A Comparison of Parametric Space vs. Real Space Triangular Meshing Algorithms

Karl G. Merkley[1] and Mark L. Dewey[2]

[1] Elemental Technologies, Inc karl@elemtech.com
[2] Elemental Technologies Inc. mildewey@elemtech.com

Summary. We present a triangulation algorithm that is a hybrid between a Delaunay triangulation and an advancing front algorithm. The Delaunay triangulation is performed in parametric space and Steiner points are inserted based on an advancing front criteria. This Advancing Steiner Point (ASP) algorithm is compared with the three-space advancing front algorithm that is currently implemented in Cubit, a mesh generation tool developed at Sandia National Laboratories. We find that the two algorithms are comparable in both speed and quality on analytic surfaces while the ASP algorithm has significant speed advantages on parametric and complex surfaces.

Key words: Triangle meshing, parametric space, advancing front mesh

1 Introduction

The Cubit meshing toolkit [1] is a research and production meshing tool developed at Sandia National Laboratories. In recent years, most of the Cubit development efforts have been directed at developing quadrilateral and hexahedral meshing methods. Cubit has supported triangular and tetrahedral meshing but this has not been a primary focus. However, as the Cubit user base increases there are more requests to support improved triangular meshing methods. Cubit has supported Delaunay methods [2] but pure Delaunay is limited to planar surfaces. It has also supported an advancing front algorithm [3][4] that is very robust but in some cases it is very slow compared with commercial meshing packages.

In this paper we introduce a parametric space triangular meshing algorithm that is being developed as part of the Cubit Adaptive Meshing Algorithms Library (CAMAL) . The algorithm introduced here is a hybrid of Delaunay and advancing front techniques. It creates a Delaunay triangulation in parametric space and inserts Steiner points into the triangulation based on an advancing front technique. The methods used by the Advancing Steiner Point (ASP) algorithm are similar to AFLR developed by Marcum.[5][6].

2 Advancing Steiner Points

The Advancing Steiner Point (ASP) algorithm is a combination of the Delaunay algorithm and the advancing front algorithm. The algorithm creates an initial Delaunay triangulation as in steps 1 to 4 of the Delaunay algorithm and then inserts Steiner points chosen based on the techniques of the advancing front algorithm. The algorithm proceeds as follows.

1. Specify node placement on the surface boundaries.
2. Create a bounding polygon extending beyond the surface boundaries
3. Generate a valid triangulation of the bounding polygon including the nodes on the surface boundaries and recover the boundary.
4. Assign a point distribution function to each boundary point based on the local point spacing. 5) Determine boundary point normals and add additional boundary normals where the boundaries are discontinuous.
5. Based on the normals at the boundary insert points along the front. Stop point insertion if the spacing has grown to the desired size. Also, stop if fronts overlap.
6. Subdivide elements of the initial triangulation at the inserted point.
7. Optimize the connectivity using recursive edge swaps. For each element pair, compare the reconnection criterion (min-max) for all possible connectivities and swap edges using the most optimal one.
8. Repeat steps 7 and 8 until the stopping criteria are met.
9. Smooth the nodal coordinates using a simple Laplacian algorithm.
10. Check for any other required edge swaps to optimize the grid.

The smoothing and edge swap optimizations in steps 9 and 10 improve the overall quality of elements in the grid. Care must be taken to avoid swapping across areas of high curvature and smoothing should be performed in parametric space to avoid time-consuming computations in real space.

3 Results

The Advancing Steiner Point algorithm outperforms the existing Cubit advancing front algorithm in both speed and element quality in every case except for a simple square. In the case of the square the two algorithms are nearly equal with a slight speed advantage to the advancing front algorithm. In every other case the ASP algorithm is faster than the advancing front algorithm. The following cases document the performance of the both algorithms in a variety of cases. Speed is measured as user experience. This measures the time the meshing process is started until the graphics display is completed.

3.1 Well Field

Figures 1 and 2 show the projection of a complex well field onto the z-plane. The 1 shows the results of an ASP grid using a natural neighbor [7] sizing algorithm. Since there is an underlying grid that exists at every point in the algorithm this is an obvious algorithm for calculating element sizes. The existing linear sizing algorithm in Cubit is based on a linear weighting of the points on the boundary. In this example, the combination of algorithm and sizing function forces a rapid change in element size near the boundary as shown in figure 2. As opposed to the more gradual change in element size shown in figure 1.

