | Number of constraints. |

Definition at line 268 of file RSVScalc.cpp.

#### 7.40.2.5 CalcTriangle()

```
void RSVScalc::CalcTriangle (
             const triangle & triIn,
             const triangulation & triRSVS,
             bool isObj = true,
             bool isConstr = true,
             bool isDeriv = true )
```

Calculates the properties of single triangle.

These values are returned to the class math arrays.

**Parameters**

| | | |
|---|---|---|
| in | *triIn* | The triangle to measure. |
| in | *triRSVS* | The containing triangulation object. |
| in | *isObj* | Calculate objective? |
| in | *isConstr* | Calculate constraint? |
| in | *isDeriv* | Calculate derivatives? |

Definition at line 61 of file RSVScalc_core.cpp.

#### 7.40.2.6 CalcTriangleDirectVolume()

```
void RSVScalc::CalcTriangleDirectVolume (
             const triangle & triIn,
             const triangulation & triRSVS,
             bool isObj = true,
             bool isConstr = true,
             bool isDeriv = true )
```

Calculates the properties of single triangle using direct calculation.

These values are returned to the class math arrays.

**Parameters**

| | | |
|---|---|---|
| in | *triIn* | The triangle to measure. |
| in | *triRSVS* | The containing triangulation object. |
| in | *isObj* | Calculate objective? |
| in | *isConstr* | Calculate constraint? |
| in | *isDeriv* | Calculate derivatives? |

<-—Change assignement

<-—Change assignement

Definition at line 436 of file RSVScalc_core.cpp.

#### 7.40.2.7 CalcTriangleEdgeLength()

```
void RSVScalc::CalcTriangleEdgeLength (
             const triangle & triIn,
             const triangulation & triRSVS,
             bool isObj = true,
             bool isConstr = true,
             bool isDeriv = true )
```

Calculates the properties of single triangle for 2D RSVS.

These values are returned to the class math arrays.

**Parameters**

| | | |
|---|---|---|
| in | *triIn* | The triangle to measure. |
| in | *triRSVS* | The containing triangulation object. |
| in | *isObj* | Calculate objective? |
| in | *isConstr* | Calculate constraint? |
| in | *isDeriv* | Calculate derivatives? |

Definition at line 705 of file RSVScalc_core.cpp.

#### 7.40.2.8 CalcTriangleFD()

```
void RSVScalc::CalcTriangleFD (
            const triangle & triIn,
            const triangulation & triRSVS,
            bool isObj = true,
            bool isConstr = true,
            bool isDeriv = true )
```

Calculates the properties of single triangle using Finite difference.

These values are returned to the class math arrays.

**Parameters**

| | | |
|---|---|---|
| in | *triIn* | The triangle to measure. |
| in | *triRSVS* | The containing triangulation object. |
| in | *isObj* | Calculate objective? |
| in | *isConstr* | Calculate constraint? |
| in | *isDeriv* | Calculate derivatives? |

Definition at line 251 of file RSVScalc_core.cpp.

#### 7.40.2.9 CalculateMesh()

```
void RSVScalc::CalculateMesh (
            mesh & meshin )
```

Calculates the mesh volumes.

**Parameters**

| | |
|---|---|
| *meshin* | The mesh. |

Definition at line 132 of file RSVScalc.cpp.

#### 7.40.2.10 CalculateTriangulation()

```
void RSVScalc::CalculateTriangulation (
            const triangulation & triRSVS,
            int derivMethod = 0 )
```

Calculates the triangulation volume and area derivatives.

**Parameters**

| | | |
|---|---|---|
| in | *triRSVS* | The triangle rsvs |
| in | *derivMethod* | The differentiation method to use. 1 : Finite Difference, 2 : Direct calculation, all others : differentiation. |

Definition at line 76 of file RSVScalc.cpp.

#### 7.40.2.11 CheckAndCompute()

```
void RSVScalc::CheckAndCompute (
            int calcMethod = 0 )
```

Prepare the active arrays for SQP calculation and calculate the SQP step.

**Parameters**

| | | |
|---|---|---|
| in | *calcMethod* | Calculation method for SQP. Check :meth:RSVScalc::ComputeSQPstep for detail. |

Definition at line 98 of file RSVScalc_SQP.cpp.

#### 7.40.2.12 ComputeSQPstep()

```
void RSVScalc::ComputeSQPstep (
            int calcMethod,
            MatrixXd & dConstrAct,
            RowVectorXd & dObjAct,
            VectorXd & constrAct,
            VectorXd & lagMultAct )
```

Calculates the next SQP step.

In normal operation the constraint should be 0 through 4. With 0 the default. By adding 10 to these values the "constraint only" mode is enabled which performs a gradient descent step based on the constraint.

**Parameters**

| | | |
|---|---|---|
| in | *calcMethod* | The calculation method. 10 can be added to all values to enable the "constraint only" mode. Values correspond to the following: `Eigen::HouseholderQR` (1); ∗ `Eigen::ColPivHouseholderQR` (2) - Default; Eigen::LLT<MatrixXd> (3); `Eigen::PartialPivLU` (4); |
| | *dConstrAct* | The active constraint Jacobian |
| | *dObjAct* | The active objective Jacobian |
| | *constrAct* | The active constraint values |
| | *lagMultAct* | The active lagrangian multipliers. |

Definition at line 123 of file RSVScalc_SQP.cpp.

#### 7.40.2.13 ConvergenceLog()

```
void RSVScalc::ConvergenceLog (
             ofstream & out,
             int loglvl = 3 ) const
```

Print convergence information to file stream.

**Parameters**

| | | |
|---|---|---|
| | *out* | The output filestream |
| in | *loglvl* | The logging detail to output. <1 nothing, ==1 Vector statistics, ==2 ...and constraint vectors, >2 ...and snaxel velocity vector. |

Definition at line 336 of file RSVScalc.cpp.

#### 7.40.2.14 numConstr()

```
int RSVScalc::numConstr ( )  [inline]
```

Getter for the number of constraints.

**Returns**

The number of constraints.

Definition at line 307 of file RSVScalc.hpp.

#### 7.40.2.15 PrepareMatricesForSQP()

```
bool RSVScalc::PrepareMatricesForSQP (
             MatrixXd & dConstrAct,
             MatrixXd & HConstrAct,
             MatrixXd & HObjAct,
             RowVectorXd & dObjAct,
             VectorXd & constrAct,
             VectorXd & lagMultAct )
```

Prepares the matrices needed for the SQP step calculation.

**Parameters**

| | |
|---|---|
| *dConstrAct* | The active constraint Jacobian |
| *HConstrAct* | The active constraint hessian |
| *HObjAct* | The active objective hessian |
| *dObjAct* | The active objective Jacobian |
| *constrAct* | The active constraint values |
| *lagMultAct* | The active lagrangian multipliers. |

**Returns**

Returns wether the calculation should be performed or not.

Definition at line 28 of file RSVScalc_SQP.cpp.

### 7.40.2.16 PrepTriangulationCalc()

```
void RSVScalc::PrepTriangulationCalc (
             const triangulation & triRSVS )
```

Groups actions needed before the calculation of triangular quantities.

**Parameters**

| | | |
|---|---|---|
| in | *triRSVS* | The triangulation object. |

Definition at line 22 of file RSVScalc.cpp.

### 7.40.2.17 Print2Screen()

```
void RSVScalc::Print2Screen (
             int outType = 0 ) const
```

Prints different amounts of `RSVScalc` owned data to the screen.

**Parameters**

| | | |
|---|---|---|
| in | *outType* | The output type to print, values [2,3,4]. |

Definition at line 171 of file RSVScalc.cpp.

#### 7.40.2.18 ReturnConstrToMesh() [1/2]

```
void RSVScalc::ReturnConstrToMesh (
            triangulation & triRSVS ) const
```

Returns a constraint to the triangulation::meshDep.

**Parameters**

| | |
|---|---|
| *triRSVS* | The triangulation object. |

Definition at line 231 of file RSVScalc.cpp.

#### 7.40.2.19 ReturnConstrToMesh() [2/2]

```
void RSVScalc::ReturnConstrToMesh (
            mesh & meshin,
            double volu::* mp = &volu::volume ) const
```

Returns a constraint to the mesh.

**Parameters**

| | | |
|---|---|---|
| | *meshin* | The input mesh. |
| in | *volu* | The volumetric field that data needs to be returned to. It is a member point of class volu. |

Definition at line 255 of file RSVScalc.cpp.

#### 7.40.2.20 ReturnVelocities()

```
void RSVScalc::ReturnVelocities (
            triangulation & triRSVS )
```

Returns velocities to the snaxels.

Returns velocities to the snake in the triangulation object.

**Parameters**

| | |
|---|---|
| *triRSVS* | The triangulation object, affects the triangulation::snakeDep attribute. |
| *triRSVS* | The triangulation object of the RSVS |

Definition at line 119 of file RSVScalc.cpp.

#### 7.40.2.21 SnakDVcond()

```
bool RSVScalc::SnakDVcond (
            const triangulation & triRSVS,
            int ii )
```

Returns wether a snaxel is a design variable or not.

If the snaxel is frozen and all its neighbours are frozen, it is not a design variable.

**Parameters**

| | | |
|---|---|---|
| in | *triRSVS* | The triangulation which is being calculated |
| in | *ii* | the snaxel subscript. |

**Returns**

wether the snaxel is design variable or not.

Definition at line 53 of file RSVScalc.cpp.

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

- incl/RSVScalc.hpp
- src/rsvs/RSVScalc.cpp
- src/rsvs/RSVScalc_core.cpp
- src/rsvs/RSVScalc_SQP.cpp

## 7.41 integrate::RSVSclass Class Reference

### Public Attributes

- param::parameters **paramconf**
- tecplotfile **outSnake**
- snake **rsvsSnake**
- mesh **snakeMesh**
- mesh **voluMesh**
- triangulation **rsvsTri**
- RSVScalc **calcObj**
- std::ofstream **logFile**
- std::ofstream **coutFile**
- std::ofstream **cerrFile**

### 7.41.1 Detailed Description

Definition at line 37 of file RSVSclass.hpp.

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

- incl/RSVSclass.hpp

## 7.42 selfint_event Class Reference

### Public Attributes

- int **e_type**
- int **f_marker1**
- int **s_marker1**
- int **f_vertices1** [3]
- int **f_marker2**
- int **s_marker2**
- int **f_vertices2** [3]
- REAL **int_point** [3]

### 7.42.1 Detailed Description

Definition at line 2330 of file tetgen.h.

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

- modules/tetgen/tetgen.h

## 7.43 snake Class Reference

### Public Member Functions

- void **disp** () const
- void **displight** () const
- bool **isready** () const
- void **PrepareForUse** (bool needOrder=true)
- void **Init** (mesh ∗snakemesh, int nSnax, int nEdge, int nSurf, int nVolu)
- void **reserve** (int nSnax, int nEdge, int nSurf, int nVolu)
- void **GetMaxIndex** (int ∗nVert, int ∗nEdge, int ∗nSurf, int ∗nVolu) const
- void **HashArray** ()
- void **HashArrayNM** ()
- void **HashParent** ()
- void **SetMaxIndex** ()
- void **SetMaxIndexNM** ()
- void **Concatenate** (const snake &other, int isInternal=0)
- bool **Check3D** () const
- void **MakeCompatible_inplace** (snake &other) const
- snake **MakeCompatible** (snake other) const
- void **ChangeIndices** (int nVert, int nEdge, int nSurf, int nVolu)
- void **ChangeIndicesSnakeMesh** (int nVert, int nEdge, int nSurf, int nVolu)
- void **ForceCloseContainers** ()
- void **UpdateDistance** (double dt, double maxDstep=1.0)
- void **UpdateDistance** (const vector< double > &dt, double maxDstep=1.0)
- void **CalculateTimeStep** (vector< double > &dt, double dtDefault, double distDefault=1.0)
- void **SnaxImpactDetection** (vector< int > &isImpact)
- void **SnaxAlmostImpactDetection** (vector< int > &isImpact, double dDlim)
- void **UpdateCoord** ()

- void **Flip** ()
- grid::limits **Scale** (const grid::limits &newSize)
- void **OrderEdges** ()
- void **SetSnaxSurfs** ()
- void **OrientFaces** ()
- int **FindBlockSnakeMeshVerts** (vector< int > &vertBlock) const
- void **AssignInternalVerts** ()
- void **CheckConnectivity** () const
- void **VertIsIn** (int vertInd, bool isIn=true)
- void **VertIsIn** (vector< int > vertInd, bool isIn=true)
- bool **ReturnFlip** () const
- void **read** (FILE ∗fid)
- void **write** (FILE ∗fid) const
- int **read** (const char ∗str)
- int **write** (const char ∗str) const

## Public Attributes

- snaxarray **snaxs**
- snaxedgearray **snaxedges**
- snaxsurfarray **snaxsurfs**
- mesh **snakeconn**
- mesh ∗ **snakemesh** =NULL
- vector< bool > **isMeshVertIn**

## Private Member Functions

- void **SetLastIndex** ()
- void **OrientSurfaceVolume** ()
- void **OrientEdgeSurface** ()

## Private Attributes

- bool **is3D** =true
- bool **isFlipped** =false

### 7.43.1 Detailed Description

Definition at line 88 of file snake.hpp.

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

- incl/snake.hpp
- src/snake/snake.cpp

## 7.44 param::snaking Class Reference

Parameters controlling tuning parameters for the stepping of the restricted surface.

```
#include <parameters.hpp>
```

## Public Member Functions

- void **PrepareForUse** ()

## Public Attributes

- double arrivaltolerance

  *Distance along edge at which a vertex is considered arrived regardless of "d" and "v".*
- double multiarrivaltolerance

  *Distance along edge at which converging snaxels are considered arrived.*
- double snaxtimestep

  *maximum snake time step length*
- double snaxdiststep

  *maximum snaxel distance movement*
- int initboundary

  *Initialisation boundary (either 0 or 1)*
- int maxsteps

  *maximum number of steps*

### 7.44.1 Detailed Description

Parameters controlling tuning parameters for the stepping of the restricted surface.

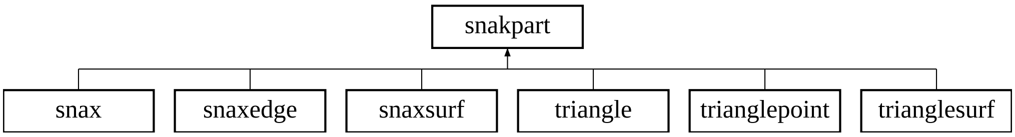
Definition at line 74 of file parameters.hpp.

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

- incl/parameters.hpp
- src/parameters.cpp

## 7.45 snakpart Class Reference

Inheritance diagram for snakpart:



## Public Member Functions

- virtual int **KeyParent** () const =0

### 7.45.1 Detailed Description

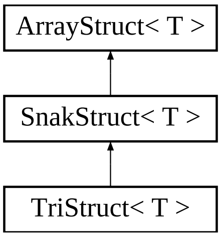
Definition at line 384 of file arraystructures.hpp.

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

- incl/arraystructures.hpp

## 7.46 SnakStruct< T > Class Template Reference

Inheritance diagram for SnakStruct< T >:



### Public Member Functions

- int **findparent** (int key) const
- void **findsiblings** (int key, vector< int > &siblings) const
- int **countparent** (int key) const
- void **HashParent** ()
- void **DeHashParent** (const int pos)
- bool **memberIsHashParent** (const int pos) const
- void **Init** (int n)
- void **push_back** (T &newelem)
- void **clear** ()
- bool **checkready** ()
- void **ForceArrayReady** ()
- void **PrepareForUse** ()
- void **Concatenate** (const SnakStruct< T > &other)
- void **remove** (const vector< int > &sub)
- T & **operator[ ]** (const int a)

### Protected Attributes

- unordered_multimap< int, int > **hashParent**
- int **isHashParent** =0

### Friends

- class **snake**

## Additional Inherited Members

### 7.46.1 Detailed Description

**template<class T>**
**class SnakStruct< T >**

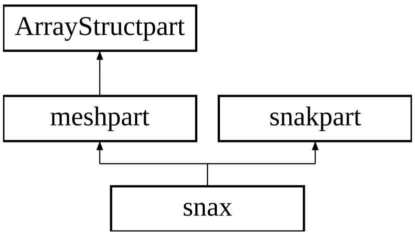
Definition at line 71 of file arraystructures.hpp.

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

- incl/arraystructures.hpp
- incl/snakstruct_incl.cpp

## 7.47 snax Class Reference

Inheritance diagram for snax:



### Public Member Functions

- void **disp** () const
- void **disptree** (const mesh &meshin, int n) const
- void **disptree** (const snake &snakein, int n) const
- int **Key** () const
- int **KeyParent** () const
- void **ChangeIndices** (int nVert, int nEdge, int nSurf, int nVolu)
- void **ChangeIndicesSnakeMesh** (int nVert, int nEdge, int nSurf, int nVolu)
- void **PrepareForUse** ()
- bool **isready** (bool isInMesh) const
- void **read** (FILE ∗fid)
- void **write** (FILE ∗fid) const
- void **set** (int index, double d, double v, int fromvert, int tovert, int edgeind, int isfreeze, int orderedge)
- void **SwitchIndex** (int typeInd, int oldInd, int newInd)
- void **TightenConnectivity** ()

### Public Attributes

- double **d** =0.0
- double **v** =0.0
- int **fromvert** =0
- int **tovert** =0
- int **edgeind** =0
- int **isfreeze** =0
- int **orderedge** =0

### 7.47.1 Detailed Description

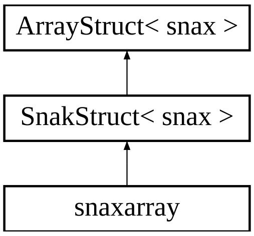
Definition at line 156 of file snake.hpp.

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

- incl/snake.hpp
- src/snake/snake.cpp

## 7.48 snaxarray Class Reference

Inheritance diagram for snaxarray:



### Public Member Functions

- void **ReorderOnEdge** ()
- void **OrderOnEdge** ()
- void **CalculateTimeStepOnEdge** (vector< double > &dt, vector< bool > &isSnaxDone, int edgeInd)
- void **DetectImpactOnEdge** (vector< int > &isImpact, vector< bool > &isSnaxDone, int edgeInd)
- bool **checkready** ()
- void **ForceArrayReady** ()
- void **PrepareForUse** ()
- void **Concatenate** (const snaxarray &other)
- snax & **operator[ ]** (const int a)

### Protected Attributes

- int **isOrderedOnEdge** =0

### Friends

- class **snake**
- void **SpawnArrivedSnaxelsDir** (snake &fullsnake, snake &partSnake, const vector< int > &isImpact, int dir)

### Additional Inherited Members

### 7.48.1 Detailed Description

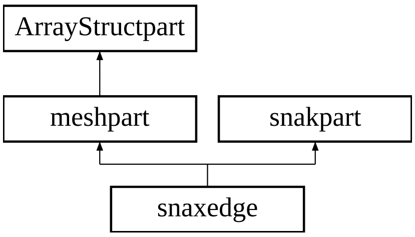
Definition at line 57 of file snake.hpp.

