

Geopack++ Meshing Operations

Barry Joe

Zhou Computing Services Inc.

Partial address: Abbotsford, BC, Canada

E-mail: bjoe@shaw.ca

Web site: members.shaw.ca/bjoe

Technical Report ZCS2010-01

May 2010

Abstract

Geopack++ is a mesh generation package for performing various meshing operations on 2-D regions and meshes, 3-D regions and meshes, and surface meshes. The operations include generating a mesh given a region, generating a 3-D mesh given a surface mesh, improving a given mesh, and refining a given mesh. Currently, only unstructured isotropic mesh generation is available, but the meshes may be linear or quadratic. The description and usage of the meshing operations are provided in this document.

1 Introduction

Geopack++ is the object-oriented C++ successor of Geopack90 (written in standard Fortran 90 and first released in June 1999), where Geopack90 is the substantially enhanced successor of the original Fortran 77 mathematical software package GEOMPACK [Joe91c] for the GEneration Of two- and three-dimensional finite element Meshes using efficient GEOMETric algorithms. Due to the efficient use of classes and data structures, Geopack++ runs 1.5 to 2 times faster than Geopack90 on Windows systems. Geopack++ also does more error and consistency checking on region, mesh, and curve/surface file data, is less sensitive to tolerance problems in floating point arithmetic, and has some algorithmic enhancements.

The recent enhancements include

- (a) a new meshing operation for subdividing the edges of a 3-D region into mesh edges (with update to region),
- (b) a new meshing operation for generating a surface mesh using the advancing front approach, starting from mesh edges produced on 3-D region edges (where the user has the option of providing these mesh edges),
- (c) a new meshing operation for generating a tetrahedral mesh in a 3-D region, where a surface mesh is first generated using the advancing front approach from mesh edges produced on 3-D region edges,
- (d) various minor changes, some in response to data files received from Geopack++ users.

Future planned enhancements include

- (a) a new meshing operation for generating a hexahedral-dominant mesh in a 3-D region, where a surface mesh is first generated using the sweep and advancing front approaches from mesh edges produced on 3-D region edges, and then subregions are meshed using sweep, advancing front, and possibly other approaches (this operation would include a combination of approaches used in current meshing operations),

- (b) improvements to the hexahedral-dominant meshing operations such as producing fewer pyramids and tetrahedra and better quality elements in the interior of the region (these are difficult problems).

For noncommercial and evaluation purposes, Geompack++ is provided as 32-bit Windows and Linux executables, that can be considered to be a black-box component that interfaces with applications via special file formats. The subroutines in a typical mathematical software package are replaced by files that specify the meshing operations. The subroutine parameter lists are replaced by operation-dependent meshing fields and input data plus region, mesh, and curve/surface files (so the data structures used for representing regions and meshes are hidden from the user). For commercial purposes, Geompack++ is available in other forms in which the data in the input and output files can be provided in an alternative equivalent way through a simple interface class (see file `geompack.h` in the Geompack++ zip file containing documents and samples).

The Geompack++ executable (either `zgpymm.exe` for Windows or `zgpymm_linux` for Linux) can be invoked as an operating system command in a Windows Command Prompt window or Linux Command Line Terminal window or using the standard library function `system` in a C or C++ routine. For easier usage, the provided Geompack++ executable can be copied into a directory that has been added to the search path for executable files, and renamed as `zgp.exe` in Windows or `zgp` in Linux. With this renaming, the `zgp` command has two arguments, an input file name followed by an output file name. For example, using the sample files `oper.in` and `err.out`, the full command is

```
zgp oper.in err.out
```

The input file is a character (ASCII) file containing the meshing operation code, operation-dependent meshing fields, possibly a nonscalar sequence of operation-dependent numerical data, and the names of region, mesh, or curve/surface files. The output file is a character (ASCII) file containing a one-line error message if there is an error; it is empty if there is no error. The general format of the input file is given in Section 2, with specific meshing operations described in Sections 3 to 5. Region, mesh, and curve/surface files are described in the document “Geompack++ File Formats for Regions and Meshes” [Joe08a]. Each error message begins with `Enn` where `nn` is between 01 and 11. The types of error messages are as follows:

- E01 – memory allocation error (e.g. no more space)
- E02 – error from opening file (e.g. nonexistent input file)
- E03 – error from reading file (e.g. end of file or invalid data)
- E04 – invalid meshing field value (e.g. positive number required)
- E05 – invalid range for input data (e.g. negative number or invalid size)
- E06 – inconsistent entity labels or types (e.g. for edges or faces)
- E07 – geometric error in input data (e.g. too few noncollinear vertices)
- E08 – geometric error during processing (e.g. due to input or roundoff error)
- E09 – cannot continue processing due to unsplitable entity (e.g. face)
- E10 – error due to tolerance (try increasing or decreasing relative tolerance)
- E11 – internal error (algorithm limitation or bug in procedure)

2 Input file format

```
opcode reg.in.type cs.in.type mesh.in.type reg.out.type cs.out.type mesh.out.type
reg.in.file_name
cs.in.file_name
mesh.in.file_name
reg.out.file_name
cs.out.file_name
mesh.out.file_name
```

toler dbglvl msglvl
operation_dependent_meshing_fields
operation_dependent_nonscalar_sequence_of_numerical_data

The first record of the input file contains 7 integer fields. The first field *opcode* is the meshing operation code, which specifies the meshing operation to be performed (see below). The other 6 fields specify the type of the region input file, curve/surface input file, mesh input file, region output file, curve/surface output file, and mesh output file, respectively. The type is ≤ 0 if the file is not needed (each meshing operation only needs a subset of the 6 files), 1 if the file is in character (ASCII) form, or ≥ 2 if the file is in binary form.

The next 6 records contain the character file name of the region input file, curve/surface input file, mesh input file, region output file, curve/surface output file, and mesh output file, respectively. File names may have blanks or spaces in the name, but each file name must be put on a separate line. The file name record is omitted if the corresponding type is ≤ 0 .

The eighth record contains the real number field *toler* and two integer fields that can be arbitrarily set (e.g. to 0). *toler* is the relative tolerance for comparison with the number zero. (Because of roundoff errors in floating point arithmetic, relative tolerance tests are used to decide if points are collinear, coplanar, etc.) The actual relative tolerance used is the maximum of *toler* and 100 times machine epsilon (the maximum *toler* value allowed is 0.001). *toler* should be set based on the number of correct significant digits in the input data (it is recommended that vertex and control point coordinates be stored to double precision accuracy if possible). For data with accuracy that is at least in the middle between single and double precision, it is recommended that *toler* be set to about 10^{-9} . If a tolerance error occurs, then secondary recommendations for *toler* are 10^{-10} , 10^{-8} , 10^{-11} , 10^{-7} .

The ninth record, which may be split into two or more lines, contains scalar meshing fields that are dependent on the particular meshing operation. For the refinement operations, the input file ends with one or more records, which may be split into two or more lines, where these extra records contain a nonscalar sequence of numerical data. The sequence consists of either a compact list of element indices specifying the elements to locally refine or some point, line segment, and distance data determining the elements to locally refine over one or more passes. The scalar meshing fields and nonscalar sequences of numerical data are described in the next 3 sections.

The meshing operations and their operation code are as follows (references are given to the parts that are described in published papers or technical reports; some new algorithms and results used in Geompack++ have not been written up in technical reports yet).

