

Algorithm xxx. gmesh2: An Interactive Graphical Two-dimensional Finite Element Mesh Generator

Robert J. Renka
University of North Texas *

June 24, 2009

gmesh2 is an interactive finite element mesh generator implemented in C, OpenGL, and GLUT. It allows interactive domain alteration, and automatically generates structured and unstructured triangle meshes, providing for both uniform refinement and Delaunay refinement, as well as mesh smoothing. All operations may be applied to the entire domain or to user-specified sub-domains.

Categories and Subject Descriptors: G.4 [**Mathematics of Computing**]: Mathematical Software

General Terms: Algorithms

Additional Key Words and Phrases: Delaunay triangulation, mesh refinement, OpenGL, triangle mesh

1 INTRODUCTION

Finite element mesh generation is an enormously important problem, and has received commensurate attention over the past three or four decades. A web search for mesh generation software will turn up several surveys and long lists of packages categorized by criteria such as the underlying methods, commercial versus non-commercial, source-code availability, structured versus unstructured, two-dimensional versus three-dimensional, and triangle-based versus quadrilateral-based in the two-dimensional case. Based on considerably less than exhaustive

*Department of Computer Science & Engineering, University of North Texas, Denton, Texas 76203-1366

research, the software package that appears to be most similar to the one discussed here is Triangle [Shewchuk 2002a], an often-cited and widely distributed code.

In addition to serving in a productivity role, at least for meshes of moderate size, `gmesh2` is designed to be a research tool for experimenting with methods of grid generation. Thus, it includes the option of testing the integrity of the mesh even though the test cannot produce a negative result (at least not by design). The purpose is to provide for debugging code changes and additions. The code is well-organized, well-documented, and based on simple data structures, thus facilitating easy alteration. Bug reports sent to the author’s email address are welcome but should, of course, be restricted to the unaltered code.

The underlying theory, algorithms, and data structures are discussed in Section 2. Section 3 treats installation, the data file format, and usage, including descriptions of the menu options provided to an interactive user. Some test results are presented in Section 4.

2 ALGORITHMS AND DATA STRUCTURES

In finite element mesh generation, the fundamental problem is to satisfy constraints on the size and shape of elements, along with constraints imposed by the domain geometry, using as few elements as possible, in order to meet the accuracy requirements of the finite element solution at minimal computational cost. The choice of quality measure for triangles is problem dependent, but with the exception of anisotropic meshing, in which there is a preferred direction (in a flow problem for example), the ideal triangle shape is equilateral. At least in the case of linear elements, small angles are problematic because they lead to ill-conditioned linear systems, and large angles are bad because they result in large interpolation errors. Refer to [Shewchuk 2002b] for a very readable treatment of these issues.

2.1 Triangle Quality Measures

We borrowed Bank’s scheme for color-coding triangles according to the following measure of triangle quality [Bank 1990]:

$$q_1(t) = 4\sqrt{3} \frac{a}{h_1^2 + h_2^2 + h_3^2},$$

where the area and side lengths of triangle t are denoted by a and h_1 , h_2 , and h_3 , respectively. Values are in the range 0 to 1 with value 0 for a degenerate triangle, and value 1 for an equilateral triangle. Triangles are categorized as follows.

good $\sqrt{3}/2 \leq q_1 \leq 1$. The largest angle is at most $\pi/2$, and the smallest angle is at least $\arccos(0.8) \approx 36.9$ degrees.

fair $0.6 \leq q_1 < \sqrt{3}/2$. The largest angle is at most $2\pi/3$, and the smallest angle is at least $\arccos(13/14) \approx 21.8$ degrees.

poor $0 \leq q_1 < 0.6$.

A second commonly used measure of quality is the ratio of inradius r_{in} to circumradius r_{circ} scaled by 2 so that an equilateral triangle again has value 1:

$$q_2(t) = \frac{2r_{\text{in}}}{r_{\text{circ}}} = \frac{(h_2 + h_3 - h_1)(h_3 + h_1 - h_2)(h_1 + h_2 - h_3)}{h_1 + h_2 + h_3}.$$

The measure that is naturally improved by Delaunay mesh refinement is the minimum angle α or the ratio of shortest edge length h_{min} to circumradius (twice the sine of the minimum angle), again normalized to $[0, 1]$ [Shewchuk 2002b]:

$$q_3(t) = \frac{2}{\sqrt{3}} \sin(\alpha) = \frac{1}{\sqrt{3}} \frac{h_{\text{min}}}{r_{\text{circ}}} = \frac{4}{\sqrt{3}} \frac{a}{h_{\text{max}}h_{\text{med}}},$$

where $h_{\text{max}}h_{\text{med}} = h_1h_2h_3/h_{\text{min}}$.