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

- incl/snake.hpp
- src/snake/snake.cpp

## 7.49 snaxedge Class Reference

Inheritance diagram for snaxedge:



### Public Member Functions

- void **PrepareForUse** ()
- void **disp** () const
- void **disptree** (const mesh &meshin, int n) const
- void **disptree** (const snake &snakein, int n) const
- int **Key** () const
- int **KeyParent** () const
- void **ChangeIndices** (int nVert, int nEdge, int nSurf, int nVolu)
- void **ChangeIndicesSnakeMesh** (int nVert, int nEdge, int nSurf, int nVolu)
- bool **isready** (bool isInMesh) const
- void **read** (FILE ∗fid)
- void **write** (FILE ∗fid) const
- void **SwitchIndex** (int typeInd, int oldInd, int newInd)
- void **TightenConnectivity** ()

### Public Attributes

- int **surfind** =0
- coordvec **normvector**

### 7.49.1 Detailed Description

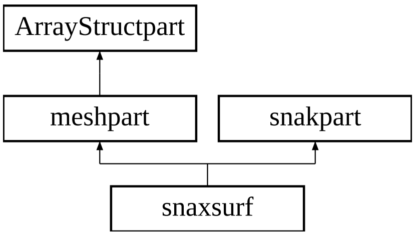
Definition at line 190 of file snake.hpp.

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

- incl/snake.hpp
- src/snake/snake.cpp

## 7.50 snaxsurf Class Reference

Inheritance diagram for snaxsurf:



### Public Member Functions

- void **PrepareForUse** ()
- void **disp** () const
- void **disptree** (const mesh &meshin, int n) const
- void **disptree** (const snake &snakein, int n) const
- int **Key** () const
- int **KeyParent** () const
- void **ChangeIndices** (int nVert, int nEdge, int nSurf, int nVolu)
- void **ChangeIndicesSnakeMesh** (int nVert, int nEdge, int nSurf, int nVolu)
- bool **isready** (bool isInMesh) const
- void **read** (FILE ∗fid)
- void **write** (FILE ∗fid) const
- void **SwitchIndex** (int typeInd, int oldInd, int newInd)
- void **TightenConnectivity** ()

### Public Attributes

- int **voluind** =0
- coordvec **normvector**

### 7.50.1 Detailed Description

Definition at line 214 of file snake.hpp.

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

- incl/snake.hpp
- src/snake/snake.cpp

## 7.51 dbg::StackFrame Struct Reference

### Public Attributes

- DWORD64 **address**
- std::string **name**
- std::string **module**
- unsigned int **line**
- std::string **file**

### 7.51.1 Detailed Description

Definition at line 82 of file data.h.

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

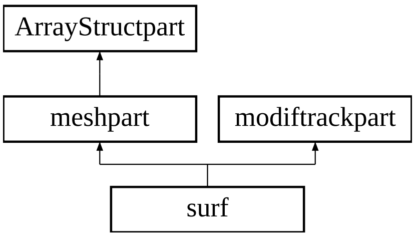
- modules/external/data.h

## 7.52 surf Class Reference

Class for surface object in a mesh.

```
#include <mesh.hpp>
```

Inheritance diagram for surf:



### Public Member Functions

- void **disp** () const
- void **disptree** (const mesh &meshin, int n) const
- void **ChangeIndices** (int nVert, int nEdge, int nSurf, int nVolu)
- void **PrepareForUse** ()
- bool **isready** (bool isInMesh) const
- void **read** (FILE ∗fid)
- void **write** (FILE ∗fid) const
- int **OrderEdges** (mesh ∗meshin)
- int **SplitSurface** (mesh &meshin, const vector< int > &fullEdgeInd)
- void **OrderedVerts** (const mesh ∗meshin, vector< int > &vertList) const
- void **TightenConnectivity** ()
- void **FlipVolus** ()
- bool **edgeconneq** (const surf &other, bool recurse=true) const
- **surf** (const surf &oldSurf)
- void **operator=** (const surf ∗other)
- int **Key** () const

### Public Attributes

- friend **surfarray**
- double **fill**
- double **target**
- double **error**
- double **area**
- vector< int > **edgeind**
- vector< int > **voluind**

### Protected Attributes

- bool **isordered**

### Friends

- class **mesh**

### 7.52.1 Detailed Description

Class for surface object in a mesh.

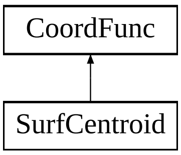
Definition at line 233 of file mesh.hpp.

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

- incl/mesh.hpp
- src/grid/mesh.cpp

## 7.53 SurfCentroid Class Reference

Inheritance diagram for SurfCentroid:



### Public Member Functions

- void **Disp** ()
- void **Calc** () override
- void **assigncentroid** (const vector< double > &vecin)
- **SurfCentroid** (int a)

## Protected Attributes

- vector< double > **centroid**
- double **edgeLength** =0.0
- vector< vector< double > const ∗ > **coords**
- double **fun**
- ArrayVec< double > **jac**
- ArrayVec< double > **hes**
- int **nCoord**

## Additional Inherited Members

### 7.53.1 Detailed Description

Definition at line 201 of file RSVSmath.hpp.

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

- incl/RSVSmath.hpp
- src/rsvs/RSVSmath.cpp

# 7.54 tecplotfile Class Reference

## Public Member Functions

- int **OpenFile** (const char ∗str, const char ∗mode="w")
- void **CloseFile** ()
- int **ZoneNum** () const
- int **PrintMesh** (const mesh &meshout, int strandID=0, double timeStep=0, int forceOutType=0, const vector< int > &vertList={})
- int **PrintSnakeInternalPts** (const snake &snakein, int strandID=0, double timeStep=0)
- int **VolDataBlock** (const mesh &meshout, int nVert, int nVolu, int nVertDat, const std::vector< int > &volu↩ List={}, const std::vector< int > &vertList={})
- int **SurfDataBlock** (const mesh &meshout, int nVert, int nSurf, int nVertDat)
- int **LineDataBlock** (const mesh &meshout, int nVert, int nEdge, int nVertDat, int nCellDat)
- int **VertDataBlock** (const mesh &meshout, int nVert, int nVertDat, int nCellDat, const vector< int > &vert↩ List={})
- int **VolFaceMap** (const mesh &meshout, int nSurf)
- int **VolFaceMap** (const mesh &meshout, const std::vector< int > &surfList, const std::vector< int > &volu↩ List, const std::vector< int > &vertList)
- int **SurfFaceMap** (const mesh &meshout, int nEdge)
- int **LineFaceMap** (const mesh &meshout, int nEdge)
- int **PrintVolumeDat** (const mesh &meshout, int shareZone, int strandID, double timeStep)
- int **DefShareZoneVolume** (int shareZone, int nVertDat)
- int **VolDataBlock** (const triangulation &triout, triarray triangulation::∗mp, int nVert, int nVolu, int nVertDat)
- int **SurfDataBlock** (const triangulation &triout, triarray triangulation::∗mp, int nVert, int nSurf, int nVertDat)
- int **LineDataBlock** (const triangulation &triout, triarray triangulation::∗mp, int nVert, int nEdge, int nVertDat, int nCellDat)
- int **LineDataBlock** (const triangulation &triout, triarray triangulation::∗mp, int nVert, int nEdge, int nVertDat, int nCellDat, const vector< int > &triList)
- int **SurfFaceMap** (const triangulation &triout, triarray triangulation::∗mp)

- int **LineFaceMap** (const triangulation &triout, triarray triangulation::∗mp)
- int **LineFaceMap** (const vector< int > &triList)
- int **VolFaceMap** (const triangulation &triout, triarray triangulation::∗mp, int nSurf)
- int **PrintTriangulation** (const triangulation &triout, triarray triangulation::∗mp, int strandID=0, double time↩ Step=0, int forceOutType=0, const vector< int > &triList={})
- int **VolDataBlock** (const triangulation &triout, trisurfarray triangulation::∗mp, int nVert, int nVolu, int nVertDat)
- int **SurfDataBlock** (const triangulation &triout, trisurfarray triangulation::∗mp, int nVert, int nSurf, int nVertDat)
- int **LineDataBlock** (const triangulation &triout, trisurfarray triangulation::∗mp, int nVert, int nEdge, int nVert↩ Dat, int nCellDat)
- int **SurfFaceMap** (const triangulation &triout, trisurfarray triangulation::∗mp)
- int **LineFaceMap** (const triangulation &triout, trisurfarray triangulation::∗mp)
- int **VolFaceMap** (const triangulation &triout, trisurfarray triangulation::∗mp, int nSurf)
- int **PrintTriangulation** (const triangulation &triout, trisurfarray triangulation::∗mp, int strandID=0, double timeStep=0, int forceOutType=0)
- int **SnakeDataBlock** (const snake &snakeout, int nVert, int nVertDat)
- int **PrintSnake** (const snake &snakeout, int strandID=0, double timeStep=0, int forceOutType=0, const vector< int > &vertList={})
- void **ZoneHeaderPolyhedron** (int nVert, int nVolu, int nSurf, int totNumFaceNode, int nVertDat, int nCellDat)
- void **ZoneHeaderPolygon** (int nVert, int nEdge, int nSurf, int nVertDat, int nCellDat)
- void **ZoneHeaderFelineseg** (int nVert, int nEdge, int nVertDat, int nCellDat)
- void **ZoneHeaderOrdered** (int nVert, int nVertDat, int nCellDat)
- void **ZoneHeaderPolyhedronSnake** (int nVert, int nVolu, int nSurf, int totNumFaceNode, int nVertDat, int nCellDat)
- void **ZoneHeaderPolygonSnake** (int nVert, int nEdge, int nSurf, int nVertDat, int nCellDat)
- void **ZoneHeaderFelinesegSnake** (int nVert, int nEdge, int nVertDat, int nCellDat)
- void **ZoneHeaderOrderedSnake** (int nVert, int nVertDat, int nCellDat)
- void **NewZone** ()
- void **StrandTime** (int strandID, double timeStep)
- int **Print** (const char ∗format,...)
- void **ResetLine** ()

### Private Attributes

- FILE ∗ **fid**
- int **lengthLine**
- int **nZones** =0

### 7.54.1 Detailed Description

Definition at line 47 of file postprocessing.hpp.

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

- incl/postprocessing.hpp
- src/postprocessing.cpp

## 7.55 tetgenbehavior Class Reference

### Public Types

- enum **objecttype** {
**NODES**, **POLY**, **OFF**, **PLY**,
**STL**, **MEDIT**, **VTK**, **MESH**,
**NEU_MESH** }

## Public Member Functions

- void **syntax** ()
- void **usage** ()
- bool **parse_commandline** (int argc, const char ∗∗argv)
- bool **parse_commandline** (const char ∗switches)

## Public Attributes

- int **plc**
- int **psc**
- int **refine**
- int **quality**
- int **nobisect**
- int **coarsen**
- int **weighted**
- int **brio_hilbert**
- int **incrflip**
- int **flipinsert**
- int **metric**
- int **varvolume**
- int **fixedvolume**
- int **regionattrib**
- int **cdtrefine**
- int **use_equatorial_lens**
- int **insertaddpoints**
- int **diagnose**
- int **convex**
- int **nomergefacet**
- int **nomergevertex**
- int **noexact**
- int **nostaticfilter**
- int **zeroindex**
- int **facesout**
- int **edgesout**
- int **neighout**
- int **voroout**
- int **meditview**
- int **vtkview**
- int **nobound**
- int **nonodewritten**
- int **noelewritten**
- int **nofacewritten**
- int **noiterationnum**
- int **nojettison**
- int **docheck**
- int **quiet**
- int **verbose**
- int **vertexperblock**
- int **tetrahedraperblock**
- int **shellfaceperblock**
- int **nobisect_nomerge**
- int **supsteiner_level**
- int **addsteiner_algo**

- int **coarsen_param**
- int **weighted_param**
- int **fliplinklevel**
- int **flipstarsize**
- int **fliplinklevelinc**
- int **reflevel**
- int **optlevel**
- int **optscheme**
- int **delmaxfliplevel**
- int **order**
- int **reversetetori**
- int **steinerleft**
- int **no_sort**
- int **hilbert_order**
- int **hilbert_limit**
- int **brio_threshold**
- REAL **brio_ratio**
- REAL **facet_separate_ang_tol**
- REAL **facet_overlap_ang_tol**
- REAL **facet_small_ang_tol**
- REAL **maxvolume**
- REAL **minratio**
- REAL **mindihedral**
- REAL **optmaxdihedral**
- REAL **optminsmtdihed**
- REAL **optminslidihed**
- REAL **epsilon**
- REAL **coarsen_percent**
- char **commandline** [1024]
- char **infilename** [1024]
- char **outfilename** [1024]
- char **addinfilename** [1024]
- char **bgmeshfilename** [1024]
- int **hole_mesh**
- char **hole_mesh_filename** [1024]
- int **apply_flow_bc**
- enum tetgenbehavior::objecttype **object**

### 7.55.1 Detailed Description

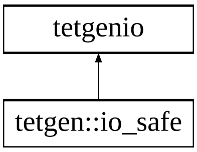
Definition at line 598 of file tetgen.h.

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

- modules/tetgen/tetgen.h
- modules/tetgen/tetgen.cpp
- modules/tetgen/tetgen.cxx

## 7.56 tetgenio Class Reference

Inheritance diagram for tetgenio:



### Classes

- struct facet
- struct pointparam
- struct polygon
- struct voroedge
- struct vorofacet

### Public Types

- typedef REAL(∗ **GetVertexParamOnEdge**) (void ∗, int, int)
- typedef void(∗ **GetSteinerOnEdge**) (void ∗, int, REAL, REAL ∗)
- typedef void(∗ **GetVertexParamOnFace**) (void ∗, int, int, REAL ∗)
- typedef void(∗ **GetEdgeSteinerParamOnFace**) (void ∗, int, REAL, int, REAL ∗)
- typedef void(∗ **GetSteinerOnFace**) (void ∗, int, REAL ∗, REAL ∗)
- typedef bool(∗ **TetSizeFunc**) (REAL ∗, REAL ∗, REAL ∗, REAL ∗, REAL ∗, REAL)

### Public Member Functions

- bool **load_node_call** (FILE ∗infile, int markers, int uvflag, char ∗)
- bool **load_node** (const char ∗)
- bool **load_edge** (const char ∗)
- bool **load_face** (const char ∗)
- bool **load_tet** (const char ∗)
- bool **load_vol** (const char ∗)
- bool **load_var** (const char ∗)
- bool **load_mtr** (const char ∗)
- bool **load_pbc** (const char ∗)
- bool **load_poly** (const char ∗)
- bool **load_off** (const char ∗)
- bool **load_ply** (const char ∗)
- bool **load_stl** (const char ∗)
- bool **load_vtk** (const char ∗)
- bool **load_medit** (const char ∗, int)
- bool **load_neumesh** (const char ∗, int)
- bool **load_plc** (const char ∗, int)
- bool **load_tetmesh** (const char ∗, int)
- void **save_nodes** (const char ∗)
- void **save_elements** (const char ∗)
- void **save_faces** (const char ∗)
- void **save_edges** (const char ∗)

- void **save_neighbors** (const char ∗)
- void **save_poly** (const char ∗)
- void **save_faces2smesh** (char ∗)
- char ∗ **readline** (char ∗string, FILE ∗infile, int ∗linenumber)
- char ∗ **findnextfield** (char ∗string)
- char ∗ **readnumberline** (char ∗string, FILE ∗infile, char ∗infilename)
- char ∗ **findnextnumber** (char ∗string)
- void **initialize** ()
- void **deinitialize** ()

## Static Public Member Functions

- static void **init** (polygon ∗p)
- static void **init** (facet ∗f)

## Public Attributes

- int **firstnumber**
- int **mesh_dim**
- int **useindex**
- REAL ∗ **pointlist**
- REAL ∗ **pointattributelist**
- REAL ∗ **pointmtrlist**
- int ∗ **pointmarkerlist**
- int ∗ **point2tetlist**
- pointparam ∗ **pointparamlist**
- int **numberofpoints**
- int **numberofpointattributes**
- int **numberofpointmtrs**
- int ∗ **tetrahedronlist**
- REAL ∗ **tetrahedronattributelist**
- REAL ∗ **tetrahedronvolumelist**
- int ∗ **neighborlist**
- int ∗ **tet2facelist**
- int ∗ **tet2edgelist**
- int **numberoftetrahedra**
- int **numberofcorners**
- int **numberoftetrahedronattributes**
- facet ∗ **facetlist**
- int ∗ **facetmarkerlist**
- int **numberoffacets**
- REAL ∗ **holelist**
- int **numberofholes**
- REAL ∗ **regionlist**
- int **numberofregions**
- REAL ∗ **facetconstraintlist**
- int **numberoffacetconstraints**
- REAL ∗ **segmentconstraintlist**
- int **numberofsegmentconstraints**
- int ∗ **trifacelist**
- int ∗ **trifacemarkerlist**
- int ∗ **o2facelist**
- int ∗ **face2tetlist**

- int ∗ **face2edgelist**
- int **numberoftrifaces**
- int ∗ **edgelist**
- int ∗ **edgemarkerlist**
- int ∗ **o2edgelist**
- int ∗ **edge2tetlist**
- int **numberofedges**
- REAL ∗ **vpointlist**
- voroedge ∗ **vedgelist**
- vorofacet ∗ **vfacetlist**
- int ∗∗ **vcelllist**
- int **numberofvpoints**
- int **numberofvedges**
- int **numberofvfacets**
- int **numberofvcells**
- void ∗ **geomhandle**
- GetVertexParamOnEdge **getvertexparamonedge**
- GetSteinerOnEdge **getsteineronedge**
- GetVertexParamOnFace **getvertexparamonface**
- GetEdgeSteinerParamOnFace **getedgesteinerparamonface**
- GetSteinerOnFace **getsteineronface**
- TetSizeFunc **tetunsuitable**

### 7.56.1 Detailed Description

Definition at line 111 of file tetgen.h.