- 200 – Check 2-D region and curve files for partial correctness, make minor changes if needed and possible, and if OK, write out (updated) files in character or binary form.
- 201 – Decompose 2-D region into simple and (nearly) convex parts [JoS86], after first approximating any NURBS curves by piecewise linear segments. The region may have isolated vertices and edges.
- 202 – Generate 2-D (triangular or quadrilateral or quadrilateral-dominant) mesh given 2-D region [JoS86, Joe86, Joe95b], with isolated vertices and edges allowed. Any NURBS curves are first approximated by piecewise linear segments and the region is then decomposed into convex parts before being meshed. Any mesh vertices on piecewise linear approximation of curves are projected to curves after the initial mesh is generated. An all-quadrilateral mesh can be generated in any region in which all edges are splittable.
- 203 – Construct 2-D constrained Delaunay triangulation of 2-D region or planar straight-line graph, with isolated vertices and edges allowed. Any NURBS curves are first approximated by piecewise linear segments. Optionally insert new vertices on edges and in interior of subregions based on local feature size. If there is no embedded region (i.e. no outer boundary or hole loops), triangulation is in convex hull of specified vertices, with a region code of 0 used for all elements.

- 204 – Improve quality of 2-D mesh by smoothing and flipping [Joe08b, Joe08c], after possible conversion to order or element type of mesh (among linear triangle, quadratic triangle, linear quadrilateral, quadratic quadrilateral). A triangular mesh can only be converted to a quadrilateral-dominant mesh in which a few triangles may remain.
- 205 – Locally refine 2-D mesh by bisecting or trisecting edges of specified elements (chosen either explicitly or by distance to specified points and line segments) and possibly other elements to maintain a conforming mesh [Joe08d].
- 206 – Subdivide 2-D region edges into mesh edges, with update to region. Isolated vertices and edges are allowed in region. For an all-quadrilateral mesh, an even number of mesh edges is generated on each subregion boundary (provided unsplitable region edges do not prevent this).
- 207 – Generate 2-D mesh given 2-D region, using advancing front approach. Isolated vertices and edges are allowed in region. The user has the option of specifying that no region edges are to be split, i.e. the region edges are already split into mesh edges using operation 206 (possibly with subsequent changes) or a different method. If region edges are allowed to be split, then the approach of operation 206 is used to subdivide region edges into mesh edges. If there are unsplitable region edges, then an all-quadrilateral mesh may not be possible. An all-quadrilateral mesh can be generated if there is an even number of mesh edges on each subregion boundary. The quality of the generated mesh depends on the sizes of the mesh edges as well as their variation.
- 250 – Generate (triangular or quadrilateral or quadrilateral-dominant) surface mesh given 3-D region, with isolated faces allowed (any isolated vertices or edges are ignored). Any NURBS curves and surfaces are first approximated by piecewise linear segments and planar subfaces, and the faces are then decomposed into convex parts before being meshed based on local feature size. Any mesh vertices on polygonal approximation of curves and surfaces are projected to curves and surfaces after the initial mesh is generated. An all-quadrilateral mesh can be generated in any region in which all edges are splittable. The surface elements satisfy the orientation properties given below in operations 305, 306, 351, and 352.
- 251 – Improve quality of surface mesh by smoothing and flipping [Joe08b, Joe08c], after possible conversion to order or element type of mesh (among linear triangle, quadratic triangle, linear quadrilateral, quadratic quadrilateral). A triangular mesh can only be converted to a quadrilateral-dominant mesh in which a few triangles may remain.
- 252 – Locally refine surface mesh by bisecting or trisecting edges of specified elements (chosen either explicitly or by distance to specified points and line segments) and possibly other elements to maintain a conforming mesh [Joe08d].
- 253 – Subdivide 3-D region edges into mesh edges, with update to region. Isolated vertices, edges, and faces are allowed in region. Some nonplanar faces may be subdivided into nearly planar subfaces. For an all-quadrilateral mesh, an even number of mesh edges is generated on boundary of each face (provided unsplitable region edges do not prevent this).
- 254 – Generate surface mesh given 3-D region, using advancing front approach, with isolated faces allowed (any isolated vertices or edges are ignored). The user has the option of specifying that no region edges are to be split, i.e. the region edges are already split into mesh edges using operation 253 (possibly with subsequent changes) or a different method. If region edges are allowed to be split, then any NURBS curves and surfaces are first approximated by piecewise linear segments and nearly planar subfaces, and then the approach of operation 253 is used to subdivide region edges into mesh edges. If region edges are not allowed to be split, then some nonplanar faces may be subdivided into nearly planar subfaces, and any new interior surface edges are subdivided into mesh edges. If there are unsplitable region edges, then an all-quadrilateral mesh may not be possible. An all-quadrilateral mesh can be

generated if there is an even number of mesh edges on boundary of each face. The quality of the generated mesh depends on the sizes of the mesh edges as well as their variation. The surface elements satisfy the orientation properties given below in operations 305, 306, 351, and 352.

- 300 – Check 3-D region and curve/surface files for partial correctness, make minor changes if needed and possible, and if OK, write out (updated) files in character or binary form.
- 301 – Decompose 3-D region into simple and (nearly) convex parts [Joe94], after first approximating any NURBS curves and surfaces by piecewise linear segments and planar subfaces. The region must have no isolated vertices, edges, or faces.
- 302 – Generate tetrahedral mesh given 3-D region, with isolated vertices, edges, and faces allowed. Any NURBS curves and surfaces are first approximated by piecewise linear segments and planar subfaces, and the faces are then decomposed into convex parts before being meshed based on local feature size. Any mesh vertices on polygonal approximation of curves and surfaces are projected to curves and surfaces after the initial surface mesh is generated. Subregions are meshed using 3-D constrained triangulation [Joe08e] and extra interior vertices are inserted based on local feature size.
- 303 – Construct 3-D Delaunay or improved-quality triangulation given set of vertices [Joe89, Joe91a, Joe95a]. The region must only consist of isolated vertex lists, and the triangulation is in the convex hull of specified vertices or in a bounding polyhedron containing the vertices in its interior.
- 304 – Construct 3-D constrained triangulation of 3-D region, with isolated vertices, edges, and faces allowed [Joe08e]. All faces of region must be planar, and a minimal triangulation is generated on non-triangular faces (using no new vertices). Optionally insert new vertices in interior of subregions based on local feature size. If there are only isolated entities, then the triangulation is in a bounding polyhedron containing the specified vertices in its interior.
- 305 – Generate tetrahedral mesh given surface mesh. All elements of the surface mesh that are on the region boundary must be oriented counterclockwise when viewed from outside the region. Region is meshed using 3-D constrained triangulation [Joe08e] and extra interior vertices are inserted based on local feature size.
- 306 – Generate tetrahedral mesh given surface mesh; region code values for elements are determined from 3-D region file if possible (else a region code of 0 is used for all elements). All elements of the surface mesh that are on the region boundary must be oriented counterclockwise when viewed from outside the region. A surface element in the interior of the region must be oriented counterclockwise when viewed from outside the subregion with smaller region code value (if the two region code values are different). Subregions are meshed using 3-D constrained triangulation [Joe08e] and extra interior vertices are inserted based on local feature size.
- 307 – Improve quality of tetrahedral mesh by smoothing and flipping [Joe89, Joe91b, Joe95a, LiJ94b], after possible conversion to order of mesh (between linear and quadratic tetrahedron).
- 308 – Locally refine tetrahedral mesh by bisecting edges of specified elements (chosen either explicitly or by distance to specified points and line segments) and possibly other elements to maintain a conforming mesh [LiJ94a, LiJ95].
- 309 – Convert hexahedral-dominant mesh to tetrahedral mesh by subdividing non-tetrahedral elements. It is possible that a hexahedron or wedge cannot be subdivided into valid tetrahedra.
- 310 – Generate tetrahedral mesh given 3-D region, with isolated vertices, edges, and faces allowed. The user has the option of specifying that no region edges are to be split, i.e. the region edges are already split into mesh edges using operation 253 (possibly with subsequent changes) or a different method. If region edges are allowed to be split, then any NURBS curves and surfaces are first approximated by

piecewise linear segments and nearly planar subfaces, and then the approach of operation 253 is used to subdivide region edges into mesh edges. If region edges are not allowed to be split, then some nonplanar faces may be subdivided into nearly planar subfaces, and any new interior surface edges are subdivided into mesh edges. Triangular surface submeshes are generated on region faces using advancing front approach of operation 254. Subregions are then meshed using 3-D constrained triangulation [Joe08e] and extra interior vertices are inserted based on local feature size.

