

NASA/TM-1999-208782



## **On the Use of CAD-Native Predicates and Geometry in Surface Meshing**

*M. J. Aftosmis*

---

**August 1999**

## The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the Lead Center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- **TECHNICAL PUBLICATION.** Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA's counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.
- **TECHNICAL MEMORANDUM.** Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.
- **CONTRACTOR REPORT.** Scientific and technical findings by NASA-sponsored contractors and grantees.

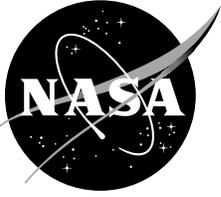
- **CONFERENCE PUBLICATION.** Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.
- **SPECIAL PUBLICATION.** Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- **TECHNICAL TRANSLATION.** English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results . . . even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at <http://www.sti.nasa.gov>
- E-mail your question via the Internet to [help@sti.nasa.gov](mailto:help@sti.nasa.gov)
- Fax your question to the NASA Access Help Desk at (301) 621-0134
- Telephone the NASA Access Help Desk at (301) 621-0390
- Write to:  
NASA Access Help Desk  
NASA Center for AeroSpace Information  
7121 Standard Drive  
Hanover, MD 21076-1320

NASA/TM-1999-208782



## **On the Use of CAD-Native Predicates and Geometry in Surface Meshing**

*M. J. Aftosmis*

*Ames Research Center, Moffett Field, California*

National Aeronautics and  
Space Administration

Ames Research Center  
Moffett Field, California 94035-1000

---

**August 1999**

### **Acknowledgments**

The CAPRI API was developed by R. Haimes under NASA Lewis Contract NAG3-2019; his participation in this work is gratefully acknowledged. The author also extends thanks to M. Delanaye, T. Pulliam, Y. Liu, and P. Walsh at NASA Ames for many productive and insightful discussions over the course of this work. Special thanks also go to J. Ruppert of Axys Pharmaceuticals for his comments on the quality surface meshing algorithm and insight into the PSLG method.

Available from:

NASA Center for AeroSpace Information  
7121 Standard Drive  
Hanover, MD 21076-1320  
(301) 621-0390

National Technical Information Service  
5285 Port Royal Road  
Springfield, VA 22161  
(703) 487-4650

# On the Use of CAD-Native Predicates and Geometry in Surface Meshing

M. J. AFTOSMIS  
Ames Research Center

## Summary

Several paradigms for accessing Computer Aided Design (CAD) geometry during surface meshing for Computational Fluid Dynamics (CFD) are discussed. File translation, inconsistent geometry engines and non-native point construction are all identified as sources of non-robustness. The paper argues in favor of accessing CAD parts and assemblies in their native format, without translation, and for the use of CAD-native predicates and constructors in surface mesh generation. The discussion also emphasizes the importance of examining the computational requirements for exact evaluation of triangulation predicates during surface meshing.

The *native* approach is demonstrated through an algorithm for the generation of closed manifold surface triangulations from CAD geometry. CAD parts and assemblies are used in their native format, and a part's native geometry engine is accessed through a modeler-independent application programming interface (API). In seeking a robust and fully automated procedure, the algorithm is based on a new physical space manifold triangulation technique specially developed to avoid robustness issues associated with poorly conditioned mappings. In addition, this approach avoids the usual ambiguities associated with floating-point predicate evaluation on constructed coordinate geometry in a mapped space. The technique is incremental, so that each new site improves the triangulation by some well defined quality measure. The algorithm terminates after achieving a prespecified measure of mesh quality and produces a triangulation such that no angle is less than a given angle bound,  $\alpha$ , or greater than  $\pi - 2\alpha$ . This result also sets bounds on the maximum vertex degree, triangle aspect-ratio and maximum stretching rate for the triangulation. In addition to the output triangulations for a variety of CAD parts, the discussion presents related theoretical results which assert the existence of such an angle bound, and demonstrate that maximum bounds of between 25° and 30° may be achieved in practice.

