NASA-TM-103,979

NASA Technical Memorandum 103979

NASA-TM-103979 19940028440

Incremental Triangulation via Edge Swapping and Local Optimization

N. Lyn Wiltberger

January 1994

National Aeronautics and
Space Administration

NASA Technical Memorandum 103979

Incremental Triangulation via Edge Swapping and Local Optimization

N. Lyn Wiltberger, Ames Research Center, Moffett Field, California

January 1994

National Aeronautics and Space Administration

Ames Research Center
Moffett Field, California 94035-1000

TABLE OF CONTENTS

Page

Incremental Triangulation via Edge Swapping and Local Optimization

N. Lyn Wiltberger
Ames Research Center

1 GETTING STARTED

1.1 INTRODUCTION

This document is intended to serve as an installation, usage, and basic theory guide for the two-dimensional triangulation software "HARLEY" written for the Silicon Graphics IRIS workstation. This code consists of an incremental triangulation algorithm based on point insertion and local edge swapping. Using this basic strategy, several types of triangulations can be produced depending on user selected options. For example, local edge swapping criteria can be chosen which minimizes the maximum interior angle (a MinMax triangulation), or which maximizes the minimum interior angle (a MaxMin or Delaunay triangulation). It should be noted that the MinMax triangulation is generally only locally optimal (not globally optimal) in this measure. The MaxMin triangulation, however, is both locally and globally optimal. In addition, Steiner triangulations can be constructed by inserting new sites at triangle circumcenters followed by edge swapping based on the MaxMin criteria. Incremental insertion of sites also provides flexibility in choosing cell refinement criteria. For instance, by choosing a refinement measure based on solution error, the code can be utilized for solution adaptive grid generation. A dynamic heap structure has been implemented in the code so that once a refinement measure is specified (i.e. maximum aspect ratio or some measure of a solution gradient for the solution adaptive grid generation) the cell with the largest value of this measure is continually removed from the top of the heap and refined. The heap refinement strategy allows the user to specify either the number of cells desired or refine the mesh until all cell refinement measures satisfy a user specified tolerance level. Since the dynamic heap structure is constantly updated, the algorithm always refines the particular cell in the mesh with the largest refinement criteria value. The code allows the user to:

1) triangulate a cloud of prespecified points (sites).
2) triangulate a set of prespecified interior points constrained by prespecified boundary curve(s).
3) Steiner triangulate the interior/exterior of prespecified boundary curve(s).
4) refine existing triangulations based on solution error measures.
5) partition meshes based on the Cuthill-McKee, Spectral, and Coordinate bisection strategies.

1.2 GETTING SET UP

The code, examples, and documentation are shipped via shell archive. Assuming that the archive is correctly installed, subdirectories will be created which contain the source code (/src), executable (/bin), examples (/examples), and documentation (/documentation). To display the command line options available simply type the program name:

```
IRIS.HOST> harley
 harley usage:
     -i [-I]    input point file name              <default:none>
     -o [-O]    outfile grid file                  <default:grid.dat>
     -ig [-IG]  input grid file (-a active)        <default:none>
     -is [-IS]  input solution file (-a active)    <default:none>
     -os [-OS]  output solution file (-a active) <default:soln.new>
     -minmax  [-MINMAX]                            <default:maxmin>
     -p       [-partition]                         <default:false>
     -a       [-adapt]     (-ig -is required)      <default:false>
     -r ival  [-refine ival]                       <default:ival=6>
     ival = 1  -> Velocity Sobolev measure
     ival = 2  -> Velocity Divided Difference
     ival = 3  -> Pressure Sobolev measure
     ival = 4  -> Pressure Divided Difference
     ival = 5  -> Density Sobolev measure
     ival = 6  -> Density Divided Difference
     ival = 7  -> Entropy Sobolev measure
     ival = 8  -> Entropy Divided Difference
     ival = 9  -> Stag Enthalpy Sobolev measure
     ival =10  -> Stag Enthalpy Divided Difference
```

Input Flags

In general, the code requires one or more input files in order to generate a triangulation. Command line flags for possible input files begin with the letter i, i.e. -i -ig -is.

-i This flag precedes input file names containing point data such as interior sites and bounding curves (specified as a collection of contiguous points).

-ig This flag precedes input file names containing initial grid data for use with solution adaptive meshing. This option must be used in conjunction with the -a and -is flags.

-is This flag precedes input file names containing initial solution data for use with solution adaptive meshing. This option must be used in conjunction with the -a and -ig flags.

Output Flags

In general, the code produces one or more output files upon completion of a triangulation. Command line flags for possible output files begin with the letter o, i.e. -o -os.

-o This flag precedes output file names to which the final grid data will be written. A default file name of "grid.dat" will be used if this flag is not specified.

-os This flag precedes output file names to which the interpolated solution data will be written in the case of solution adaptive meshing. The newly interpolated solution data file corresponds to the final grid generated by the code. A default file name of "soln.new" will be used if this flag is not specified. This option is used in conjunction with the -a, -ig, and -is flags.

Triangulation Flags

Two different types of triangulations (MaxMin or MinMax) can be constructed depending on the choice of edge swapping criteria. The default edge swapping criteria used in the code produces a MaxMin (Delaunay) triangulation.

`-minmax` The invocation of this flag results in a locally MinMax triangulation.

Partitioning Flag

`-p` This flag enables partitioning of the newly generated mesh.

Solution Adaptive Meshing Flags

`-a` This flag enables solution adaptive meshing and must be specified in conjunction with the `-ig`, `-is`, and `-r ival` flags.

`-r ival` The `-r` flag in conjunction with the `ival` parameter specifies the refinement measure used during the solution adaptive meshing. The default value of `ival` is 6.

User Prompted Input

In addition to command line flags, the program also requires other types of user response during the course of a triangulation. User input is required when the graphics window is active. (The graphics window is usually active after the terminal bell sounds and the message "HIT ESC TO EXIT" appears in the text window.) When the graphics window is active, the graphics object can be manipulated via the mouse. To exit this graphics manipulation mode and return to the text window, hit the escape key found on the terminal keyboard. The second type of user input involves ASCII data input, i.e. the number of cells desired, refinement tolerances, smoothing parameters, etc. Entering a "/" from the terminal allows the program to proceed using default values. In most cases this effectively bypasses the relevant option. One or more of the following user queries may appear during a typical triangulation:

`Enter max AR and max cells:` These parameters control the degree of mesh refinement in Steiner triangulations. The user can specify the largest cell aspect ratio allowable and/or the maximum number of cells generated. The refinement terminates when one of the two constraints is satisfied. For example, entering "0.0, 5000" would generate a mesh with 5000 cells.

`Enter max error and max cells:` These parameters control the degree of mesh refinement in solution adaptive triangulations. The user can specify the largest cell refinement measure allowable and/or the maximum number of cells generated. The refinement terminates when one of the two constraints is satisfied.

`Enter power [0,1]:` This parameter controls the length scaling used in the solution adaptation refinement measure. See later discussions for further details on the adaptation measures.

`Enter smoothing coefficient [0,1]:` This parameter controls the amount of smoothing performed on a mesh. A value of 1 allows maximum mesh smoothing; a value of 0 results in no mesh movement.

`Enter # of smoothing iterations:` This integer parameter controls the number of smoothing iterations applied. The default value of this parameter is 0.

`Enter dimension of coloring:` This integer parameter controls the number of mesh partitions generated in the graph partitioning option. The dimension refers to the hypercube dimension which means that the number of partitions generated is 2^{dim}.

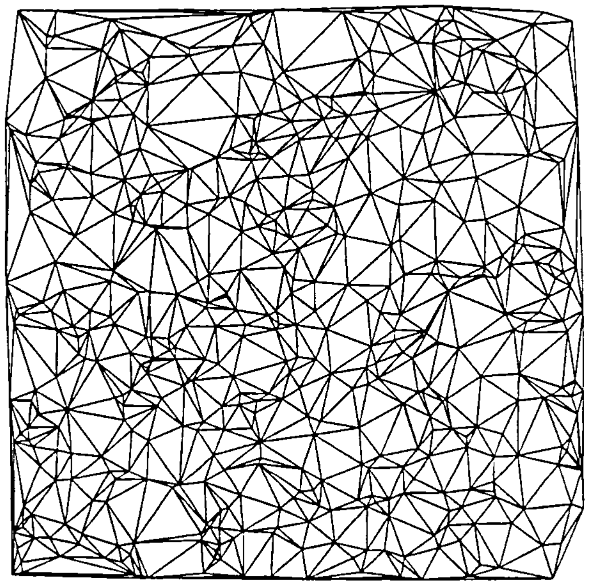
`Enter partitioning type`: This parameter selects the type of graph partitioning used. Several options are available as discussed in section 2.2.

1.3 CASE EXAMPLES

In this section, several case examples are given to illustrate typical usage of the HARLEY software. The input files necessary to recreate these examples can be found in the /examples directory created as part of the software installation.

1.3.1 Case 1: Triangulation of a point cloud

Triangulate a specified cloud of points. Given a set of points, form the triangulation of the points with no specification of bounding curves (i.e. the outer boundary). For the Delaunay triangulation, the resulting outer boundary formed from the triangulation coincides with the convex hull of the point set. No additional points will be added. Local edge swapping is determined either by the the MaxMin or MinMax criteria as selected by the user. See Section 3.2.1 File Formats - Case 1 for input file format.



Case 1: Delaunay triangulation of random points

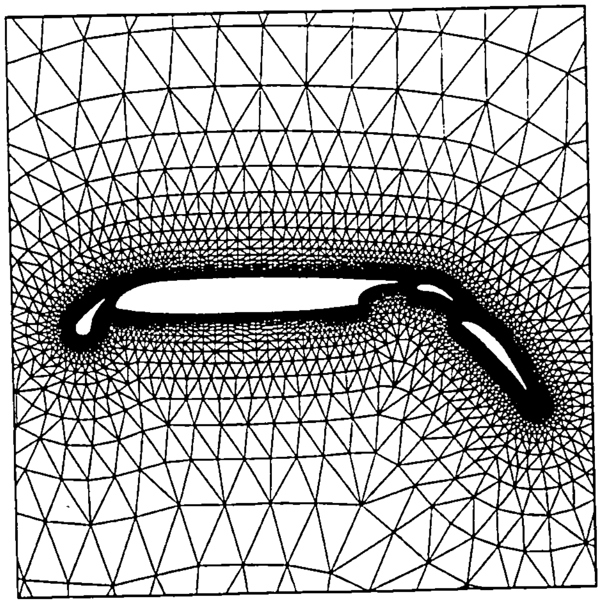
Screen output:

```
IRIS.HOST> harley -i case1.input -o case1.grid
 You have chosen a Max-Min triangulation

 Reading curve            1 Number of points          500
 HIT ESC TO EXIT
 HIT ESC TO EXIT
 HIT ESC TO EXIT
 Number of cells        979
 Number of edges       1478
 Number of unreachable cells was:    0
 Grid write complete
IRIS.HOST>
```

1.3.2 Case 2: Constrained triangulation of a point cloud with prespecified bounding curves

Triangulate a prespecified set of interior sites constrained by prespecified boundary curve(s). Given the boundary curves, represented by contiguous points, an initial triangulation of the boundary curves is formed. The remaining interior sites are then inserted and local edge swapping occurs depending on whether a MinMax or MaxMin triangulation is desired. Typically the MinMax triangulation will produce better results for meshes containing highly stretched cells. (See section 2.1.3.2 for further discussion.) The option exists for additional interior points to be inserted after the initial specified points have been inserted. In the case of additional point insertion, the user may specify either the largest aspect ratio cell desired or the number of total cells desired. Additional points will be inserted at the cell circumcenters. Refinement occurs in the cell with the largest aspect ratio as determined by the dynamic heap structure . A "Poor Man's" Laplacian smoothing is also available for postprocessing of triangulations. See section 3.2.2 File Formats - Case 2 for input file format.