- 350 – Generate hexahedral-dominant mesh given 3-D region, with isolated faces allowed (but not isolated vertices or edges). Any NURBS curves and surfaces are first approximated by piecewise linear segments and planar subfaces, and the faces are then decomposed into convex parts before being meshed based on local feature size. Any mesh vertices on polygonal approximation of curves and surfaces are projected to curves and surfaces after the initial surface mesh is generated. A partial decomposition and sweep submeshes are generated where possible. Remaining subregions are meshed using advancing front approach followed by improvement procedures.
- 351 – Generate hexahedral-dominant mesh given surface mesh. All elements of the surface mesh that are on the region boundary must be oriented counterclockwise when viewed from outside the region. Region is meshed using advancing front approach followed by improvement procedures.
- 352 – Generate hexahedral-dominant mesh given surface mesh; region code values for elements are determined from 3-D region file if possible (else a region code of 0 is used for all elements). All elements of the surface mesh that are on the region boundary must be oriented counterclockwise when viewed from outside the region. A surface element in the interior of the region must be oriented counterclockwise when viewed from outside the subregion with smaller region code value (if the two region code values are different). Subregions are meshed using advancing front approach followed by improvement procedures.
- 353 – Improve quality of hexahedral-dominant mesh by smoothing, merging, and flipping, with possible conversion to order of mesh (between linear and quadratic element).
- 354 – Locally refine hexahedral-dominant mesh by bisecting edges of specified elements (chosen either explicitly or by distance to specified points and line segments) and possibly other elements to maintain a conforming mesh. It is possible that a non-tetrahedral element cannot be subdivided into valid subelements.

3 Input files for meshing operations

The fields of the meshing operations are given in the following input file templates. A few fields are for the author’s use only, and are set to constants for general usage. The description of the scalar fields, some of them common to several operations, is given in the next section. The description of the nonscalar sequences of numerical data for the refinement operations is given in Section 5.

Operation 200 – Check 2-D region and curve files

```

200 reg_in_type cs_in_type 0 reg_out_type cs_out_type 0
reg_in_file_name
cs_in_file_name
reg_out_file_name
cs_out_file_name
toler 0 0
0

```

Operation 201 – Decompose 2-D region

```
201 reg_in.type cs_in.type 0 reg_out.type 0 0  
reg_in.file_name  
cs_in.file_name  
reg_out.file_name  
toler 0 0  
2 0 angcir grdtol lrdtol angles angspc angtol
```

Operation 202 – Generate 2-D mesh given region, using convex polygon decomposition

```
202 reg_in.type cs_in.type 0 0 0 mesh_out.type  
reg_in.file_name  
cs_in.file_name  
mesh_out.file_name  
toler 0 0  
4 implmidn angcir grdtol lrdtol angspc angtol munif dmin nmin nelemd  
nodelem mtype quadmu nimpiter nsmpas optmu
```

Operation 203 – Generate 2-D constrained Delaunay triangulation

```
203 reg_in.type cs_in.type 0 0 0 mesh_out.type  
reg_in.file_name  
cs_in.file_name  
mesh_out.file_name  
toler 0 0  
3 implmidn angcir grdtol lrdtol binexp hscal nodelem mtype nimpiter  
nsmpas optmu
```

Operation 204 – Improve/convert 2-D mesh

```
204 0 cs_in.type mesh_in.type 0 0 mesh_out.type  
cs_in.file_name  
mesh_in.file_name  
mesh_out.file_name  
toler 0 0  
implmidn nodelem mtype quadmu nimpiter nsmpas optmu
```

Operation 205 – Refine 2-D mesh

```
205 0 cs_in.type mesh_in.type 0 0 mesh_out.type  
cs_in.file_name  
mesh_in.file_name  
mesh_out.file_name  
toler 0 0  
implmidn sizelist mtype quadmu nimpiter nsmpas optmu  
compact_list_of_element_indices or nonscalar_sequence_of_distance_data
```

Operation 206 – Subdivide 2-D region edges into mesh edges (with update to region)

```
206 reg_in.type cs_in.type 0 reg_out.type 0 0  
reg_in.file_name  
cs_in.file_name  
reg_out.file_name  
toler 0 0  
0 angcir grdtol lrdtol hemult nodelem
```

Operation 207 – Generate 2-D mesh given region, using advancing front approach

```
207 reg_in.type cs_in.type 0 0 0 mesh_out.type  
reg_in.file_name  
cs_in.file_name  
mesh_out.file_name  
toler 0 0  
3 implmidn angcir grdtol lrdtol hemult nodelem mtype quadmu nimpiter  
nsmpas optmu
```

Operation 250 – Generate surface mesh given region, using convex face decomposition

```
250 reg_in.type cs_in.type 0 0 0 mesh_out.type  
reg_in.file_name  
cs_in.file_name  
mesh_out.file_name  
toler 0 0  
4 implmidn angcir grdtol lrdtol angfac aspc2d atol2d lsratio hfratio hfmult  
nodelem mtype quadmu nimpiter nsmpas optmu
```

Operation 251 – Improve/convert surface mesh

251 0 *cs_in.type mesh_in.type 0 0 mesh_out.type*
cs_in.file_name
mesh_in.file_name
mesh_out.file_name
toler 0 0
implmidn nodelem mtype quadmu nimpiter nsmpas optmu

Operation 252 – Refine surface mesh

252 0 *cs_in.type mesh_in.type 0 0 mesh_out.type*
cs_in.file_name
mesh_in.file_name
mesh_out.file_name
toler 0 0
implmidn sizlist mtype quadmu nimpiter nsmpas optmu
compact_list_of_element_indices or nonscalar_sequence_of_distance_data

Operation 253 – Subdivide 3-D region edges into mesh edges (with update to region)

253 *reg_in.type cs_in.type 0 reg_out.type 0 0*
reg_in.file_name
cs_in.file_name
reg_out.file_name
toler 0 0
0 angcir grdtol lrdtol hemult nodelem

Operation 254 – Generate surface mesh given region, using advancing front approach

254 *reg_in.type cs_in.type 0 0 0 mesh_out.type*
reg_in.file_name
cs_in.file_name
mesh_out.file_name
toler 0 0
3 implmidn angcir grdtol lrdtol hemult nodelem mtype quadmu nimpiter
nsmpas optmu