## 1. Introduction

Mesh generation has long been recognized as a bottleneck in the CFD process.<sup>(ref. 1)</sup> The last decade has witnessed a myriad of international and domestic conferences and symposiums aimed at focusing research on this impediment. Unstructured, hybrid, and Cartesian mesh methods are all aimed at simplifying the mesh generation task for complex configurations. The success of these approaches is well represented in the literature and with an appropriate initial surface triangulation, the volume mesh generation can generally be accomplished in a relatively automated fashion in minutes-to-hours on an engineering workstation.<sup>(refs. 2-7)</sup> As faster processors continue to shrink the wall-clock time required for both mesh generation and flow solution, the man-hour intensive task of extracting an initial surface discretization from a CAD geometry promises to become an ever larger fraction of the bottleneck. Additionally, if the user must be involved in the extraction of surface data from CAD, then mesh adaptation - which involves enriching the discretization on the body surface - will remain an elusive goal.

Historically, surface discretization has been one of the least automated steps in the numerical simulation cycle, and for good reason. Due to its dependence on implicitly defined surfaces and curves, CAD data is by its nature imprecise. Various geometry engines typically demonstrate discrepancies in their interpretations of the same entities. As a result, "repair" of CAD surfaces has become an area of substantial research.<sup>89</sup> This problem is exacerbated when CAD models are output in many of the standard formats, since such files frequently do not include important topological and construction information along with the entity geometry.

In response to these and other requirements for user assistance, many in the research and industrial CFD communities have adopted an interactive paradigm for surface mesh generation. The commercial unstructured mesh generators in references 7,10 and 11 all interact with CAD data through files which have been translated from their CAD native environment to some standardized format (namely IGES<sup>(ref. 12)</sup>, STEP<sup>(ref. 13)</sup> or STL<sup>(ref. 14)</sup>).<sup>1</sup>

This paper examines an alternative paradigm. The approach interfaces with the CAD system in a dynamic manner via calls to CAD native routines. By accessing the model in its native environment, this approach avoids translation to a format which can deplete the model of topological information. This is important since it avoids the consistency conflicts that can occur when two different geometry engines attempt to infer topological information from imprecise data.

To avoid placing CAD specific calls in the software, we argue in favor of wrapping these calls in a standardized Application Programming Interface (API) such as CAPRI.<sup>(ref. 15)</sup> This library presents a standardized interface to the application program for various CAD systems.<sup>2</sup> CAPRI supports a variety of operations like truth testing, geometry construction, and entity queries. This strategy also divides the task of software maintenance between tasks associated with the CAD system and those associated with the surface mesher.

Maintaining the consistency of the models by direct manipulation of CAD parts and assemblies is the first step that this work takes toward building a robust method for surface triangulation. A basic premise of this approach is that we resolve consistency problems on as simple a model as possible, and maintain this consistency as the triangulation evolves and becomes more complex.

---

1 Recent releases of some of this software now supports "direct" interfaces which do read parts in their native formats, however, this practice is not the norm.

2 CAPRI currently supports ProEngineer™ Unigraphics™ and SDRC I-DEAS™ with CATIA™ support in beta test.

## 1.1. Abstract Geometric Structures

Following the approach of Yap,<sup>(ref. 16)</sup> our approach toward generating a robust geometric algorithm contends that a geometric structure,  $D$ , consists of four elements:<sup>23</sup>

$$D = \left( G, \lambda, \Phi(\mathbf{z}), \mathbf{z} \right) \quad (1)$$

Where the graph,  $G = (V, E)$  is a directed set of vertices,  $V$ , and edges,  $E$ .  $\lambda$  is a function describing the index labels of the graph.  $\Phi$  is a geometric operator which represents the consistency predicates for the connectivity and is a function of the actual coordinates  $\mathbf{z}$ .  $G$  represents a tessellation of the vertices and is therefore purely combinatorial. A structure is said to be *consistent* if the predicate  $\Phi(\mathbf{z})$  holds.