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

- modules/tetgen/tetgen.h
- modules/tetgen/tetgen.cpp
- modules/tetgen/tetgen.cxx

## 7.57 tetgenmesh Class Reference

### Classes

- class arraypool
- class badface
- class face
- class flipconstraints
- class insertvertexflags
- class memorypool
- class optparameters
- class triface

## Public Types

- enum **verttype** {
**UNUSEDVERTEX**, **DUPLICATEDVERTEX**, **RIDGEVERTEX**, **ACUTEVERTEX**,
**FACETVERTEX**, **VOLVERTEX**, **FREESEGVERTEX**, **FREEFACETVERTEX**,
**FREEVOLVERTEX**, **NREGULARVERTEX**, **DEADVERTEX** }
- enum **interresult** {
**DISJOINT**, **INTERSECT**, **SHAREVERT**, **SHAREEDGE**,
**SHAREFACE**, **TOUCHEDGE**, **TOUCHFACE**, **ACROSSVERT**,
**ACROSSEDGE**, **ACROSSFACE** }
- enum **locateresult** {
**UNKNOWN**, **OUTSIDE**, **INTETRAHEDRON**, **ONFACE**,
**ONEDGE**, **ONVERTEX**, **ENCVERTEX**, **ENCSEGMENT**,
**ENCSUBFACE**, **NEARVERTEX**, **NONREGULAR**, **INSTAR**,
**BADELEMENT** }
- typedef REAL ∗∗ **tetrahedron**
- typedef REAL ∗∗ **shellface**
- typedef REAL ∗ **point**

## Public Member Functions

- void **inittables** ()
- tetrahedron **encode** (triface &t)
- tetrahedron **encode2** (tetrahedron ∗ptr, int ver)
- void **decode** (tetrahedron ptr, triface &t)
- void **bond** (triface &t1, triface &t2)
- void **dissolve** (triface &t)
- void **esym** (triface &t1, triface &t2)
- void **esymself** (triface &t)
- void **enext** (triface &t1, triface &t2)
- void **enextself** (triface &t)
- void **eprev** (triface &t1, triface &t2)
- void **eprevself** (triface &t)
- void **enextesym** (triface &t1, triface &t2)
- void **enextesymself** (triface &t)
- void **eprevesym** (triface &t1, triface &t2)
- void **eprevesymself** (triface &t)
- void **eorgoppo** (triface &t1, triface &t2)
- void **eorgoppoself** (triface &t)
- void **edestoppo** (triface &t1, triface &t2)
- void **edestoppoself** (triface &t)
- void **fsym** (triface &t1, triface &t2)
- void **fsymself** (triface &t)
- void **fnext** (triface &t1, triface &t2)
- void **fnextself** (triface &t)
- point **org** (triface &t)
- point **dest** (triface &t)
- point **apex** (triface &t)
- point **oppo** (triface &t)
- void **setorg** (triface &t, point p)
- void **setdest** (triface &t, point p)
- void **setapex** (triface &t, point p)
- void **setoppo** (triface &t, point p)
- REAL **elemattribute** (tetrahedron ∗ptr, int attnum)

- void **setelemattribute** (tetrahedron ∗ptr, int attnum, REAL value)
- REAL **volumebound** (tetrahedron ∗ptr)
- void **setvolumebound** (tetrahedron ∗ptr, REAL value)
- int **elemindex** (tetrahedron ∗ptr)
- void **setelemindex** (tetrahedron ∗ptr, int value)
- int **elemmarker** (tetrahedron ∗ptr)
- void **setelemmarker** (tetrahedron ∗ptr, int value)
- void **infect** (triface &t)
- void **uninfect** (triface &t)
- bool **infected** (triface &t)
- void **marktest** (triface &t)
- void **unmarktest** (triface &t)
- bool **marktested** (triface &t)
- void **markface** (triface &t)
- void **unmarkface** (triface &t)
- bool **facemarked** (triface &t)
- void **markedge** (triface &t)
- void **unmarkedge** (triface &t)
- bool **edgemarked** (triface &t)
- void **marktest2** (triface &t)
- void **unmarktest2** (triface &t)
- bool **marktest2ed** (triface &t)
- int **elemcounter** (triface &t)
- void **setelemcounter** (triface &t, int value)
- void **increaseelemcounter** (triface &t)
- void **decreaseelemcounter** (triface &t)
- bool **ishulltet** (triface &t)
- bool **isdeadtet** (triface &t)
- void **sdecode** (shellface sptr, face &s)
- shellface **sencode** (face &s)
- shellface **sencode2** (shellface ∗sh, int shver)
- void **spivot** (face &s1, face &s2)
- void **spivotself** (face &s)
- void **sbond** (face &s1, face &s2)
- void **sbond1** (face &s1, face &s2)
- void **sdissolve** (face &s)
- point **sorg** (face &s)
- point **sdest** (face &s)
- point **sapex** (face &s)
- void **setsorg** (face &s, point pointptr)
- void **setsdest** (face &s, point pointptr)
- void **setsapex** (face &s, point pointptr)
- void **sesym** (face &s1, face &s2)
- void **sesymself** (face &s)
- void **senext** (face &s1, face &s2)
- void **senextself** (face &s)
- void **senext2** (face &s1, face &s2)
- void **senext2self** (face &s)
- REAL **areabound** (face &s)
- void **setareabound** (face &s, REAL value)
- int **shellmark** (face &s)
- void **setshellmark** (face &s, int value)
- void **sinfect** (face &s)
- void **suninfect** (face &s)
- bool **sinfected** (face &s)

- void **smarktest** (face &s)
- void **sunmarktest** (face &s)
- bool **smarktested** (face &s)
- void **smarktest2** (face &s)
- void **sunmarktest2** (face &s)
- bool **smarktest2ed** (face &s)
- void **smarktest3** (face &s)
- void **sunmarktest3** (face &s)
- bool **smarktest3ed** (face &s)
- void **setfacetindex** (face &f, int value)
- int **getfacetindex** (face &f)
- void **tsbond** (triface &t, face &s)
- void **tsdissolve** (triface &t)
- void **stdissolve** (face &s)
- void **tspivot** (triface &t, face &s)
- void **stpivot** (face &s, triface &t)
- void **tssbond1** (triface &t, face &seg)
- void **sstbond1** (face &s, triface &t)
- void **tssdissolve1** (triface &t)
- void **sstdissolve1** (face &s)
- void **tsspivot1** (triface &t, face &s)
- void **sstpivot1** (face &s, triface &t)
- void **ssbond** (face &s, face &edge)
- void **ssbond1** (face &s, face &edge)
- void **ssdissolve** (face &s)
- void **sspivot** (face &s, face &edge)
- int **pointmark** (point pt)
- void **setpointmark** (point pt, int value)
- enum verttype **pointtype** (point pt)
- void **setpointtype** (point pt, enum verttype value)
- int **pointgeomtag** (point pt)
- void **setpointgeomtag** (point pt, int value)
- REAL **pointgeomuv** (point pt, int i)
- void **setpointgeomuv** (point pt, int i, REAL value)
- void **pinfect** (point pt)
- void **puninfect** (point pt)
- bool **pinfected** (point pt)
- void **pmarktest** (point pt)
- void **punmarktest** (point pt)
- bool **pmarktested** (point pt)
- void **pmarktest2** (point pt)
- void **punmarktest2** (point pt)
- bool **pmarktest2ed** (point pt)
- void **pmarktest3** (point pt)
- void **punmarktest3** (point pt)
- bool **pmarktest3ed** (point pt)
- tetrahedron **point2tet** (point pt)
- void **setpoint2tet** (point pt, tetrahedron value)
- shellface **point2sh** (point pt)
- void **setpoint2sh** (point pt, shellface value)
- point **point2ppt** (point pt)
- void **setpoint2ppt** (point pt, point value)
- tetrahedron **point2bgmtet** (point pt)
- void **setpoint2bgmtet** (point pt, tetrahedron value)
- void **setpointinsradius** (point pt, REAL value)

- REAL **getpointinsradius** (point pt)
- bool **issteinerpoint** (point pt)
- void **point2tetorg** (point pt, triface &t)
- void **point2shorg** (point pa, face &s)
- point **farsorg** (face &seg)
- point **farsdest** (face &seg)
- void **tetrahedrondealloc** (tetrahedron ∗)
- tetrahedron ∗ **tetrahedrontraverse** ()
- tetrahedron ∗ **alltetrahedrontraverse** ()
- void **shellfacedealloc** (memorypool ∗, shellface ∗)
- shellface ∗ **shellfacetraverse** (memorypool ∗)
- void **pointdealloc** (point)
- point **pointtraverse** ()
- void **makeindex2pointmap** (point ∗&)
- void **makepoint2submap** (memorypool ∗, int ∗&, face ∗&)
- void **maketetrahedron** (triface ∗)
- void **makeshellface** (memorypool ∗, face ∗)
- void **makepoint** (point ∗, enum verttype)
- void **initializepools** ()
- REAL **insphere_s** (REAL ∗, REAL ∗, REAL ∗, REAL ∗, REAL ∗)
- REAL **orient4d_s** (REAL ∗, REAL ∗, REAL ∗, REAL ∗, REAL ∗, REAL, REAL, REAL, REAL, REAL)
- int **tri_edge_2d** (point, point, point, point, point, point, int, int ∗, int ∗)
- int **tri_edge_tail** (point, point, point, point, point, point, REAL, REAL, int, int ∗, int ∗)
- int **tri_edge_test** (point, point, point, point, point, point, int, int ∗, int ∗)
- int **tri_edge_inter_tail** (point, point, point, point, point, REAL, REAL)
- int **tri_tri_inter** (point, point, point, point, point, point)
- REAL **dot** (REAL ∗v1, REAL ∗v2)
- void **cross** (REAL ∗v1, REAL ∗v2, REAL ∗n)
- bool **lu_decmp** (REAL lu[4][4], int n, int ∗ps, REAL ∗d, int N)
- void **lu_solve** (REAL lu[4][4], int n, int ∗ps, REAL ∗b, int N)
- REAL **incircle3d** (point pa, point pb, point pc, point pd)
- REAL **orient3dfast** (REAL ∗pa, REAL ∗pb, REAL ∗pc, REAL ∗pd)
- REAL **norm2** (REAL x, REAL y, REAL z)
- REAL **distance** (REAL ∗p1, REAL ∗p2)
- void **facenormal** (point pa, point pb, point pc, REAL ∗n, int pivot, REAL ∗lav)
- REAL **shortdistance** (REAL ∗p, REAL ∗e1, REAL ∗e2)
- REAL **triarea** (REAL ∗pa, REAL ∗pb, REAL ∗pc)
- REAL **interiorangle** (REAL ∗o, REAL ∗p1, REAL ∗p2, REAL ∗n)
- void **projpt2edge** (REAL ∗p, REAL ∗e1, REAL ∗e2, REAL ∗prj)
- void **projpt2face** (REAL ∗p, REAL ∗f1, REAL ∗f2, REAL ∗f3, REAL ∗prj)
- bool **tetalldihedral** (point, point, point, point, REAL ∗, REAL ∗, REAL ∗)
- void **tetallnormal** (point, point, point, point, REAL N[4][3], REAL ∗volume)
- REAL **tetaspectratio** (point, point, point, point)
- bool **circumsphere** (REAL ∗, REAL ∗, REAL ∗, REAL ∗, REAL ∗cent, REAL ∗radius)
- bool **orthosphere** (REAL ∗, REAL ∗, REAL ∗, REAL ∗, REAL, REAL, REAL, REAL, REAL ∗, REAL ∗)
- void **tetcircumcenter** (point tetorg, point tetdest, point tetfapex, point tettapex, REAL ∗circumcenter, REAL ∗radius)
- void **planelineint** (REAL ∗, REAL ∗, REAL ∗, REAL ∗, REAL ∗, REAL ∗, REAL ∗)
- int **linelineint** (REAL ∗, REAL ∗, REAL ∗, REAL ∗, REAL ∗, REAL ∗, REAL ∗, REAL ∗)
- REAL **tetprismvol** (REAL ∗pa, REAL ∗pb, REAL ∗pc, REAL ∗pd)
- bool **calculateabovepoint** (arraypool ∗, point ∗, point ∗, point ∗)
- void **calculateabovepoint4** (point, point, point, point)
- void **report_overlapping_facets** (face ∗, face ∗, REAL dihedang=0.0)
- int **report_selfint_edge** (point, point, face ∗sedge, triface ∗searchtet, enum interresult)
- int **report_selfint_face** (point, point, point, face ∗sface, triface ∗iedge, int intflag, int ∗types, int ∗poss)

- void **flip23** (triface ∗, int, flipconstraints ∗fc)
- void **flip32** (triface ∗, int, flipconstraints ∗fc)
- void **flip41** (triface ∗, int, flipconstraints ∗fc)
- int **flipnm** (triface ∗, int n, int level, int, flipconstraints ∗fc)
- int **flipnm_post** (triface ∗, int n, int nn, int, flipconstraints ∗fc)
- int **insertpoint** (point, triface ∗, face ∗, face ∗, insertvertexflags ∗)
- void **insertpoint_abort** (face ∗, insertvertexflags ∗)
- void **transfernodes** ()
- void **hilbert_init** (int n)
- int **hilbert_split** (point ∗vertexarray, int arraysize, int gc0, int gc1, REAL, REAL, REAL, REAL, REAL, REAL)
- void **hilbert_sort3** (point ∗vertexarray, int arraysize, int e, int d, REAL, REAL, REAL, REAL, REAL, REAL, int depth)
- void **brio_multiscale_sort** (point ∗, int, int threshold, REAL ratio, int ∗depth)
- unsigned long **randomnation** (unsigned int choices)
- void **randomsample** (point searchpt, triface ∗searchtet)
- enum locateresult **locate** (point searchpt, triface ∗searchtet, int chkencflag=0)
- void **flippush** (badface ∗&, triface ∗)
- int **incrementalflip** (point newpt, int, flipconstraints ∗fc)
- void **initialdelaunay** (point pa, point pb, point pc, point pd)
- void **incrementaldelaunay** (clock_t &)
- void **flipshpush** (face ∗)
- void **flip22** (face ∗, int, int)
- void **flip31** (face ∗, int)
- long **lawsonflip** ()
- int **sinsertvertex** (point newpt, face ∗, face ∗, int iloc, int bowywat, int)
- int **sremovevertex** (point delpt, face ∗, face ∗, int lawson)
- enum locateresult **slocate** (point, face ∗, int, int, int)
- enum interresult **sscoutsegment** (face ∗, point, int, int, int)
- void **scarveholes** (int, REAL ∗)
- int **triangulate** (int, arraypool ∗, arraypool ∗, int, REAL ∗)
- void **unifysegments** ()
- void **identifyinputedges** (point ∗)
- void **mergefacets** ()
- void **removesmallangles** ()
- void **meshsurface** ()
- void **interecursive** (shellface ∗∗subfacearray, int arraysize, int axis, REAL, REAL, REAL, REAL, REAL, REAL, int ∗internum)
- void **detectinterfaces** ()
- void **makesegmentendpointsmap** ()
- enum interresult **finddirection** (triface ∗searchtet, point endpt)
- enum interresult **scoutsegment** (point, point, face ∗, triface ∗, point ∗, arraypool ∗)
- int **getsteinerptonsegment** (face ∗seg, point refpt, point steinpt)
- void **delaunizesegments** ()
- int **scoutsubface** (face ∗searchsh, triface ∗searchtet, int shflag)
- void **formregion** (face ∗, arraypool ∗, arraypool ∗, arraypool ∗)
- int **scoutcrossedge** (triface &crosstet, arraypool ∗, arraypool ∗)
- bool **formcavity** (triface ∗, arraypool ∗, arraypool ∗, arraypool ∗, arraypool ∗, arraypool ∗, arraypool ∗)
- void **delaunizecavity** (arraypool ∗, arraypool ∗, arraypool ∗, arraypool ∗, arraypool ∗, arraypool ∗)
- bool **fillcavity** (arraypool ∗, arraypool ∗, arraypool ∗, arraypool ∗, arraypool ∗, arraypool ∗, triface ∗crossedge)
- void **carvecavity** (arraypool ∗, arraypool ∗, arraypool ∗)
- void **restorecavity** (arraypool ∗, arraypool ∗, arraypool ∗, arraypool ∗)
- void **flipcertify** (triface ∗chkface, badface ∗∗pqueue, point, point, point)
- void **flipinsertfacet** (arraypool ∗, arraypool ∗, arraypool ∗, arraypool ∗)
- int **insertpoint_cdt** (point, triface ∗, face ∗, face ∗, insertvertexflags ∗, arraypool ∗, arraypool ∗, arraypool ∗, arraypool ∗, arraypool ∗, arraypool ∗)

- void **refineregion** (face &, arraypool ∗, arraypool ∗, arraypool ∗, arraypool ∗, arraypool ∗, arraypool ∗)
- void **constrainedfacets** ()
- void **constraineddelaunay** (clock_t &)
- int **checkflipeligibility** (int fliptype, point, point, point, point, point, int level, int edgepivot, flipconstraints ∗fc)
- int **removeedgebyflips** (triface ∗, flipconstraints ∗)
- int **removefacebyflips** (triface ∗, flipconstraints ∗)
- int **recoveredgebyflips** (point, point, face ∗, triface ∗, int fullsearch)
- int **add_steinerpt_in_schoenhardtpoly** (triface ∗, int, int chkencflag)
- int **add_steinerpt_in_segment** (face ∗, int searchlevel)
- int **addsteiner4recoversegment** (face ∗, int)
- int **recoversegments** (arraypool ∗, int fullsearch, int steinerflag)
- int **recoverfacebyflips** (point, point, point, face ∗, triface ∗)
- int **recoversubfaces** (arraypool ∗, int steinerflag)
- int **getvertexstar** (int, point searchpt, arraypool ∗, arraypool ∗, arraypool ∗)
- int **getedge** (point, point, triface ∗)
- int **reduceedgesatvertex** (point startpt, arraypool ∗endptlist)
- int **removevertexbyflips** (point steinerpt)
- int **suppressbdrysteinerpoint** (point steinerpt)
- int **suppresssteinerpoints** ()
- void **recoverboundary** (clock_t &)
- void **carveholes** ()
- void **reconstructmesh** ()
- int **search_face** (point p0, point p1, point p2, triface &tetloop)
- int **search_edge** (point p0, point p1, triface &tetloop)
- int **scoutpoint** (point, triface ∗, int randflag)
- REAL **getpointmeshsize** (point, triface ∗, int iloc)
- void **interpolatemeshsize** ()
- void **out_points_to_cells_map** ()
- void **insertconstrainedpoints** (point ∗insertarray, int arylen, int rejflag)
- void **insertconstrainedpoints** (tetgenio ∗addio)
- void **collectremovepoints** (arraypool ∗remptlist)
- void **meshcoarsening** ()
- void **makefacetverticesmap** ()
- int **segsegadjacent** (face ∗, face ∗)
- int **segfacetadjacent** (face ∗checkseg, face ∗checksh)
- int **facetfacetadjacent** (face ∗, face ∗)
- void **save_segmentpoint_insradius** (point segpt, point parentpt, REAL r)
- void **save_facetpoint_insradius** (point facpt, point parentpt, REAL r)
- void **enqueuesubface** (memorypool ∗, face ∗)
- void **enqueuetetrahedron** (triface ∗)
- int **checkseg4encroach** (point pa, point pb, point checkpt)
- int **checkseg4split** (face ∗chkseg, point &, int &)
- int **splitsegment** (face ∗splitseg, point encpt, REAL, point, point, int, int)
- void **repairencsegs** (int chkencflag)
- int **checkfac4encroach** (point, point, point, point checkpt, REAL ∗, REAL ∗)
- int **checkfac4split** (face ∗chkfac, point &encpt, int &qflag, REAL ∗ccent)
- int **splitsubface** (face ∗splitfac, point, point, int qflag, REAL ∗ccent, int)
- void **repairencfacs** (int chkencflag)
- int **checktet4split** (triface ∗chktet, int &qflag, REAL ∗ccent)
- int **splittetrahedron** (triface ∗splittet, int qflag, REAL ∗ccent, int)
- void **repairbadtets** (int chkencflag)
- void **delaunayrefinement** ()
- long **lawsonflip3d** (flipconstraints ∗fc)
- void **recoverdelaunay** ()
- int **gettetrahedron** (point, point, point, point, triface ∗)