2.2 Triangle Mesh Definition and Data Structures

A triangle mesh is a set of triangles in which the intersection of two triangles, if not empty, is a vertex of both or an edge of both, and whose union is a connected region with one or more simple closed non-intersecting boundary curves. Boundary curves are oriented so that the triangles are on the left as the sequence of boundary vertices is traversed. Thus each boundary vertex has a unique predecessor and successor.

The domain geometry is defined by designating a subset of the boundary vertices as being non-removable. These form the corners in the domain boundary. Note the simplicity of this definition. It is not necessary to specify the connection topology of the non-removable vertices; the connections are implicitly defined by the triangle mesh.

Memory management is also made particularly simple by specifying storage requirements in terms of a single parameter — the maximum number of vertices N_{max} . Each of the following mesh parameters has an upper bound that is expressible as a multiple of N_{max} .

- N_v : Number of vertices.
- $N_b \leq N_v$: Number of boundary vertices (boundary edges).
- $N_c \leq N_b/3$: Number of boundary curves.
- $N_n \leq N_b$: Number of non-removable vertices.
- $N_t = 2N_v - N_b + 2N_c - 4 \leq 2N_v - 5$: Number of triangles.
- $N_e = 3N_v - N_b + 3N_c - 6 \leq 3N_v - 6$: Number of edges.

The primary data structure is a vertex list containing the vertex coordinates, and a triangle list stored as an array dimensioned $2 \times N_{\text{max}}$ by 9, and containing

three vertex indices, three neighboring triangle indices, and three edge indices for each triangle. Vertices are stored in cyclical counterclockwise order, and the arbitrary choice of first vertex determines the ordering of the neighboring triangles and edges: the first edge is opposite the first vertex and is shared by the first neighboring triangle. In the case of a boundary edge, the non-existent neighboring triangle is represented by index -1.

If an array of values, edge midpoints in the case of 6-node triangles for example, is associated with edges, it is necessary to order the edges, assign them indices, and store the relationship between edge indices and vertex indices. This is the reason that edge indices are included in the triangle list even though they are redundant for the purpose of mesh generation.

Several additional arrays are needed for efficiency: a list that maps each vertex index to one of the triangles containing the vertex (uniquely determined in the case of a boundary vertex), an edge list that maps an edge index to one of the triangles containing the edge (the one with larger index), a boundary curve list containing a boundary vertex index for each boundary curve, and a list of non-removable vertex indices, along with a Boolean flag specifying removable/non-removable for each vertex. One additional array of length $3 \times N_{max}$ is used by several functions as temporary storage for vertices, triangles, or edges.

2.3 Triangle Mesh Subsets: Polygon **R**

The user may select a polygonal region **R** to be refined, optimized, coarsened, or removed from the mesh. The polygon boundary is a simple closed unoriented curve defined by a sequence of three or more vertices. Vertices of **R** may be added, deleted, or moved by mouse button presses and mouse motion. The relationship between **R** and the mesh objects is defined by a simple and elegant method using a list of Boolean flags that specify, for each vertex in the mesh, whether or not it is contained in **R**. This list is re-created whenever **R** is altered, and is updated with every change in the triangle mesh. For expediency, we define edges and triangles to be contained in **R** if and only if all their vertices are in **R**, and a new vertex inserted into an edge or triangle that is contained in **R** is always flagged as being in **R**.

2.4 Delaunay Refinement

A Delaunay triangulation is a triangle mesh in which the triangle circumcircles (the circles defined by the three vertices of each triangle) contain no vertices in their interiors. A constrained Delaunay triangulation is a triangle mesh in which, if the circumcircle of a triangle has a vertex in its interior, that vertex is separated from (every point of) the triangle interior by a boundary edge. An edge swap is the replacement of a pair of triangles forming a convex quadrilateral by the pair of triangles associated with the other diagonal. A swap test is the local application of the Delaunay empty circumcircle criterion to a pair of adjacent triangles. The swap test is well-defined and selects the pair of triangles that maximizes the minimum angle over the two pairs [Lawson 1977].

An efficient and numerically stable implementation of the swap test is described in [Cline and Renka 1984].