As an example, a 2-D Delaunay triangulation algorithm usually makes use of an *inCircle* predicate<sup>(ref. 17)</sup>,  $\Phi_{inCircle}$  which establishes  $G$  by insisting that the circumcircle of the triangle  $\Delta_{a,b,c}$  can contain no other vertex in the graph. If such a predicate holds for every triangle in  $G$ , then this *instance* of the geometric structure  $D$  is said to be consistent. This interpretation offers direct insights into the formulation of a robust algorithm for creating triangulations of CAD volumes.

## 1.2. Robustness

### *Consistent CAD Models*

The rational B-splines used to describe surfaces in most CAD systems are implicitly defined for physical space coordinates of the geometry. Therefore, the *constructors* for vertex geometry generally require a Newton solve carried to some internal tolerance. Since the results of this construction will be subject to both tolerance and round-off error, the system may then “nudge” the constructed point,  $z_i$ , to some nearby exactly representable location (on an integer grid, for example). If the geometry engine’s predicate,  $\Phi_{onSurf}(z, S)$ , for determining if  $z_i$  is on a surface,  $S$  is consistent then it will return “true” when later queried if  $z_i$  lies on the surface. However, if  $\tilde{\Phi}_{onSurf}(z, S)$  now represents some user-defined predicate which may be ignorant of the systems construction rules, then it is very unlikely to return consistent results.

The CAPRI API ensures the maintenance of a consistent representation of the model by providing access to a subset of the CAD geometry engine’s constructors, queries and predicates. Our implementation adopts a multi-threaded programming approach which runs the geometry engine on its own thread in order to respond to queries from the main triangulation thread.

### *Physical Space Triangulation*

A variety of existing surface meshing techniques adopt a *mapped-space* approach for generating surface triangulations. In this approach, a surface and its bounding curves are triangulated in a 2-D parameter space, which may or may not have some additional scaling imposed. Reference 18 pro-

vides a mathematical description of the Non-Uniform Rational B-Spline (NURBS) surfaces typically used in CAD systems. Here we note only that an iterative method is required to solve for the physical space coordinates of a position specified on the surface in the parameter space. This process involves division of two (generally) high-order polynomials, and is therefore subject to both error associated with finite-precision arithmetic and error associated with tolerancing for the convergence of the iterative solve. As a result, computed coordinates in the mapped space are necessarily noisy and cannot be considered exact values. Two consequences of this approach are:

1. Since the error bounds on the input,  $\mathbf{z}$ , are unknown, evaluation of the triangulation predicates (e.g.  $\tilde{\Phi}_{inCircle}$ ) are unlikely to robustly produce consistent results (see figure 19 in reference 17; also references 19 and 20).
2. The polynomial basis for the NURBS may be high-order, and therefore small errors in parameter space may produce dramatic results in physical space – even within the subspace for which the surface is defined. The likelihood of encountering poorly conditioned mappings is the primary reason that CAD repair software generally attempts to renormalize the NURBS surface and recast it using basis polynomials with as low an order as possible<sup>(ref. 8)</sup>.

These two observations motivate an examination of physical space triangulation techniques. In this approach, we construct a manifold triangulation on the surface, and evaluate the triangulation predicates in  $\mathfrak{R}^3$ . New sites are constructed by the CAD geometry engine and, since this output is consistent with the system’s internal predicates, it is considered exact by the external predicates of the triangulation algorithm. The presentation in §3 emphasizes both minimization and tracing of the floating-point error in evaluation of the triangulation predicates. Computational requirements for exact evaluation are presented.