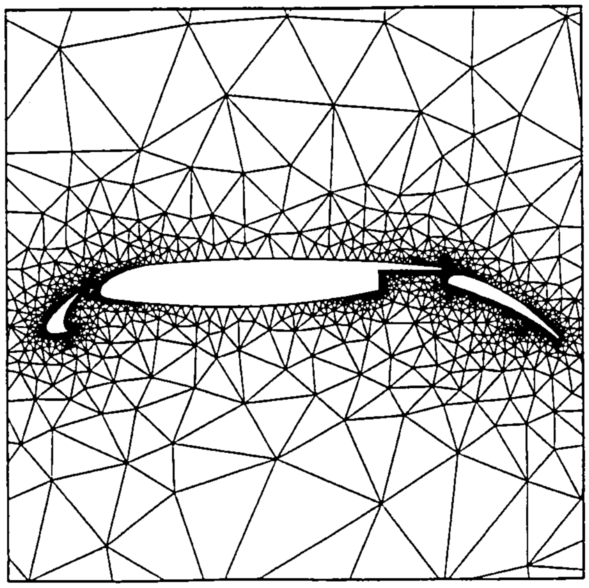


Case 2: Constrained triangulation, prespecified points and curves
Screen output:

```
IRIS.HOST> harley -i case2.input -o case2.grid -minmax
 You have chosen a Min-Max triangulation
 Reading curve            1 Number of points          160
 Reading curve            2 Number of points          200
 Reading curve            3 Number of points          160
 Reading curve            4 Number of points          160
 Reading curve            5 Number of points           80
 read interior pts, number of pts          16274
 HIT ESC TO EXIT
 HIT ESC TO EXIT
 HIT ESC TO EXIT
 Number of cells so far:        33314
 Maximum cell aspect ratio:    951.0345690283474
 Enter max AR and max cells: /
 -------------------------# of Points, # of cells,  Max Aspect Ratio-
 Just made the final pass       17034       33314   951.0345690283474
 HIT ESC TO EXIT
 Enter smoothing coefficient [0,1]: /
 Enter # of smoothing iterations: /
 HIT ESC TO EXIT
 Number of cells       33314
 Number of edges       50351
 Number of unreachable cells was:    0
 Grid write complete
IRIS.HOST>
```

1.3.3 Case 3: Steiner triangulation with prespecified boundary curves

Steiner triangulate the interior/exterior of specified boundary curve(s). In this case, an initial triangulation is constructed from the boundary curve(s), which are represented as contiguous points. Steiner points are inserted into the interior of the outer boundary curve (the last curve read in) and to the exterior of the inner boundary curve(s). New cells are generated by refining the cell with the largest aspect ratio, as determined by the dynamic heap structure. Each new point is placed at the circumcenter of the cell which is being refined. The MinMax or MaxMin options may be used to determine if local edge swapping occurs after a new point is inserted. The user can specify either the maximum aspect ratio desired or the specific number of cells desired. A "Poor Man's" Laplacian smoothing is also available for postprocessing of triangulations. See section 3.2.3 File Formats - Case 3 for input file format.



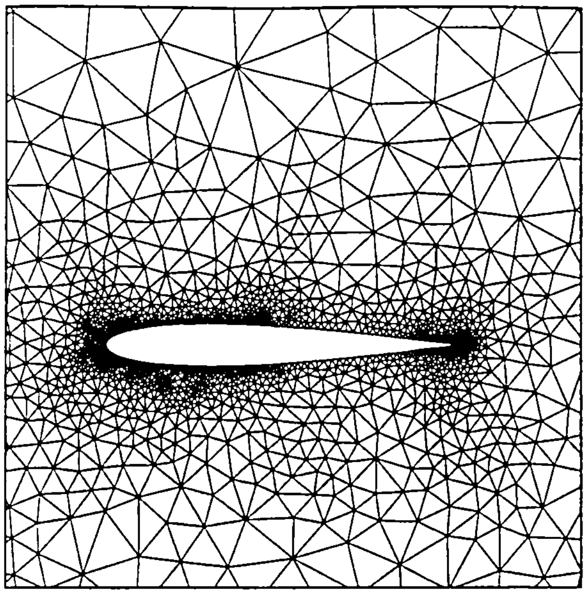
Case 3: Steiner triangulation with prespecified boundary curves

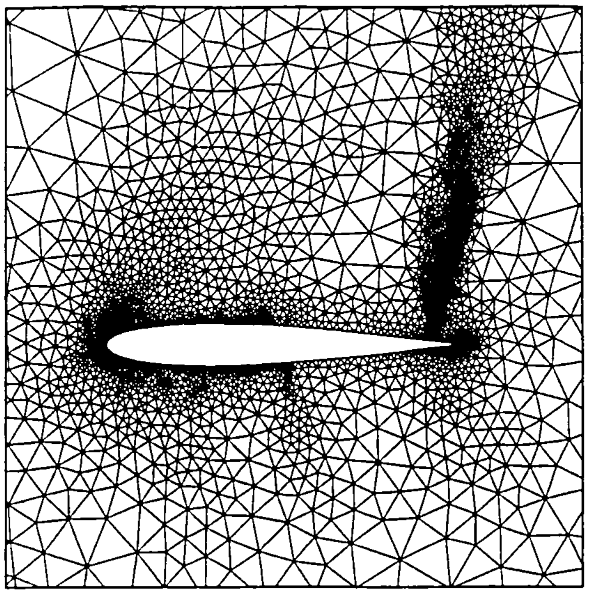
Screen output:

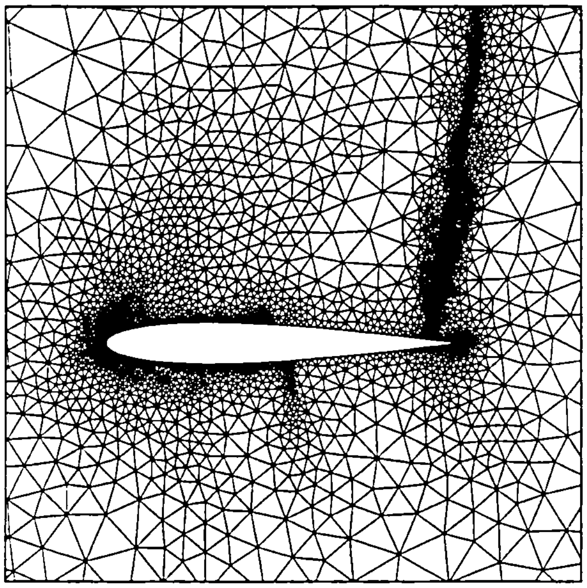
```
IRIS.HOST> harley -i case3.input -o case3.grid
 You have chosen a Max-Min triangulation
 Reading curve              1 Number of points           180
 Reading curve              2 Number of points           240
 Reading curve              3 Number of points           180
 Reading curve              4 Number of points            80
 HIT ESC TO EXIT
 HIT ESC TO EXIT
 Number of cells so far:          684
 Maximum cell aspect ratio:    818911.8266297410
 Enter max AR and max cells: 1.5 5000
 -------------------------# of Points, # of cells,  Max Aspect Ratio-
 Just made another pass          959        1242     5.452872734359350
 Just made another pass         1847        3018     1.836270018454983
 Just made another pass         2229        3782     1.742575760831057
 Just made another pass         2677        4678     1.785552596845508
 Just made the final pass       2838        5000     1.607958714670921
 HIT ESC TO EXIT
 Enter smoothing coefficient [0,1]: /
 Enter # of smoothing iterations: /
 HIT ESC TO EXIT
 Number of cells          5000
 Number of edges          7840
 Number of unreachable cells was:     0
 Grid write complete
IRIS.HOST>
```

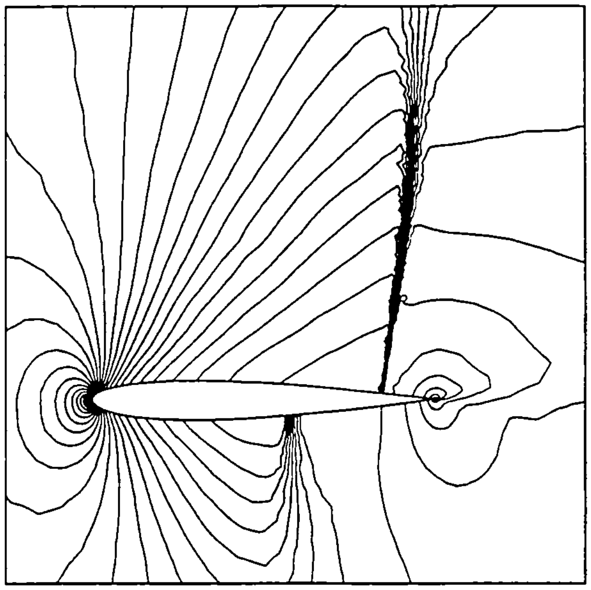
1.3.4 Case 4: Solution adaptive triangulation

Refine existing triangulations based on solution error measures. In this case, initial mesh and solution files are required. Steiner points are inserted into cells in the interior of the mesh. Cells are chosen for refinement based on a measure of solution error. The MinMax or MaxMin options may be used to determine what type of local edge swapping occurs after a point is inserted. The user can specify either the maximum solution error desired or the specific number of cells generated. Cells are refined until one of the two criteria is satisfied. After the newly adapted mesh is generated, the code will also output a new solution file with interpolated values at the new mesh points. Refinement measures are calculated in the function refine_measure. The user can easily modify this function for other possible error measures. A "Poor Man's" Laplacian smoothing is also available for postprocessing of triangulations. See Section 3.2.4 File Formats - Case 4 for input file format.


Case 4 (a) initial grid


Case 4 (b) grid after 1 cycle


Case 4 (c) grid after 2 cycles


Case 4 (d) solution after 2 cycles

Screen output:

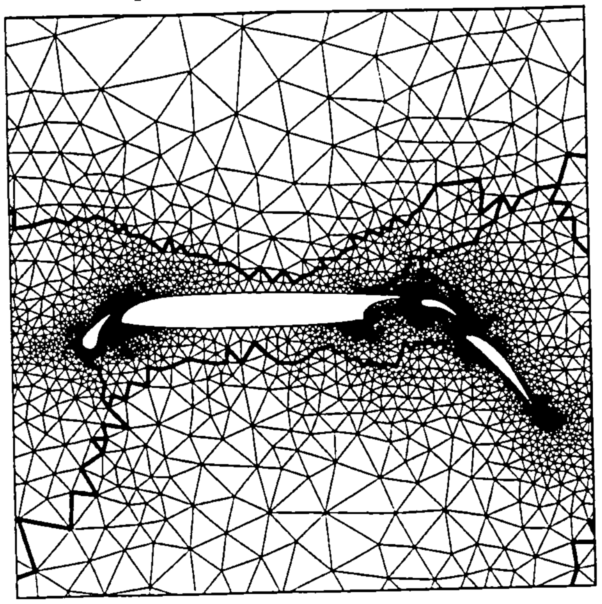
```
    IRIS.HOST>harley -ig case4.grid.1 -is q.1 -og case4.grid.2 -os q.1.new -a -r 6
     Refinement type =            6  -> Density Divided Difference
     You have chosen a Max-Min triangulation
     Begin grid read
     Number of cells        =        5000
     Number of nodes        =        2655
     Number of edges        =        7655
     Number of bc edges     =         310
     Number of curves       =           2
     Grid read complete
     Solution read complete
```

```
Freestream Mach Number =   0.8500000
HIT ESC TO EXIT
Divided difference refinement measure was chosen
The form of the refinement measure in each cell is:
error measure = max(function divided difference)*(length)^power
length = sqrt(cell area)
Enter power[0,1]: .05
Number of cells so far:          5000
Maximum error   0.2706142854949063
Enter max error and/or max cells: 0 10000
-------------------------# of Points, # of cells, Max Refinement Value--
Just made another pass          2756        5200  0.1078620972206629
Just made another pass          2957        5600  6.8420049508985227E-02
Just made another pass          3158        6000  5.3211269196278507E-02
Just made another pass          3360        6400  4.6488102699898110E-02
Just made another pass          3560        6800  4.1857333467829391E-02
Just made another pass          3760        7200  3.8550165709022268E-02
Just made another pass          3960        7600  3.5883312867324539E-02
Just made another pass          4160        8000  3.3396920241482246E-02
Just made another pass          4361        8400  3.7104946134550842E-02
Just made another pass          4561        8800  3.0038342055240216E-02
Just made another pass          4761        9200  3.6224232731339447E-02
Just made the final pass        5162        9999  2.6390731462883296E-02
HIT ESC TO EXIT
Enter smoothing coefficient [0,1]: .9
Enter # of smoothing iterations: 3
HIT ESC TO EXIT
HIT ESC TO EXIT
HIT ESC TO EXIT
HIT ESC TO EXIT
Number of cells        9999
Number of edges       15161
Number of unreachable cells was:     0
Grid write complete
Solution write complete
IRIS.HOST>
IRIS.HOST>harley -ig case4.grid.2 -is q.2 -og case4.grid.3 -os q.2.new -a -r 6
 Refinement type =             6  -> Density Divided Difference
 You have chosen a Max-Min triangulation
 Begin grid read
 Number of cells          =        9999
 Number of nodes          =        5162
 Number of edges          =       15161
 Number of bc edges       =         325
 Number of curves         =           2
 Grid read complete
 Solution read complete
```

```
Freestream Mach Number =   0.8500000
HIT ESC TO EXIT
Divided difference refinement measure was chosen
The form of the refinement measure in each cell is:
error measure = max(function divided difference)*(length)^power
length = sqrt(cell area)
Enter power[0,1]: .2
Number of cells so far:          9999
Maximum error   0.1266147313334715
Enter max error and/or max cells: 0 15000
-------------------------# of Points, # of cells, Max Refinement Value--
Just made another pass          5363       10400  7.0304236197733809E-02
Just made another pass          5563       10800  5.4593397958585754E-02
Just made another pass          6564       12800  3.1269508738813693E-02
Just made another pass          6764       13200  2.8677292377648551E-02
Just made another pass          6965       13600  2.6975830698469089E-02
Just made another pass          7165       14000  2.5359550981042922E-02
Just made another pass          7365       14400  2.4198127105300599E-02
Just made the final pass        7665       14999  2.2304001575539640E-02
HIT ESC TO EXIT
Enter smoothing coefficient [0,1]: .8
Enter # of smoothing iterations: 3
HIT ESC TO EXIT
HIT ESC TO EXIT
HIT ESC TO EXIT
HIT ESC TO EXIT
Number of cells        14999
Number of edges        22664
Number of unreachable cells was:     0
Grid write complete
Solution write complete
IRIS.HOST>
```