Our primary mesh-refinement algorithm is Rupert’s Delaunay refinement method [Rupert 1995] as elaborated on and extended by Shewchuk [Shewchuk 2002a] and Üngör [Üngör 2009]. A constrained Delaunay triangulation is converted to a Delaunay triangulation by inserting new vertices (Steiner points) at circumcenters or off-centers of bad triangles and near midpoints of boundary edges, and applying Delaunay edge swaps. Bad triangles are those that are too large or that have an angle smaller than a user-specified minimum. The circumcenter of a bad triangle is a reasonable choice for a new vertex because it splits the bad triangle without creating short edges, which imply small angles. The idea behind the off-center is that, by choosing an insertion point between the circumcenter and the midpoint of the shortest edge of the triangle being split, it is often possible to avoid creating a new triangle that would fail the minimum angle criterion. Thus, when off-centers are used in place of circumcenters, depending on the boundary constraints, it may be possible to satisfy the size and shape criteria with fewer triangles. This is discussed further in Section 4.

When a boundary vertex or potential new vertex (circumcenter or off-center) encroaches upon a boundary edge (lies on or inside the diametral circle of the edge), the edge is split by inserting its midpoint (or a nearby point chosen to avoid an infinite sequence of edge splits that can occur when adjacent boundary edges form a small angle). Following each insertion of a new vertex, the swap test and appropriate edge swaps are applied to the edges opposite the new vertex. This sequence of swaps is the key element of Lawson’s incremental algorithm for constructing a Delaunay triangulation [Lawson 1977].

The encroached boundary edges are held in a queue stored as a circular array of boundary edge structures, each consisting of an edge index and a pair of endpoint vertex indices. The vertex indices are used to determine if the edge is still present in the mesh when it is dequeued. The bad triangles are stored in a priority queue with priority defined by the length of the shortest edge. This was found to result in fewer triangles than assigning priority by smallest angle ([Üngör 2009]). The priority queue is implemented as a binary heap with an array of pointers into an array of triangle structures, each containing a triangle index, the three vertex indices, the squared length of the shortest edge, and the index (reverse pointer) into the array of pointers.

The use of a binary heap entails an $O(N_v \times \log(N_v))$ operation count, making Delaunay refinement asymptotically less efficient than it could be. All other operations are linear in N_v . The priority queue could be eliminated at a cost of requiring more triangles to satisfy the size and shape constraints. The use of a priority queue would have to be eliminated if the code were to be parallelized.

2.5 Other Operations

The package provides two simple uniform mesh-refinement methods: edge-splitting at midpoints, and insertion of triangle barycenters. When an edge is split, its midpoint is joined to the opposite vertices (or vertex in the case of a bound-

ary edge). Following the edge splits, an edge swap is applied to the first edge inserted into each of the original triangles, so that each triangle in \mathbf{R} is partitioned into four similar equal-area subtriangles. In the case of the barycenter insertions, no edge swaps are automatically applied, and each triangle in \mathbf{R} is partitioned into three equal-area subtriangles which are similar to each other but not similar to the parent triangle. The subtriangles are not well-shaped in this case, but a good mesh can be restored by applying an edge optimization operation.

The edge optimization operation consists of several passes through the edges in \mathbf{R} , applying Delaunay swap tests and the appropriate swaps. The operation is terminated with a constrained Delaunay triangulation when a pass through the edges results in no swaps. However, the number of passes is limited to four — usually, but not necessarily sufficient. This operation is an option that should be user-selected following uniform refinement by barycenter insertion. It is automatically applied as the first step of Delaunay refinement, and it is automatically applied before each smoothing pass.

In addition to optimization with topological changes (edge swaps), the program provides a smoothing option based on centroidal Voronoi tessellations [Chen 2004] in which removable vertices in \mathbf{R} may be moved. Each removable boundary vertex is moved to the midpoint of its predecessor and successor (unless they are adjacent). In the case of an interior vertex, the potential new position is a convex combination of the centroids of the triangles in its star (the triangles that contain the vertex), with triangle areas relative to the star area as the coefficients. The vertex is moved to the new position if it is viable (all the neighboring vertices are visible, so that no change in mesh topology would be required). A boundary layer option is provided, in which each interior vertex adjacent to a boundary vertex is moved to a location that depends only on the triangles that contain the vertex and contain a boundary vertex, so that the new location is usually closer to the boundary. The smoothing operation involves a single sweep through the vertices (preceded by an edge optimization operation), and might have to be applied several times before changes become insignificant. The definition of convergence is thus left to the user.