- long **improvequalitybyflips** ()
- int **smoothpoint** (point smtpt, arraypool ∗, int ccw, optparameters ∗opm)
- long **improvequalitybysmoothing** (optparameters ∗opm)
- int **splitsliver** (triface ∗, REAL, int)
- long **removeslivers** (int)
- void **optimizemesh** ()
- int **checkmesh** (int topoflag)
- int **checkshells** ()
- int **checksegments** ()
- int **checkdelaunay** (int perturb=1)
- int **checkregular** (int)
- int **checkconforming** (int)
- void **printfcomma** (unsigned long n)
- void **qualitystatistics** ()
- void **memorystatistics** ()
- void **statistics** ()
- void **jettisonnodes** ()
- void **highorder** ()
- void **indexelements** ()
- void **numberedges** ()
- void **outnodes** (tetgenio ∗)
- void **outmetrics** (tetgenio ∗)
- void **outelements** (tetgenio ∗)
- void **outfaces** (tetgenio ∗)
- void **outhullfaces** (tetgenio ∗)
- void **outsubfaces** (tetgenio ∗)
- void **outedges** (tetgenio ∗)
- void **outsubsegments** (tetgenio ∗)
- void **outneighbors** (tetgenio ∗)
- void **outvoronoi** (tetgenio ∗)
- void **outsmesh** (char ∗)
- void **outmesh2medit** (char ∗)
- void **outmesh2vtk** (char ∗)
- void **initializetetgenmesh** ()
- void **freememory** ()

## Public Attributes

- tetgenio ∗ **in**
- tetgenio ∗ **addin**
- tetgenbehavior ∗ **b**
- tetgenmesh ∗ **bgm**
- memorypool ∗ **tetrahedrons**
- memorypool ∗ **subfaces**
- memorypool ∗ **subsegs**
- memorypool ∗ **points**
- memorypool ∗ **tet2subpool**
- memorypool ∗ **tet2segpool**
- memorypool ∗ **badtetrahedrons**
- memorypool ∗ **badsubfacs**
- memorypool ∗ **badsubsegs**
- memorypool ∗ **flippool**
- arraypool ∗ **unflipqueue**
- badface ∗ **flipstack**

- arraypool ∗ **cavetetlist**
- arraypool ∗ **cavebdrylist**
- arraypool ∗ **caveoldtetlist**
- arraypool ∗ **cavetetshlist**
- arraypool ∗ **cavetetseglist**
- arraypool ∗ **cavetetvertlist**
- arraypool ∗ **caveencshlist**
- arraypool ∗ **caveencseglist**
- arraypool ∗ **caveshlist**
- arraypool ∗ **caveshbdlist**
- arraypool ∗ **cavesegshlist**
- arraypool ∗ **subsegstack**
- arraypool ∗ **subfacstack**
- arraypool ∗ **subvertstack**
- arraypool ∗ **encseglist**
- arraypool ∗ **encshlist**
- int ∗ **idx2facetlist**
- point ∗ **facetverticeslist**
- point ∗ **segmentendpointslist**
- point **dummypoint**
- triface **recenttet**
- face **recentsh**
- point ∗ **highordertable**
- int **numpointattrib**
- int **numelemattrib**
- int **sizeoftensor**
- int **pointmtrindex**
- int **pointparamindex**
- int **point2simindex**
- int **pointmarkindex**
- int **pointinsradiusindex**
- int **elemattribindex**
- int **volumeboundindex**
- int **elemmarkerindex**
- int **shmarkindex**
- int **areaboundindex**
- int **checksubsegflag**
- int **checksubfaceflag**
- int **checkconstraints**
- int **nonconvex**
- int **autofliplinklevel**
- int **useinsertradius**
- long **samples**
- unsigned long **randomseed**
- REAL **cosmaxdihed**
- REAL **cosmindihed**
- REAL **cossmtdihed**
- REAL **cosslidihed**
- REAL **minfaceang**
- REAL **minfacetdihed**
- REAL **tetprism_vol_sum**
- REAL **longest**
- REAL **minedgelength**
- REAL **xmax**
- REAL **xmin**

- REAL **ymax**
- REAL **ymin**
- REAL **zmax**
- REAL **zmin**
- long **insegments**
- long **hullsize**
- long **meshedges**
- long **meshhulledges**
- long **steinerleft**
- long **dupverts**
- long **unuverts**
- long **nonregularcount**
- long **st_segref_count**
- long **st_facref_count**
- long **st_volref_count**
- long **fillregioncount**
- long **cavitycount**
- long **cavityexpcount**
- long **flip14count**
- long **flip26count**
- long **flipn2ncount**
- long **flip23count**
- long **flip32count**
- long **flip44count**
- long **flip41count**
- long **flip31count**
- long **flip22count**
- unsigned long **totalworkmemory**
- int **transgc** [8][3][8]
- int **tsb1mod3** [8]

## Static Public Attributes

- static REAL **PI** = 3.14159265358979323846264338327950288419716939937510582
- static int **bondtbl** [12][12] = {{0,},}
- static int **fsymtbl** [12][12] = {{0,},}
- static int **esymtbl** [12] = {9, 6, 11, 4, 3, 7, 1, 5, 10, 0, 8, 2}
- static int **enexttbl** [12] = {0,}
- static int **eprevtbl** [12] = {0,}
- static int **enextesymtbl** [12] = {0,}
- static int **eprevesymtbl** [12] = {0,}
- static int **eorgoppotbl** [12] = {0,}
- static int **edestoppotbl** [12] = {0,}
- static int **facepivot1** [12] = {0,}
- static int **facepivot2** [12][12] = {{0,},}
- static int **orgpivot** [12] = {7, 7, 5, 5, 6, 4, 4, 6, 5, 6, 7, 4}
- static int **destpivot** [12] = {6, 4, 4, 6, 5, 6, 7, 4, 7, 7, 5, 5}
- static int **apexpivot** [12] = {5, 6, 7, 4, 7, 7, 5, 5, 6, 4, 4, 6}
- static int **oppopivot** [12] = {4, 5, 6, 7, 4, 5, 6, 7, 4, 5, 6, 7}
- static int **tsbondtbl** [12][6] = {{0,},}
- static int **stbondtbl** [12][6] = {{0,},}
- static int **tspivottbl** [12][6] = {{0,},}
- static int **stpivottbl** [12][6] = {{0,},}
- static int **ver2edge** [12] = {0, 1, 2, 3, 3, 5, 1, 5, 4, 0, 4, 2}

- static int **edge2ver** [6] = {0, 1, 2, 3, 8, 5}
- static int **epivot** [12] = {4, 5, 2, 11, 4, 5, 2, 11, 4, 5, 2, 11}
- static int **sorgpivot** [6] = {3, 4, 4, 5, 5, 3}
- static int **sdestpivot** [6] = {4, 3, 5, 4, 3, 5}
- static int **sapexpivot** [6] = {5, 5, 3, 3, 4, 4}
- static int **snextpivot** [6] = {2, 5, 4, 1, 0, 3}

### 7.57.1 Detailed Description

Definition at line 849 of file tetgen.h.

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

- modules/tetgen/tetgen.h
- modules/tetgen/tetgen.cpp
- modules/tetgen/tetgen.cxx

## 7.58 tri2mesh Class Reference

### Public Attributes

- vector< int > **celltarg**
- vector< double > **constrinfluence**

### 7.58.1 Detailed Description

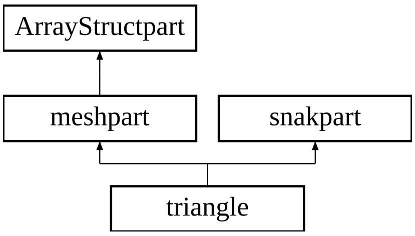
Definition at line 101 of file triangulate.hpp.

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

- incl/triangulate.hpp

## 7.59 triangle Class Reference

Inheritance diagram for triangle:



### Public Member Functions

- void **disp** () const override
- void **disptree** (const mesh &meshin, int n) const override
- int **Key** () const override
- int **KeyParent** () const override
- void **ChangeIndices** (int nVert, int nEdge, int nSurf, int nVolu) override
- void **PrepareForUse** () override
- bool **isready** (bool isInMesh) const override
- void **SwitchIndex** (int typeInd, int oldInd, int newInd)
- void **read** (FILE ∗fid) override
- void **write** (FILE ∗fid) const override
- void **TightenConnectivity** () override
- void **SetPointType** (int a, int b, int c)

### Public Attributes

- vector< int > **pointtype**
- vector< int > **pointind**
- int **parentsurf** =0
- int **parenttype** =0
- tri2mesh **connec**

### Private Attributes

- bool **isTriangleReady** =false

### 7.59.1 Detailed Description

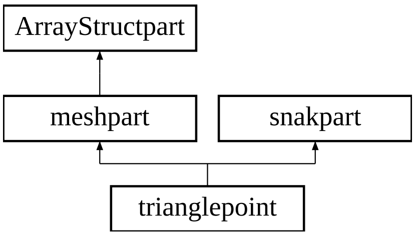
Definition at line 108 of file triangulate.hpp.

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

- incl/triangulate.hpp
- src/snake/triangulate.cpp

## 7.60 trianglepoint Class Reference

Inheritance diagram for trianglepoint:



### Public Member Functions

- void **disp** () const override
- void **disptree** (const mesh &meshin, int n) const override
- int **Key** () const override
- int **KeyParent** () const override
- void **ChangeIndices** (int nVert, int nEdge, int nSurf, int nVolu) override
- void **ChangeIndicesSnakeMesh** (int nVert, int nEdge, int nSurf, int nVolu)
- void **PrepareForUse** () override
- bool **isready** (bool isInMesh) const override
- void **SwitchIndex** (int typeInd, int oldInd, int newInd)
- void **read** (FILE ∗fid) override
- void **write** (FILE ∗fid) const override
- void **TightenConnectivity** () override

### Public Attributes

- coordvec **coord**
- int **parentsurf** =0
- int **parentType** =0
- int **nInfluences** =0

### 7.60.1 Detailed Description

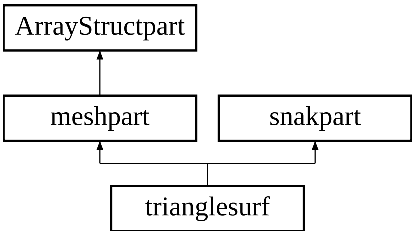
Definition at line 149 of file triangulate.hpp.

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

- incl/triangulate.hpp
- src/snake/triangulate.cpp

## 7.61 trianglesurf Class Reference

Inheritance diagram for trianglesurf:



### Public Member Functions

- void **disp** () const override
- void **disptree** (const mesh &meshin, int n) const override
- int **Key** () const override
- int **KeyParent** () const override
- void **ChangeIndices** (int nVert, int nEdge, int nSurf, int nVolu) override
- void **ChangeIndicesSnakeMesh** (int nVert, int nEdge, int nSurf, int nVolu)
- void **PrepareForUse** () override
- bool **isready** (bool isInMesh) const override
- void **SwitchIndex** (int typeInd, int oldInd, int newInd)
- void **read** (FILE ∗fid) override
- void **write** (FILE ∗fid) const override
- void **TightenConnectivity** () override

### Public Attributes

- vector< int > **indvert**
- vector< int > **typevert**
- vector< int > **voluind**
- int **parentsurfmesh** =0

### 7.61.1 Detailed Description

Definition at line 176 of file triangulate.hpp.

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

- incl/triangulate.hpp
- src/snake/triangulate.cpp

## 7.62 triangulation Class Reference

### Public Member Functions

- void **disp** () const
- void **PrepareForUse** ()
- void **CleanDynaTri** ()
- void **CalcTriVertPosDyna** (int ii)
- void **CalcTriVertPosDyna** ()
- void **CalcTriVertPos** (int ii)
- void **CalcTriVertPos** ()
- void **SetActiveStaticTri** ()
- void **SetConnectivity** ()
- void **SetConnectivityStat** (int ii)
- void **SetConnectivityInter** (int ii)
- void **SetConnectivityDyna** (int ii)
- **triangulation** (mesh &meshin)

### Public Attributes

- vector< int > **acttri**
- triarray **stattri**
- triarray **dynatri**
- triarray **intertri**
- tripointarray **trivert**
- trisurfarray **trisurf**
- snake ∗ **snakeDep** =NULL
- mesh ∗ **meshDep** =NULL

### 7.62.1 Detailed Description

Definition at line 62 of file triangulate.hpp.

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

- incl/triangulate.hpp
- src/snake/triangulate.cpp

## 7.63 tetgenmesh::triface Class Reference

### Public Member Functions

- triface & **operator=** (const triface &t)

### Public Attributes

- tetrahedron ∗ **tet**
- int **ver**

### 7.63.1 Detailed Description

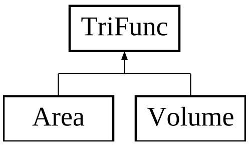
Definition at line 974 of file tetgen.h.

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

- modules/tetgen/tetgen.h

## 7.64 TriFunc Class Reference

Inheritance diagram for TriFunc:



### Public Member Functions

- bool **CheckValid** ()
- bool **MakeValid** ()
- void **PreCalc** ()
- void **assign** (const vector< double > &in0, const vector< double > &in1, const vector< double > &in2)
- void **assign** (const vector< double > ∗in0, const vector< double > ∗in1, const vector< double > ∗in2)
- void **assign** (int pRepI, const vector< double > &pRep)
- void **ReturnDatPoint** (double ∗∗a, ArrayVec< double > ∗∗b, ArrayVec< double > ∗∗c)
- virtual void **Calc** ()=0
- **TriFunc** (int a)

### Protected Member Functions

- bool **MakeValidField** (vector< double > ∗TriFunc::∗mp)

### Protected Attributes

- vector< double > const ∗ **p0** =NULL
- vector< double > const ∗ **p1** =NULL
- vector< double > const ∗ **p2** =NULL
- double **fun**
- ArrayVec< double > **jac**
- ArrayVec< double > **hes**
- bool **isReady**
- bool **isCalc**
- int **nTarg**

### 7.64.1 Detailed Description

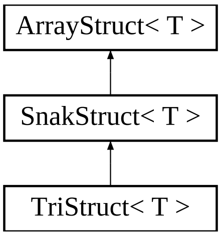
Definition at line 30 of file RSVSmath.hpp.

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

- incl/RSVSmath.hpp
- src/rsvs/RSVSmath.cpp

## 7.65 TriStruct< T > Class Template Reference

Inheritance diagram for TriStruct< T >:



## Friends

- class **triangulation**

## Additional Inherited Members

### 7.65.1 Detailed Description

**template<class T>**
**class TriStruct< T >**

Definition at line 43 of file triangulate.hpp.

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

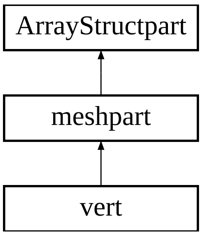
- incl/triangulate.hpp

## 7.66 vert Class Reference

Class for a vertex in a mesh.

`#include <mesh.hpp>`

Inheritance diagram for vert:



## Public Member Functions

- void **disp** () const
- void **disptree** (const mesh &meshin, int n) const
- void **ChangeIndices** (int nVert, int nEdge, int nSurf, int nVolu)
- void **PrepareForUse** ()
- bool **isready** (bool isInMesh) const
- void **read** (FILE ∗fid)
- void **write** (FILE ∗fid) const
- void **TightenConnectivity** ()
- **vert** (const vert &oldEdge)
- void **operator=** (const vert ∗other)
- int **Key** () const

## Public Attributes

- vector< int > **edgeind**
- vector< double > **coord**

### 7.66.1 Detailed Description

Class for a vertex in a mesh.

Definition at line 369 of file mesh.hpp.

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

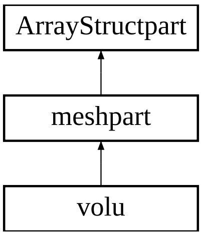
- incl/mesh.hpp
- src/grid/mesh.cpp

## 7.67 volu Class Reference

Class for volume cell objects in a mesh.

```
#include <mesh.hpp>
```

Inheritance diagram for volu:



## Public Member Functions

- void **ChangeIndices** (int nVert, int nEdge, int nSurf, int nVolu)
- void **disp** () const
- void **disptree** (const mesh &meshin, int n) const
- void **PrepareForUse** ()
- bool **isready** (bool isInMesh) const
- void **read** (FILE ∗fid)
- void **write** (FILE ∗fid) const
- void **TightenConnectivity** ()
- **volu** (const volu &oldVolu)
- void **operator=** (const volu ∗other)
- int **Key** () const

### Public Attributes

- double **fill**
- double **target**
- double **error**
- double **volume**
- vector< int > **surfind**

### 7.67.1 Detailed Description

Class for volume cell objects in a mesh.