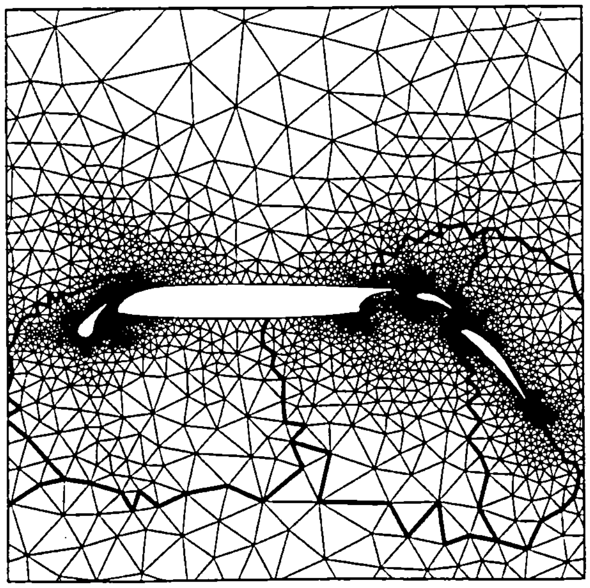
1.3.5 Case 5: Steiner triangulation with graph partitioning

Steiner triangulate the interior/exterior of specified boundary curve(s) and partition the resulting mesh. In this case, an initial triangulation is constructed from the boundary curve(s), which are represented as contiguous points. Steiner points are inserted into the interior of the outer boundary curve (the last curve read in) and to the exterior of the inner boundary curves. New cells are generated by refining the cell with the largest aspect ratio, as determined by the dynamic heap structure. Each new point is placed at the circumcenter of the cell which is being refined. The MinMax or MaxMin options may be used to determine if local edge swapping occurs after a new point is inserted. The user can specify either the maximum aspect ratio desired or the specific number of cells desired. A "Poor Man's" Laplacian smoothing is also available for postprocessing of triangulations. The resulting mesh is partitioned using one of the available partitioning methods (Cuthill-McKee, Spectral, and Coordinate bisection). The user is queried for the hypercube dimension; which means that the number of partitions generated is 2^{dim}. The corresponding theory for the partitioning strategies can be found in section 2.2. See section 3.2.3 File Formats - Case 3 for input file format.



Case 5: Mesh Partitioning - Cuthill-McKee (8 partitions).
Screen output:

```
IRIS.HOST>harley -i case5.input -o case5.cm.grid -p
 You have chosen a Max-Min triangulation
 Reading curve            1 Number of points        160
 Reading curve            2 Number of points        200
 Reading curve            3 Number of points        160
 Reading curve            4 Number of points        160
 Reading curve            5 Number of points         80
 HIT ESC TO EXIT
 HIT ESC TO EXIT
 Number of cells so far:          766
 Maximum cell aspect ratio:    5758719.974066261
 Enter max AR and max cells: 0 12000
 -------------------------# of Points, # of cells,  Max Aspect Ratio-
 Just made another pass        1097        1440    4.863169589305104
 Just made another pass        1323        1892    3.064135153182815
 Just made another pass        2600        4446    1.791479553061699
 Just made another pass        2661        4568    2.552360505432250
 Just made another pass        2894        5034    1.626761070576061
 Just made another pass        3048        5342    2.487128429433772
 Just made another pass        3301        5848    1.786355839666233
 Just made another pass        3302        5850    1.550959518336859
 Just made another pass        3395        6036    2.447001889739652
 Just made another pass          3396        6038    1.537563800683395
 Just made another pass          4205        7656    1.797896940612256
 Just made another pass          4252        7750    1.488079533428099
 Just made another pass          4326        7898    1.823219088351195
 Just made another pass          4776        8798    1.635043888200062
 Just made another pass          4777        8800    1.552025883011375
 Just made another pass          4799        8844    2.793676636319880
 Just made another pass          5209        9664    3.401915281402041
 Just made the final pass        6377       12000    1.478897132303701
 HIT ESC TO EXIT
 Enter smoothing coefficient [0,1]: .1
 Enter # of smoothing iterations: 3
 HIT ESC TO EXIT
 HIT ESC TO EXIT
 HIT ESC TO EXIT
 HIT ESC TO EXIT
 Number of cells       12000
 Number of edges       18380
 Number of unreachable cells was:     0
 Grid write complete
 Enter dimension of coloring: 3
  1 :   Coordinate Bisection
  2 :   Cuthill-McKee
  3 :   Spectral
 Enter partitioning type: 2
 HIT ESC TO EXIT
 Partitioning write complete
IRIS.HOST>
```



Case 5 Mesh Partitioning - Spectral (8 partitions).

Screen output:

```
IRIS.HOST>harley -i case5.input -o case5.sp.grid -p
 You have chosen a Max-Min triangulation
 Reading curve               1 Number of points        160
 Reading curve               2 Number of points        200
 Reading curve               3 Number of points        160
 Reading curve               4 Number of points        160
 Reading curve               5 Number of points         80
 HIT ESC TO EXIT
 HIT ESC TO EXIT
 Number of cells so far:          766
 Maximum cell aspect ratio:    5758719.974066261
 Enter max AR and max cells: 0 12000
 -------------------------# of Points, # of cells,  Max Aspect Ratio-
 Just made another pass          1097        1440   4.863169589305104
 Just made another pass          1323        1892   3.064135153182815
 Just made another pass          2600        4446   1.791479553061699
 Just made another pass          2661        4568   2.552360505432250
 Just made another pass          2894        5034   1.626761070576061
 Just made another pass          3048        5342   2.487128429433772
 Just made another pass          3301        5848   1.786355839666233
 Just made another pass          3302        5850   1.550959518336859
 Just made another pass          3395        6036   2.447001889739652
 Just made another pass          3396        6038   1.537563800683395
 Just made another pass          4205        7656   1.797896940612256
 Just made another pass          4252        7750   1.488079533428099
 Just made another pass          4326        7898   1.823219088351195
 Just made another pass          4776        8798   1.635043888200062
 Just made another pass          4777        8800   1.552025883011375
 Just made another pass          4799        8844   2.793676636319880
 Just made another pass          5209        9664   3.401915281402041
 Just made the final pass        6377       12000   1.478897132303701
 HIT ESC TO EXIT
 Enter smoothing coefficient [0,1]: .1
 Enter # of smoothing iterations: 3
 HIT ESC TO EXIT
 HIT ESC TO EXIT
 HIT ESC TO EXIT
 HIT ESC TO EXIT
 Number of cells       12000
 Number of edges       18380
 Number of unreachable cells was:     0
 Grid write complete
 Enter dimension of coloring: 3
  1 :   Coordinate Bisection
  2 :   Cuthill-McKee
  3 :   Spectral
 Enter partitioning type: 3
  number of partitions =           8
  Fiedler  Value =     -1.67828261852264E+00
  Fiedler  Value =     -1.74138307571411E+00
  Fiedler  Value =     -1.62928771972656E+00
  Fiedler  Value =     -1.74765408039093E+00
  Fiedler  Value =     -1.71586239337921E+00
  Fiedler  Value =     -1.59011912345886E+00
  Fiedler  Value =     -1.79067003726959E+00
 HIT ESC TO EXIT
 Partitioning write complete
IRIS.HOST>
```

2 SOME THEORY

In this section, we give a brief discussion of the relevant theory associated with Delaunay and MinMax triangulations. We also discuss the incremental insertion algorithm from which the HARLEY triangulation code is constructed. Finally, we briefly mention the partitioning problem and the partitioning algorithms included in the HARLEY code.

2.1 TRIANGULATION METHODS

Although many algorithms exist for triangulating sites (points) in an arbitrary number of space dimensions, only a few have been used on practical problems. In particular, Delaunay triangulation has proven to be a very useful triangulation technique. This section will present some of the basic concepts surrounding Delaunay and related triangulations.

2.1.1 Voronoi Diagram and Delaunay Triangulation

Recall the definition of the Dirichlet tessellation in a plane. The Dirichlet tessellation of a point set is the pattern of convex regions, each being closer to some point P in the point set than to any other point in the set. These Dirichlet regions are also called Voronoi regions. The edges of Voronoi polygons comprise the Voronoi diagram, see figure 2.1.1. The idea extends naturally to higher dimensions.

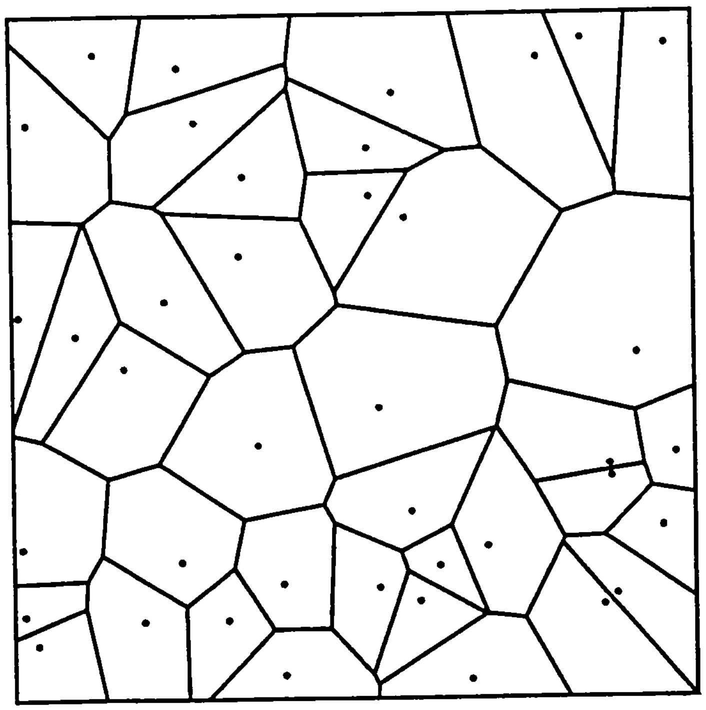


Figure 2.1.1 Voronoi diagram of 40 random sites.

Voronoi diagrams have a rich mathematical theory. *The Voronoi diagram is believed to be one of the most fundamental constructs defined by discrete data.* Voronoi diagrams have been independently discovered in a wide variety of disciplines. Computational geometricians have a keen interest in Voronoi diagrams. It is well known that Voronoi diagrams are related to convex hulls via stereographic projection. Point location in a Voronoi diagram can be performed in $O(\log(n))$ time with $O(n)$ storage for n regions. This is useful in solving post-office or related problems in optimal time. Another example of the Voronoi diagram which occurs in the natural sciences can be visualized by placing crystal "seeds" at random sites in 3-space. Let the crystals grow at the same rate in all directions. When two crystals collide simply stop their growth. The crystal formed for each site would represent that volume of space which is closer to that site than to any other site. This would effectively construct a Voronoi diagram. We now consider the role of Voronoi diagrams in Delaunay triangulation.

Definition: The Delaunay triangulation of a point set is defined as the dual of the Voronoi diagram of the set.

The Delaunay triangulation in two space dimensions is formed by connecting two points if and only if their Voronoi regions have a common border segment. If no four or more points are cocircular, then we have that *vertices of the Voronoi are circumcenters of the triangles.* This is true because vertices of the Voronoi represent locations that are equidistant to three (or more) sites. Also note that from the definition of duality, edges of the Voronoi are in one-to-one correspondence to edges of the Delaunay triangulation (ignoring boundaries). Because edges of the Voronoi diagram are the locus of points equidistant to two sites, each edge of the Voronoi diagram is perpendicular to the corresponding edge of the Delaunay triangulation. This duality extends to three dimensions in a straightforward way. The Delaunay triangulation possesses several alternate characterizations and many properties of importance. Unfortunately, not all of the two dimensional characterizations have three-dimensional extensions. To avoid confusion, properties and algorithms for construction of two dimensional Delaunay triangulations will be considered here.

2.1.2 Properties of a 2-D Delaunay Triangulation

(1) *Uniqueness.* The Delaunay triangulation is unique. This assumes that no four sites are cocircular. The uniqueness follows from the uniqueness of the Dirichlet tessellation.