The smoothing method was chosen for simplicity and efficiency. A better smoothing algorithm, not implemented, would be to move a vertex if and only if the new position is both viable and improves some measure of triangle quality.

The problem of mesh coarsening is difficult, and we offer only a crude solution. A randomly selected subset of the removable vertices in \mathbf{R} , with relative size arbitrarily chosen to be about 70%, is removed from the mesh. Following a mesh coarsening operation, mesh quality can be restored by smoothing (with edge optimization included).

3 INSTALLING AND USING THE SOFTWARE

3.1 Installation

The program consists of a single source code file, `glnesh2.c`, along with a sample input file, `glnesh2.in` containing a simple triangle mesh in the required format (described below). The program is started by specifying the name of the executable file followed by an input file name and an output file name on the command line. Omitting the input or output file name results in a description of the input file format appearing in the terminal window. Once started, the right mouse button brings up a menu of options. When the program is terminated, the triangle mesh is written to the output file. Additional intermediate output files can be obtained by selecting the write output option, and renaming the output file.

The computing platform must include OpenGL [OpenGL 1997] and GLUT [Kilgard 1996]. OpenGL is a standard interface to the graphics capability of all modern graphics adapters, and the OpenGL Utility Toolkit (GLUT) provides a simple means of window creation and event handling (keyboard and mouse input) that is independent of the hardware, operating system, and window management system. OpenGL is installed with the operating system or compiler on most computers, and GLUT, or one of several similar alternatives may be freely downloaded from the web. Header comments in the `glnesh2` source code include instructions for compiling and linking with the appropriate libraries on systems running Microsoft Windows, Linux, and Apple OS X.

3.2 Data File Format

The triangle mesh file format contains no redundant information. It includes only N_v , the vertex list, N_t , and the first three columns of the triangle list. The adjacency information is determined by a search, and the edges are assigned indices in a systematic fashion.

A data set consists of the following sequence of at least $N_v + N_t + 2$ records.

Optional comment lines, each beginning with `#`

N_v

x_0 y_0

x_1 y_1

...

x_{N_v-1} y_{N_v-1}

Optional comment lines, each beginning with `#`

N_t

$i1_0$ $i2_0$ $i3_0$

$i1_1$ $i2_1$ $i3_1$

...

$i1_{N_t-1}$ $i2_{N_t-1}$ $i3_{N_t-1}$

with one or more spaces separating vertex components and vertex indices, and vertex indices in the range 0 to $N_v - 1$.

The output file includes the creation date and the triangle mesh parameter values in the beginning comment lines, and has two additional values, u_i v_i , following x_i y_i for each vertex. These are vertex function values, such as velocity components, with values that default to zeros, but may be altered by a menu option.

3.3 Menu Options

Upon execution, the input triangle mesh is displayed in a new window with the input data file name in the title bar. A right mouse button click displays a menu of options in a popup window. The menu entries are more or less self-explanatory, but necessarily cryptic. We provide more complete descriptions here.

Note that the price of portability across windowing systems is a less than perfect user interface. The menu-handling capabilities of GLUT, in particular, are quite limited. Also, in the interest of simplicity, all text entry is through the terminal from which the program was started. This entails the somewhat awkward process of switching back and forth between the primary display window and the text-entry window.

Each menu entry has an associated key, allowing the option to be selected either from the menu (mouse button) or by a single keypress. There is a main menu and three sub-menus. The main menu options are as follows:

- B:** Set **R** to contain the Bounding Box. **R** is initialized to an axis-aligned rectangle that contains all of the vertices. This is a good first step for operating on the entire grid.
- c:** Cycle through triangle Color-fill options. There are four options for rendering triangles: outline (only the edges are drawn — the default), color-filled using four alternating colors (a 4-coloring of the mesh, useful only for making pretty pictures), color-filled only if in **R** (useful for making it clear which triangles are in **R**), and color-coded by quality (red, green, and blue correspond to poor, good, and fair, respectively, as defined for q_1).
- e:** Toggle annotation of Edges with indices.
- F:** Output the triangle mesh to a File. The contents of the file whose name was specified on the command line are overwritten. Intermediate output files must therefore be renamed. This option is automatically selected on termination of the program.