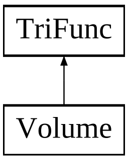
Definition at line 165 of file mesh.hpp.

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

- incl/mesh.hpp
- src/grid/mesh.cpp

## 7.68 Volume Class Reference

Inheritance diagram for Volume:



### Public Member Functions

- void **Calc** () override
- void **CalcFD** ()

### Private Member Functions

- **TriFunc** ()
- **TriFunc** (int a)
- void **PreCalc** ()

### Private Attributes

- vector< double > const ∗ **p0**
- vector< double > const ∗ **p1**
- vector< double > const ∗ **p2**
- double **fun**
- ArrayVec< double > **jac**
- ArrayVec< double > **hes**

### Additional Inherited Members

### 7.68.1 Detailed Description

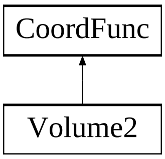
Definition at line 144 of file RSVSmath.hpp.

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

- incl/RSVSmath.hpp
- src/rsvs/RSVSmath.cpp

## 7.69 Volume2 Class Reference

Inheritance diagram for Volume2:



### Public Member Functions

- void **Calc** () override

### Private Member Functions

- void **PreCalc** ()

### Private Attributes

- vector$<$ vector$<$ double $>$ const $*$ $>$ **coords**
- double **fun**
- ArrayVec$<$ double $>$ **jac**
- ArrayVec$<$ double $>$ **hes**

### Additional Inherited Members

### 7.69.1 Detailed Description

Definition at line 188 of file RSVSmath.hpp.

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

- incl/RSVSmath.hpp
- src/rsvs/RSVSmath.cpp

## 7.70 tetgenio::voroedge Struct Reference

### Public Attributes

- int **v1**
- int **v2**
- REAL **vnormal** [3]

### 7.70.1 Detailed Description

Definition at line 142 of file tetgen.h.

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

- modules/tetgen/tetgen.h

## 7.71 tetgenio::vorofacet Struct Reference

### Public Attributes

- int **c1**
- int **c2**
- int ∗ **elist**

### 7.71.1 Detailed Description

Definition at line 154 of file tetgen.h.

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

- modules/tetgen/tetgen.h

## 7.72 param::voronoi Class Reference

Class for handling of voronoi VOS meshing parameters.

```
#include <parameters.hpp>
```

### Public Member Functions

- void **PrepareForUse** ()
- void **ReadPoints** ()

### Public Attributes

- std::vector< double > inputpoints

  *Vector of input points, 4 datums per point.*
- double distancebox

  *Distance at which to build the bounding box of the mesh.*
- std::string pointfile

  *A string pointing to a file containing the set of inputpoints.*
- double snakecoarseness

  *The coarseneness level of the snaking mesh that will be generated.*

### 7.72.1 Detailed Description

Class for handling of voronoi VOS meshing parameters.

Definition at line 115 of file parameters.hpp.

### 7.72.2 Member Data Documentation

#### 7.72.2.1 snakecoarseness

```
double param::voronoi::snakecoarseness
```

The coarseneness level of the snaking mesh that will be generated.

1 -> same as VOS, 0.1 -> 1 tenth the edge length of the VOS.

Definition at line 126 of file parameters.hpp.

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

- incl/parameters.hpp
- src/parameters.cpp

## 7.73 param::voxel Class Reference

Parameters controlling cartesian grid properties.

```
#include <parameters.hpp>
```

### Public Member Functions

- void **PrepareForUse** ()

## Public Attributes

- std::array< int, 3 > gridsizebackground

  *Size of the Background grid on which the VOS is defined.*
- std::array< int, 3 > gridsizesnake

  *Size of the Snaking grid on which the snake is defined.*

### 7.73.1 Detailed Description

Parameters controlling cartesian grid properties.

Definition at line 100 of file parameters.hpp.

### 7.73.2 Member Data Documentation

#### 7.73.2.1 gridsizesnake

```
std::array<int, 3> param::voxel::gridsizesnake
```

Size of the Snaking grid on which the snake is defined.

final size = gridsizebackground∗gridsizesnake

Definition at line 107 of file parameters.hpp.

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

- incl/parameters.hpp
- src/parameters.cpp

# Chapter 8

# File Documentation

## 8.1 incl/arraystructures.hpp File Reference

Provide vector container with hashed index mapping.

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cstdlib>
#include <string>
#include <sstream>
#include <stdexcept>
#include <unordered_map>
#include <functional>
#include "warning.hpp"
#include "arraystructures_incl.cpp"
#include "snakstruct_incl.cpp"
```

### Classes

- class ArrayStruct< T >
- class HashedVector< T, Q, R >
- class SnakStruct< T >
- class ArrayStruct< T >
- class SnakStruct< T >
- class ModiftrackArray< T >
- class HashedVector< T, Q, R >
- class HashedMap< T, Q, R >
- class HashedVectorSafe< T, Q, R >
- class ArrayStructpart
- class snakpart
- class modiftrackpart

### Typedefs

- typedef unsigned int **unsigned_int**

## Functions

- template<class T >
  int **TestTemplate_ArrayStruct** ()
- bool **CompareFuncOut** (function< void()> func1, function< void()> func2)
- template<typename T >
  void **sort** (vector< T > &vec)
- template<typename T >
  void **unique** (vector< T > &vec)
- template<typename T >
  void **set_intersection** (vector< T > &targVec, const vector< T > &vec1, const vector< T > &vec2, bool isSort)
- template<class T >
  vector< int > **FindSubList** (const vector< T > &keyFind, const vector< T > &keyList, unordered_↩ multimap< T, int > &hashTable)
- template<class T >
  vector< int > **FindSubList** (const vector< T > &keyFind, const vector< T > &keyList, const unordered_↩ multimap< T, int > &hashTable)
- template<class T , class Q >
  void **HashVector** (const vector< T > &elems, unordered_multimap< T, Q > &hashTable, const vector< Q > &targElems={})
- template<class T >
  int **FindSub** (const T &key, const unordered_multimap< T, int > &hashTable)
- template<class T >
  void **ConcatenateVector** (vector< T > &vecRoot, const vector< T > &vecConcat)
- template<class T , class R >
  vector< R > **ReturnDataEqualRange** (T key, const unordered_multimap< T, R > &hashTable)
- template<class T , class R >
  void **ReturnDataEqualRange** (T key, const unordered_multimap< T, R > &hashTable, vector< R > &sub↩ List)
- template<class T >
  bool **CompareDisp** (T ∗mesh1, T ∗mesh2)
- template<class T >
  int **TestReadyness** (T &stackT, const char ∗txt, bool errTarg)
- template<class T >
  void **DisplayVector** (vector< T > vec)
- template<class T >
  void **DisplayVectorStatistics** (vector< T > vec)
- template<class T , class R >
  R **ConcatenateVectorField** (const ArrayStruct< T > &arrayIn, R T::∗mp, const vector< int > &subList)
- template<class T , class R >
  vector< R > **ConcatenateScalarField** (const ArrayStruct< T > &arrayIn, R T::∗mp, const vector< int > &subList)
- template<class T , class R >
  R **ConcatenateVectorField** (const ArrayStruct< T > &arrayIn, R T::∗mp, int rStart, int rEnd)
- template<class T , class R >
  vector< R > **ConcatenateScalarField** (const ArrayStruct< T > &arrayIn, R T::∗mp, int rStart, int rEnd)
- template<class T , class R , class U , class V >
  void **OperArrayStructMethod** (const ArrayStruct< T > &arrayIn, const vector< int > &subList, R T::∗mp, U &out, V oper)
- template<template< class Q, class R > class T, class Q , class R >
  void **EraseKeyPair** (T< Q, R > hashTable, Q key, R pos)

### 8.1.1 Detailed Description

Provide vector container with hashed index mapping.

This file provides classes and methods for the handling of groups of mesh subcomponents...

## 8.2 incl/filesystem.hpp File Reference

Custom filesystem header Faff about with filesystem depending on version To give a readable compile time error if incompatible things are attempted.

```
#include <boost/filesystem.hpp>
```

### 8.2.1 Detailed Description

Custom filesystem header Faff about with filesystem depending on version To give a readable compile time error if incompatible things are attempted.

Basically a workaround gcc on windows hating me.

## 8.3 incl/main.hpp File Reference

file containing the main functions and the command line parser.

```
#include <string>
```

### Namespaces

- param

  *Namespace containing the parameter classes used to control execution of the 3D-RSVS program.*

### Functions

- int **RSVSExecution** (int argc, char ∗argv[ ])
- void **NoExecution** (int execFlow, param::parameters &paramconf)
- int **parse::CommandLineParser** (int argc, char ∗argv[ ], param::parameters &paramconf)
- void **parse::config::useconfig** (const std::string &confCase, param::parameters &paramconf)
- void **parse::config::loadconfig** (const std::string &confCase, param::parameters &paramconf)

### 8.3.1 Detailed Description

file containing the main functions and the command line parser.

## 8.4 incl/matrixtools.hpp File Reference

Tools to support conversion, display and derivatives of Eigen matrices.

```
#include <vector>
#include <iostream>
#include <fstream>
#include <string>
#include "vectorarray.hpp"
```

## Functions

- template<class T >
  void **PrintMatrixFile** (const std::vector< T > &mat, const char ∗name)
- void **Deriv1stChainScalar** (const Eigen::MatrixXd &dSdc, const Eigen::MatrixXd &dcdd, Eigen::MatrixXd &dSdd)
- void **Deriv2ndChainScalar** (const Eigen::MatrixXd &dSdc, const Eigen::MatrixXd &dcdd, const Eigen::↩ MatrixXd &HSc, const Eigen::MatrixXd &Hcd, Eigen::MatrixXd &HSd)
- void **VecBy3DimArray** (const Eigen::MatrixXd &vec, const Eigen::MatrixXd &arr3dim, Eigen::MatrixXd &retArray)
- void **ArrayVec2MatrixXd** (const ArrayVec< double > &arrayIn, Eigen::MatrixXd &matOut)
- void **PrintMatrix** (const Eigen::MatrixXd &mat)
- void **PrintMatrixFile** (const Eigen::MatrixXd &mat, const char ∗name)
- void **PrintMatrix** (const Eigen::RowVectorXd &mat)
- void **PrintMatrix** (const Eigen::VectorXd &mat)
- double **StreamStatistics** (const Eigen::VectorXd &&vec, std::ofstream &out, const std::string &&sep=std↩ ::string(", "))
- void **StreamOutVector** (const Eigen::VectorXd &&vec, std::ofstream &out, const std::string &&sep=std↩ ::string(", "))

### 8.4.1 Detailed Description

Tools to support conversion, display and derivatives of Eigen matrices.

## 8.5 incl/mesh.hpp File Reference

Provides all the mesh tools used for the generation of 3D grids and geometries.

```
#include <iostream>
#include <array>
#include <vector>
#include <algorithm>
#include <cstdlib>
#include <string>
#include <sstream>
#include <stdexcept>
#include <unordered_map>
#include <functional>
#include <cmath>
#include "arraystructures.hpp"
```

## Classes

- class coordvec

  *Handles the use and norm of a vector for which the norm and the unit value might be needed.*
- class meshpart

  */Abstract class to ensure mesh interfaces are correct.*
- class volu

  *Class for volume cell objects in a mesh.*
- class surf

*Class for surface object in a mesh.*

- class edge

  *Class for an edge object in a mesh.*
- class vert

  *Class for a vertex in a mesh.*
- class ConnecRemv

  *Class containing the information needed to trim objects from a mesh.*
- class meshdependence

  *Class for connecting meshes.*
- class mesh

  *Class for mesh handling.*

## Typedefs

- typedef ModiftrackArray< surf > **surfarray**
- typedef ArrayStruct< volu > **voluarray**
- typedef ModiftrackArray< edge > **edgearray**
- typedef ArrayStruct< vert > **vertarray**
- typedef std::array< std::array< double, 2 >, 3 > **grid::limits**
- typedef std::array< std::array< double, 3 >, 3 > grid::transformation

  *Defines a linear transformation to the mesh where for each dimension: {new minimum, old minimum , scaling}.*

## Functions

- void **ConnVertFromConnEdge** (const mesh &meshin, const vector< int > &edgeind, vector< int > &vertind)
- int **OrderMatchLists** (const vector< int > &vec1, const vector< int > &vec2, int p1, int p2)
- void CropMeshGreedy (mesh &meshin, const std::vector< double > &lb, const std::vector< double > &ub)

  *Crops a mesh to only the elements inside the cropBox.*
- int OrderEdgeList (vector< int > &edgeind, const mesh &meshin, bool warn=true, bool errout=true, const vector< int > ∗edgeIndOrigPtr=NULL, const surf ∗surfin=NULL)

  *Orders a list of edge to be connected.*
- double VertexDistanceToPlane (const vector< double > &planeVert1, const vector< double > &planeVert2, const vector< double > &planeVert3, const vector< double > &testVertex, coordvec &temp1, coordvec &temp2)

  *Calculates the distance from a vertex to a plane.*
- vector< double > VerticesDistanceToPlane (const vector< double > &planeVert1, const vector< double > &planeVert2, const vector< double > &planeVert3, const vector< double > &testVertices, coordvec &temp1, coordvec &temp2)

  *Calculates the distance from a set of vertices to a plane.*
- double **VertexDistanceToPlane** (const vector< double > &planeVert1, const vector< double > &plane↩Vert2, const vector< double > &planeVert3, const vector< double > &testVertex)
- vector< double > **VerticesDistanceToPlane** (const vector< double > &planeVert1, const vector< double > &planeVert2, const vector< double > &planeVert3, const vector< double > &testVertices)
- mesh Points2Mesh (const std::vector< double > &vecPts, int nProp=3)

  *Takes in a set of points and returns a mesh of points ready for voronisation.*
- double **PlanesDotProduct** (const vector< double > &planeVert1, const vector< double > &planeVert2, const vector< double > &planeVert3, const vector< double > &planeVert4, const vector< double > &planeVert5, const vector< double > &planeVert6, bool normalize=true)
- void PlaneNormal (const vector< double > &planeVert1, const vector< double > &planeVert2, const vector< double > &planeVert3, coordvec &normal, coordvec &temp1)

*Calculates a plane's normal vector.*

- template<class T , class V , class W >
  double **meshhelp::ProjectRay** (int count, const W &&boundBox, const T &dir, const V &orig, double min↩ Dist=0.0)
- void **meshhelp::PlaceBorderVertex** (const std::vector< double > &coordIn, const std::vector< double > &coordOut, const std::vector< double > &lb, const std::vector< double > &ub, std::vector< double > &coordTarg)
- void **meshhelp::SplitBorderSurfaceEdgeind** (const mesh &meshin, const std::vector< bool > &edgeOut, std::vector< int > &vecconnIn, std::vector< int > &vecconnOut)
- void **meshhelp::SplitBorderVolumeSurfind** (const mesh &meshin, const std::vector< bool > &edgeOut, std::vector< int > &vecconnIn, std::vector< int > &vecconnOut)
- void **meshhelp::HandleMultiSurfaceSplit** (mesh &meshin, vector< int > &edgeindOld, vector< int > &edgeindNew, vector< int > &vertindNew)

  *Handles case for AddBoundary where a surface more than one split.*
- std::vector< int > **meshhelp::FindVertInFromEdgeOut** (const mesh &meshin, const std::vector< bool > &vertOut, const std::vector< int > &edgeList, const std::vector< int > &edgeListCheck)
- std::vector< int > **meshhelp::FindEdgeInFromSurfOut** (const mesh &meshin, const std::vector< bool > &edgeOut, std::vector< int > surfList)
- double **meshhelp::VerticesDistanceSquared** (const mesh &meshin, const vector< int > &vertind)
- double **meshhelp::VerticesDistance** (const mesh &meshin, const vector< int > &vertind)
- bool **meshhelp::IsVerticesDistance0** (const mesh &meshin, const vector< int > &vertind, double eps)
- int **meshhelp::VertexInVolume** (const mesh &meshin, const vector< double > testCoord, bool needFlip)
- int **meshhelp::Get3PointsInSurface** (const mesh &meshin, int surfCurr, std::array< int, 3 > &surfacePoints)

  *Gets 3 unique points in the surface.*
- int **Test_ArrayStructures** ()
- int **Test_Volu** ()
- int **Test_Surf** ()
- int **Test_Vert** ()
- int **Test_Edge** ()
- int **Test_Mesh** ()
- int **Test_Crop** ()

## 8.5.1 Detailed Description

Provides all the mesh tools used for the generation of 3D grids and geometries.

This file provides the mesh class and it's associated sub-component. These can be used to robustly control changes in geometry.

## 8.5.2 Function Documentation

### 8.5.2.1 CropMeshGreedy()

```
void CropMeshGreedy (
            mesh & meshin,
            const std::vector< double > & lb,
            const std::vector< double > & ub )
```

Crops a mesh to only the elements inside the cropBox.

Anything impinging on the cropBox is deleted. Steps: 1 - Find vertices out of the box 2 - Delete those vertices 3 - Propagate the deletion to the higher level containers 4 - Propagate back to the lower level containers removing empty connectors.

**Parameters**

| | | |
|---|---|---|
| | *meshin* | Input mesh |
| in | *lb* | lower bound vector |
| in | *ub* | upper bound vector |

Definition at line 4659 of file mesh.cpp.

#### 8.5.2.2 Get3PointsInSurface()

```
int meshhelp::Get3PointsInSurface (
            const mesh & meshin,
            int surfCurr,
            std::array< int, 3 > & surfacePoints )
```

Gets 3 unique points in the surface.

These points are guaranteed to be separate

**Parameters**

| | | |
|---|---|---|
| in | *meshin* | The input mesh |
| in | *surfCurr* | current surface index to analyse. |
| out | *surfacePoints* | The surface points indices. |

**Returns**

The number of points which have been found.

Definition at line 4954 of file mesh.cpp.

#### 8.5.2.3 HandleMultiSurfaceSplit()

```
void meshhelp::HandleMultiSurfaceSplit (
            mesh & meshin,
            vector< int > & edgeindOld,
            vector< int > & edgeindNew,
            vector< int > & vertindNew )
```

Handles case for AddBoundary where a surface more than one split.

**Parameters**

| | |
|---|---|
| *meshin* | The mesh |
| *edgeindOld* | the edgeind internal |
| *edgeindNew* | the edgeind out of the boundary (that will be split) |
| *vertindNew* | the vertex index that must be trimmed to be of size 2 |

Definition at line 4791 of file mesh.cpp.

### 8.5.2.4 OrderEdgeList()