Operation 300 – Check 3-D region and curve/surface files

```
300 reg_in.type cs_in.type 0 reg_out.type cs_out.type 0  
reg_in.file_name  
cs_in.file_name  
reg_out.file_name  
cs_out.file_name  
toler 0 0  
0
```

Operation 301 – Decompose 3-D region

```
301 reg_in.type cs_in.type 0 reg_out.type 0 0  
reg_in.file_name  
cs_in.file_name  
reg_out.file_name  
toler 0 0  
4 0 angcir grdtol lrdtol angfac angles aspc2d atol2d angacc rdacc  
atol2dlo angacclo rdacclo
```

Operation 302 – Generate tetrahedral mesh given region, using convex face decomposition

```
302 reg_in.type cs_in.type 0 0 0 mesh_out.type  
reg_in.file_name  
cs_in.file_name  
mesh_out.file_name  
toler 0 0  
4 implmidn angcir grdtol lrdtol angfac aspc2d atol2d lsratio hfratio hfmult  
nodelem mtype hscal nimpiter nsmpas optmu poormu scalmu
```

Operation 303 – Construct 3-D Delaunay or improved-quality triangulation

```
303 reg_in.type 0 0 0 0 mesh_out.type  
reg_in.file_name  
mesh_out.file_name  
toler 0 0  
0 mtype binexp bndpol
```

Operation 304 – Construct 3-D constrained triangulation

```
304 reg_in.type cs_in.type 0 0 0 mesh_out.type  
reg_in.file_name  
cs_in.file_name  
mesh_out.file_name  
toler 0 0  
3 implmidn nodelem mtype hscal nimpiter nmpas optmu poormu scalmu
```

Operation 305 – Generate tetrahedral mesh given surface mesh

```
305 0 cs_in.type mesh_in.type 0 0 mesh_out.type  
cs_in.file_name  
mesh_in.file_name  
mesh_out.file_name  
toler 0 0  
3 implmidn regcod nodelem mtype hscal nimpiter nmpas optmu poormu scalmu
```

Operation 306 – Generate tetrahedral mesh given surface mesh and region

```
306 reg_in.type cs_in.type mesh_in.type 0 0 mesh_out.type  
reg_in.file_name  
cs_in.file_name  
mesh_in.file_name  
mesh_out.file_name  
toler 0 0  
3 implmidn nodelem mtype hscal nimpiter nmpas optmu poormu scalmu
```

Operation 307 – Improve/convert tetrahedral mesh

```
307 0 cs_in.type mesh_in.type 0 0 mesh_out.type  
cs_in.file_name  
mesh_in.file_name  
mesh_out.file_name  
toler 0 0  
implmidn nodelem mtype nimpiter nmpas optmu poormu scalmu
```

Operation 308 – Refine tetrahedral mesh

```
308 0 cs_in.type mesh_in.type 0 0 mesh_out.type  
cs_in.file_name  
mesh_in.file_name  
mesh_out.file_name  
toler 0 0  
implmidn sizlist mtype nimpiter nmpas optmu poormu scalmu  
compact_list_of_element_indices or nonscalar_sequence_of_distance_data
```

Operation 309 – Convert hexahedral-dominant mesh to tetrahedral mesh

```
309 0 cs_in.type mesh_in.type 0 0 mesh_out.type  
cs_in.file_name  
mesh_in.file_name  
mesh_out.file_name  
toler 0 0  
implmidn nodelem mtype nimpiter nmpas optmu poormu scalmu
```

Operation 310 – Generate tetrahedral mesh given region, using advancing front approach for surface mesh

```
310 reg_in.type cs_in.type 0 0 0 mesh_out.type  
reg_in.file_name  
cs_in.file_name  
mesh_out.file_name  
toler 0 0  
3 implmidn angcir grdtol lrdtol hemult nodelem mtype hscal nimpiter  
nmpas optmu poormu scalmu
```

Operation 350 – Generate hexahedral-dominant mesh given region, using convex face decomposition

```
350 reg_in.type cs_in.type 0 0 0 mesh_out.type  
reg_in.file_name  
cs_in.file_name  
mesh_out.file_name  
toler 0 0  
4 implmidn angcir grdtol lrdtol angfac angres aspc2d atol2d angacc rdacc  
atol2dlo angacclo rdacclo lsratio hfratio hfmult nodelem mtype quadmu  
nimpiter nmpas optmu mergmu
```

Operation 351 – Generate hexahedral-dominant mesh given surface mesh

```
351 0 cs_in_type mesh_in_type 0 0 mesh_out_type  
cs_in_file_name  
mesh_in_file_name  
mesh_out_file_name  
toler 0 0  
3 implmidn regcod nodelem mtype quadmu nimpiter nmpas optmu mergmu
```

Operation 352 – Generate hexahedral-dominant mesh given surface mesh and region

```
352 reg_in_type cs_in_type mesh_in_type 0 0 mesh_out_type  
reg_in_file_name  
cs_in_file_name  
mesh_in_file_name  
mesh_out_file_name  
toler 0 0  
3 implmidn nodelem mtype quadmu nimpiter nmpas optmu mergmu
```

Operation 353 – Improve/convert hexahedral-dominant mesh

```
353 0 cs_in_type mesh_in_type 0 0 mesh_out_type  
cs_in_file_name  
mesh_in_file_name  
mesh_out_file_name  
toler 0 0  
implmidn nodelem mtype nimpiter nmpas optmu mergmu
```

Operation 354 – Refine hexahedral-dominant mesh

```
354 0 cs_in_type mesh_in_type 0 0 mesh_out_type  
cs_in_file_name  
mesh_in_file_name  
mesh_out_file_name  
toler 0 0  
implmidn sizlist mtype nimpiter nmpas optmu mergmu  
compact_list_of_element_indices or nonscalar_sequence_of_distance_data
```

The following changes have been made to the original Geopack++ and Geopack90 versions (in increasing order of year):

- (a) 2002: In comparison to similar input file templates for Geopack90, the fields *isoved* of operations 200 and 300, *minrarea* and *maxtri* of operation 203, and *angres*, *angacc*, *rdacc*, *atol2dlo*, *angacclo*, and *rdacclo* of operation 302 occur in Geopack90 but have been dropped from Geopack++.

- (b) 2006: Operations 201 and 202 now allow isolated vertices and edges in regions.
- (c) 2006: The refinement operations 205 and 252 now allow the choice of bisecting or trisecting edges for linear all-quadrilateral meshes (for other types of meshes, only bisection is allowed). The use of trisection instead of bisection means that the local refinement may not extend as far into the non-selected elements. Trisection is chosen by adding 300 to the value of the *mtype* field (see the next section).
- (d) 2006: In addition to explicitly selecting the elements to locally refine, the refinement operations 205, 252, 308, and 354 now allow the selection of elements based on distances of element vertices to specified points and line segments (possibly over multiple refinement passes).
- (e) 2008: For operation 303, the field *bndtet* has been renamed to *bndpol*. The bounding polyhedron used for 3-D Delaunay and constrained triangulations has been changed to a bounding box from a bounding tetrahedron, since the former allows for fewer numerical difficulties when the number of vertices in triangulation gets larger.
- (f) 2008: Operations 206 and 207 have been added.
- (g) 2010: Operations 253, 254, and 310 have been added.