- o:** Draw a circle (drag with left mouse button). This option is used, along with option **O**, to create a domain with circular holes and/or rounded corners.
- P:** Print triangle mesh data structure. The triangle mesh data structures, along with triangle quality measures, are printed in the terminal window from which the program was started.
- q:** Query current state (print parameters). Parameters defining the current state of the program, along with mesh parameters, are written to the terminal window.
- r:** Restore the default viewing volume (which is altered by pan and zoom options): an axis-aligned rectangle containing the vertices. The Home key is an alternative trigger for this option.
- R:** Select a new polygon **R** (mouse button presses). Each button press adds a vertex, and the Enter or Escape key terminates the process, connecting the last vertex to the first. Intersecting polygon edges are not permitted. Once **R** has been created by option **R** or **B**, a vertex may be selected and dragged with the left button down, inserted by left clicking on an edge of **R**, or deleted with a shifted left mouse button press.
- t:** Toggle annotation of Triangles with indices.
- u:** Enter uv-edit mode (select vertex with mouse). Allows user- specified vertex function values to be written to the output file.
- v:** Toggle annotation of Vertices with indices.
- V:** Toggle annotation of non-removable Vertices. Non-removable vertex indices are written in a different text color to distinguish them from removable vertices when both options **v** and **V** are toggled on. By default (when the program is started), the only indices displayed are those of the non-removable vertices.
- Z/z:** Zoom in (**z**) or out (**Z**). This option, used in conjunction with one of the panning options, enables arbitrarily precise navigation. The Page Up and Page Down keys provide alternatives. The arrow keys pan by moving the center of the viewing volume. More precise motion, needed when zoomed in, is obtained with the shifted arrow keys.
- esc:** Cancel mode (**D/E/R**), or terminate program. The Escape key (or Enter key) terminates the operation associated with keys **D**, **E**, and **R**, or terminates the program if one of those modes is not in effect. A confirmation dialog box is displayed before the program is terminated. Confirmation then causes the triangle mesh to be written to the output file, and the program to be terminated.

A submenu labeled ‘Additional Options’ provides the following:

- a:** Toggle preservation of Aspect ratio. If the aspect ratio is not preserved, the triangle mesh is stretched (and distorted) to fill the window. This is useful for displaying domains with aspect ratios that differ significantly from 1.
- b:** Toggle display of the Bounding Box. This option toggles display of a wire-frame rectangle containing the mesh, and annotated with the coordinates of two diagonally opposite corners.
- H:** Toggle off-center/circumcenter in refineRd. The Delaunay refinement procedure uses off-centers by default, but can be forced to use circumcenters instead.
- i:** Triangulation Integrity check (debugging). The triangle mesh is tested for validity, and the program is terminated with an error message in the case of failure.
- j:** Toggle antialiasing (Jaggies). The ‘staircasing’ effect is smoothed out in the display of triangle edges.
- K/k:** Increase/decrease line width. By default, edges are drawn with a line width of two pixels.
- M/m:** Increase/decrease minimum angle for refineRd. The minimum angle used by the Delaunay refinement procedure may be chosen in the range 1 to 35 degrees (with 20 as the default). Note that it may be necessary to increase the maximum allowable number of Steiner points (option X) in order to achieve the minimum angle (but it may not be achievable with any number of Steiner points).
- p:** Pan (select new center with mouse button). A single left mouse button press selects the center of the viewing volume.
- w:** Select a new view volume (left mouse button). A left mouse button press selects a corner of the new viewing volume, and motion with the button down drags a diagonally opposite corner. This is equivalent to a combined pan and zoom.
- X/x:** Double/halve the maximum number of Steiner points for refineRd. Refer to the M/m option.

A submenu labeled ‘Triangulation Updates’ provides the following:

- C:** Replace non-removable vertex set by Corners. The set of non-removable vertices is replaced by the set of all corners, where a corner is taken to be a boundary vertex that does not lie on the line defined by its predecessor and successor. Three points that are close to collinear, using a small relative tolerance, are treated as lying on the same line. When the program is started, the list of non-removable vertices is initialized with the corners in the input triangle mesh.