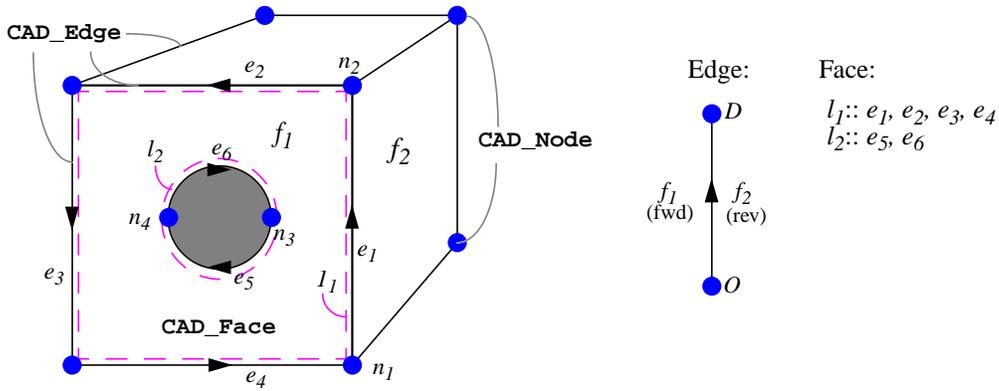
## 2. The CAPRI API

Our basic approach is to take a crude manifold triangulation of each closed volume in the CAD assembly and improve it until it either satisfies a preset measure of mesh quality or produces a preset number of triangles. A variety of mesh quality measures may be defined within this framework, and this preliminary investigation examines two such criterion: (1) the mesh must be free from small angles (sliver triangles); (2) edges in the triangulation must not deviate from the underlying model by more than a prescribed tolerance.

### 2.1. CAPRI Volumes

CAD entities are accessed through the CAPRI programming interface.<sup>(ref. 15)</sup> This API provides a layer of indirection such that CAD system specific data may be accessed by an application program using CAD system neutral function calls. Figure 1 presents an abstract view of the entities that CAPRI

3 Reference[16] actually writes eq. (1) as  $D = (G, \lambda, \Phi(\mathbf{z}), I)$  where  $I$  is a mapping from the input parameters  $\mathbf{c}$  to  $\mathbf{z}$ ,  $I: \mathbf{z} \rightarrow \mathbf{c} = (c_1, \dots, c_n) \in \mathfrak{R}^d$ .



**Figure 1.** CAPRI data structured demonstrated on a simple volume with a cylindrical cutout.

provides. A *cad\_node* is the lowest dimensional entity and corresponds to a point in 3-space. A *cad\_edge* has a *cad\_node* at both ends. Each edge is directed from its origin,  $O$ , to its destination,  $D$ . *Cad\_edges* are not assumed to be simplicial and may follow a general curve in space (see  $e_5$  and  $e_6$  in Fig.1). Each edge is connected to two *cad\_face* entities. In general, these faces are composed of several loops and are not assumed planar, since they follow the underlying parameterization of the surface. A *cad\_face* is composed of one or many *loops*, which are collections of oriented edges. Loops are oriented such that the surface of the *cad\_face* lies inside them when they are traversed in a counterclockwise circuit when viewed from a point outside the solid. This convention permits holes in a surface to be described by a clockwise loop. In Figure 1, *cad\_face*,  $f_1$ , consists of loops  $l_1$  and  $l_2$ , each of which is composed of *cad\_edge* entities. The edge ordering of  $l_2$  indicates that it is clockwise, and therefore describes a hole in  $f_1$ . Edges and faces have an underlying parameterization, and while points may be queried for their parametric values  $(u, v)$ , details of this ruling are not otherwise exposed to the application program.

**2.2. Initial Manifold Triangulation**

A central theme in the present approach is the maintenance of a closed volume throughout the procedure. For each volume, CAPRI returns a simplicial decomposition of each of the  $m$  *cad\_face* entities,  $S_i$ , where  $S_i \in \{S_1, S_2, \dots, S_m\}$ . Each of these triangulations are manifold within their respective *cad\_edges*. In addition, an indexing function,  $\lambda_C$ , is returned

for each *cad\_volume*. Therefore a simplicial, manifold representation of each *cad\_volume*,  $S_C$  may be constructed by taking the union of the decompositions of all the *cad\_face* entities of a volume, subject to the indexing  $\lambda_C$ .