Table 1 shows the resulting time and element quality for the resulting mesh. We see that the ASP algorithm is about three times faster. In addition the minimum angle is quality criteria is much better for the ASP algorithm.

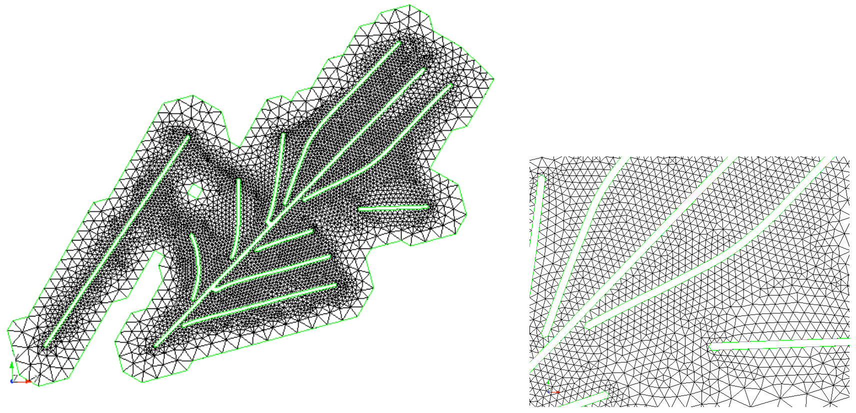

Fig. 1. Well field projected onto the z-plane and meshed with the ASP algorithm and a close-up of the resulting mesh.

Table 1. Advancing Front vs. Advancing Steiner Point on planar well mesh.

Algorithm	Time (sec)	Minimum Angle (degrees)
Advancing Front	7.27	8.4
ASP	2.34	20.46

3.2 Wing Surface

The wing surface shown in figure 3 is a NURBS surface with a high curvature on the leading edge. Table 2 shows that there is an order of magnitude in speed

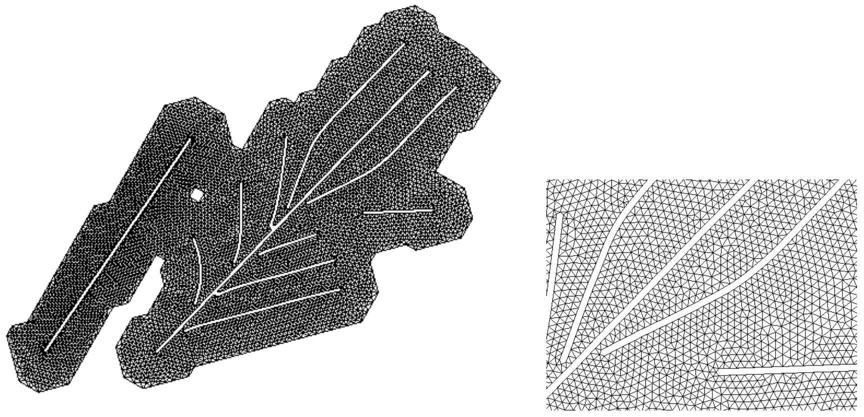

Fig. 2. Well field projected onto the z-plane and meshed with the Cubit advancing front algorithm and a closeup of the resulting mesh.

between the advancing front algorithm and the ASP algorithm. The difference in speed derives from several points. First, the ASP algorithm computes intersections in parameter space. Second, the ASP algorithm does not require an inverted element check that calculates normals on the surface for every triangle. Finally, the ASP algorithm performs smoothing in parametric space. Smoothing in real space requires a calculation to ensure that smoothed points are moved to the surface. By reducing the number of real space evaluations on the NURBS surface we see a dramatic improvement in performance.

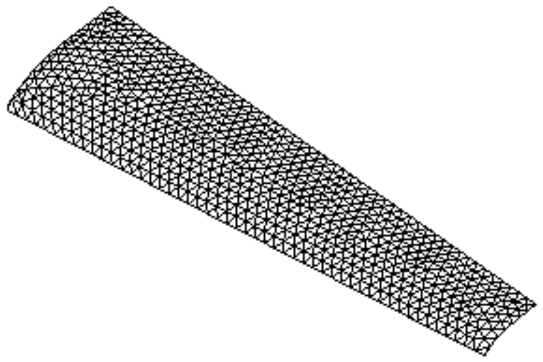

Fig. 3. NURBS wing surface meshed with the ASP algorithm.