- D:** Alter Domain (drag non-removable vertices). This option initiates a mode in which the non-removable vertices can be dragged with the left mouse button. The mode is terminated by the Enter key or Escape key.
- E:** Select Edges for swapping (left mouse button). This option initiates a mode in which selected edges may be swapped (if shared by pairs of triangles that form convex quadrilaterals). The mode is terminated by the Enter or Escape key. This manual edge swapping mode could be used to eliminate a triangle that has all three vertices on a portion of the boundary on which Dirichlet boundary conditions are imposed.
- f:** Fill concavities in **R** with triangles. A concavity occurs where three adjacent boundary vertices form a reentrant corner at a removable vertex. The first and last vertices are connected by a new boundary edge. Each invocation of this operation makes a single pass through all the boundary curves, and therefore may not fill all of the concavities.
- L:** Toggle boundary layer (smoothing method). If the boundary layer option is selected, the smoothing method is modified to create thin triangles near the boundary.
- N/n:** Convert vertices in **R** to Non-removable/removable. All boundary vertices in **R** are made non-removable (N) or removable (n).
- O:** Move non-removable vertices in **R** to circle. All non-removable vertices in **R** that can be moved without destroying the mesh are (centrally) projected onto the current circle (if any). Refer to the o option.
- s:** Optimize **R** with Delaunay edge Swaps. Up to four passes through the edges in **R** are made, swapping edges as required by the empty circumcircle criterion.
- S:** Optimize **R** by Smoothing (moving vertices). Following application of the Delaunay edge swapping optimization procedure, one smoothing pass through all the vertices in **R** is executed.
- &:** Refine **R** using Delaunay refinement.
- +:** Refine **R** by adding triangle barycenters.
- >:** Refine **R** by adding edge midpoints.
- :** Unrefine by reversing **&**, **+**, or **>** if possible. The vertices inserted by the previous unreversed refinement, if any, and assuming the refinement was not rendered non-reversible by deletion of a vertex, are deleted in the reverse of the order in which they were inserted. Since options **&** and **>** involve edge swaps, which are not recorded, unrefinement does not actually reverse the sequence of mesh alterations, and may fail to delete all of the vertices that were inserted by refinement.

<: Coarsen **R** by removing vertices. About 70% of the removable vertices in **R**, chosen at random, are deleted. All removable vertices can be deleted by several invocations of this option, intermixed with the fill (f) option — necessary to convert boundary vertices (that form reentrant corners) to interior vertices.

Backspace or Delete: Remove triangles in **R**. All removable triangles that can be removed from **R** without destroying the triangle mesh are removed, with new boundary curves created as necessary. This option can be used to create a domain with holes.

A third submenu, labeled ‘Title and Font’, contains the following three options along with nine additional options, associated with keys 0, 1, ..., 8, which select the font used for indices and the title. The most useful fonts for the indices (vertex, triangle, and edge indices) are the default 10-point characters (option 2), but sometimes it is convenient to temporarily switch to option 3 (a 24-point font) for readability. Regardless of the font, it may be necessary to zoom in when indices are too crowded.

T: Replace Text (keyboard string entry). A title (line of text with up to 80 characters) may be written to the window that contains the mesh. The text is entered in the terminal window.

l: Change title Location (left button). If a text string has been entered (option T), this option allows its location to be selected or altered by placing the mouse cursor and clicking the left button.

!: Toggle font selection: indices/title. By default, the font selection options change the font used for annotation indices. They can be applied to the title font instead.

4 Test Results

The initial motivation for the work presented here was the need for a front end to a finite element code, currently being developed, for solution of the Navier-Stokes equations in two dimensions. In fact, some of the `gmesh2` code will be used in the finite element code as part of a multigrid linear solver for the pressure-Poisson equations, with grids obtained by uniform mesh refinement using edge midpoints. (The reason for writing `gmesh2` in C is that the finite element code will be parallelized using an extension of C.) Maximum efficiency for a given fine-grid mesh width is obtained by maximizing the number of grids. The ideal initial grid is therefore as coarse as possible within the constraints of the domain geometry and the shape criterion. (A finite difference method would be very inefficient unless the domain is simple.) The Delaunay refinement method, with no size criterion specified, is perfectly suited to this problem.

We tested the Delaunay refinement method on two sample data sets. Unfortunately, the optimal value of the minimum angle is problem dependent and

will have to be determined experimentally in the context of the finite element code. In our tests the criterion defining good triangles is a minimum angle chosen to be as large as possible up to 35 degrees. The algorithm generally fails to terminate (or would fail if the number of Steiner points were not limited) for minimum angles larger than 34 degrees. A recent paper describes a Delaunay refinement method that usually terminates for minimum angles as large as 42 degrees [Erten and Üngör 2009]. Our test results confirm the substantial advantage of off-centers over circumcenters in terms of requiring fewer Steiner points (and hence fewer elements) when the minimum angle is large.