```
int OrderEdgeList (
            vector< int > & edgeind,
            const mesh & meshin,
            bool warn,
            bool errout,
            const vector< int > * edgeIndOrigPtr,
            const surf * surfin )
```

Orders a list of edge to be connected.

This list of edges is ordered in place. THis will not work for self crossing lists.

**Parameters**

| | | |
|---|---|---|
| | *edgeind* | The edgeind |
| in | *meshin* | The meshin |
| in | *warn* | The warning |
| in | *errout* | The errout |

**Returns**

> the return value is a flag which can be: 0 - the edges have been ordered and closed 1 - the edges have been ordered and closed but the list was truncated. <0 - The edges are ordered but not closed. (need errout to be false) for edgeind of size -<return val>="">

Definition at line 2421 of file mesh.cpp.

### 8.5.2.5 PlaneNormal()

```
void PlaneNormal (
            const vector< double > & planeVert1,
            const vector< double > & planeVert2,
            const vector< double > & planeVert3,
            coordvec & normal,
            coordvec & temp1 )
```

Calculates a plane's normal vector.

**Parameters**

| | | |
|---|---|---|
| in | *planeVert1* | The plane vertex 1 |
| in | *planeVert2* | The plane vertex 2 |
| in | *planeVert3* | The plane vertex 3 |
| | *normal* | The normal |
| | *temp1* | The temporary 1 |

Definition at line 224 of file mesh.cpp.

#### 8.5.2.6 Points2Mesh()

```
mesh Points2Mesh (
             const std::vector< double > & vecPts,
             int nProp )
```

Takes in a set of points and returns a mesh of points ready for voronisation.

**Parameters**

| | | |
|---|---|---|
| in | *vecPts* | The vector of points |
| in | *nProp* | number of properties per point |

**Returns**

The points in the mesh format

Definition at line 4688 of file mesh.cpp.

#### 8.5.2.7 VertexDistanceToPlane()

```
double VertexDistanceToPlane (
             const vector< double > & planeVert1,
             const vector< double > & planeVert2,
             const vector< double > & planeVert3,
             const vector< double > & testVertex,
             coordvec & temp1,
             coordvec & temp2 )
```

Calculates the distance from a vertex to a plane.

calculates the distance from a plane to a vertex, with the plane defined by three vertices.

Two optional arguments can be provided to avoid the need for memory allocation if this is called in a loop. For max speedup if testing a surface multiple times against many vertices temp2 can be reused

**Parameters**

| | | |
|---|---|---|
| in | *planeVert1* | The plane vertex 1 |
| in | *planeVert2* | The plane vertex 2 |
| in | *planeVert3* | The plane vertex 3 |
| in | *testVertex* | The test vertex |
| | *[in\|out]* | temp1 The temporary array 1 |
| | *[in\|out]* | temp2 The temporary array 2 |

**Returns**

the distance from the plane to the vertex

Definition at line 256 of file mesh.cpp.

#### 8.5.2.8 VerticesDistanceToPlane()

```
vector<double> VerticesDistanceToPlane (
            const vector< double > & planeVert1,
            const vector< double > & planeVert2,
            const vector< double > & planeVert3,
            const vector< double > & testVertices,
            coordvec & temp1,
            coordvec & temp2 )
```

Calculates the distance from a set of vertices to a plane.

Calculates the distance from a plane to a vertex, with the plane defined by three vertices.

Two optional arguments can be provided to avoid the need for memory allocation if this is called in a loop. For max speedup if testing a surface multiple times against many vertices temp2 is reused internally allowing surface properties to only be calculated once.

**Parameters**

| | | |
|---|---|---|
| in | *planeVert1* | The plane vertex 1 |
| in | *planeVert2* | The plane vertex 2 |
| in | *planeVert3* | The plane vertex 3 |
| in | *testVertices* | The test vertices |
| | *[in\|out]* | temp1 The temporary array 1 |
| | *[in\|out]* | temp2 The temporary array 2 |

**Returns**

the distance from the plane to the vertex

Definition at line 294 of file mesh.cpp.

## 8.6 incl/meshprocessing.hpp File Reference

Tools for the mathematical processing of meshes.

```
#include <vector>
#include "arraystructures.hpp"
#include "mesh.hpp"
#include "snake.hpp"
```

## Functions

- void **FlattenBoundaryFaces** (mesh &meshin)
- std::vector< int > **FindHolesInSnake** (const snake &snakein, const HashedVector< int, int > &uncertain↩ Vert)
- void PrepareSnakeForCFD (const snake &snakein, double distanceTol, mesh &meshgeom, std::vector< double > &holeCoords)

  *Prepares the snake to be used for CFD, removes duplicate points and triangulates it.*
- HashedVector< int, int > **GroupCloseSnaxels** (const snake &snakein, double distTol)
- void **TestVertClose** (int vertIndIn, std::vector< bool > &isSnaxDone, const mesh &meshin, double distTol, std::vector< int > &sameEdges)
- HashedVector< int, int > **GroupCloseVertices** (const mesh &meshin, double distTol)
- int **FindVertexHole** (int vertInd, const mesh &meshin, const std::vector< bool > &vertIn, const HashedVector< int, int > &uncertainVert, std::vector< bool > &vertExplored)
- double **DomInter** (double x, double y1, double y2)
- mesh **BuildDomain** (const std::array< double, 3 > &lowerB, const std::array< double, 3 > &upperB, double tolInner=0.0)
- mesh BuildCutCellDomain (const std::array< double, 3 > &outerLowerB, const std::array< double, 3 > &outerUpperB, const std::array< double, 3 > &innerLowerB, const std::array< double, 3 > &innerUpperB, int nSteps, std::vector< int > &vertPerSubDomain)

  *Builds a series of domains with different edge properties controlling the interpolation of the metric.*
- double PseudoSurfaceAngle (const mesh &meshin, const std::array< int, 2 > &surfInds)

  *Calculates the pseudo surface angle.*
- std::vector< double > CalculateEdgeCurvature (const mesh &meshin)

  *Calculates the angles between the surfaces connected at an edge.*
- std::vector< double > CalculateVertexCurvature (const mesh &meshin, int smoothingSteps)

  *Calculates the vertex curvature.*
- std::vector< double > CalculateVertexMinEdgeLength (const mesh &meshin)

  *Calculates the vertex minimum edge length.*
- std::vector< double > CalculateVertexMeanEdgeLength (const mesh &meshin)

  *Calculates the vertex mean edge length.*
- std::vector< double > CalculateEdgeLengths (const mesh &meshin)

  *Calculates the edge lengths.*

## 8.6.1 Detailed Description

Tools for the mathematical processing of meshes.

## 8.6.2 Function Documentation

### 8.6.2.1 BuildCutCellDomain()

```
mesh BuildCutCellDomain (
             const std::array< double, 3 > & outerLowerB,
             const std::array< double, 3 > & outerUpperB,
             const std::array< double, 3 > & innerLowerB,
             const std::array< double, 3 > & innerUpperB,
             int nSteps,
             std::vector< int > & vertPerSubDomain )
```

Builds a series of domains with different edge properties controlling the interpolation of the metric.

**Parameters**

| | | |
|---|---|---|
| in | *outerLowerB* | The outer lower b |
| in | *outerUpperB* | The outer upper b |
| in | *innerLowerB* | The inner lower b |
| in | *innerUpperB* | The inner upper b |
| in | *nSteps* | The steps |
| | *vertPerSubDomain* | The vertical per sub domain |

**Returns**

The cut cell domain.

These also serve to avoid having a badly conditioned initial triangulation with very small edges.

`nSteps` is the number of total domains. 0 will return an empty mesh, 1 will return a mesh of the inner bound 2 will return inner and outer bounds,

```
{
    {DomInter(x, innerLowerB[0], outerLowerB[0]),
    DomInter(x, innerUpperB[0], outerUpperB[0])},
    {DomInter(x, innerLowerB[1], outerLowerB[1]),
    DomInter(x, innerUpperB[1], outerUpperB[1])},
    {DomInter(x, innerLowerB[2], outerLowerB[2]),
    DomInter(x, innerUpperB[2], outerUpperB[2])}
}
```

meshtemp = BuildDomain( {DomInter(x, innerLowerB[0], outerLowerB[0]), DomInter(x, innerLowerB[1], outerLowerB[1]), DomInter(x, innerLowerB[2], outerLowerB[2])}, {DomInter(x, innerUpperB[0], outerUpperB[0]), DomInter(x, innerUpperB[1], outerUpperB[1]), DomInter(x, innerUpperB[2], outerUpperB[2])} );

Definition at line 445 of file meshprocessing.cpp.

### 8.6.2.2 CalculateEdgeCurvature()

```
std::vector<double> CalculateEdgeCurvature (
             const mesh & meshin )
```

Calculates the angles between the surfaces connected at an edge.

To work the faces need have a common orientation

**Parameters**

| | | |
|---|---|---|
| in | *meshin* | The input mesh |

**Returns**

The edge angles.

Definition at line 551 of file meshprocessing.cpp.

#### 8.6.2.3 CalculateEdgeLengths()

```
std::vector<double> CalculateEdgeLengths (
             const mesh & meshin )
```

Calculates the edge lengths.

**Parameters**

| | | |
|---|---|---|
| in | *meshin* | The meshin |

**Returns**

The edge lengths.

Definition at line 686 of file meshprocessing.cpp.

#### 8.6.2.4 CalculateVertexCurvature()

```
std::vector<double> CalculateVertexCurvature (
             const mesh & meshin,
             int smoothingSteps )
```

Calculates the vertex curvature.

**Parameters**

| | | |
|---|---|---|
| in | *meshin* | The input mesh |
| in | *smoothingSteps* | The number of metric smoothing steps |

**Returns**

The vertex curvature.

Definition at line 581 of file meshprocessing.cpp.

#### 8.6.2.5 CalculateVertexMeanEdgeLength()

```
std::vector<double> CalculateVertexMeanEdgeLength (
             const mesh & meshin )
```

Calculates the vertex mean edge length.

**Parameters**

|  |  |  |
|---|---|---|
| in | *meshin* | The meshin |

**Returns**

The vertex mean edge length.

Definition at line 656 of file meshprocessing.cpp.

### 8.6.2.6 CalculateVertexMinEdgeLength()

```
std::vector<double> CalculateVertexMinEdgeLength (
            const mesh & meshin )
```

Calculates the vertex minimum edge length.

**Parameters**

|  |  |  |
|---|---|---|
| in | *meshin* | The meshin |

**Returns**

The vertex minimum edge length.

Definition at line 628 of file meshprocessing.cpp.

### 8.6.2.7 PrepareSnakeForCFD()

```
void PrepareSnakeForCFD (
            const snake & snakein,
            double distanceTol,
            mesh & meshgeom,
            std::vector< double > & holeCoords )
```

Prepares the snake to be used for CFD, removes duplicate points and triangulates it.

**Parameters**

|  |  |  |
|---|---|---|
| in | *snakein* | The snakein |
| in | *distanceTol* | The distance tolerance |
|  | *meshgeom* | The meshgeom |
|  | *holeCoords* | The hole coordinates |

Definition at line 137 of file meshprocessing.cpp.

#### 8.6.2.8 PseudoSurfaceAngle()

```
double PseudoSurfaceAngle (
            const mesh & meshin,
            const std::array< int, 2 > & surfInds )
```

Calculates the pseudo surface angle.

This pseudo angle is the dot product between the normal, i

**Parameters**

|  |  |  |
|---|---|---|
| in | *meshin* | The input mesh |
| in | *surfInds* | The surface indices |

**Returns**

dot product between surface normals if facing outwards

Definition at line 499 of file meshprocessing.cpp.

## 8.7 incl/meshrefinement.hpp File Reference

Tools for the refinement and coarsening of meshes.

```
#include "mesh.hpp"
```

### Functions

- void **CoarsenMesh** (const mesh &meshchild, mesh &newparent, const vector< int > &elmMapping)
- void **CartesianMapping** (const mesh &meshin, vector< int > &elmMapping, vector< int > &dims)
- void **CartesianMapping2D** (const mesh &meshin, vector< int > &elmMapping, vector< int > &dims)
- void **CartesianMapping3D** (const mesh &meshin, vector< int > &elmMapping, vector< int > &dims)
- int **Test_MeshRefinement** ()

### 8.7.1 Detailed Description

Tools for the refinement and coarsening of meshes.

## 8.8 incl/parameters.hpp File Reference

Parameters for the integrated 3DRSVS.

```
#include <cstdlib>
#include <array>
#include <string>
#include <vector>
#include <ctime>
```

### Classes

- struct param::filltype< T >

  *The input type of fill information.*
- class param::rsvs

  *Parameters related to the Velocity calculation and VOS steps.*
- class param::snaking

  *Parameters controlling tuning parameters for the stepping of the restricted surface.*
- class param::voxel

  *Parameters controlling cartesian grid properties.*
- class param::voronoi

  *Class for handling of voronoi VOS meshing parameters.*
- class param::grid

  *Class for parameters of the grid generation.*
- class param::ioin

  *Class containing the input configuration these are files to load.*
- class param::ioout

  *Class containing the output configuration these are files to store and where to store them.*
- class param::files

  *Class containing all parameter settings for file operations.*
- class param::parameters

  *Root class for all the parameters.*

### Namespaces

- param

  *Namespace containing the parameter classes used to control execution of the 3D-RSVS program.*
- param::io

  *Provide functions for reading and writing of the parameter structure.*
- param::test

  *Tests for the parameter implementation.*

### Typedefs

- typedef std::array< double, 2 > param::realbounds

  *Collects a lower and an upper bound.*
- typedef std::vector< std::pair< std::string, std::string > > param::exports

  *Collects the export settings which is a vector of pairs of strings.*

### Functions

- void **param::io::read** (const std::string &fileName, parameters &p)
- void **param::io::readflat** (const std::string &fileName, parameters &p)
- void **param::io::write** (const std::string &fileName, const parameters &p)
- void **param::io::writeflat** (const std::string &fileName, const parameters &p)
- int **param::io::updatefromstring** (const std::vector< std::string > &flatjsonKeyVal, parameters &p, const std::string &&sep=std::string(":"))
- void **param::io::defaultconf** ()
- int **param::test::base** ()
- int **param::test::io** ()
- int **param::test::ioflat** ()
- int **param::test::ipartialread** ()
- int **param::test::prepareforuse** ()
- int **param::test::autoflat** ()
- int **param::test::symmetry** ()

### 8.8.1 Detailed Description

Parameters for the integrated 3DRSVS.

## 8.9 incl/postprocessing.hpp File Reference

Provide tecplot file formating for mesh and snake outputs.

```
#include <iostream>
#include <stdarg.h>
#include "arraystructures.hpp"
#include "triangulate.hpp"
```

### Classes

- class tecplotfile

### Functions

- int **Test_tecplotfile** ()
- int **TestCompareReadWrite** (const char ∗fileToOpen, mesh &blockGrid, tecplotfile &outmesh1)

### 8.9.1 Detailed Description

Provide tecplot file formating for mesh and snake outputs.

## 8.10 incl/RSVSalgorithm.hpp File Reference

Functions which are part of the RSVS algorithm but not core to the snaking process.

```
#include <vector>
```

### Functions

- std::vector< int > **FindSpawnVerts** (const mesh &meshin, std::vector< int > &vertList, std::vector< int > &voluOutList, int outerBorder=1)
- void **SpawnRSVS** (snake &snakein, int outerBorder=1)
- void **RemoveSnakeInVolu** (snake &snakein, std::vector< int > &voluInd, int outerBorder)
- void **RemoveSnakeInSurf** (snake &snakein, std::vector< int > &voluInd, int outerBorder)
- void **SpawnSnakeAndMove** (snake &snakein, std::vector< int > vertSpawn)
- int **Test_RSVSalgo_init** ()
- int **Test_RSVSalgo** ()
- int **Test_RSVSalgoflat** ()

### 8.10.1 Detailed Description

Functions which are part of the RSVS algorithm but not core to the snaking process.

## 8.11 incl/RSVScalc.hpp File Reference

Provides the infrastructure for calculation of the RSVS equations.

```
#include <iostream>
#include <fstream>
#include <vector>
#include "vectorarray.hpp"
#include "RSVSmath.hpp"
#include "mesh.hpp"
#include "snake.hpp"
#include "triangulate.hpp"
#include <Eigen>
```

### Classes

- class RSVScalc

  *Class to handle the RSVS calculation.*

## Functions

- void ResizeLagrangianMultiplier (const RSVScalc &calcobj, VectorXd &lagMultAct, bool &isLarge, bool &is←Nan)

  *Resizes the lagrangian multiplier LagMultAct based on whether any of its values are nan or too large.*
- template<class T >
  bool SQPstep (const RSVScalc &calcobj, const MatrixXd &dConstrAct, const RowVectorXd &dObjAct, const VectorXd &constrAct, VectorXd &lagMultAct, VectorXd &deltaDVAct, bool &isNan, bool &isLarge, bool attemptConstrOnly=true)

  *Template for calculation of an SQP step.*

## 8.11.1 Detailed Description

Provides the infrastructure for calculation of the RSVS equations.

## 8.11.2 Function Documentation

### 8.11.2.1 ResizeLagrangianMultiplier()

```
void ResizeLagrangianMultiplier (
             const RSVScalc & calcobj,
             VectorXd & lagMultAct,
             bool & isLarge,
             bool & isNan )
```

Resizes the lagrangian multiplier LagMultAct based on whether any of its values are nan or too large.

This uses the RSVScalc object to guide the resizing operation if it is needed.

**Parameters**

| | | |
|---|---|---|
| in | *calcobj* | The calculation object. |
| in,out | *lagMultAct* | The vector of active lagrangian multipliers. |
| out | *isLarge* | Returns if lagMultAct is too large. |
| out | *isNan* | Returns if lagMultAct has Nan values. |

Definition at line 29 of file RSVScalc_core.cpp.

### 8.11.2.2 SQPstep()

```
template<class T >
bool SQPstep (
             const RSVScalc & calcobj,
             const MatrixXd & dConstrAct,
             const RowVectorXd & dObjAct,
             const VectorXd & constrAct,
             VectorXd & lagMultAct,
             VectorXd & deltaDVAct,
             bool & isNan,
             bool & isLarge,
             bool attemptConstrOnly = true )
```

Template for calculation of an SQP step.

This template cannot be deduced and needs the developer to pass the required solver class when it is called.

Instantiation options: Eigen::HouseholderQR Eigen::ColPivHouseholderQR Eigen::LLT<MatrixXd> (∗) <- needs a full type to be defined (see below) Eigen::PartialPivLU

For stability info https://eigen.tuxfamily.org/dox/group__TutorialLinearAlgebra.↩ html https://eigen.tuxfamily.org/dox/group__DenseDecompositionBenchmark.↩ html

**Parameters**

| | | |
|---|---|---|
| in | *calcobj* | The calculation object |
| in | *dConstrAct* | Active constraint jacobian dh/dx |
| in | *dObjAct* | Active objective jacobian dJ/dx |
| in | *constrAct* | Active constraint vector |
| | *lagMultAct* | The active lagrangian multipliers |
| | *deltaDVAct* | The active SQP step to take |
| out | *isNan* | Indicates if lagMult is nan |
| out | *isLarge* | Indicates if lagMult is large |
| in | *attemptConstrOnly* | Should the step algorithm attempt using only the constraint to step. |

**Template Parameters**

| | |
|---|---|
| *T* | The Eigen object template type to use. A full type will be defined using T<MatrixXd>. |

**Returns**

(isLarge || isNan), if true some form of failure was detected.

This template cannot be deduced and needs the developer to pass the required solver class when it is called.

Instantiation options: Eigen::HouseholderQR<MatrixXd> Eigen::ColPivHouseholderQR<MatrixXd> Eigen::LL↩ T<MatrixXd> Eigen::PartialPivLU<MatrixXd>

For stability info https://eigen.tuxfamily.org/dox/group__TutorialLinearAlgebra.↩ html https://eigen.tuxfamily.org/dox/group__DenseDecompositionBenchmark.↩ html

**Parameters**

| | | |
|---|---|---|
| in | *calcobj* | The calculation object |
| in | *dConstrAct* | Active constraint jacobian dh/dx |

**Parameters**

| | | |
|---|---|---|
| in | *dObjAct* | Active objective jacobian dJ/dx |
| in | *constrAct* | Active constraint vector |
| | *lagMultAct* | The active lagrangian multipliers |
| | *deltaDVAct* | The active SQP step to take |
| out | *isNan* | Indicates if lagMult is nan |
| out | *isLarge* | Indicates if lagMult is large |
| in | *attemptConstrOnly* | Should the step algorithm attempt using only the constraint to step. |

**Template Parameters**

| | |
|---|---|
| *T* | The Eigen object type to use. Should take a RSVScalc::HLag as a constructor and support a solve method. |

**Returns**

(isLarge || isNan), if true some form of failure was detected.

Definition at line 427 of file RSVScalc.hpp.

## 8.12 incl/RSVSclass.hpp File Reference

Simple class containing all the information needed for the 3D-RSVS execution.