(2) *The circumcircle criteria.* A triangulation of $N \geq 2$ sites is Delaunay if and only if the circumcircle of every interior triangle is point-free. For if this was not true, the Voronoi regions associated with the dual would not be convex and the Dirichlet tessellation would be invalid. Related to the circumcircle criteria is the *incircle* test for four points as shown in figures 2.1.2(a)-2.1.2(b)

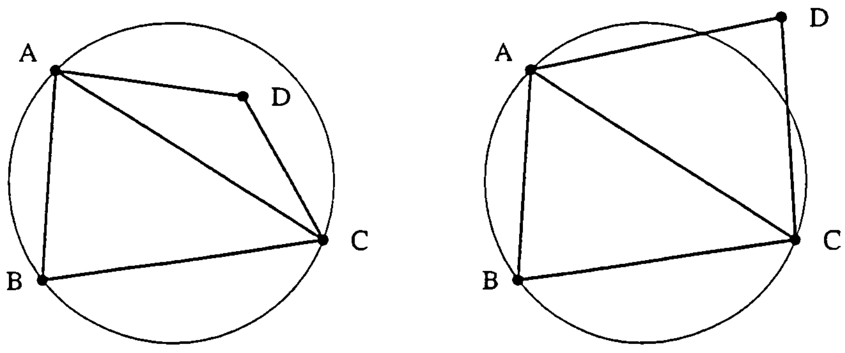


Figure 2.1.2 Incircle test for $\triangle ABC$ and D. **(left)** (true), **(right)** (false).

This test is true if point D lies interior to the circumcircle of $\triangle ABC$ which is equivalent to testing whether $\angle ABC + \angle CDA$ is less than or greater than $\angle BCD + \angle BAD$. More precisely we have that

$$\angle ABC + \angle CDA = \begin{cases} < 180^\circ & \text{incircle false} \\ \;\; 180^\circ & \text{A,B,C,D cocircular} \\ > 180^\circ & \text{incircle true} \end{cases}$$

Since interior angles of the quadrilateral sum to 360°, if the circumcircle of $\triangle ABC$ contains D then swapping the diagonal edge from position $A - C$ into $B - D$ guarantees that the new triangle pair satisfies the circumcircle criteria. Furthermore, this process of diagonal swapping is local, i.e. it does not disrupt the Delaunayhood of any triangles adjacent to the quadrilateral.

(3)*Edge circle property.* A triangulation of sites is Delaunay if and only if there exists *some* circle passing through the endpoints of each and every edge which is point-free. This characterization is very useful because it also provides a mechanism for defining a *constrained* Delaunay triangulation where certain edges are prescribed *a priori*. A triangulation of sites is a constrained Delaunay triangulation if for each and every edge of the mesh there exists some circle passing through its endpoints containing no other site in the triangulation which is *visible* to the edge. In figure 2.1.2(c), site d is not visible to the segment a-c because of the constrained edge a-b.

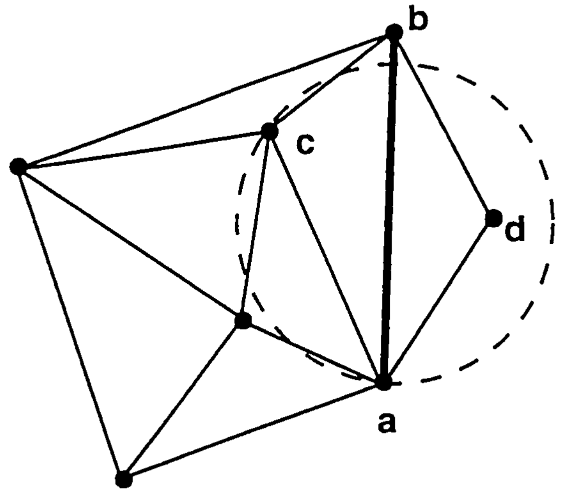


Figure 2.1.2(c) Constrained Delaunay triangulation. Site d is not visible to a-c due to constrained segment a-b.

(4)*Equiangularity property.* Delaunay triangulation maximizes the minimum angle of the triangulation. For this reason Delaunay triangulation is often called the MaxMin triangulation. This property is also locally true for all adjacent triangle pairs which form a convex quadrilateral. This is the basis for the local edge swapping algorithm of Lawson [1].

(5)*Minimum Containment Circle.* A recent result by Rajan [2] shows that the Delaunay triangulation minimizes the maximum containment circle over the entire triangulation. The containment circle is defined as the smallest circle enclosing the three vertices of a triangle. This is identical to the circumcircle for acute triangles and a circle with diameter equal to the longest side of the triangle for obtuse triangles (see figure 2.1.2(d)).

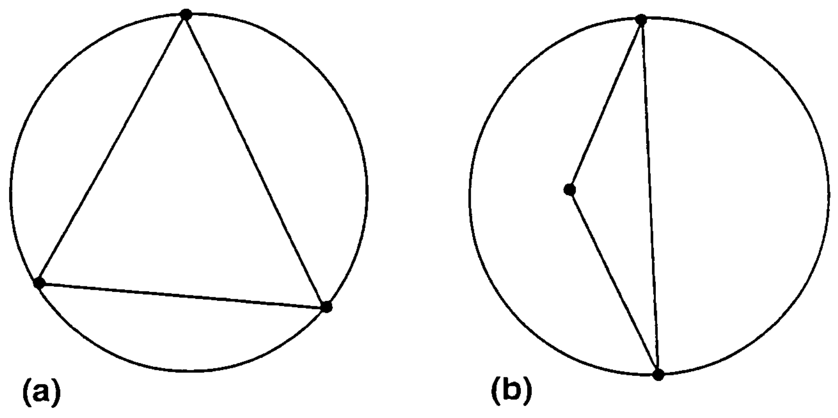


Figure 2.1.2(d) Containment circles for acute (a) and obtuse (b) triangles.

This property extends to n dimensions. Unfortunately, the result does not hold lexicographically.

(6)*Nearest neighbor property.* An edge formed by joining a vertex to its nearest neighbor is an edge of the Delaunay triangulation. This property makes Delaunay triangulation a powerful tool in solving the closest proximity problem. Note that the nearest neighbor edges do not describe all edges of the Delaunay triangulation.

(7)*Minimal roughness.* The Delaunay triangulation is a minimal roughness triangulation for arbitrary sets of scattered data, Rippa [3]. Given arbitrary data f_i at all vertices of the mesh and a triangulation of these points, a unique piecewise linear interpolating surface can be constructed. The Delaunay triangulation has the property that of all triangulations it minimizes the roughness of this surface as measured by the following Sobolev semi-norm:

$$\int_T \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2 \right] dx\, dy$$

This is a interesting result as it does not depend on the actual form of the data. This also indicates that Delaunay triangulation approximates well those functions which minimize this Sobolev norm. One example would be the harmonic functions satisfying Laplace's equation with suitable boundary conditions which minimize exactly this norm. In a later section, we will prove that a Delaunay triangulation guarantees a maximum principle for the discrete Laplacian approximation (with linear elements).

2.1.3 Incremental Insertion Algorithm

For simplicity, assume that the site to be added lies within a bounding polygon of the existing triangulation. If we desire a triangulation from a new set of sites, three initial phantom points can always be added which define a triangle large enough to enclose all points to be inserted. In addition, interior boundaries are usually temporarily ignored for purposes of the Delaunay triangulation. After completing the triangulation, spurious edges are then deleted as a postprocessing step. Incremental insertion algorithms begin by inserting a new site into an existing Delaunay triangulation. This introduces the task of point location in the triangulation. Our incremental algorithm requires knowing which triangle the new site falls within. A search technique based on mesh walking (traversal) is implemented due to its simplicity. This method works extremely well when successively added points are close together. The basic idea is start at the location in the mesh of the previously inserted point and move one edge (or cell) at a time in the general direction of the newly added point. In the worst case, each point insertion requires $O(N)$ walks. This would result in a worst case overall complexity $O(N^2)$. For randomly distributed points, the average point insertion requires $O(N^{\frac{1}{2}})$ walks which gives an overall complexity $O(N^{\frac{3}{2}})$. For many applications where successive points tend to be close together, the number of walks is roughly constant and this simple algorithm can be very competitive when compared to more sophisticated tree search algorithms. In a mesh adaptation and refinement scenario the particular cell for site insertion is determined as part of the mesh adaptation algorithm, thereby reducing the burden of point location.

2.1.3.1 Green and Sibson Algorithm - The MaxMin Triangulation

Implementation of the Green and Sibson [4] algorithm is relatively straightforward. The first step is location, i.e. find the triangle containing point Q. Once this is done, three edges are then created connecting Q to the vertices of this triangle as shown in figure 2.1.3.1(a). If the point falls on an edge, then the edge is deleted and four edges are created connecting to vertices of the newly created quadrilateral. Using the circumcircle criteria it can be shown that the newly created edges (3 or 4) are automatically Delaunay. Unfortunately, some of the original edges are now incorrect. We need to somehow find all "suspect" edges which could possibly fail the circle test. Given that this can be done (described below), each suspect edge is viewed as a diagonal of the quadrilateral formed from the two adjacent triangles. The circumcircle test is applied to either one of the two adjacent triangles of the quadrilateral. If the fourth point of the quadrilateral is interior to this circumcircle, the suspect edge is then swapped as shown in figure 2.1.3.1(b), two more edges then become suspect. At any given time we can immediately identify all suspect edges. To do this, first consider the subset of all triangles which share Q as a vertex. One can guarantee at all times that all initial edges incident to Q are Delaunay and any edge made incident to Q by swapping must be Delaunay. Therefore, we need only consider the remaining edges of this subset which form a polygon about Q as suspect and subject to the incircle test. The process terminates when all suspect edges pass the circumcircle test.

The algorithm can be summarized as follows:

Algorithm: Incremental Delaunay Triangulation, Green and Sibson [17]

Step 1. Locate existing cell enclosing point Q.

Step 2. Insert site and connect to 3 or 4 surrounding vertices.

Step 3. Identify suspect edges.

Step 4. Perform edge swapping of all suspect edges failing the incircle test.

Step 5. Identify new suspect edges.

Step 6. If new suspect edges have been created, go to step 3.

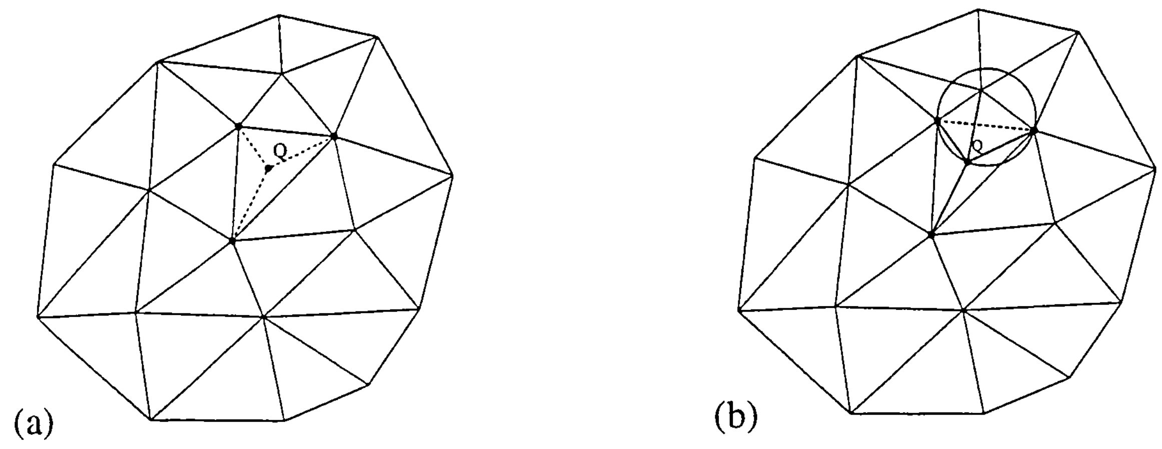


Figure 2.1.3.1 (a) Inserting of new vertex, (b) Swapping of suspect edge.

The Green and Sibson algorithm can be implemented using standard recursive programming techniques. The heart of the algorithm is the recursive procedure which would take the following form for the configuration shown in figure 2.1.3.1(c):

```
procedure swap[ $v_q$, $v_1$, $v_2$, $v_3$, $edges$]
if($incircle[v_q,v_1,v_2,v_3]$ = TRUE)then
    call reconfig_edges[$v_q$, $v_1$, $v_2$, $v_3$, $edges$]
    call swap[$v_q$, $v_1$, $v_4$, $v_2$, $edges$]
    call swap[$v_q$, $v_2$, $v_5$, $v_3$, $edges$]
endif
endprocedure
```

This example illustrates an important point. The nature of Delaunay triangulation guarantees that any edges swapped incident to Q will be final edges of the Delaunay triangulation. This means that we need only consider *forward propagation* in the recursive procedure. In a later section, we will consider incremental insertion and edge swapping for generating non-Delaunay triangulations based on other swapping criteria. This algorithm can also be programmed recursively but requires *backward propagation* in the recursive implementation. For the Delaunay triangulation algorithm, the insertion algorithm would simplify to the following three steps:

Recursive Algorithm: Incremental Delaunay Triangulation, Green and Sibson

Step 1. Locate existing cell enclosing point Q.

Step 2. Insert site and connect to surrounding vertices.

Step 3. Perform recursive edge swapping on newly formed cells (3 or 4).