4 Description of scalar fields of meshing operations

The scalar fields of the meshing operations are described in this section, in alphabetical order of field name. In the field descriptions, the type of a field is specified as either I for integer or R for real number. Recommended values are given for some fields.

angacc (R) – Positive value is initial (highest) minimum acceptable dihedral angle (in degrees) created by separator faces. Recommended value is 25.0 for operation 301, 50.0 for operation 350.

angacclo (R) – Positive value is lowest minimum acceptable dihedral angle (in degrees) created by separator faces. If unresolved reflex edges remain, the acceptable dihedral angle is incrementally decreased from *angacc* to *angacclo*. If *angacclo* > *angacc* then no simple or convex polyhedron decomposition occurs. Recommended value is 10.0 or lower for operation 301 (if convex polyhedron decomposition is desired), 50.0 for operation 350.

angcir (R) – Positive value is angle (in degrees) for which circular arcs are subdivided in piecewise linear approximation of region curves so subarcs subtend at most this angle. Recommended value is 45.0.

angfac (R) – Positive value is minimum desirable angle (in degrees) for faces created in polygonal approximation of nonplanar surfaces. Recommended value is 15.0.

angres (R) – Magnitude ≥ 180.0 is angle (in degrees) such that any reflex vertex in (piecewise linear approximation of) 2-D region with larger interior angle is resolved, and any reflex edge in (polygonal approximation of) 3-D region with larger dihedral angle is attempted to be resolved (i.e. a separator edge or face is introduced from reflex vertex or edge). For operation 201, sign is always positive. For operations 301 and 350, the sign is positive if only trapezoidal separator faces are acceptable after simple decomposition is obtained, and the sign is negative if arbitrary separator faces are acceptable. Recommended value is 180.0 for operation 201 (if convex polygon decomposition is desired), -180.0 for operation 301 (if convex polyhedron decomposition is desired), 180.0 for operation 350.

angspc (R) – Positive value is spacing angle (in degrees) used in controlling extra edge points to be considered as separator endpoints. Recommended value is 30.0.

angtol (R) – Positive value is tolerance angle (in degrees) used in accepting separators. Recommended value is 20.0.

aspc2d (R) – Positive value is spacing angle (in degrees) used in controlling extra edge points to be considered as separator endpoints for simple or convex decomposition of faces. Recommended value is 30.0.

atol2d (R) – Positive value is tolerance angle (in degrees) used in accepting separator edges for simple or convex decomposition of faces and initial (highest) tolerance angle for acceptable face vertex angles created by separator faces. Recommended value is 20.0.

atol2dlo (R) – Positive value is lowest tolerance angle (in degrees) for acceptable face vertex angles created by separator faces. If unresolved reflex edges remain, the tolerance angle is incrementally decreased from *atol2d* to *atol2dlo*. If *atol2dlo* > *atol2d* then no simple or convex polyhedron decomposition occurs. Recommended value is 10.0 or lower for operation 301 (if convex polyhedron decomposition is desired), 20.0 for operation 350.

binexp (R) – Value is the exponent for the number of bins used for constructing Delaunay triangulation. If *binexp* > 0.0, vertices are inserted in order given by a bin sort using about n^{binexp} bins where n is the number of vertices. If *binexp* ≤ 0.0, vertices are inserted in arbitrary order. Recommended value is 0.4 or 0.5.

bndpol (I) – Value is > 0 if vertices of a bounding polyhedron are to be added to triangulation and inserted first to create initial tetrahedra, or ≤ 0 otherwise. A bounding polyhedron is needed for 3-D Delaunay triangulation if tolerance errors occur when 3-D Delaunay triangulation is constructed without bounding polyhedron. The bounding polyhedron currently used is a bounding box with 8 extra vertices (previously used was a bounding tetrahedron with 4 extra vertices). Recommended value is 1 if vertices are not nicely spaced or do not have full double precision accuracy.

dmin (R) – Positive value is used to determine if variation of mesh distribution function in convex polygon is “sufficiently high”. Recommended value is 0.4.

grdtol (R) – Positive value is global relative distance tolerance of each NURBS curve to its piecewise linear approximation. For 3-D region, this field also controls fineness/coarseness of subdivision of nonplanar faces into convex planar or nearly planar subfaces (in convex face decomposition or advancing front approach, respectively). Recommended value depends on complexity of region. For 2-D regions and simple 3-D regions with small to medium number of elements to be generated, 0.04 is recommended. For complicated 3-D regions or finer meshes, *grdtol* should be decreased by up to one or two (or more) orders of magnitude. If this field is too large then some piecewise linear curves may intersect or triangles or quadrilaterals near narrow parts may intersect. In the convex face decomposition approach, with a too large *grdtol* value, some vertices may not be projected back to NURBS curves/surfaces due to creation of invalid elements. In the advancing front approach, with a too large *grdtol* value, it may not be possible to generate a surface mesh. If this field is too small, then piecewise linear approximation of curves may contain too many segments or subdivision of nonplanar faces may contain too many subfaces. Currently, there is no automatic way to choose an “optimal” value. But in the convex face or polygon decomposition approach, the given value may be automatically decreased by a limited factor if it is estimated to be too large.

hemult (R) – If value is positive, then it is multiplier applied to local feature size to get mesh spacing for each edge (smaller *hemult* value yields shorter mesh edges). If value is nonpositive, then region edges are not split (mesh edges are assumed to be already generated on region edges).

hfmult (R) – Positive value is multiplier to be applied to local feature size to get mesh spacing for each face. A smaller value means that smaller triangles or quadrilaterals are generated on each face (and more volume elements are generated). If *hfratio* < 1.0, then feature size is based on length of edges, width of convex faces, and narrowness of subvolumes (in polygonal approximation) of region. If *hfratio* is 1.0, then feature size is based on length of edges and width of convex faces (narrowness of subvolumes is not considered). So a smaller *hfmult* value may need to be used if *hfratio* is 1.0. For mixed hexahedral meshes, a smaller *hfmult* should yield a larger percentage of volume occupied by hexahedron elements (if *hfmult* is sufficiently large).

hfratio (R) – Positive value is minimum ratio (≤ 1.0) allowed between mesh spacings (length scale for one dimension of element) for adjacent faces (in polygonal approximation) of region, where ratio is smaller spacing divided by larger spacing. Adjustments to mesh spacings are made upwards for smaller spacing to satisfy *hfratio*. *hfratio* is 1.0 for a uniform mesh spacing on all faces. Recommended value is 0.5, with larger values for more complicated regions.

hscal (R) – If value is positive, then extra interior vertices are to be inserted in triangulation based on

spacing of vertices of constrained edges and faces and isolated vertices (if any). $hscal \leq 0.0$ produces no extra vertices. $hscal = 1.0$ produces default spacing, $0.0 < hscal < 1.0$ produces denser spacing, and $hscal > 1.0$ produces sparser spacing of extra vertices. Recommended value is 1.0.

implmidn (I) – For quadratic mesh, $implmidn \geq 0$ for implicit midnodes (i.e. midnodes at midpoint of edge joining corner nodes are not generated) or $implmidn < 0$ for explicit midnodes. Value is arbitrary for linear mesh.

lrdtol (R) – Positive value is local relative distance tolerance of each non-circular NURBS curve to its piecewise linear approximation. Enough subdivision is performed so maximum distance $\leq lrdtol \cdot (\text{length scale of control polygon})$. Recommended value is 0.2.

lsratio (R) – Positive value is length scale ratio (≥ 2.0) used to determine whether length scales of edges of a convex face are sufficiently varied to attempt a further subdivision of face. A larger value means that there are fewer subdivision attempts. Recommended value is 4.0 for triangular surface or tetrahedral meshes or 8.0 for quadrilateral surface or hexahedral meshes.