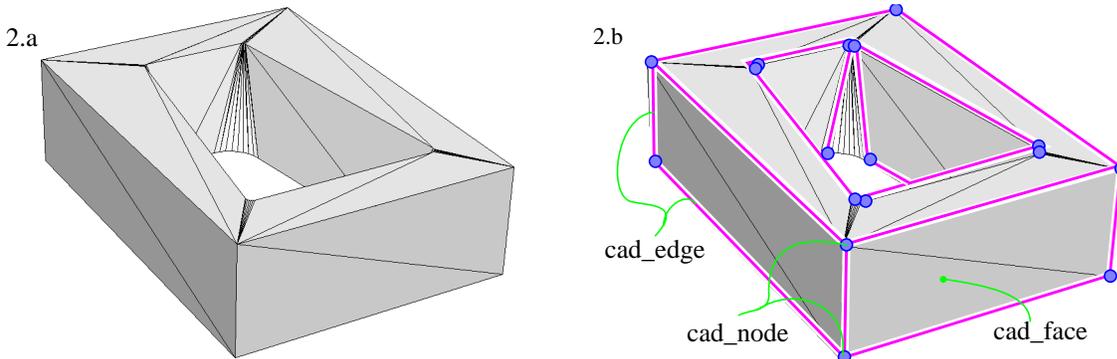
$$S_C = \bigcup_{i=1}^m S_i \tag{2}$$

Figure 2 displays an example of this initial triangulation for a simple part. The manufacturing die shown has 14 *cad\_face* entities and the initial triangulation,  $S_C$ , has 270 triangles. As is typical, this triangulation is quite irregular, and planar regions are decomposed into as few triangles as possible. Extremely high aspect ratio triangles are common in these boundary triangulations. Figure 2.b labels selected CAD entities on this triangulation. Notice that although some *cad\_face* sites may be present, this initial triangulation is essentially a boundary triangulation and the number of triangles is proportional to the number of *cad\_edges*.

Despite the poor quality of the triangulation, the structure in Figure 2 has several desirable properties. Namely, it is consistent, manifold, oriented and closed. We wish to improve this triangulation by adding sites on both the *cad\_edge* and *cad\_face* entities and by enforcing an external predicate governing the type of triangulation.

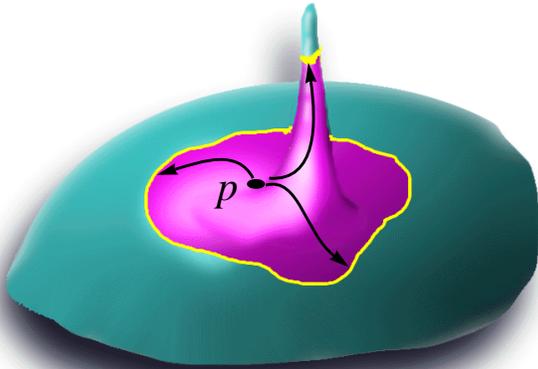
**3. Mesh Improvement**

Our approach for manifold surface triangulation traces its roots to work on *quality triangulations* of Planar Straight Line Graphs (PSLGs)<sup>(refs. 21, 22, 23)</sup> and related work on qual-



**Figure 2.** Initial closed, manifold, surface triangulation,  $S_C$ , of a CAD model for a manufacturing die. Underlying CAD entities exposed to application program are labeled in the frame on the right.





**Figure 4.** An example of nonintuitive consequences of a straightforward interpretation of the inCircle PSLG predicate on curved surfaces. With distances measured using the geodesic distance along the surface, the loci of points equidistant from  $p$  reaches around the spire but does not include its tip.

PSLG method to curved surfaces. alg.

