Delaunay Triangulation

Generated by Doxygen 1.8.5

Wed Nov 25 2015 16:50:57

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 Delaunay3d Class Reference

Public Member Functions

- **Delaunay3d** (Matrix< double > *_points, int num_points)
- **Delaunay3d** (const Delaunay3d &other)
- Delaunay3d & **operator=** (const Delaunay3d &other)
- void **setup** (Matrix< double > *_points, int num_points)
- double **l2norm** (const vector< double > &a)
- double **dot** (const vector< double > &a, const vector< double > &b)
- void compute_jacobian (Tet &elem)
- void cross_product3 (vector< double > &c, const vector< double > &a, const vector< double > &b)
- void **compute_circumsphere** (Tet &elem)
- double det4 (int ielem, int i, const vector< double > &r) const
- int **find_containing_tet** (const vector< double > &r, int startelement) const
- int check_face_tet (const Tet &elem, const Face &face) const
- void compute_circumcircle (Tet &elem)
- void bowyer_watson ()
- void clear ()

  Reset the Delaunay3d object, except for input data.
- void **writeGmsh2** (string mfile)
- Walkdata **find_containing_tet_and_barycentric_coords** (const vector< double > &rr, int startelement) const
- void **compute_jacobians** ()
- bool **detect_negative_jacobians** ()

Public Attributes

- Matrix< double > **points**
- std::vector< Point > nodes

  List of nodes in the Delaunay graph.
- std::vector< Tet > elems

  List of all elements (tetrahedra) in the Delaunay graph.
- std::vector< int > badelems

  Collection of 'bad elements', that are to be removed while adding a point; its are integers that index members index elems.
- std::vector< Face > faces

  List of all faces in the Delaunay graph.

- std::vector< int > voidpoly

  Collection of faces that bounds the void obtained after removing bad elements while adding a point; its members are integers that index faces.
- Matrix< double > **jacobians**
- int **npoints**

2.1.1 Member Function Documentation

2.1.1.1 void Delaunay3d::bowyer_watson ( )

Computes the Delaunay triangulation (tetrahedralization, in this case). Make sure 'points' has space for three more points when passing to this sub. 'N' is the actual number of real points. First, find the element containing the new point

Second, search among neighbors for other triangles whose circumcircles contain this point

Third, store the faces that will be obtained after removal of bad triangles

Delete faces which are between two bad elements. NOTE: This is one place that is ineffecient because of use of array stacks (std::vectors) as it needs deletion of arbitrary members.

Fourth, delete bad elements. This is another place where array stacks (vectors) of elems, badelems etc make the program slower.

Fifth, add new elements; these are formed by the faces in voidpoly and the new point. Also correspondingly update 'faces'.

2.1.1.2 int Delaunay3d::check_face_tet (const Tet & *elem,* const Face & *face*) const [inline]

Returns the local face number (number of node opposite to the face) of elem's face that is the same as the second argument. If no faces of elem match, it returns -1.

2.1.1.3 void Delaunay3d::compute_circumcircle (Tet & *elem*)

Computes circumcentre and circumradius of a tetrahedron. NOTE: The tetrahedron's jacobian (2∗volume) should be stored in [elem.D](&ref D) beforehand. The center and radius of the circumsphere are calculated as follows. The circumcenter is

$$\mathbf{O} = \frac{|\mathbf{a}^2(\mathbf{b}\times\mathbf{c}) + \mathbf{b}^2(\mathbf{c}\times\mathbf{a}) + \mathbf{c}^2(\mathbf{a}\times\mathbf{b})|}{12V}$$

and the radius $R = |\mathbf{O}|$.

2.1.1.4 void Delaunay3d::compute_jacobian (Tet & *elem*)

Computes 6∗volume of any tetrahedron.

2.1.1.5 void Delaunay3d::cross_product3 (vector< double > & *c,* const vector< double > & *a,* const vector< double > & *b*) [inline]

Computes cross product c of two 3-vectors a and b.

2.1.1.6 double Delaunay3d::det4 (int *ielem,* int *i,* const vector< double > & *r*) const

Calculates the jacobian of the tetrahedron formed by point r and a face of tetrahedron ielem. The face is selected by i between 0 and 3. Face i is the face opposite to local node i of the tetrahedron.

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

- /home/aditya/Myprojects/amovemesh/include/abowyerwatson3d.hpp

2.2 Face Struct Reference

`#include <abowyerwatson3d.hpp>`

Public Attributes

- int **p** [3]
- int **elem** [2]

2.2.1 Detailed Description

Structure for triangular face.

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

- /home/aditya/Myprojects/amovemesh/include/abowyerwatson3d.hpp

2.3 Tet Class Reference

`#include <abowyerwatson3d.hpp>`

Public Member Functions

- **Tet** (const Tet &other)

Public Attributes

- int p [4]

  Indices of vertices.
- Point centre

  Coords of circumcenter of the tet.
- int surr [4]

  Indices of surrounding tets. Note that the neighbor corresponding to surr[3] is opposite the vertex p[3].
- double D

  6∗volume of tet.
- double radius

  Radius of circumcircle of tet.

2.3.1 Detailed Description

Class representing a tetrahedron.

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

- /home/aditya/Myprojects/amovemesh/include/abowyerwatson3d.hpp

2.4 Walkdata Struct Reference

`#include <abowyerwatson3d.hpp>`

Public Attributes

- int **elem**
- double **areacoords** [4]

2.4.1 Detailed Description

Intended to encapsulate data required by 'walk-through' algorothms. Not needed for mesh generation, but in independent application of the walk-through subroutine find_containing_triangle_and_area_coords().

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

- /home/aditya/Myprojects/amovemesh/include/abowyerwatson3d.hpp

Index