mergmu (R) – Positive quantity is such that new elements formed by merging/collapsing others in mixed hexahedral mesh are acceptable if their minimum measure value is $\geq \min(mergmu, \text{measure of old elements})$. No merging/collapsing is done if $mergmu \geq 1.0$. If merging/collapsing is desired, recommended value is 0.05 for $mtype \geq 3$, 0.015 for $mtype = 2$, or 0.005 for $mtype = 1$.

mtype (I) – Value is element shape measure type: ≤ 1 for minimum (solid) angle, 2 for radius ratio, 3 for mean ratio, or ≥ 4 for quadratic mean ratio. Measure values increase as quality of element increases, and range from 0.0 to 1.0. The quadratic mean ratio is a heuristic measure so valid measure values may be slightly larger than 1.0. Invalid elements have measure ≤ 0.0 , and are not allowed in input meshes and are not created by any operation (if a quadratic element has positive linear measure but nonpositive quadratic measure, then one or more of its midnodes are moved to get positive quadratic measure). Recommended value is 3 for linear 3-D mesh, 1 or 3 for linear 2-D or surface mesh, or 4 for quadratic mesh. In the special cases of operations 205 and 252 for an all-quadrilateral mesh: *mtype* is negated to value ≤ -1 if triangles are allowed in the refinement (in this case, the magnitude of *mtype* is used for the measure type); or for a linear mesh, *mtype* is incremented by 300 to value ≥ 300 if trisection of edges is to be used instead of bisection (in this case, $mtype - 300$ is used for the measure type). In the special case of operation 303, $mtype \leq 0$ for Delaunay triangulation and $mtype > 0$ for improved-quality triangulation based on the measure type 1 for minimum solid angle, 2 for radius ratio, or ≥ 3 for mean ratio.

munif (R) – Value is mesh uniformness parameter in interval $[0.0, 1.0]$ for variation in element size. $munif = 0.0$ yields largest variation in size, and $munif = 1.0$ results in uniform size throughout region. Recommended value is 0.25, unless uniform size is desired.

nelemd (I) – Positive value is desired number of elements in 2-D mesh.

nimpiter (I) – Nonnegative value is number of improvement iterations applied to mesh. Each iteration consists of projection of vertices/midnodes to curves and surfaces (if needed), constrained Laplacian and optimization-based smoothing, and flipping/merging (a merge is an operation that combines elements with fewer nodes into fewer elements with more nodes, e.g. two triangles can be merged into a quadrilateral). In the last iteration, no flipping/merging is performed, except a few merges may be done for hexahedral-dominant meshes if $nimpiter > 1$. Recommended value is 2 or 3, unless no improvement is desired.

nmin (I) – Positive value is used to determine if “sufficiently large” number of elements are generated in convex polygon. Recommended value is 10.

nodelem (I) – Magnitude is maximum number of nodes per element of mesh; sign is negative for a mixed mesh. For 2-D or surface mesh, *nodelem* is 3 for linear triangle, 4 for linear quadrilateral, 6 for quadratic triangle, 8 for quadratic quadrilateral, -4 for linear quadrilateral/triangle, and -8 for quadratic quadrilateral/triangle. For 3-D mesh, *nodelem* is 4 for linear tetrahedron, 8 for linear hexahedron, 10 for quadratic tetrahedron, 20 for quadratic hexahedron, -8 for linear hexahedron/wedge/pyramid/tetrahedron, or -20 for quadratic hexahedron/wedge/pyramid/tetrahedron. Since an all-hexahedral mesh cannot be guaranteed to be generated, 8 and 20 are currently equivalent to -8 and -20 . For operations 204, 251, 307, and 353, an invalid *nodelem* value may be specified to indicate that the element type is unchanged from the input mesh file. For the other operations, an invalid *nodelem* value defaults to 3 (linear triangle), 4 (linear

tetrahedron), or 8 (linear hexahedron) depending on the type of mesh. For operations 206 and 253, *nodelem* must be 4 or 8 in order to get an even number of mesh edges generated on boundary of each 2-D subregion or 3-D face (which is needed for an all-quadrilateral mesh).

nsmpas (I) – Magnitude is number of smoothing passes through vertices each time that smoothing is performed. Sign is negative if curve or curve/surface vertices are not to be smoothed for 2-D/surface or 3-D meshes, respectively. Recommended value is 2.

optmu (R) – Quantity is maximum local measure value at which optimization-based smoothing is attempted for movable interior and surface vertices, i.e. optimization-based smoothing is attempted if $\min(\text{measures of elements incident on vertex}) \leq \text{optmu}$. Recommended value is 0.3 to 0.4 if optimization-based smoothing is desired, or 0.0 if no optimization-based smoothing is desired.

poormu (R) – Quantity is such that any tetrahedron with measure $< \text{poormu}$ is deemed poorly shaped. Vertex insertion schemes may be used to remove poorly shaped tetrahedra. Recommended value is 0.1 for $mtype \geq 3$, 0.03 for $mtype = 2$, or 0.01 for $mtype = 1$.

quadmu (R) – Nonzero magnitude is minimum measure value at which a quadrilateral is preferred over two triangles, i.e. a quadrilateral with measure $> |\text{quadmu}|$ is preferred to its split into two triangles. For the 2-D and surface mesh operations, the sign is negative if no attempt is to be made to obtain or keep a layer of adjacent quadrilaterals next to constrained edges. For operations 350, 351, and 352, the sign must be positive. Recommended value is 0.1 for $mtype \geq 3$, 0.03 for $mtype = 2$, or 0.01 for $mtype = 1$.

rdacc (R) – Positive value is initial (highest) minimum acceptable relative distance between a separator plane and vertices not on plane. Recommended value is 0.2.

rdacclo (R) – Positive value is lowest minimum acceptable relative distance between a separator plane and vertices not on plane. If unresolved reflex edges remain, the acceptable relative distance is incrementally decreased from *rdacc* to *rdacclo*. If $\text{rdacclo} > \text{rdacc}$ then no simple or convex polyhedron decomposition occurs. Recommended value is 0.05 or lower for operation 301 (if convex polyhedron decomposition is desired), 0.05 for operation 350.

regcod (I) – Nonnegative value is region code for elements of 3-D mesh.

scalmu (R) – Value is scale factor in interval (0.0, 1.0] such that a vertex is inserted on an edge of a poorly-shaped tetrahedron if (resulting local minimum measure) $> \text{scalmu} \cdot (\text{existing local minimum measure})$, i.e. the tetrahedra are allowed to become worse if $\text{scalmu} < 1.0$. Recommended value is 1.0. Try increasing *nimpiter* or decreasing *scalmu* to 0.5 if poorly shaped tetrahedra remain in mesh.

sizlist (I) – If nonnegative, value is size of *compact_list_of_element_indices* (see the next section). If negative, $-\text{sizlist}$ is the number of refinement passes for choosing elements to refine based on distances to specified points and line segments.