Alg. Q has two salient aspects. (1) The triangulation is constrained Delaunay. (2) New sites are added at circumcenters. Chew observes that a straightforward definition of a circle on a surface is the loci of points on the surface which are equidistant from another point on the surface, where all distances are measured using the geodesic distance along the surface. While straightforward, this definition is problematic. Distances along the surface must be measured in physical space, and will therefore be expensive to compute on NURBS surfaces. In addition, due to the inherent error in finding the coordinates of a point on such a surface, robust predicates based upon this definition will be difficult to formulate. Finally, Chew notes that this definition has less subtle and non-intuitive consequences. As shown in Figure 4, a circle whose center lies near the base of a sharp spire, for example, may reach completely around the spire without also including the tip of the spire.

To circumvent such difficulties, the method in reference 24 makes use of an alternative definition of a circle. In the plane, the three vertices of a triangle define a unique circle. In three dimensions, however, an infinite family of spheres may be passed through those three points. Connecting the line through the centers of this family of spheres and intersecting this line with the surface identifies a particular sphere in this family. The circumcenter of the triangle may be defined to be the loci of points at the intersection of this particular sphere and the surface. Once this sphere is found, then the  $\tilde{\Phi}_{inCircle}$  triangulation predicate may be evaluated by simply computing distances in three dimensions as a vector magnitude.

Despite the effort, problems still exist with this predicate. (1) If the triangulation does not resolve the underlying surface closely enough, the line of circumsphere centers will not necessarily intersect the surface. Alternatively, it may also intersect the surface in multiple locations. (2) Computing the intersection of this line with the surface will require an itera-

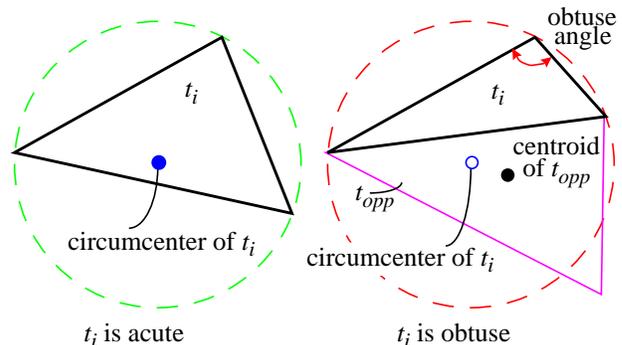
tive method and will therefore be computationally intensive. (3) Once this intersection is successfully located on the surface, one must determine which triangle the point falls inside. Since the triangulation only matches the surface at the vertices, ambiguous situations may arise when two triangles claim ownership of the same site (near a ridge, for example). (4) On a curved surface, the circumcenters of two triangles with a shared edge may not be consistent. Specifically, when  $\tilde{\Phi}_{inCircle}$  tests one of the triangles against the opposite vertex of the other, the results may not agree when the roles of the triangles are reversed. Lemma 5 in reference 24 shows this situation will arise if the normal vectors of the triangles vary by more than  $\pi/2$ . A successful algorithm must guard against  $\tilde{\Phi}_{inCircle}$  becoming inconsistent.

### 3.3. A Physical-Space Surface Meshing Algorithm

These outstanding ambiguities and computational expense motivated a search for a more manageable algorithm. Our method makes two fundamental changes.

- To avoid the ambiguity associated with the definition of  $\tilde{\Phi}_{inCircle}$ , we do not attempt a Delaunay triangulation of the curved surface. Instead we seek a triangulation which is everywhere locally maxmin. While this may seem to constitute a dramatic relaxation, recall that the goal is a practical algorithm, and we have no reason to prefer strictly Delaunay output triangulations. In addition, since one property of a Delaunay triangulation is that it is maxmin, this choice is worth investigating.
- When all angles of a triangle are acute, the circumcenter falls within the triangle. Ownership of the new site can then be uniquely assigned to this triangle. However, when a triangle is obtuse, ownership can become less clear. Therefore we make a simple choice: When a triangle is obtuse, we insert the new site at the centroid of the *other* triangle sharing the edge opposite the obtuse angle. Figure 5. illustrates this insertion rule. If  $t_i$  is an obtuse triangle, then the new sites are added at the centroid of  $t_{opp}$ .