The first data set is a three-character logo surrounded by a rectangle. The triangle mesh parameter values on input are $N_c = 4$, $N_b = N_v = N_n = 70$, $N_t = 74$, and $N_e = 146$. The refined grid is depicted in Figure 1. It has minimum angle 35 degrees and parameters $N_b = 301$, $N_v = 1346$, $N_t = 2395$, and $N_e = 3743$. With minimum angle 34 degrees, the values are $N_b = 194$, $N_v = 399$, $N_t = 608$, and $N_e = 1009$. The small decrease in minimum angle resulted in a very large reduction of the mesh size.

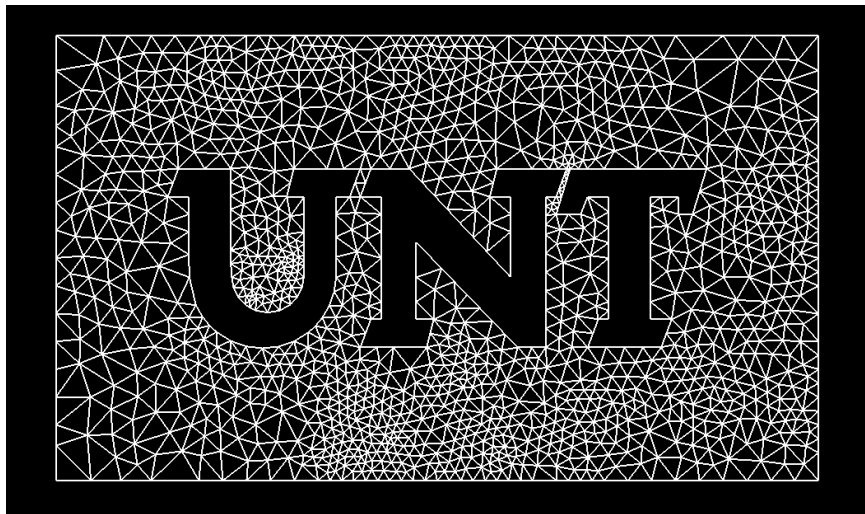


Figure 1: Logo: Minimum angle = 35 degrees, 2395 triangles.

With circumcenters used in place of off-centers, the minimum angle had to be reduced to 34 degrees, and the parameter values are $N_b = 257$, $N_v = 871$, $N_t = 1489$, and $N_e = 2362$. The use of off-centers thus reduced the number of triangles from 1489 to 608.

The second data set was created from 98 points digitized from a NACA4412 airfoil, along with four points defining a containing rectangle. The input parameter values are $N_c = 2$, $N_b = N_v = N_n = 102$, $N_t = 102$, and $N_e = 204$. Figure 2 depicts the result of Delaunay refinement using off-centers. The minimum angle is 34 degrees, and the changed mesh parameters are $N_b = 155$, $N_v = 630$,

$N_t = 1105$, and $N_e = 1735$.

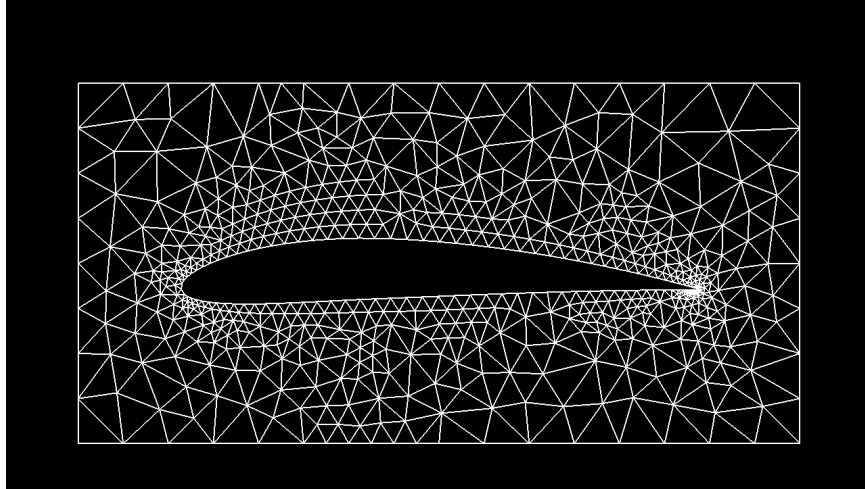


Figure 2: NACA4412 airfoil: Minimum angle = 34 degrees, 1105 triangles.

When circumcenters are used in place of off-centers, the largest minimum angle achievable is 31 degrees, and the mesh parameters are $N_b = 157$, $N_v = 874$, $N_t = 1591$, and $N_e = 2465$. With off-centers and largest minimum angle set to 31 degrees, the values are $N_b = 146$, $N_v = 538$, $N_t = 930$, and $N_e = 1468$ — a 41.5% reduction in N_t .