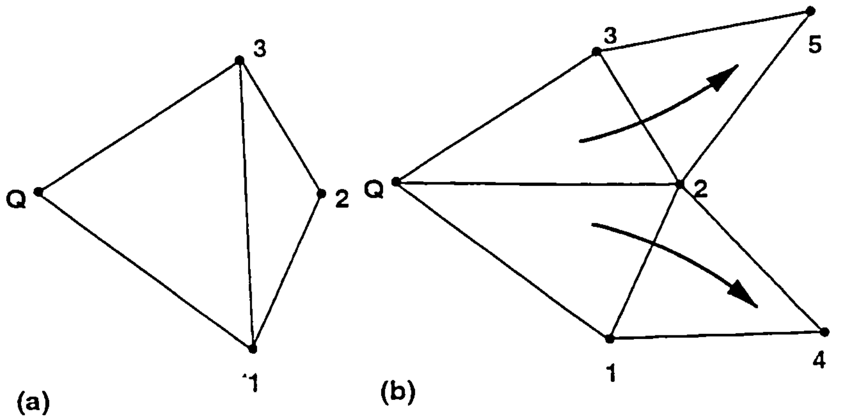


Figure 2.1.3.1(c) Edge swapping with forward propagation.

2.1.3.2 The MinMax Triangulation

As Babuška and Aziz [5] point out, from the point of view of finite elements the MaxMin (Delaunay) triangulation is not essential. What is essential is that no angle be too close to 180°. In other words, triangulations which minimize the maximum angle are more desirable. These triangulations are referred to as MinMax triangulations. In this case, the diagonal position for convex pairs of triangles is chosen which minimizes the maximum interior angle for both triangles. This type of algorithm is guaranteed to converge in a finite number of steps using arguments similar to Delaunay triangulation. Figures 2.1.3.2(a) and 2.1.3.2(b) present a Delaunay (MaxMin) and MinMax triangulation for 100 random points.

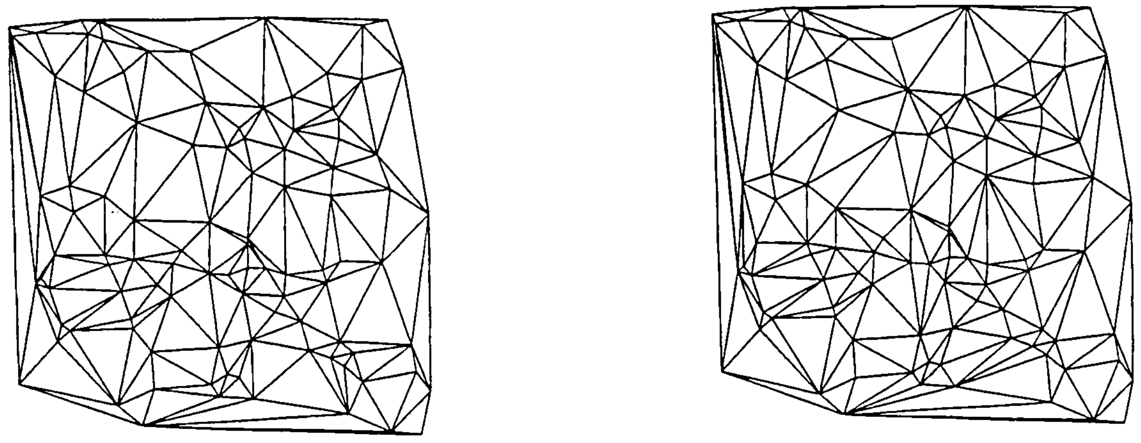


Figure 2.1.3.2 (a) Delaunay Triangulation (b) MinMax Triangulation

We have implemented a version of the Green and Sibson algorithm [4] which has been modified to produce locally optimal MinMax triangulations using incremental insertion and local edge swapping. The algorithm is implemented using recursive programming with complete forward and backward propagation (contrast figures 2.1.3.2(c) and 2.1.3.1(c)). This is a necessary step to produce locally optimized meshes.

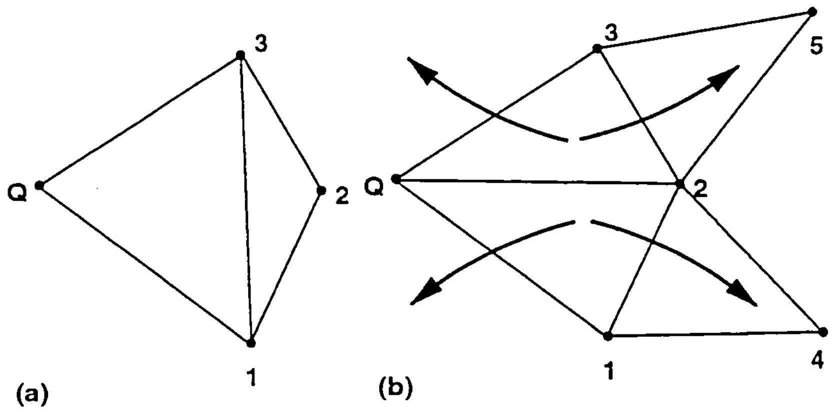


Figure 2.1.3.2(c) Edge swapping with forward and backward propagation

The MinMax triangulation has proven to be very useful in CFD. Figure 2.1.3.2(d) shows the Delaunay triangulation near the trailing edge region of an airfoil with extreme point clustering.

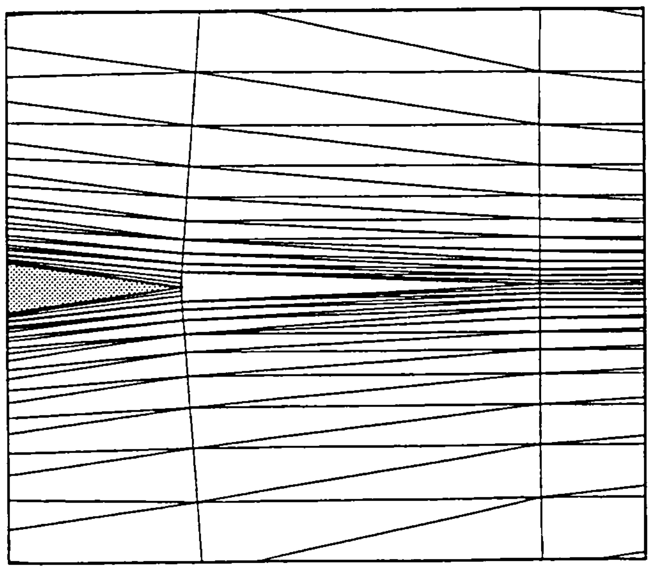


Figure 2.1.3.2(d) Delaunay triangulation near trailing edge of airfoil.

Upon first inspection, the mesh appears flawed near the trailing edge of the airfoil. Further inspection and extreme magnification near the trail edge of the airfoil (figure 2.1.3.2(e)) indicates that the grid is a mathematically correct Delaunay triangulation. Because the Delaunay triangulation does not control the maximum angle, the cells near the trailing edge have angles approaching 180°. The presence of nearly collapsed triangles leaves considerable doubt as to the accuracy of any numerical solutions computed in the trailing edge region.

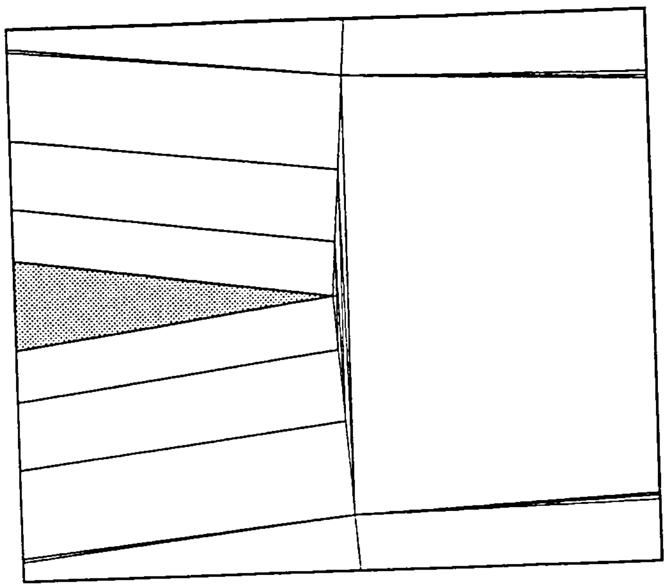


Figure 2.1.3.2(e) Extreme closeup of Delaunay triangulation near trailing edge of airfoil.

Edge swapping based on the MinMax criteria via incremental insertion produces the desired result as shown in figure 2.1.3.2(f).

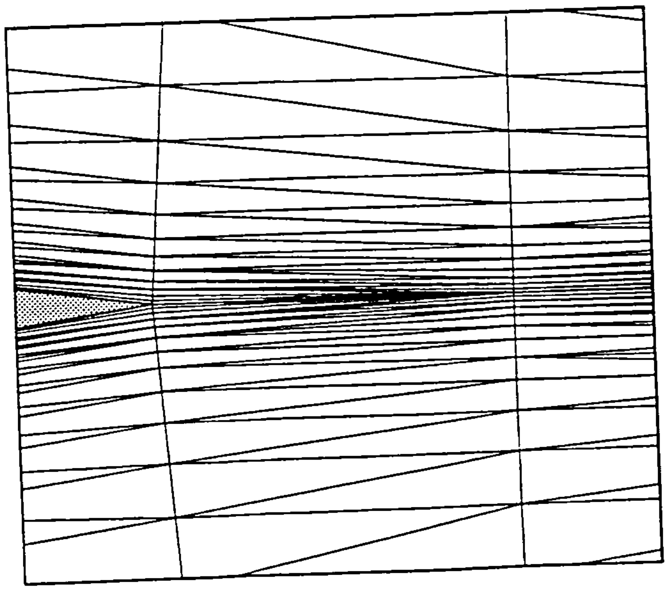


Figure 2.1.3.2(f) MinMax triangulation near trailing edge of airfoil.

2.1.4 Steiner Triangulation

Definition: A Steiner triangulation is any triangulation that adds additional sites to an existing triangulation to improve some measure of grid quality.

The insertion algorithm described earlier provides a simple mechanism for generating Steiner triangulations. Holmes [6] demonstrated the feasibility of inserting sites at circumcenters of Delaunay triangles into an existing 2-D triangulation to improve measures of grid quality. This has the desired effect of placing the new site in a position that guarantees that no other site in the triangulation can lie closer than the radius of the circumcircle, see figure 2.1.4(a). In a loose sense, the new site is placed as far away from other nearby sites as conservatively possible.

Warren *et al* [7] and Anderson [8] further demonstrated the utility of this type of Steiner triangulation in the generation and adaptive refinement of 2-D meshes. The algorithm discussed herein also permits Steiner triangulations based on either MinMax or MaxMin (Delaunay) insertion. Only in the latter case is the insertion at triangle circumcenters truly justifiable.

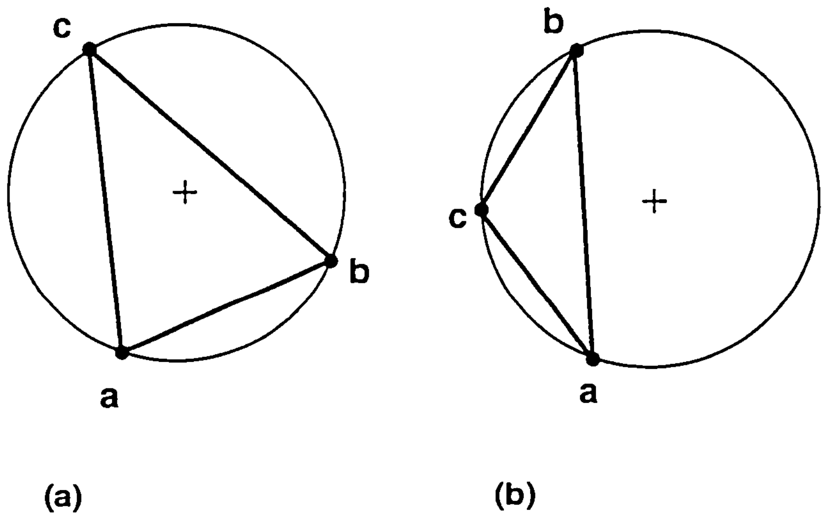


Figure 2.1.4(a) Inserting site at circumcenter of acute and obtuse triangles.

The 2-D Steiner point grid generation algorithm described in [6,7,8] consists of the following steps. The first step is the Delaunay triangulation of the boundary data. Usually three or four points are placed in the far field with convex hull enclosing all the boundary points. Starting with a triangulation of these points, sites corresponding to boundary curves are incrementally inserted using Sibson's algorithm in [4] as shown in figure 2.1.4(b). The initial triangulation does not guarantee that all boundary edges are members of the triangulation. Local edge swapping is performed so as to produce a *constrained* Delaunay triangulation which guarantees that all boundary edges are actual edges of the triangulation.