```
#include <iostream>
#include <fstream>
#include "mesh.hpp"
#include "snake.hpp"
#include "postprocessing.hpp"
#include "parameters.hpp"
#include "triangulate.hpp"
#include "RSVScalc.hpp"
```

### Classes

- class integrate::RSVSclass

### 8.12.1 Detailed Description

Simple class containing all the information needed for the 3D-RSVS execution.

## 8.13 incl/RSVSintegration.hpp File Reference

Integration into the full 3 dimensional Restricted Snake Volume of Solid method.

```
#include <vector>
#include <fstream>
#include <tuple>
```

## Classes

- class integrate::iteratereturns

## Namespaces

- param

  *Namespace containing the parameter classes used to control execution of the 3D-RSVS program.*

## Functions

- void **SnakeConnectivityUpdate** (snake &testSnake, vector< int > &isImpact, double impactAlmost↩ Range=0.2)
- void **SnakeConnectivityUpdate_2D** (snake &testSnake, vector< int > &isImpact)
- void **SnakeConnectivityUpdate_legacy** (snake &snakein, vector< int > &isImpact)
- void **SnakeConnectivityUpdate_robust** (snake &snakein, vector< int > &isImpact)
- int **TimeStamp** (const char ∗str, int start_s)
- void **integrate::Prepare** (RSVSclass &RSVSobj)
- void **integrate::prepare::Mesh** (const param::grid &gridconf, mesh &snakeMesh, mesh &voluMesh)
- void **integrate::prepare::Snake** (const param::snaking &snakconf, const param::rsvs &rsvsconf, mesh &snakeMesh, mesh &voluMesh, snake &rsvsSnake)
- void **integrate::prepare::Triangulation** (mesh &snakeMesh, snake &rsvsSnake, triangulation &rsvsTri)
- void **integrate::prepare::Output** (const param::parameters &paramconf, const param::parameters &origcong, tecplotfile &outSnake, std::ofstream &logFile, std::ofstream &coutFile, std::ofstream &cerrFile)
- void **integrate::prepare::grid::Voxel** (const param::grid &gridconf, mesh &snakeMesh, mesh &voluMesh)
- void **integrate::prepare::grid::Voronoi** (const param::grid &gridconf, mesh &snakeMesh, mesh &voluMesh)
- void **integrate::prepare::grid::Load** (const param::grid &gridconf, mesh &snakeMesh, mesh &voluMesh)
- void **integrate::execute::All** (integrate::RSVSclass &RSVSobj)
- iteratereturns **integrate::execute::RSVSiterate** (RSVSclass &RSVSobj)
- void **integrate::execute::Logging** (RSVSclass &RSVSobj, double totT, int nVoluZone, int stepNum)
- void **integrate::execute::PostProcessing** (RSVSclass &RSVSobj, double totT, int nVoluZone, int stepNum)
- void **integrate::execute::Exporting** (RSVSclass &RSVSobj)
- void **integrate::execute::logging::Log** (std::ofstream &logFile, RSVScalc &calcObj, int loglvl)
- void **integrate::execute::logging::Snake** (tecplotfile &outSnake, snake &rsvsSnake, mesh &voluMesh, double totT, int nVoluZone)
- void **integrate::execute::logging::FullTecplot** (tecplotfile &outSnake, snake &rsvsSnake, triangulation &rsvsTri, mesh &voluMesh, double totT, int nVoluZone, int stepNum)
- void **integrate::execute::postprocess::Log** (std::ofstream &logFile, RSVScalc &calcObj, int loglvl)
- void **integrate::execute::postprocess::Snake** (snake &rsvsSnake, mesh &voluMesh, param::parameters &paramconf)
- void **integrate::execute::postprocess::FullTecplot** (tecplotfile &outSnake, snake &rsvsSnake, triangulation &rsvsTri, mesh &voluMesh, double totT, int nVoluZone, int stepNum)
- void **integrate::execute::exporting::SU2** (std::string exportStr, snake &rsvsSnake, param::parameters &paramconf)
- int **integrate::test::Prepare** ()
- int **integrate::test::All** ()

### 8.13.1 Detailed Description

Integration into the full 3 dimensional Restricted Snake Volume of Solid method.

## 8.14 incl/rsvsjson.hpp File Reference

Interface between the RSVS project and the `JSON for Modern C++ library`.

```
#include <cstdlib>
#include <array>
#include <string>
#include "json.hpp"
```

### Classes

- struct param::filltype< T >

  *The input type of fill information.*

### Namespaces

- param

  *Namespace containing the parameter classes used to control execution of the 3D-RSVS program.*

### Typedefs

- using **rsvsjson::json** = nlohmann::json

### Functions

- template<class T >
  void **param::to_json** (rsvsjson::json &j, const filltype< T > &p)
- template<class T >
  void **param::from_json** (const rsvsjson::json &j, filltype< T > &p)
- void **param::to_json** (rsvsjson::json &j, const rsvs &p)
- void **param::from_json** (const rsvsjson::json &j, rsvs &p)
- void **param::to_json** (rsvsjson::json &j, const snaking &p)
- void **param::from_json** (const rsvsjson::json &j, snaking &p)
- void **param::to_json** (rsvsjson::json &j, const voxel &p)
- void **param::from_json** (const rsvsjson::json &j, voxel &p)
- void **param::to_json** (rsvsjson::json &j, const voronoi &p)
- void **param::from_json** (const rsvsjson::json &j, voronoi &p)
- void **param::to_json** (rsvsjson::json &j, const grid &p)
- void **param::from_json** (const rsvsjson::json &j, grid &p)
- void **param::to_json** (rsvsjson::json &j, const parameters &p)
- void **param::from_json** (const rsvsjson::json &j, parameters &p)
- void **param::to_json** (rsvsjson::json &j, const ioin &p)
- void **param::from_json** (const rsvsjson::json &j, ioin &p)
- void **param::to_json** (rsvsjson::json &j, const ioout &p)
- void **param::from_json** (const rsvsjson::json &j, ioout &p)
- void **param::to_json** (rsvsjson::json &j, const files &p)
- void **param::from_json** (const rsvsjson::json &j, files &p)
- void **rsvsjson::flatupdate** (json &jfin, json &jnew, bool isFlatFin, bool isFlatNew)
- void **tetgen::to_json** (rsvsjson::json &j, const apiparam &p)
- void **tetgen::from_json** (const rsvsjson::json &j, apiparam &p)

### 8.14.1 Detailed Description

Interface between the RSVS project and the `JSON for Modern C++ library`.

## 8.15 incl/RSVSmath.hpp File Reference

Performs Volume and Area calculations for the RSVS process.

```
#include <vector>
#include <cmath>
#include "vectorarray.hpp"
#include "RSVSmath_automatic.hpp"
```

### Classes

- class TriFunc
- class CoordFunc
- class Volume
- class Area
- class LengthEdge
- class Volume2
- class SurfCentroid

### 8.15.1 Detailed Description

Performs Volume and Area calculations for the RSVS process.

This provides a simple(ish) interface to the "RSVSmath_automatic.hpp" header which is auto generated by matlab's symbolic toolbox.

## 8.16 incl/snake.hpp File Reference

Provides the core restricted surface snake container.

```
#include <iostream>
#include <vector>
#include <cmath>
#include <cfloat>
#include <unordered_map>
#include "arraystructures.hpp"
#include "mesh.hpp"
```

## Classes

- class snaxarray
- class snake
- class snax
- class snaxedge
- class snaxsurf

## Typedefs

- typedef SnakStruct< snaxedge > **snaxedgearray**
- typedef SnakStruct< snaxsurf > **snaxsurfarray**

## Functions

- double **SnaxImpactDt** (const snax &snax1, const snax &snax2)
- bool **IsAproxEqual** (double d1, double d2)
- int **CompareSnakeInternalStatus** (const vector< bool > &thisVec, bool thisFlipped, const vector< bool > &otherVec, bool otherFlipped)
- int **Test_SnakeStructures** ()
- int **Test_coordvec** ()
- int **Test_snax** ()
- int **Test_snaxedge** ()
- int **Test_snake** ()
- int **Test_snakeinit** ()
- int **Test_snakeinit_MC** ()
- int **Test_snakeOrderEdges** ()
- int **Test_snakeinitflat** ()
- void **Test_stepalgo** (snake &testSnake, vector< int > &isImpact)
- void **Test_stepalgo_mergeclean** (snake &testSnake, vector< int > &isImpact)

### 8.16.1 Detailed Description

Provides the core restricted surface snake container.

This container allows efficient and robust geometry and topology evolution.

## 8.17 incl/snakeengine.hpp File Reference

Functions needed to evolve the r-surface snake.

```
#include <iostream>
#include <vector>
#include <unordered_map>
#include "arraystructures.hpp"
#include "snake.hpp"
#include "postprocessing.hpp"
```

## Functions

- void **SpawnAtVertex** (snake &snakein, int indVert)
- void **SpawnAtVertexVert** (snake &newsnake, int nVert, int indVert, int subVert, const vector< int > &surfInds, const vector< int > &edgeInds, const vector< int > &edgeSubs, unordered_multimap< int, int > &hashSurfInds)
- void **SpawnAtVertexEdge** (snake &newsnake, int nEdge, const vector< int > &surfInds, const vector< int > &edgeInds, const vector< int > &voluInds, const vector< int > &surfSubs, unordered_multimap< int, int > &hashEdgeInds, unordered_multimap< int, int > &hashVoluInds)
- void **SpawnAtVertexSurf3D** (snake &newsnake, int nSurf, const vector< int > &surfInds, const vector< int > &voluInds, const vector< int > &voluSubs, unordered_multimap< int, int > &hashSurfInds)
- void **SpawnAtVertexSurf2D** (snake &newsnake, int nEdge, const vector< int > &voluInds)
- void **SpawnAtVertexVolu** (snake &newsnake, int nSurf)
- void **MergeAllContactVertices** (snake &fullsnake, vector< int > &isImpact)
- void **SpawnArrivedSnaxels** (snake &fullsnake, const vector< int > &isImpact)
- void **SpawnArrivedSnaxelsDir** (snake &fullsnake, snake &partSnake, const vector< int > &isImpact, int dir, HashedVector< int, int > &vertNoSpawn)
- void **MergeCleanSnake** (snake &fullsnake, vector< int > &isImpact)
- void **CleanupSnakeConnec** (snake &snakein)
- void **IdentifyMergEdgeSameSurfConnec** (const snake &snakein, vector< ConnecRemv > &connecEdit)
- void **IndentifyEdgeSameSurf** (const snake &snakein, int currSub, int &stepCheck, vector< int > &tempSub, vector< int > &tempSub2, vector< int > &tempSub3, HashedVector< int, int > &tempIndHash, HashedVector< int, int > &edge2Surf, vector< int > tempCount)
- void **IdentifyMergEdgeConnec** (const snake &snakein, vector< ConnecRemv > &connecEdit)
- void **IdentifyMergeEdgeGeneral** (const snake &snakein, vector< bool > &isObjDone, vector< ConnecRemv > &connecEdit, ConnecRemv &tempConnec, ConnecRemv &tempConnec2, vector< int > &tempSub, vector< int > &tempSub2, vector< int > &tempCount, HashedVector< int, int > &tempIndHash)
- void **IdentifyMergeEdgeGeneralChain** (const snake &snakein, vector< bool > &isObjDone, vector< ConnecRemv > &connecEdit, ConnecRemv &tempConnec, ConnecRemv &tempConnec2, vector< int > &tempSub, vector< int > &tempSub2, vector< int > &tempCount, HashedVector< int, int > &tempIndHash, int jjStart)
- void **IdentifyMergSurfConnec** (const snake &snakein, vector< ConnecRemv > &connecEdit)
- void **IdentifyMergeSurfGeneral** (const snake &snakein, vector< bool > &isObjDone, vector< ConnecRemv > &connecEdit, ConnecRemv &tempConnec, vector< int > &tempSub, vector< int > &tempSub2, vector< int > &tempCount, HashedVector< int, int > &edge2Surf, HashedVector< int, int > &tempIndHash)
- void **IdentifyMergeSurfRecursive** (const snake &snakein, vector< bool > &isObjDone, vector< int > &tempCount, const HashedVector< int, int > &edge2Surf, const HashedVector< int, int > &tempIndHash, ConnecRemv &tempConnec, const vector< int > &tempSub, const vector< int > &tempSub2, int excludeSub)
- void **ModifyMergVoluConnec** (snake &snakein, vector< ConnecRemv > &connecEdit, const vector< int > &indRmvVert)
- void **ModifyMergSurf2DConnec** (snake &snakein, vector< ConnecRemv > &connecEdit)
- void **SnaxEdgeConnecDetection** (snake &snakein, vector< ConnecRemv > &connecEdit)
- void **SnaxNoConnecDetection** (const mesh &snakeconn, vector< ConnecRemv > &connecEdit)
- void **dispconnrmv** (vector< ConnecRemv > conn)
- void **CheckSnakeRemovalsVert** (const snake &snakein, const vector< int > &indRmvVert)
- void **CheckSnakeRemovalsEdge** (const snake &snakein, const vector< int > &indRmvEdge)
- void **CheckSnakeRemovalsSurf** (const snake &snakein, const vector< int > &indRmvSurf)
- void **CheckSnakeRemovalsVolu** (const snake &snakein, const vector< int > &indRmvVolu)

### 8.17.1 Detailed Description

Functions needed to evolve the r-surface snake.

## 8.18 incl/snakstruct_incl.cpp File Reference

File for the implementation of the class template SnakStruct this .cpp file is INCLUDED as part of arraystructures.hpp and cannot be compiled on its own.

```
#include "arraystructures.hpp"
```

### 8.18.1 Detailed Description

File for the implementation of the class template SnakStruct this .cpp file is INCLUDED as part of arraystructures.hpp and cannot be compiled on its own.

This file adds the support for a second hashed variable called by KeyParent

## 8.19 incl/test.hpp File Reference

Provides the custom testing system used by the RSVS3D project.

```
#include <iostream>
#include <stdarg.h>
#include <ctime>
#include <fstream>
#include <string>
```

### Classes

- class rsvstest::customtest

  *Class for customtest.*

### Namespaces

- rsvstest

  *Namespace for rsvs tests.*

### Functions

- int **rsvstest::maintest** ()
- int **rsvstest::newtest** ()

### 8.19.1 Detailed Description

Provides the custom testing system used by the RSVS3D project.

## 8.20 incl/tetgenrsvs.hpp File Reference

Interface between the RSVS project and `tetgen`.