5 Description of nonscalar sequences of numerical data

The two cases of nonscalar sequences of numerical data for the refinement operations are described in this section.

compact_list_of_element_indices (integer array) – The list of $\text{sizlist} \geq 0$ signed integer element indices (labels) may contain negated indices to indicate the end of a range of elements, e.g. 2 -5 10 20 -28 is short for the elements with indices 2, ..., 5, 10, 20, ..., 28. The expanded list of element indices specifies the elements to locally refine.

nonscalar_sequence_of_distance_data – The number of refinement passes is $-\text{sizlist}$. Multiple passes can be used to produce much smaller elements near specified points or line segments. An element is chosen for refinement in a pass if all its vertices (or corner nodes) are within the specified distance of a specified point or line segment. The data for the first pass follows the scalar fields. The data for each further pass (if any) follows that for the previous pass. Each pass contains the below fields in the given order, except that the z_i and zz_i fields are omitted for operation 205. *npt* and *nls* are integer fields. The other fields have real number values. $\text{npt} \geq 0$ and $\text{nls} \geq 0$ are the number of points and line segments, respectively. (x_i, y_i) and (xx_i, yy_i) are coordinates of points or line segment endpoints in 2-D plane for operation 205. (x_i, y_i, z_i) and

(xx_i, yy_i, zz_i) are coordinates of points or line segment endpoints in 3-D space for operations 252, 308, and 354. $d_i \geq 0.0$ are distances. Distance data for points precedes that for line segments.

```

npt  nls
x1  y1  z1  d1
  :
xnpt  ynpt  znpt  dnpt
xnpt+1  ynpt+1  znpt+1  xxnpt+1  yynpt+1  zznpt+1  dnpt+1
  :
xnpt+nls  ynpt+nls  znpt+nls  xxnpt+nls  yynpt+nls  zznpt+nls  dnpt+nls

```

6 Algorithmic notes

Some of the new algorithms and research results used in Geompack++ appear in the technical reports [Joe08b, Joe08c, Joe08d, Joe08e]. Others will appear in future technical reports. Some brief algorithmic notes are provided in the following subsections.

Surface meshing using convex face decomposition

This is the approach used in operations 250, 302, and 350. Surface meshes are generated by approximating nonlinear curves and nonplanar surfaces by piecewise linear segments and planar subfaces, decomposing planar faces into convex subfaces, generating submeshes on the planar convex subfaces using the 2-D algorithms of operation 202, and projecting vertices of submeshes back to nonlinear curves and nonplanar surfaces (if elements maintain a positive shape measure). For complicated regions with nonplanar surfaces, many convex subfaces may result in decomposition to avoid small angles in faces. The positive shape measure criterion means that some vertices of type 4 or 6 may remain in mesh.

Surface meshing using advancing front approach

This is the approach used in operations 254 and 310. Surface meshes are generated by approximating nonlinear curves and nonplanar surfaces by piecewise linear segments and nearly planar subfaces (that may be nonconvex or nonsimple with face holes), generating submeshes on planar faces or planar approximation of nearly planar subfaces using the 2-D algorithms of operation 207, and projecting vertices of submeshes back to nonplanar surface if planar approximation of face is used. This approach produces fewer subfaces (much fewer for complicated regions) than the convex face decomposition approach since subdivision of faces does not go as far. If there are projection problems or invalid elements result from projection, then either invalid elements are fixed by moving vertices or the nonplanar subface is further split if possible. Because projection is done before checking for valid elements (opposite order to convex face decomposition approach), vertices of type 4 or 6 do not remain in mesh. But the meshing process may fail if (a) *grdtol* is too large for complicated regions, or (b) the domain curve $(u(t), v(t))$ for a nonplanar NURBS surface $S(u, v)$ is such that $S(u(t), v(t))$ is not a close enough approximation to the corresponding space curve $C(t)$, or (c) other geometric difficulties occur.

Fields *grdtol*, *lrtdtol*, and *angcir*

The fields *grdtol*, *lrtdtol*, and *angcir* affect the closeness of polygonal approximation to the actual curves and surfaces of 2-D and 3-D regions. Decreasing *grdtol* affects the approximation of all curves and nonplanar surfaces, while decreasing *lrtdtol* or *angcir* affects the approximation of non-circular NURBS curves and circular arcs, respectively. If *grdtol* is too large relative to the sizes of generated elements, then there may be difficulty projecting some vertices all the way back to the actual curve or surface (convex polygon or face decomposition approach), or invalid elements may result after projection back to nonplanar surface (advancing front approach for surface meshes). The projection is harder for quadrilateral meshes than

triangular meshes since it is harder for a quadrilateral to maintain a positive shape measure as its vertices are moved. If there are some unprojected vertices or some surface elements have a size that is large relative to the local narrowness of 3-D region, then the surface mesh may self-intersect causing failure to generate a 3-D mesh. For the hexahedral-dominant mesh, near self-intersection of the quadrilateral surface mesh can also cause failure. If these problems arise, then decreasing *grdtol* and/or increasing *hfratio* may help. Because the two approaches use a different amount of subdivision of nonplanar faces and do projection of mesh vertices differently, different *grdtol* values may be needed to get a satisfactory mesh for complicated 3-D regions using the two approaches.

Mesh edge generation for advancing front approach

Operations 206, 207, 253, 254, and 310 use local feature size and heuristics to determine mesh edge sizes on region edges. For a region edge, the subdivision into mesh edges could be uniform or geometric 1-sided (mesh edge size gradually increases from one endpoint to the other) or geometric 2-sided (mesh edge size gradually increases from both endpoints to middle of region edge). Since no method can generate optimal mesh edge sizes for all regions and applications, *hemult* ≤ 0.0 may be specified for operation 207, 254, or 310 and the user may use his or her own mesh edge generation method in the input region or modify the output region of operation 206 or 253.

To help guide the advancing front mesh generation, the user can add (unconstrained) isolated edges in the interior of a 2-D region or split (nonplanar) faces of a 3-D region into subfaces.

Unsplittable edges may be specified by marking the appropriate edges in 2-D or 3-D region, in which case these edges remain unsplittable throughout all meshing stages. For operations 207, 254, and 310, *hemult* ≤ 0.0 may be used to specify that all edges are unsplittable during the advancing front generation of the 2-D or surface mesh, but any edges marked as splittable may be split during the postprocessing improvement stage of meshing.

When all-quadrilateral mesh is possible

If there are no unsplittable edges, then operations 202, 206, 207, 250, 253, and 254 can generate an even number of mesh edges on the boundary of each 2-D subregion or 3-D face (if *nodelem* value of 4 or 8 is specified). The following enhancements were added in the 2010 version, in the case there is at least one unsplittable edge. If there is a subregion or face (possibly the union of smaller adjacent subregions or subfaces) with an odd number of unsplittable edges and no splittable edges on its boundary, then obviously it is not possible to generate an even number of mesh edges on the boundary of this subregion or face, so an all-quadrilateral mesh cannot be generated for this subregion or face. For operations 202, 206, and 207, this is the only situation where the even number is not possible for all subregions. For operations 250, 253, and 254, the even number may also not be possible for all faces if there is a splittable edge incident on 3 or more faces (and at least one unsplittable edge).

Other notes on advancing front approach for 2-D and surface meshes

The advancing front approach allows for more variability in the size of mesh edges generated on the edges of 2-D subregions or 3-D faces. So it is better than the convex polygon or face decomposition approach in generating mesh elements with larger local change in element sizes. But if the variability in mesh edge sizes is too large over a subregion or face, some poor-quality elements may be generated.

The advancing front approach is much better than the convex polygon or face decomposition approach at generating 2-D and surface meshes when there are unsplittable edges specified in region. In particular, operation 250 (and also operations 302 and 350) may generate a poor surface submesh if a nonplanar face has unsplittable boundary edges.