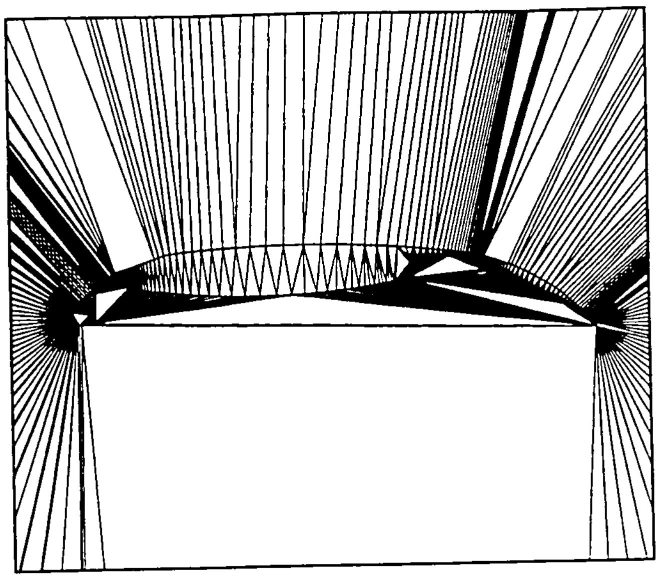


Figure 2.1.4(b) Initial triangulation of boundary points.

The boundary edges are marked so that they cannot be removed as the triangulation is refined. The user must specify some measure of quality for triangle refinement (aspect ratio, area, containment circle radius, for example) and a threshold value for the measure. Some care must be taken to insure that measures are chosen which are guaranteed to be reduced when the refinement takes place. Figure 2.1.4(c) shows a Steiner triangulation with MaxMin insertion and refinement based on maximum aspect ratio.

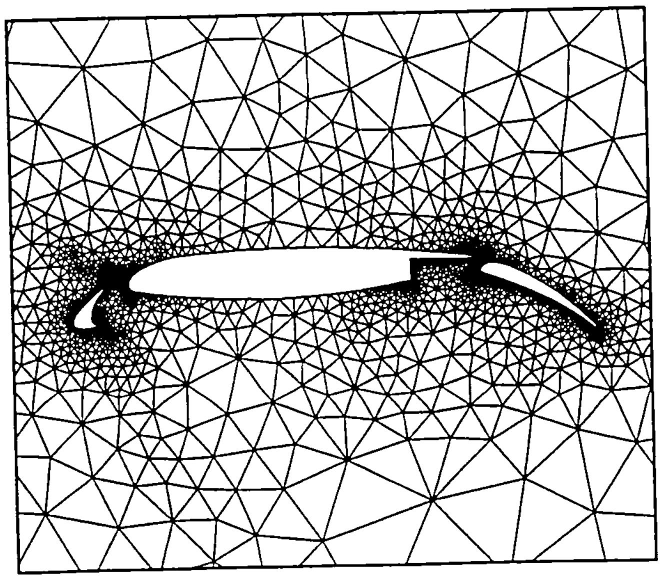


Figure 2.1.4(c) Steiner triangulation with sites inserted at circumcenters to reduce maximum cell aspect ratio.

In the present algorithm, a *dynamic heap* data structure of the quality measure is maintained. (Heap structures are a very efficient way of keeping a sorted list of entries with insertion and query time $O(\log N)$ for N entries.) The triangle with the largest value of the specified measure will be located at the top of the heap at all times during the triangulation. This makes implementation of a Steiner triangulation which *minimizes the maximum value* of the measure very efficient (and unique). In this implementation, the user can either specify the number of triangles to be generated or a threshold value of the measure. Note that multiple measures can be refined lexicographically.

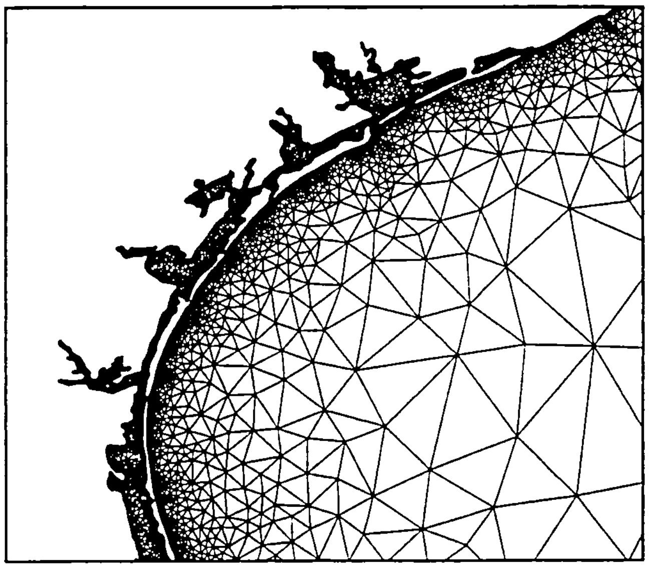


Figure 2.1.4(d) Steiner triangulation of Texas coast and the Gulf of Mexico.

This triangulation has proven to be very flexible. For instance, figure 2.1.4(d) shows a Steiner triangulation of the Texas coast and Gulf of Mexico.

2.1.5 Solution Adaptive Triangulation

The incremental insertion algorithm also permits mesh refinement based on data (i.e. solution) dependent measures. An initial mesh and data are required. In all cases, the solution data is assumed to be given at vertices of the mesh and varies linearly within each triangle (see section 3.2.4 for the appropriate file format). For example in figure 2.1.5, solution values f_1, f_2, f_3 are shown located at the vertices of triangle T_{123}.

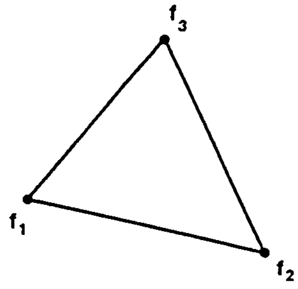


Figure 2.1.5 Generic triangle depicting solution at vertices (f_1, f_2, f_3).

From the solution data, refinement measures are computed which are usually some estimate of the solution error in a numerical computation. A dynamic heap structure is created to facilitate a refinement sequence which refines triangles with the largest refinement measure first. Monotone cubic Hermite spline interpolants of bounding curves are used for placing new vertices on boundary curves as needed in the refinement process. As triangles are refined and vertices added, the solution is linearly interpolated onto the new vertices for purposes of producing a new solution data file corresponding to the refined mesh. The current implementation assumes that a 4-vector, U, of solution data is given at vertices of the mesh. This 4-vector corresponds to the conserved flow variables of the compressible Euler equations (mass, Cartesian momentum components, and energy):

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}$$

From this vector, several scalar functions, f_i, can be formed as described in the usage section 1.2. From these scalar functions, two refinement measures are available:
<u>Sobolev Semi-Norm Measure</u>

$$M(T_{123}) = \int_{T_{123}} \left((f_x)^2 + (f_y)^2\right) da$$

<u>Scaled Edge Gradients</u>

$$M(T_{123}) = \max(|f_2 - f_1|, |f_3 - f_2|, |f_3 - f_1|)L^p, \quad L = \sqrt{Area_{123}}$$

In the latter measure, the parameter p is user specified. Typical values of p range from 0 to 1. Other triangle-wise measures can be incorporated by modifying the function "refine_measure."

2.2 GRAPH PARTITIONING FOR PARALLEL COMPUTING

An efficient partitioning of a mesh for distributed memory machines is one that ensures an even distribution of computational workload among the processors and minimizes the amount of time spent in interprocessor communications. The former requirement is termed *load balancing*. For if the load were not evenly distributed, some processors will have to sit idle at synchronization points waiting for other processors to catch up. The second requirement comes from the fact that communication between processors takes time and it is not always possible to hide this latency in data transfer. In our parallel implementation of a finite-volume flow solver on unstructured grids, data for the nodes that lie on the boundary between two processors is exchanged, hence requiring a bidirectional data-transfer. On many systems, a synchronous exchange of data can yield a higher performance than when done asynchronously. To exploit this fact, edges of the communication graph are colored such that no vertex has more than one edge of a certain color incident upon it. A communication graph is a graph in which the vertices are the processors and an edge connects two vertices if the two corresponding processors share an interprocessor boundary. The colors in the graph represent separate communication cycles. For instance, the mesh partitioned amongst four processors as shown in figure 2.2(a), would produce the communication graph shown in figure 2.2(b).

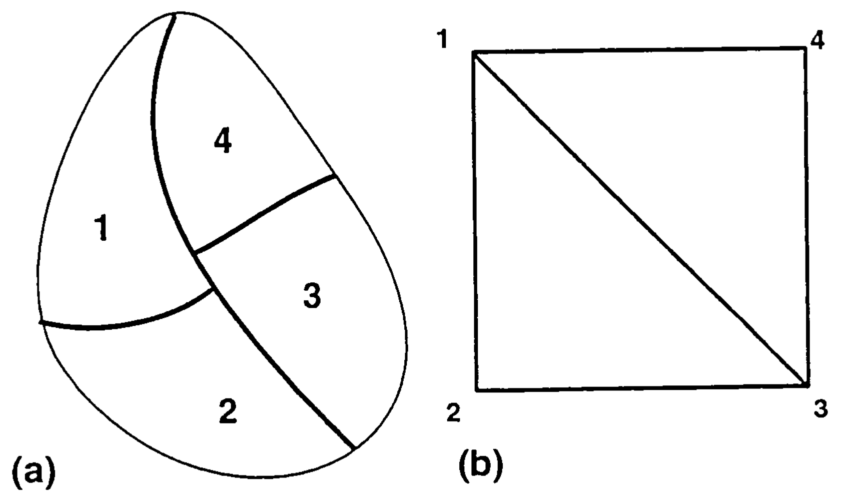


Figure 2.2 (a) Four partition mesh. (b) Communication graph.

The graph shown in figure 2.2(b) can be colored edgewise using three colors. For example, in the first communication cycle, processors (1.4) could perform a synchronous data exchange as would processors (2.3). In the second communication cycle, processors (1,2) and (3.4) would exchange and in the third cycle, processors (1,3) would exchange while processors 2 and 4 sit idle. Vizing's theorem proves that any graph of maximum vertex degree Δ (number of edges incident upon a vertex) can be colored using n colors such that $\Delta \leq n \leq \Delta + 1$. Hence, any operation that calls for every processor to exchange data with its neighbors will require n communication cycles.

The actual cost of communication can often be accurately modeled by the linear relationship:

$$Cost = \alpha + \beta m$$

where α is the time required to initiate a message, β is the rate of data-transfer between two processors and m is the message length. For n messages, the cost would be

$$Cost = \sum_{n} (\alpha + \beta m_n).$$

This cost can be reduced in two ways. One way is to reduce Δ thereby reducing n. The alternative is to reduce the individual message lengths. The bounds on n are $2 \leq N \leq P-1$ for $P \geq 3$ where P is the total number of processors. Figure 2.2(c) shows the partitioning of a mesh which reduces Δ, and 2.2(d) shows a partitioning which minimizes the message lengths. For the mesh in figure 2.2(c), $\Delta = 2$ while in figure 2.2(d), $\Delta = 3$. However. the average message length for the partitions shown in figure 2.2(d) is about half as much as that in figure 2.2(c).

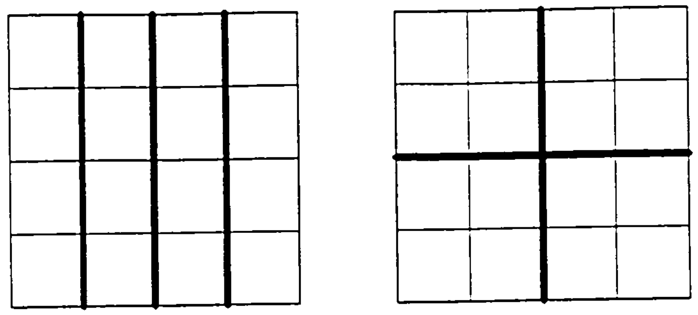


Figure 2.2 (c) Mesh partitioning with minimized Δ, (d) Mesh with minimizes message length.

In practice, it is difficult to partition an unstructured mesh while simultaneously minimizing the number and length of messages. In the following paragraphs, a few of the most popular partitioning algorithms which approximately accomplish this task will be discussed. All the algorithms discussed below: coordinate bisection, Cuthill-McKee, and spectral partitioning are evaluated in the paper by Venkatakrishnan, Simon, and Barth [9]. This paper evaluates the partitioning techniques within the confines of an explicit, unstructured finite-volume Euler solver. Spectral partitioning has been extensively studied by Simon [10].

Note that for the particular applications that we have in mind (a finite-volume scheme with solution unknowns at vertices of the mesh), it makes sense to partition the domain such that the separators correspond to edges of the mesh. Also note that the partitioning algorithms all can be implemented recursively. The mesh is first divided into two sub-meshes of nearly equal size. Each of these sub-meshes is subdivided into two more sub-meshes and the process in repeated until the desired number of partitions P is obtained (P is a integer power of 2). Since we desire the separator of the partitions to coincide with edges of the mesh, the division of a sub-mesh into two pieces can be viewed as a 2-coloring of *faces* of the sub-mesh. For the Cuthill-McKee and spectral partitioning techniques, this amounts to supplying these algorithms with the *dual* of the graph for purposes of the 2-coloring. The balancing of each partition is usually done cellwise; although an edgewise balancing is more appropriate in the present applications. Due to the recursive nature of partitioning, the algorithms outlined below represent only a single step of the process.