```
#include <array>
#include <vector>
#include <string>
#include <algorithm>
#include "tetgen.h"
#include "mesh.hpp"
```

### Classes

- class tetgen::io_safe

  *Class for memory safe interface with tetgen.h.*
- class tetgen::apiparam

### Typedefs

- typedef std::array< std::array< double, 3 >, 2 > tetgen::dombounds

  *Type defining domain boundaries.*

### Functions

- std::vector< int > tetgen::RSVSVoronoiMesh (const std::vector< double > &vecPts, mesh &vosMesh, mesh &snakMesh, tetgen::apiparam &inparam)

  *Genrates Snaking and VOS RSVS meshes from the voronoi diagram of a set of points.*
- void tetgen::SnakeToSU2 (const snake &snakein, const std::string &fileName, tetgen::apiparam &inparam)

  *Genrates an SU2 mesh file from a snake.*
- void **tetgen::input::POINTGRIDS** (const mesh &meshdomain, tetgen::io_safe &tetin, const tetgen::apiparam &tetgenParam, bool generateVoroBound=false)
- void **tetgen::input::RSVSGRIDS** (const mesh &meshdomain, tetgen::io_safe &tetin, const tetgen::apiparam &tetgenParam)
- void **tetgen::input::RSVSGRIDS** (const mesh &meshdomain, const mesh &meshboundary, tetgen::io_safe &tetin, const tetgen::apiparam &tetgenParam)
- void **tetgen::input::RSVS2CFD** (const snake &snakein, tetgen::io_safe &tetin, const tetgen::apiparam &tetgenParam)
- mesh **tetgen::output::VORO2MESH** (tetgen::io_safe &tetout)
- void **tetgen::output::SU2** (const char ∗fileName, const tetgenio &tetout)
- dombounds **tetgen::output::GetBoundBox** (io_safe &tetout)
- mesh tetgen::output::TET2MESH (tetgen::io_safe &tetout)

  *Translates a tetgen output to the RSVS native mesh format.*
- void **tetgen::internal::CloseVoronoiMesh** (mesh &meshout, tetgen::io_safe &tetout, std::vector< int > &rayEdges, int DEINCR, tetgen::dombounds boundBox)
- template<class T , class V >
  double tetgen::internal::ProjectRay (int count, const tetgen::dombounds &boundBox, const T &dir, const V &orig, double minDist=0.0)

  *Project voronoi diagram rays to the bounding Box.*

- void **tetgen::internal::MeshData2Tetgenio** (const mesh &meshgeom, tetgen::io_safe &tetin, int facet↩ Offset, int pointOffset, int pointMarker, const std::vector< double > &pointMtrList, const std::vector< double > &facetConstr, int facetConstrOffset)
- void **tetgen::internal::Mesh2Tetgenio** (const mesh &meshgeom, const mesh &meshdomain, tetgen::io_safe &tetin, int numHoles)
- void **tetgen::internal::Mesh2TetgenioPoints** (const mesh &meshgeom, const mesh &meshdomain, tetgen::io_safe &tetin)
- void **tetgen::internal::PointCurvature2Metric** (std::vector< double > &vertCurvature, const tetgen::apiparam &inparam)
- std::vector< bool > **tetgen::voronoi::Points2VoroAndTetmesh** (const std::vector< double > &vecPts, mesh &voroMesh, mesh &tetMesh, const tetgen::apiparam &inparam)
- std::vector< bool > **tetgen::voronoi::BoundaryFacesFromPoints** (const mesh &meshin, const std↩ ::vector< int > &boundaryPts)
- void **tetgen::test::LoadData** (mesh &snakeMesh, mesh &voluMesh, snake &snakein, mesh &triMesh)
- int **tetgen::test::api** ()
- int **tetgen::test::call** ()
- int **tetgen::test::CFD** ()
- int **tetgen::test::RSVSVORO** ()
- int **tetgen::test::RSVSVORO_Contain** ()
- int **tetgen::test::RSVSVOROFunc** (int nPts=0, double distanceTol=0.26, const char ∗tecoutStr="../TEST↩ OUT/rsvs_voro.plt")
- int **tetgen::test::RSVSVOROFunc_contain** (int nPts=0, double distanceTol=0.26, const char ∗tecout↩ Str="../TESTOUT/rsvs_voro_contain.plt")
- int **Test_RSVSvoro_init** ()

## 8.20.1 Detailed Description

Interface between the RSVS project and `tetgen`.

## 8.20.2 Typedef Documentation

### 8.20.2.1 dombounds

```
typedef std::array<std::array<double, 3>, 2> tetgen::dombounds
```

Type defining domain boundaries.

Simple short hand for a matrix of 2∗3 doubles.

Definition at line 41 of file tetgenrsvs.hpp.

## 8.20.3 Function Documentation

#### 8.20.3.1 ProjectRay()

```
template<class T , class V >
double tetgen::internal::ProjectRay (
            int count,
            const tetgen::dombounds & boundBox,
            const T & dir,
            const V & orig,
            double minDist = 0.0 )
```

Project voronoi diagram rays to the bounding Box.

**Parameters**

|  |  |  |
|---|---|---|
| in | *count* | number of coordinates |
| in | *boundBox* | The bounds of the domain (array<array<double,3>,2>) |
| in | *dir* | vector with direction (pointing in) |
| in | *orig* | The origin of the ray |
| in | *minDist* | The minimum allowable stretch distance |

**Template Parameters**

|  |  |
|---|---|
| *T* | type of `dir`: an iterable of size `count` |
| *V* | type of `orig`: an iterable of size `count` |

**Returns**

Distance along the ray at which the boundBox is encountered.

Definition at line 258 of file tetgenrsvs.hpp.

#### 8.20.3.2 RSVSVoronoiMesh()

```
std::vector< int > tetgen::RSVSVoronoiMesh (
            const std::vector< double > & vecPts,
            mesh & vosMesh,
            mesh & snakMesh,
            tetgen::apiparam & inparam )
```

Genrates Snaking and VOS RSVS meshes from the voronoi diagram of a set of points.

**Parameters**

|  |  |  |
|---|---|---|
| in | *vecPts* | a vector of input points (3 coordinate) followed by a target volume fraction. Vecpts is a 1D vector with 4 values per point. |
|  | *vosMesh* | The vos mesh |
|  | *snakMesh* | The snaking mesh |
|  | *inparam* | The tetgen interface parameter at input. |

**Returns**

Returns the mapping of the original points to the snake mesh volumes.

Definition at line 837 of file tetgenrsvs.cpp.

#### 8.20.3.3 SnakeToSU2()

```
void tetgen::SnakeToSU2 (
            const snake & snakein,
            const std::string & fileName,
            tetgen::apiparam & inparam )
```

Genrates an SU2 mesh file from a snake.

Uses tetgen to generate a volume mesh around a snake and outputs it to the SU2 format.

**Parameters**

| | | |
|---|---|---|
| in | *snakein* | A snake which needs to be meshed |
| in | *fileName* | The file name |
| | *inparam* | tetgen interface parameter object. Used to define boundary growth rate and element sizes. |

Definition at line 968 of file tetgenrsvs.cpp.

#### 8.20.3.4 TET2MESH()

```
mesh tetgen::output::TET2MESH (
            tetgen::io_safe & tetout )
```

Translates a tetgen output to the RSVS native mesh format.

**Parameters**

| | |
|---|---|
| *tetout* | the tetgenio object containing a mesh to be translated to the native RSVS mesh format. |

**Returns**

mesh object containing the translated grid.

@raises invalid_argument if tetout was generated without passing the neighbour flag to tetgen (`-nn`)

Definition at line 994 of file tetgenrsvs.cpp.

## 8.21 incl/triangulate.hpp File Reference

Provides a triangulation for snake and meshes.

```
#include <vector>
#include "arraystructures.hpp"
#include "snake.hpp"
#include "mesh.hpp"
```

## Classes

- class TriStruct< T >
- class TriStruct< T >
- class triangulation
- class tri2mesh
- class triangle
- class trianglepoint
- class trianglesurf

## Typedefs

- typedef TriStruct< triangle > **triarray**
- typedef TriStruct< trianglepoint > **tripointarray**
- typedef TriStruct< trianglesurf > **trisurfarray**

## Functions

- void **CalculateSnakeVel** (snake &snakein)
- void **CalculateSnakeVelRand** (snake &snakein)
- void **CalculateSnakeVelUnit** (snake &snakein)
- void **CalculateSnakeVelFast** (snake &snakein)
- void **CalculateNoNanSnakeVel** (snake &snakein, double deltaStep=0.01)
- void **TriangulateSurface** (const surf &surfin, const mesh &meshin, triarray &triangul, tripointarray &trivert, const int typeMesh, int trivertMaxInd)
- void **TriangulateTriSurface** (const trianglesurf &surfin, triarray &triangul, tripointarray &trivert, const int typeMesh, int trivertMaxInd)
- void **TriangulateContainer** (const mesh &meshin, triangulation &triangleRSVS, const int typeMesh, const vector< int > &subList={})
- void **TriangulateSnake** (snake &snakein, triangulation &triangleRSVS)
- void **TriangulateMesh** (mesh &meshin, triangulation &triangleRSVS)
- void **MeshTriangulation** (mesh &meshout, const mesh &meshin, triarray &triangul, tripointarray &trivert)
- void **MaintainTriangulateSnake** (triangulation &triangleRSVS)
- void **SnakeSurfaceCentroid_fun** (coordvec &coord, const surf &surfin, const mesh &meshin)
- void **HybridSurfaceCentroid_fun** (coordvec &coord, const trianglesurf &surfin, const mesh &meshin, const mesh &snakeconn)
- void **Test_stepalgoRSVS** (snake &testSnake, triangulation &RSVStri, vector< double > &dt, vector< int > &isImpact, RSVScalc &calcObj, tecplotfile &outSnake2, double totT)
- void **BuildTriSurfGridSnakeIntersect** (triangulation &triangleRSVS)
- int **FollowVertexConnection** (int actVert, int prevEdge, const HashedVector< int, int > &edgeSurfInd, const HashedVector< int, int > &vertSurfInd, const snake &snakeRSVS, const mesh &meshRSVS, int &returnIndex, int &returnType, int &nextEdge)
- int **FollowSnaxelDirection** (int actSnax, const snake &snakeRSVS, int &returnIndex, int &returnType, int &actEdge)
- bool **FollowSnaxEdgeConnection** (int actSnax, int actSurf, int followSnaxEdge, const snake &snakeRSVS, vector< bool > &isSnaxEdgeDone, int &returnIndex)
- mesh **TriarrayToMesh** (const triangulation &triangul, const triarray &triin)
- void **FlattenBoundaryFaces** (mesh &meshin)
- int **Test_snakeRSVS** ()
- int **Test_surfcentre** ()
- int **Test_snakeRSVS_singlevol** ()
- int **Test_RSVSalgo_singlevol** ()
- int **Test_MeshOrient** ()

### 8.21.1 Detailed Description

Provides a triangulation for snake and meshes.

This links an active snake and mesh to their triangulated representation needed to compute areas and volumes.

## 8.22 incl/vectorarray.hpp File Reference

Provides a 2D std::vector based container.

```
#include <iostream>
#include <vector>
#include "warning.hpp"
#include "vectorarray_incl.cpp"
```

### Classes

- class ArrayVec< T >

  *Template class for vector of vectors (matrix).*

### 8.22.1 Detailed Description

Provides a 2D std::vector based container.

## 8.23 incl/vectorarray_incl.cpp File Reference

File for the implementation of the class template vectorarray this .cpp file is INCLUDED as part of vectorarray.hpp and cannot be compiled on its own.

```
#include "vectorarray.hpp"
```

### 8.23.1 Detailed Description

File for the implementation of the class template vectorarray this .cpp file is INCLUDED as part of vectorarray.hpp and cannot be compiled on its own.

## 8.24 incl/voxel.hpp File Reference

Generation of cartesian grids.

```
#include <iostream>
#include <numeric>
#include <Eigen>
#include "arraystructures.hpp"
#include "postprocessing.hpp"
```

## Functions

- template<class T >
  T cumsum (const T &matIn, int d)

  *template which applies cumulative sum to Eigen Matrix.*
- template<class T >
  T cumprod (const T &matIn, int d)

  *template which applies cumulative product to Eigen Matrix.*
- int **BuildBlockGrid** (std::array< int, 3 > &dimGrid, mesh &blockGrid)
- int **BuildBlockGrid** (RowVector3i dimGrid, mesh &blockGrid)
- int **BuildBlockVert** (RowVector3i dimGrid, mesh &blockGrid, int nVert, Matrix3i edgeProp, RowVector3i n↩ EdgeDim)
- int **BuildBlockEdge** (RowVector3i dimGrid, mesh &blockGrid, int nEdge, RowVector3i nEdgeDim, Row↩ Vector3i nSurfDim, Matrix3i edgeProp, Matrix3i surfProp)
- int **BuildBlockSurf** (RowVector3i dimGrid, int nSurf, mesh &blockGrid, Matrix3i surfProp, Matrix3i edgeProp, RowVector3i nSurfDim, RowVector3i nEdgeDim)
- int **BuildBlockVolu** (RowVector3i dimGrid, int nVolu, mesh &blockGrid, RowVector3i nSurfDim, Matrix3i surfProp)
- int **Test_BuildBlockGrid_noout** ()
- int **Test_MeshOut** ()

### 8.24.1 Detailed Description

Generation of cartesian grids.

### 8.24.2 Function Documentation

#### 8.24.2.1 cumprod()

```
template<class T >
T cumprod (
            const T & matIn,
            int d )
```

template which applies cumulative product to Eigen Matrix.

Cumprod is applied row-wise for d=0 and col-wise for d=1

**Parameters**

| | | |
|---|---|---|
| in | *mat↩ In* | The matrix input |
| in | *d* | dimension to use 0-row wise, 1 col-wise |

**Template Parameters**

| | |
|---|---|
| *T* | Eigen type |

**Returns**

The cumulative product.

Definition at line 98 of file voxel.hpp.

#### 8.24.2.2 cumsum()

```
template<class T >
T cumsum (
            const T & matIn,
            int d )
```

template which applies cumulative sum to Eigen Matrix.

Cumprod is applied row-wise for d=0 and col-wise for d=1

**Parameters**

| | | |
|---|---|---|
| in | *mat↩<br>In* | The matrix input |
| in | *d* | dimension to use 0-row wise, 1 col-wise |

**Template Parameters**

| | |
|---|---|
| *T* | Eigen type |

**Returns**

The cumulative sum.

Definition at line 67 of file voxel.hpp.

## 8.25 incl/warning.hpp File Reference

Provides the error and warning system used by the RSVS3D project.

```
#include <iostream>
#include <stdarg.h>
#include <stdexcept>
#include <fstream>
```

### Classes

- class rsvs3d::rsvs_exception

  *Exception for signaling rsvs errors.*

## Namespaces

- rsvs3d

  *Namespace for general purpose tools of the RSVS project.*

## Macros

- #define RSVS3D_ERROR(M) (rsvs3d::error(M, __PRETTY_FUNCTION__, __FILE__, __LINE__, true))

  *Throw generic rsvs errors.*
- #define RSVS3D_ERROR_NOTHROW(M) (rsvs3d::error(M, __PRETTY_FUNCTION__, __FILE__, __LINE__, false))

  *Generic rsvs warning.*
- #define RSVS3D_ERROR_TYPE(M, T) (rsvs3d::error<T>(M, __PRETTY_FUNCTION__, __FILE__, __LINE__, true))

  *Throw a specific error type.*
- #define RSVS3D_ERROR_LOGIC(M) (rsvs3d::error<std::logic_error>(M, __PRETTY_FUNCTION__, __FILE__, __LINE__, true))

  *Throw a logic_error.*
- #define RSVS3D_ERROR_ARGUMENT(M) (rsvs3d::error<std::invalid_argument>(M, __PRETTY_FUNCTION__, __FILE__, __LINE__, true))

  *Throw a invalid_argument.*
- #define RSVS3D_ERROR_RANGE(M) (rsvs3d::error<std::range_error>(M, __PRETTY_FUNCTION__, __FILE__, __LINE__, true))

  *Throw a range_error.*

## Functions

- template<class E = rsvs_exception>
  void rsvs3d::error (const char ∗message="", const char ∗caller="", const char ∗file="", int line=0, bool throwError=true)

  *Custom error function.*
- void **ThrowWarning** (const char ∗message)
- template<class T >
  void CheckFStream (const T &file, const char ∗callerID, const std::string &fileName)

  *Checks a file stream.*

## 8.25.1 Detailed Description

Provides the error and warning system used by the RSVS3D project.

## 8.25.2 Macro Definition Documentation

### 8.25.2.1 RSVS3D_ERROR

```
#define RSVS3D_ERROR(
              M ) (rsvs3d::error(M, __PRETTY_FUNCTION__, __FILE__, __LINE__, true))
```

Throw generic rsvs errors.

**Parameters**

| | |
|---|---|
| *M* | Message of the error (const char∗). |

**Exceptions**

| | |
|---|---|
| *rsvs3d::rsvs_exception* | |

Definition at line 85 of file warning.hpp.

#### 8.25.2.2 RSVS3D_ERROR_ARGUMENT

```
#define RSVS3D_ERROR_ARGUMENT(
              M ) (rsvs3d::error<std::invalid_argument>(M, __PRETTY_FUNCTION__, __FILE__, __←
LINE__, true))
```

Throw a invalid_argument.

**Parameters**

| | |
|---|---|
| *M* | Message of the error (const char∗). |

**Exceptions**

| | |
|---|---|
| *std::invalid_argument* | |

Definition at line 120 of file warning.hpp.

#### 8.25.2.3 RSVS3D_ERROR_LOGIC

```
#define RSVS3D_ERROR_LOGIC(
              M ) (rsvs3d::error<std::logic_error>(M, __PRETTY_FUNCTION__, __FILE__, __LINE_←
_, true))
```

Throw a logic_error.

**Parameters**

| | |
|---|---|
| *M* | Message of the error (const char∗). |

**Exceptions**

| | |
|---|---|
| *std::logic_error* | |

Definition at line 111 of file warning.hpp.

#### 8.25.2.4 RSVS3D_ERROR_NOTHROW

```
#define RSVS3D_ERROR_NOTHROW(
              M ) (rsvs3d::error(M, __PRETTY_FUNCTION__, __FILE__, __LINE__, false))
```

Generic rsvs warning.

**Parameters**

|  |  |
|---|---|
| *M* | Message of the warning (const char∗). |

Definition at line 92 of file warning.hpp.

#### 8.25.2.5 RSVS3D_ERROR_RANGE

```
#define RSVS3D_ERROR_RANGE(
              M ) (rsvs3d::error<std::range_error>(M, __PRETTY_FUNCTION__, __FILE__, __LINE_↩
_, true))
```

Throw a range_error.

**Parameters**

|  |  |
|---|---|
| *M* | Message of the error (const char∗). |

**Exceptions**

|  |  |
|---|---|
| *std::range_error* |  |

Definition at line 129 of file warning.hpp.

#### 8.25.2.6 RSVS3D_ERROR_TYPE

```
#define RSVS3D_ERROR_TYPE(
              M,
              T ) (rsvs3d::error<T>(M, __PRETTY_FUNCTION__, __FILE__, __LINE__, true))
```

Throw a specific error type.

**Parameters**

| | |
|---|---|
| *M* | Message of the warning (const char∗). |

**Template Parameters**

| | |
|---|---|
| *T* | Type of the exception to throw. |

**Exceptions**

| | |
|---|---|
| *T* | |

Definition at line 102 of file warning.hpp.

### 8.25.3 Function Documentation

#### 8.25.3.1 CheckFStream()

```
template<class T >
void CheckFStream (
            const T & file,
            const char * callerID,
            const std::string & fileName )
```

Checks a file stream.

**Parameters**

| | | |
|---|---|---|
| `in` | *file* | input or output file stream |
| `in` | *callerID* | the name of the caller function as given by pretty function |
| `in` | *fileName* | The name of the file opened in the stream. |

**Template Parameters**

| | |
|---|---|
| *T* | either ifstream or ofstream, needs to support method `T::is_open()` |

Definition at line 144 of file warning.hpp.

# Index