The advancing front approach takes more time for meshing than the convex polygon approach. The amount of time to generate an element at the front is approximately constant or slightly more than constant, so the total time is slightly greater than linear time, but the constant of proportionality is larger than with the convex polygon approach due to the amount of intersection and other basic operations.

For operation 207, the closing phase uses some of the algorithms of operation 202, e.g. convex polygon decomposition and triangulation or quadrilateral of convex polygon. In the 2010 version, more layers of advancing front are done before the closing phase is applied, so better elements are generated in the interior of the subregion in the case that (nearly) uniform-sized unsplitable edges are specified on the subregion boundary.

Refinement of all-quadrilateral meshes

When locally refining all-quadrilateral meshes (2-D or surface) by bisection of edges, it is possible for refinement to extend quite far from the specified elements in order to maintain a conforming refined mesh with elements of reasonable quality. If triangles are allowed in the refined mesh (by negating the *mtype* field), then the local refinement will not extend as far and the subsequent improvement stage may merge some triangles into quadrilaterals. Alternatively, if trisection of quadrilateral edges is used instead of bisection in the refined linear mesh (by incrementing the *mtype* field by 300), then the local refinement will not extend as far but each quadrilateral may be refined into up to 9 subelements (instead of 4).

Refinement of hexahedral-dominant meshes

The refinement of a hexahedral-dominant mesh is not guaranteed to produce a mesh in which all elements have positive shape measure (the same applies when converting a hexahedral-dominant mesh to a tetrahedral mesh). If any elements have nonpositive shape measure, then smoothing is applied. If there are still invalid elements after smoothing, then an error occurs. In trying to maintain a conforming refined mesh with elements of reasonable quality, it is possible that local refinement extends to global refinement.

Hexahedral-dominant mesh generation

Hexahedral-dominant mesh generation is much slower than the other operations. Sweep submeshes are used where possible (for operation 350), otherwise an advancing front approach is used starting from elements of surface submeshes. The advancing front approach takes slightly greater than linear time, but the constant of proportionality is rather large due to the amount of intersection and other basic operations. All generated elements have positive shape measure (the Jacobian determinant evaluated at corner nodes is positive, except at the apex node of a pyramid).

The Geompack++ hexahedral-dominant mesher has a lot of enhancements over the Geompack90 one. There are some faster procedures used and fewer elements generated in general (particularly fewer pyramids and tetrahedra in the interior of the mesh). Also, with the given surface mesh, the number of hexahedra in the layer next to boundary and constrained faces is close to optimal, using the criterion that hexahedra incident on adjacent surface quadrilaterals (sharing an edge whose dihedral angle is approximately in the range 120° to 240°) should share a common face.

An alternative way to generate a hexahedral-dominant mesh is to first use operation 254 to generate a quadrilateral surface mesh and then use this surface mesh as input to operation 351 or 352.

Research is continuing on techniques to produce fewer pyramids and tetrahedra and better quality elements in the interior of the region. Research is also continuing on determining sufficient and necessary conditions on the quadrilateral surface mesh in order to generate a complete layer of hexahedra next to the subregion boundaries and ultimately an all-hexahedral mesh.

Miscellaneous notes

- (a) In 2006, the 2-D decomposition and meshing operations 201 and 202 were enhanced to allow isolated vertices and edges in regions. Before proceeding with the decomposition, isolated vertices and edges are joined to subregion boundaries using short crack or cut edges that do not introduce small interior angles or short boundary edges.
- (b) For operation 301 with *angres* field set to -180.0 , the decomposition proceeds as far as possible based on the values of the fields *atol2d*, *angacc*, *rdacc*, *atol2dlo*, *angacclo*, *rdacclo*. By making the latter 3

fields sufficiently small (as low as 1.0 or 0.1 for the angles and 0.005 or 0.001 for *rdaclo*), it is likely that a convex polyhedron decomposition can be obtained, but some polyhedra will be poorly shaped.

- (c) The 3-D constrained triangulation algorithm is successful in most situations. The difficult cases are when there are non-adjacent constrained triangular faces that are much closer to each other than the length of the edges involved.

References

- [Joe86] B. Joe (1986), Delaunay triangular meshes in convex polygons, *SIAM J. Sci. Stat. Comput.*, 7, pp. 514-539.
- [Joe89] B. Joe (1989), Three-dimensional triangulations from local transformations, *SIAM J. Sci. Stat. Comput.*, 10, pp. 718-741.
- [Joe91a] B. Joe (1991), Construction of three-dimensional Delaunay triangulations using local transformations, *Computer Aided Geometric Design*, 8, pp. 123-142.
- [Joe91b] B. Joe (1991), Delaunay versus max-min solid angle triangulations for three-dimensional mesh generation, *Int. J. Numer. Methods Eng.*, 31, pp. 987-997.
- [Joe91c] B. Joe (1991), GEOMPACK – a software package for the generation of meshes using geometric algorithms, *Adv. Eng. Software*, 13, pp. 325-331.
- [Joe92] B. Joe (1992), Three-dimensional boundary-constrained triangulations, in *Artificial Intelligence, Expert Systems, and Symbolic Computing*, ed. E. N. Houstis and J. R. Rice, Elsevier Science Publishers, pp. 215-222.
- [Joe94] B. Joe (1994), Tetrahedral mesh generation in polyhedral regions based on convex polyhedron decompositions, *Int. J. Numer. Methods Eng.*, 37, pp. 693-713.
- [Joe95a] B. Joe (1995), Construction of three-dimensional improved-quality triangulations using local transformations, *SIAM J. Sci. Comput.*, 16, pp. 1292-1307.
- [Joe95b] B. Joe (1995), Quadrilateral mesh generation in polygonal regions, *Computer-Aided Design*, 27, pp. 209-222.
- [Joe08a] B. Joe (2008), Geompack++ file formats for regions and meshes, Technical Report ZCS2008-01, <http://members.shaw.ca/bjoe>.
- [Joe08b] B. Joe (2008), Shape measures for quadrilaterals, pyramids, wedges, and hexahedra, Technical Report ZCS2008-03, <http://members.shaw.ca/bjoe>.
- [Joe08c] B. Joe (2008), Flips for quadrilateral meshes, Technical Report ZCS2008-04, <http://members.shaw.ca/bjoe>.
- [Joe08d] B. Joe (2008), Local refinement of quadrilateral meshes, Technical Report ZCS2008-05, <http://members.shaw.ca/bjoe>.
- [Joe08e] B. Joe (2008), Construction of three-dimensional constrained triangulations, Technical Report ZCS2008-06, <http://members.shaw.ca/bjoe>.
- [JoS86] B. Joe and R. B. Simpson (1986), Triangular meshes for regions of complicated shape, *Int. J. Numer. Methods Eng.*, 23, pp. 751-778.
- [LiJ94a] A. Liu and B. Joe (1994), On the shape of tetrahedra from bisection, *Mathematics of Computation*, 63, pp. 141-154.

- [LiJ94b] A. Liu and B. Joe (1994), Relationship between tetrahedron shape measures, *BIT*, 34, pp. 268-287.
- [LiJ95] A. Liu and B. Joe (1995), Quality local refinement of tetrahedral meshes based on bisection, *SIAM J. Sci. Comput.*, 16, pp. 1269-1291.
- [LiJ96] A. Liu and B. Joe (1996), Quality local refinement of tetrahedral meshes based on 8-subtetrahedron subdivision, *Mathematics of Computation*, 65, pp. 1183-1200.