2.2.1 Coordinate Bisection Partitioning

In the coordinate bisection algorithm, face centroids are sorted either horizontally or vertically depending of the current level of the recursion. A separator is chosen which balances the number of faces. Faces are colored depending on which side of the separator they are located. The actual edges of the mesh corresponding to the separator are characterized as those edges which have adjacent faces of different color, see figure 2.2.1. This partitioning is very efficient to create but gives sub-optimal performance on parallel computations owing to the long message lengths than can routinely occur.

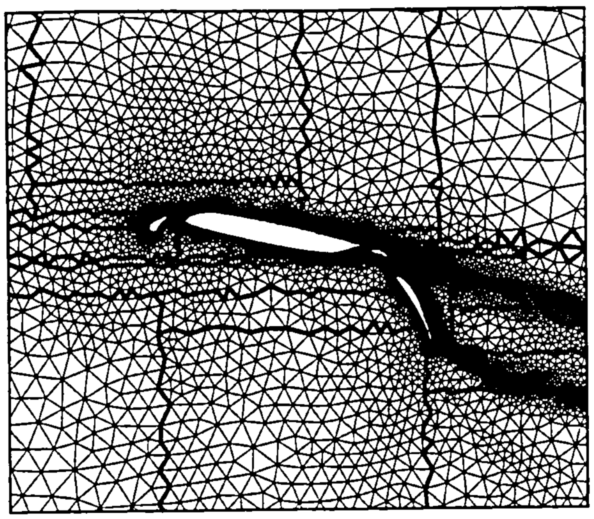


Figure 2.2.1 Coordinate bisection (16 partitions).

2.2.2 Cuthill-McKee Partitioning

The Cuthill-McKee (CM) algorithm described earlier can also be used for recursive mesh partitioning. In this case, the Cuthill-McKee ordering is performed on the *dual* of the mesh graph.

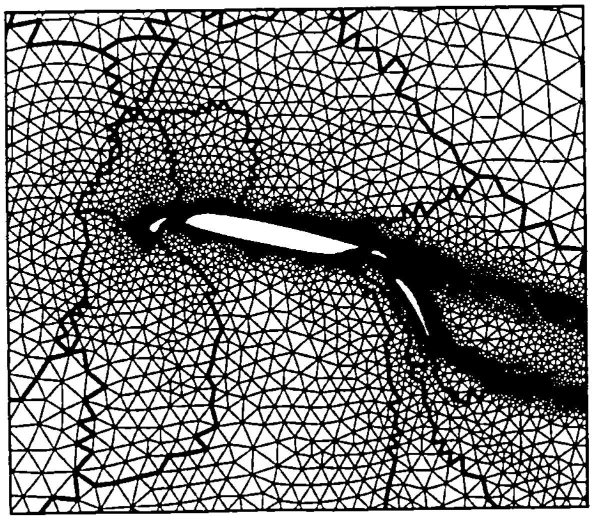


Figure 2.2.2 Cuthill-McKee partitioning of mesh (16 partitions).

A separator is chosen either at the median of the ordering (which would balance the coloring of faces of the original mesh) or the separator is chosen at the level set boundary *closest* to the median as possible. This latter technique has the desired effect of reducing the number of disconnected sub-graphs that occur during the course of the partitioning. Figure 2.2.2 shows a Cuthill-McKee partitioning for the multi-component airfoil mesh. The Cuthill-McKee ordering tends to produce long boundaries because of the way that the ordering is propagated through a mesh. The maximum degree of the communication graph also tends to be higher using the Cuthill-McKee algorithm. The results shown in ref. [9] for multi-component airfoil grids indicate a performance on parallel computations which is slightly worse than the coordinate bisection technique.

2.2.3 Spectral Partitioning

The last partitioning considered is the spectral partitioning which exploits properties of the Laplacian $\mathcal{L}$ of a graph (defined below). The algorithm consists of the following steps:

Algorithm: Spectral Partitioning.

Step 1. Calculate the matrix $\mathcal{L}$ associated with the Laplacian of the graph (dual graph in the present case).
Step 2. Calculate the eigenvalues and eigenvectors of $\mathcal{L}$.
Step 3. Order the eigenvalues by magnitude, $\lambda_1 \leq \lambda_2 \leq \lambda_3 ... \lambda_N$.
Step 4. Determine the smallest nonzero eigenvalue, λ_f and its associated eigenvector $\mathbf{x}_f$ (the Fiedler vector).
Step 5. Sort elements of the Fiedler vector.
Step 6. Choose a divisor at the median of the sorted list and 2-color vertices of the graph (or dual) which correspond to elements of the Fielder vector less than or greater than the median value.

The spectral partitioning of the multi-component airfoil is shown in figure 2.2.3(a). In reference [9], we found that parallel computations performed slightly better on the spectral partitioning than on the coordinate bisection or Cuthill-McKee. The cost of the spectral partitioning is very high (even using a Lanczos algorithm to compute the eigenvalue problem). It has yet to be determined if the spectral partitioning will have practical merit.

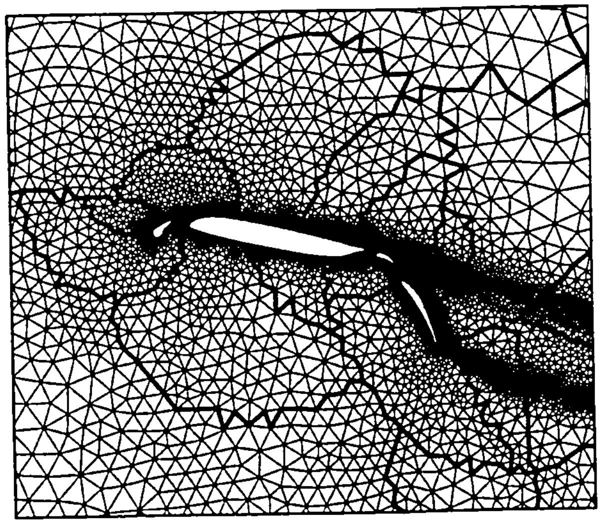


Figure 2.2.3(a) Spectral partitioning of multi-component airfoil (16 partitions).

The spectral partitioning exploits a peculiar property of the "second" eigenvalue of the

Laplacian matrix associated with a graph. The Laplacian matrix of a graph is simply

$$\mathcal{L} = -\mathcal{D} + \mathcal{A}.$$

where $\mathcal{A}$ is the standard adjacency matrix

$$\mathcal{A}_{ij} = \begin{cases} 1 & e(v_i, v_j) \in \ G \\ 0 & \text{otherwise} \end{cases}$$

and $\mathcal{D}$ is a diagonal matrix with entries equal to the degree of each vertex. $\mathcal{D}_i = d(v_i)$. From this definition, it should be clear that rows of $\mathcal{L}$ each sum to zero. Define an N-vector, $\mathbf{s} = [1, 1, 1, ...]^T$. By construction we have that

$$\mathcal{L}\mathbf{s} = 0.$$

This means that at least one eigenvalue is zero with $\mathbf{s}$ as an eigenvector.
The objective of the spectral partitioning is to divide the mesh into two partitions of equal size such that the number of edges cut by the partition boundary is approximately minimized.

Technically speaking, the smallest nonzero eigenvalue need not be the second. Graphs with disconnected regions will have more that one zero eigenvalue depending on the number of disconnected regions. For purposes of discussion, we assume that disconnected regions are not present, i.e. that λ_2 is the relevant eigenmode.

<u>Elements of the proof:</u>

Define a partitioning vector which 2-colors the vertices

$$\mathbf{p} = [+1, -1, -1, +1, +1, ..., +1, -1]^T$$

depending on the sign of elements of $\mathbf{p}$ and the one-to-one correspondence with vertices of the graph, see for example figure 2.2.3(b). Balancing the number of vertices of each color amounts to the requirement that

$$\mathbf{s} \perp \mathbf{p}$$

where we have assumed an even number of vertices.

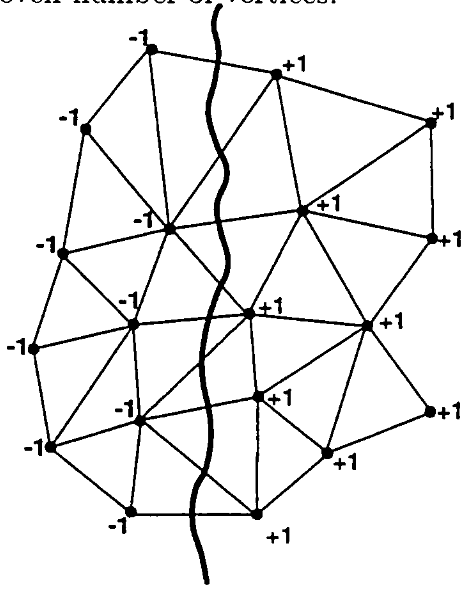


Figure 2.2.3(b) Arbitrary graph with 2-coloring showing separator and cut edges.

The key observation is that the number of cut edges, E_c, is precisely related to the L_1 norm of the Laplacian matrix multiplying the partitioning vector, i.e.

$$4E_c = \|\mathcal{L}\mathbf{p}\|_1$$

which can be easily verified. The goal is to minimize cut edges. That is to find $\mathbf{p}$ which minimizes $\|\mathcal{L}\mathbf{p}\|_1$ subject to the constraints that $\|\mathbf{p}\|_1 = N$ and $\mathbf{s} \perp \mathbf{p}$. Since $\mathcal{L}$ is a real symmetric (positive semi-definite) matrix, it has a complete set of real eigenvectors which can be orthogonalized with each other. The next step of the proof would be to extend the domain of $\mathbf{p}$ to include real numbers (this introduces an inequality) and expand $\mathbf{p}$ in terms of the orthogonal eigenvectors.

$$\mathbf{p} = \sum_{i=1}^{n} c_i \mathbf{x}_i$$

By virtue of

$$\mathcal{L}\mathbf{s} = 0$$

we have that $\mathbf{x}_1 = \mathbf{s}$. It remains to be shown that $\|\mathcal{L}\mathbf{p}\|_1$ is minimized when $\mathbf{p} = \mathbf{p}' = n\mathbf{x}_2/\|\mathbf{x}_2\|_1$,i.e. when the Fiedler vector is used. Inserting this expression for $\mathbf{p}$ we have that

$$\|\mathcal{L}\mathbf{p}'\|_1 = n\lambda_2$$

It is a simple matter to show that adding any other eigenvector component to $\mathbf{p}'$ while insisting that $\|\mathbf{p}\|_1 = N$ can only increase the L_1 norm. This would complete the proof. Figure 2.2.4(c) plots contours (level sets) of the Fiedler vector for the multi-component airfoil problem.

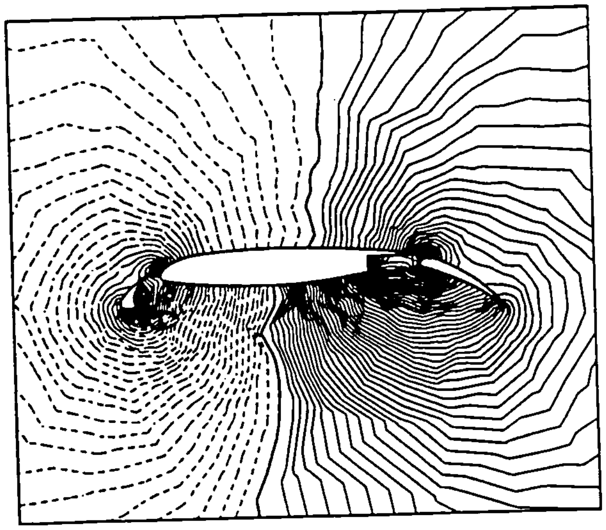


Figure 2.2.4(c) Contours of Fiedler Vector for Spectral Partitioning. Dashed lines are less than the median value.