Table 2. Advancing Front vs. Advancing Steiner Point on wing surface.

Algorithm	Time (sec)	Minimum Angle (degrees)
Advancing Front	67.03	19.1
ASP	.62	29.6

3.3 Cubit Test Suite

Cubit has an extensive set of test problems. Most of these problems are mechanical parts such as those shown in figure 4. These parts are primarily composed of analytical surfaces such as planes, cylinders, and cones. Timing results were computed for time to mesh all of the surfaces in a set of fifty-four of these test problems. The advancing front algorithm performs very well on the majority of these surfaces. There was one model that skewed the timing results. In this model the advancing front algorithm required 2500 seconds to complete the mesh. The ASP algorithm completed this same model in 20 seconds. Table 3 shows that the overall quality for these type of surfaces are comparable with the ASP algorithm being about 20% faster overall. Quality metrics for these models were also computed but the minimum angle computation is skewed because there are sliver surfaces that are not removed by either algorithm.

Table 3. Advancing Front vs. Advancing Steiner Point on 54 parts in the Cubit test suite.

Algorithm	Time (sec)	Minimum Angle (degrees)
Advancing Front	160.34	.9
ASP	198.82	.5

4 Conclusion

The current Cubit advancing front algorithm performs intersections computation in real space whereas the Advancing Steiner Point algorithm does these same calculations in parameter space. While the parameter space computation is somewhat faster, it appears that the main time drivers are related to finalizing the mesh. The Cubit advancing front algorithm has to do a normal check to ensure that the $\Re^3$ intersections have not created inverted triangles. It also does an optimization based smoothing $\Re^3$ that can be very computationally expensive. The most expensive call is a normal calculation on a NURBS surface. This same call is inexpensive on analytic surfaces and thus we see that the two algorithms are comparable on analytic surfaces.

The Advancing Steiner Point algorithm creates better elements at the closure conditions because it maintains a continuous Delaunay mesh at every

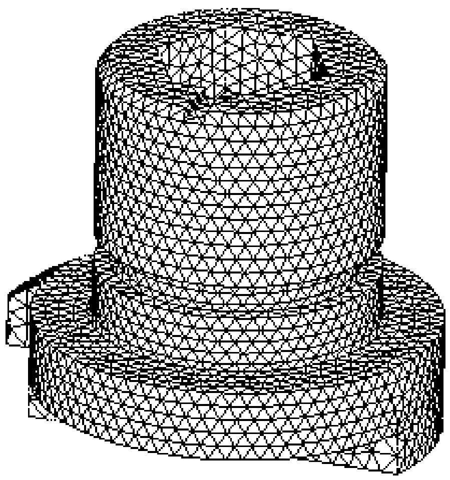

Fig. 4. Sample models from the Cubit test suite.

point in the process. Thus the closure process for the advancing is simply bounded by the element size indicating when no further points should be inserted. The Advancing Steiner Point algorithm provides both speed and improved element quality over the existing Cubit advancing front algorithm.

References

1. The CUBIT Geometry and Mesh Generation Toolkit, Sandia National Laboratories, http://cubit.sandia.gov/, 2009.
2. Alper Üngör. Off-centers: A new type of steiner points for computing size-optimal quality-guaranteed delaunay triangulations. *Comput. Geom. Theory Appl.*, 42(2):109–118, 2009.
3. S.H. Lo. A new mesh generation scheme for arbitrary planar domains. *IJNME*, 21:1403–1426, 1985.
4. T.S. Lau and S.H. Lo. Finite element mesh generation over analytical surfaces. *Computers and Structures*, 59(2):301–309, 1996.
5. Marcum and Nigel P. Weatherill. Unstructured grid generation using iterative point insertion and local reconnection. *AIAA Journal*, 33(9):1619–1625, 1995.
6. D.L. Marcum. Point insertion methods. In J.F. Thompson, B.K. Soni, and N.P. Weatherill, editors, *Handbook of grid generation*. CRC Press, Boca Raton, 1999.
7. Steven J. Owen and Sunil Saigal. Surface mesh sizing control. *International Journal for Numerical Methods in Engineering*, 47(1):497–511, 2000.