In order to demonstrate the domain creation capability of the code, we started with a uniform grid in a square domain, added a hole, rounded two of the corners, and applied the smoothing procedure. The result is depicted in Figure 3. The minimum angle is 17.4 degrees, and the parameter values are $N_b = 126$, $N_v = 580$, $N_n = 80$, $N_t = 1034$, and $N_e = 1614$.

The method for creating the hole consists of the following steps: 1) draw a circle (option o), 2) select a polygon **R** containing the triangles that have one or more vertices interior to the circle, and no other triangles (option R), 3) remove the triangles contained in **R** (option Backspace), 4) make the boundary vertices in **R** non-removable (option N), 5) project the non-removable vertices in **R** onto the circle (option O), 6) Set **R** to a bounding box containing the entire grid (option B), and 7) smooth the grid (option S).

References

- [Bank 1990] Bank, Randolph E. 1990. PLTMG: A Software Package for Solving Elliptic Partial Differential Equations, User's Guide 6.0, Society for Industrial and Applied Mathematics, Philadelphia, PA.

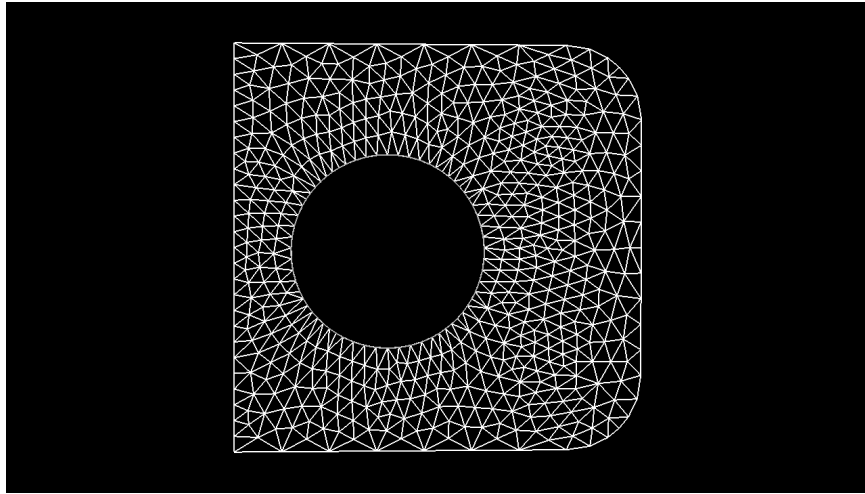


Figure 3: Square domain with hole and rounded corners, 1034 triangles.

- [Chen 2004] Chen, L. 2004. Mesh smoothing schemes based on optimal Delaunay triangulations. In Proceedings of 13th International Meshing Roundtable, 109–120.
- [Cline and Renka 1984] Cline, A. R., and Renka, R. J. 1984. A storage-efficient method for construction of a Thiessen triangulation. *Rocky Mt. J. Math.* 14, 1, 119–139.
- [Erten and Üngör 2009] Erten, H. and Üngör, A. 2009. Quality triangulations with locally optimal Steiner points. *SIAM J. Sci. Comput.* 31, 3, 2103–2130.
- [Kilgard 1996] Kilgard, M. 1996. *OpenGL Programming for the X Window System*, Addison-Wesley, Reading, MA.
- [Lawson 1977] Lawson, C. L. 1977. Software for C^1 surface interpolation. In *Mathematical Software III*, J. R. Rice, Ed. Academic Press, New York, 161–194.
- [OpenGL 1997] OpenGL Architecture Review Board. 1997. *OpenGL Programming Guide*, Addison-Wesley, Reading, MA.
- [Rupert 1995] Rupert, J. 1995. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms* 18, 3, 548–585.
- [Shewchuk 2002a] Shewchuk, J. R. 2002. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications* 22 ,1-3, 21–74.

- [Shewchuk 2002b] Shewchuk, J. R. 2002. What is a Good Linear Element? Interpolation, Conditioning, and Quality Measures. In *Proceedings, 11th International Meshing Roundtable* (September), 115–126.
- [Üngör 2009] Alper Üngör, Off-centers: a new type of Steiner points for computing size-optimal quality-guaranteed Delaunay triangulations. *Computational Geometry: Theory and Applications* 42, 2 (February), 109–118.