3 FILE FORMATS

3.1 THE GENERIC FORMAT

INPUT FILE

Generic File Structure: (FORMATTED)

```
Record 1    :  ncurves                  ! Integer number of boundary curves
Record 2    :  N1                       ! Integer number of points in curve 1
Record 3    :  x_1   y_1                !
Record 4    :  x_2   y_2                !
Record 5    :  x_3   y_3                !
.                                       ! Contiguous coordinate pairs of boundary
.                                       ! curve 1
.                                       !
Record N1+2:  x_N1  y_N1                !
Record N1+3:  N2                        ! Integer number of points in curve 2
Record N1+4:  x_1   y_1                 !
Record N1+5:  x_2   y_2                 !
Record N1+6:  x_3   y_3                 !
.                                       ! Contiguous coordinate pairs of boundary
.                                       ! curve 2
.                                       !
Record N1+N2+3:  x_N2  y_N2             !
.
.
.
etc
eof
```

OUTPUT FILE

Generic File Structure: (FORMATTED)

```
                                      ! Number of cells, Number of edges,
n_cells,n_edges,n_b_edges,n_points ! Number of boundary edges, Number of points
x_1 y_1                               ! Coordinate pairs for entire grid
x_2 y_2                               ! number of pairs = n_points
.                                     !
.                                     !
X_n_points Y_n_points                 !
e_to_c(1,1) e_to_c(2,1)               ! Edge to Cell pointers
e_to_c(1,2) e_to_c(2,2)               ! number of index pairs = n_edges
.                                     !
.                                     !
.                                     !
.                                     !
```

```
e_to_c(1,n_edges) e_to_c(2,n_edges)!
e_to_v(1,1),e_to_v(2,1)            ! Edge to Vertex pointers
e_to_v(1,2),e_to_v(2,2)            ! number of index pairs = n_edges
.                                  !
.                                  !
.                                  !
                                   !
                                   !
.
e_to_v(1,n_edges),e_to_v(2,n_edges)!
b_e(1)                             ! Boundary Edge pointers
b_e(2)                             ! number of index entries = n_b_edges
.                                  !
.                                  !
.                                  !
                                   !
                                   !
.                                  !
b_e(n_b_edges)                     !
b_bc(1),idum1,idum2                !
b_bc(2),idum1,idum2                ! Boundary Type identifier
.                                  ! (idum1, idum2 not used)
.                                  ! number of index trios = n_b_edges
.                                  !
.                                  !
.                                  !
                                   !
                                   !
.                                  !
b_bc(n_b_edges),idum1,idum2        !
ncurves                            ! Number of boundary curves-outer curve last
ncpts(1)                           ! ncpts(1)= starting address curve1
ncpts(2)                           ! ncpts(2)= starting address curve2
.                                  !
.                                  !
.                                  !
.                                  !
                                   !
ncpts(ncurves)                     ! ncpts(ncurve)= starting address last curve
ncpts(ncurves+1)                   ! ncpts(ncurves+1)= total number of
eof                                !                       boundary points + 1
```

3.2 EXAMPLE FORMATS

3.2.1 Case 1: Triangulation of a point cloud

Triangulate a specified cloud of points. Given a set of points, form the triangulation of the points with no specification of bounding curves (i.e. the outer boundary). For the Delaunay triangulation, the resulting outer boundary formed from the triangulation coincides with the convex hull of the point set. No additional points will be added. Local edge swapping is determined either by the the MaxMin or MinMax criteria as selected by the user.

Input file structure: (FORMATTED)

```
0              ! No boundary curves
N              ! Number of points to be triangulated
x_1 y_1        !
x_2 y_2        !
.              !
.              ! Coordinate pairs (not contiguous)
.              !
X_N y_N        !
eof
```

3.2.2 Case 2: Constrained triangulation of a point cloud with prespecified bounding curves

Triangulate a prespecified set of interior sites constrained by prespecified boundary curve(s). Given the boundary curves, represented by contiguous points, an initial triangulation of the boundary curves is formed. The remaining interior sites are then inserted and local edge swapping occurs depending on whether a MinMax or MaxMin triangulation is desired. Typically the MinMax triangulation will produce better results for meshes containing highly stretched cells, see section 2.1.3.2 for further discussion. The option exists for additional interior points to be inserted after the initial specified points have been inserted. In the case of additional point insertion, the user may specify either the largest aspect ratio cell desired or the number of total cells desired. Additional points will be inserted at the cell circumcenters. Refinement occurs in the cell with the largest aspect ratio as determined by the dynamic heap structure . A "Poor Man's" Laplacian smoothing is also available for postprocessing of triangulations.

Input file structure: (FORMATTED)

```
ncurves        ! Number of boundary curves, last curve is the outer boundary
N1             ! Number of points in curve 1
x_1 y_1        !
x_2 y_2        !
.              ! Coordinate pairs for curve 1
.              !
X_N1 y_N1      !
N2             ! Number of points in curve 2
x_1 y_1        !
x_2 y_2        ! Coordinate pairs for curve 2
.              !
.              !
```

```
X_N2 y_N2     !
.             !
.             !
NL            ! Number of points in last curve (outer boundary)
x_1 y_1       !
x_2 y_2       !
.             ! Coordinate pairs for last curve (outer boundary)
.             !
X_NL y_NL     !
NUMINT or 0   ! Number of specified interior points, if 0 then
x_1 y_1       ! program will self-count
x_2 y_2       !
.             !
.             ! Coordinate pairs for specified interior points.
X_L Y_L       !
eof
```

3.2.3 Case 3: Steiner triangulation with prespecified boundary curves

Steiner triangulate the interior/exterior of specified boundary curve(s). In this case, an initial triangulation is constructed from the boundary curve(s), which are represented as contiguous points. Steiner points are inserted into the interior of the outer boundary curve (the last curve read in) and to the exterior of the inner boundary curves. New cells are generated by refining the cell with the largest aspect ratio, as determined by the dynamic heap structure. Each new point is placed at the circumcenter of the cell which is being refined. The MinMax or MaxMin options may be used to determine if local edge swapping occurs after a new point is inserted. The user can specify either the maximum aspect ratio desired or the specific number of cells desired. A "Poor Man's" Laplacian smoothing is also available for postprocessing of triangulations.

Input file structure: (FORMATTED)

```
ncurves       ! Number of boundary curves, last curve is the outer boundary
N1            ! Number of points in curve 1
x_1 y_1       !
x_2 y_2       !
.             ! Coordinate pairs for curve 1
.             !
X_N1 y_N1     !
N2            ! Number of points in curve 2
x_1 y_1       !
x_2 y_2       !
.             ! Coordinate pairs for curve 2
.             !
X_N2 y_N2     !
.
```

```
.
.
NL            ! Number of points in last curve (outer boundary)
x_1 y_1       !
x_2 y_2       !
.             !
.             ! Coordinate pairs for last curve (outer boundary)
.             !
X_NL y_NL     !
eof
```

3.2.4 Case 4: Solution adaptive triangulation

Refine existing triangulations based on solution error measures. In this case, initial mesh and solution files are required. Steiner points are inserted into cells in the interior of the mesh. Cells are chosen for refinement based on a measure of solution error. The MinMax or MaxMin options may be used to determine what type of local edge swapping occurs after a point is inserted. The user can specify either the maximum solution error desired or the specific number of cells generated. Cells are refined until one of the two criteria is satisfied. After the newly adapted mesh is generated, the code will also output a new solution file with interpolated values at the new mesh points. Refinement measures are calculated in the function refine_measure. The user can easily modify this function for other possible error measures. A "Poor Man's" Laplacian smoothing is also available for postprocessing of triangulations.

Input file structure: Initial grid (FORMATTED)

```
                                   ! Number of cells, Number of edges,
n_cells,n_edges,n_b_edges,n_points ! Number of boundary edges, Number of points
x_1 y_1                            ! Coordinate pairs for entire grid
x_2 y_2                            ! number of pairs = n_points
                                   !
.                                  !
X_n_points Y_n_points              !
e_to_c(1,1) e_to_c(2,1)            ! Edge to Cell pointers
e_to_c(1,2) e_to_c(2,2)            ! number of index pairs = n_edges
.                                  !
.                                  !
.                                  !
e_to_c(1,n_edges) e_to_c(2,n_edges)!
e_to_v(1,1),e_to_v(2,1)            ! Edge to Vertex pointers
e_to_v(1,2),e_to_v(2,2)            ! number of index pairs = n_edges
.                                  !
.                                  !
                                   !
.                                  !
e_to_v(1,n_edges),e_to_v(2,n_edges)!
```

```
b_e(1)                                  ! Boundary Edge pointers
b_e(2)                                  ! number of index entries = n_b_edges
 .                                      !
 .                                      !
                                        !
                                        !
 .                                      !
b_e(n_b_edges)                          !
b_bc(1),idum1,idum2                     !
b_bc(2),idum1,idum2                     ! Boundary Type identifier
 .                                      ! (idum1, idum2 not used)
 .                                      ! number of index trios = n_b_edges
 .                                      !
                                        !
 .                                      !
b_bc(n_b_edges),idum1,idum2             !
ncurves                                 ! Number of boundary curves-outer curve last
ncpts(1)                                ! ncpts(1)= starting address curve1
ncpts(2)                                ! ncpts(2)= starting address curve2
 .                                      !
 .                                      !
 .                                      !
                                        !
ncpts(ncurves)                          ! ncpts(ncurve)= starting address last curve
ncpts(ncurves+1)                        ! ncpts(ncurves+1)= total number of
eof                                     !                         boundary points + 1
```

Input file structure: Initial solution (UNFORMATTED)

```
n_points                              ! Number of points (same as n_points in grid)
fsmach,alpha,re,time                  ! Freestream Mach, alpha, Reynolds number, time
((sol(n,j),j=1,num_points),n=1,4)! Solution variables
eof
```

3.2.5 Case 5: Steiner triangulation with graph partitioning

See Case 3 for input file format

4 REFERENCES

1. Lawson, C. L., "Software for C^1 Surface Interpolation", Mathematical Software III, (Ed., John R. Rice), Academic Press, New York, 1977.
2. Rajan, V. T., "Optimality of the Delaunay Triangulation in $\mathbf{R}^d$", Proceedings of the 7th ACM Symposium on Computational Geometry, 1991, pp. 357-372.
3. Rippa, S., "Minimal Roughness Property of the Delaunay Triangulation", CAGD, Vol. 7, No. 6., 1990, pp-489–497.
4. Green, P. J. and Sibson, R., "Computing the Dirichlet Tesselation in the Plane", The Computer Journal, Vol. 21, No. 2, 1977, pp. 168—173.
5. Babuška, I., and Aziz, A. K., "On the Angle Condition in the Finite Element Method", SIAM J. Numer. Anal., Vol. 13, No. 2, 1976.
6. Holmes, G. and Snyder, D., "The Generation of Unstructured Triangular Meshes using Delaunay Triangulation," in *Numerical Grid Generation in CFD*, pp. 643-652, Pineridge Press, 1988.
7. Warren, G., Anderson, W.K., Thomas, J.L., and Krist, S.L.,"Grid Convergence for Adaptive Methods,", AIAA paper 91-1592-CP, Honolulu, Hawaii, June 24-27,1991.
8. Anderson, W.K.,"A Grid Generation and Flow Solution Method for the Euler Equations on Unstructured Grids," NASA Langley Research Center, USA, unpublished manuscript, 1992.
9. Venkatakrishnan, V., Simon, H.D., Barth, T.J., "A MIMD Implementation of a Parallel Euler Solver for Unstructured Grids", NASA Ames R.C., Tech. Report RNR-91-024, 1991.
10. Simon, H.D., "Partitioning of Unstructured Problems for Parallel Processing," NASA Ames R.C., Tech. Report RNR-91-008, 1991.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE January 1994	3. REPORT TYPE AND DATES COVERED Technical Memorandum

4. TITLE AND SUBTITLE Incremental Triangulation via Edge Swapping and Local Optimization	5. FUNDING NUMBERS 505-5950
6. AUTHOR(S) N. Lyn Wiltberger	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Ames Research Center Moffett Field, CA 94035-1000	8. PERFORMING ORGANIZATION REPORT NUMBER A-92196
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001	10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA TM-103979

11. SUPPLEMENTARY NOTES

Point of Contact: N. Lyn Wiltberger, Ames Research Center, MS 202A-2, Moffett Field, CA 94035-1000; (415) 604-4474

12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified — Unlimited Subject Category 34	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This document is intended to serve as an installation, usage, and basic theory guide for the two-dimensional triangulation software "HARLEY" written for the Silicon Graphics IRIS workstation. This code consists of an incremental triangulation algorithm based on point insertion and local edge swapping. Using this basic strategy, several types of triangulations can be produced depending on user selected options. For example, local edge swapping criteria can be chosen which minimizes the maximum interior angle (a MinMax triangulation), or which maximizes the minimum interior angle (a MaxMin or Delaunay triangulation). It should be noted that the MinMax triangulation is generally only locally optimal (not globally optimal) in this measure. The MaxMin triangulation, however, is both locally and globally optimal. In addition, Steiner triangulations can be constructed by inserting new sites at triangle circumcenters followed by edge swapping based on the MaxMin criteria. Incremental insertion of sites also provides flexibility in choosing cell refinement criteria. For instance, by choosing a refinement measure based on solution error, the code can be utilized for solution adaptive grid generation. A dynamic heap structure has been implemented in the code so that once a refinement measure is specified (i.e., maximum aspect ratio or some measure of a solution gradient for the solution adaptive grid generation) the cell with the largest value of this measure is continually removed from the top of the heap and refined. The heap refinement strategy allows the user to specify either the number of cells desired or refine the mesh until all cell refinement measures satisfy a user specified tolerance level. Since the dynamic heap structure is constantly updated, the algorithm always refines the particular cell in the mesh with the largest refinement criteria value. The code allows the user to: (1) triangulate a cloud of prespecified points (sites), (2) triangulate a set of prespecified interior points constrained by prespecified boundary curve(s), (3) Steiner triangulate the interior/exterior of prespecified boundary curve(s), (4) refine existing triangulations based on solution error measures, and (5) partition meshes based on the Cuthill-McKee, Spectral, and Coordinate bisection strategies.

14. SUBJECT TERMS Unstructured grid generation, Delaunay triangulation	15. NUMBER OF PAGES 43
	16. PRICE CODE A03

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102