While admittedly ad hoc, the modified insertion strategy is not as arbitrary as it may initially seem. As a particular angle of  $t_i$  opens up in the transition from acute to obtuse, the circumcenter will pass from within  $t_i$  to within  $t_{opp}$ . The centroid of  $t_{opp}$  may therefore be thought of as an *approximate*



**Figure 5.** Sites are added at the centroid of a triangle if the triangle is acute. Otherwise they are added at the centroid of the triangle opposite the obtuse angle.

circumcenter of  $t_i$ . Denoting the radius of the circumcircle of  $t_i$  as  $R$ , it can be shown that if one chooses an approximate circumcenter within a distance  $fR$  of the true circumcenter, then an angle bound of

$$\alpha \leq \text{asin}\left(\frac{1-f}{2}\right) \quad (4)$$

may still be achieved. Reference 24 contends that even using approximate circumcenters, angle bounds of  $30^\circ$  may be achieved in practice.

With these changes, the manifold triangulation algorithm becomes:

**Algorithm M:**

*Quality Manifold triangulation of a CAD volume.*

**Input:** Underlying CAD volume,  $P$ , with initial triangulation  $S_C$ , and input vertices,  $V_{in}$ .  
Target angle,  $\alpha$ .

**Output:**  $\tilde{\mathcal{CN}}(V_{out})$  with all angles  $\geq \alpha$ .

**Initialize:** Compute constrained locally maxmin triangulation  $\mathcal{CN}(V_{in})$ , using `cad_edge` entities as constraints. Build minimum angle priority queue,  $PQ_{min}$  with  $t_{PQ}$  denoting triangle at head of queue, having min angle  $\theta_{PQ}$ .

1. Apply  $\tilde{\Phi}_{Encroached}$  to all constraint edges:  
While (any constraining edge is encroached){  
    Split constrained edge. Update  $\mathcal{CN}(V)$ , Update  $PQ_{min}$ .  
}
2. While ( $\theta_{PQ} < \alpha$ ){
- 2.a If ( $t_{PQ}$  is not obtuse)  
    Assign  $t := t_{PQ}$  with circumcenter  $p$ .  
    Else Assign  $t := t_{opp}$  with centroid  $p$ .
- 2.b If ( $p$  encroaches any constraining edge,  $e$ )
- 2.c Split constrained edge. Update  $\mathcal{CN}(V)$ , Update  $PQ_{min}$ .
- 2.d Else Split triangle  $t$ :  
    Add  $p$  to  $V$ . Update  $\mathcal{CN}(V)$ , Update  $PQ_{min}$ .

When the algorithm terminates, all angles are greater than  $\alpha$ . Thus, it recovers the properties of quality and size-optimality cited after the presentation of Alg. Q in §3.1. The bound on output size, however, depends on the site insertion strategy always inserting new vertices within the circumcircle of  $t_{PQ}$ ,<sup>(ref. 23)</sup> and the modified strategy will not always guarantee this, thus the algorithm sacrifices strict proof of this property.

The new algorithm requires initialization with a transformation of the part's original constrained manifold triangulation  $S_C$  to one which is everywhere locally maxmin. If we assume the existence of a triangulation predicate  $\tilde{\Phi}_{XN}$  which can be enforced for every pair of triangles sharing an edge in the triangulation (similar to the application of  $\tilde{\Phi}_{inCircle}$ ), then this initialization may be performed with edge sweeps followed by edge-swapping when a violation is encountered. While the possibility of multiple sweeps makes this a seemingly inefficient approach, we recall that the initial complexity of  $S_C$  is only proportional to the number of `cad_edges` in the geometric structure. Thus this simplistic approach is not a problem.

Site insertion proceeds in a manner comparable to the PSLG algorithm. A convenient implementation makes use of the

incremental insertion strategy of reference 25. However, since the triangulation is no longer Delaunay, both forward and reverse propagation is necessary after edge swaps. Sites from edge or triangle splitting must be projected to their actual locations on the underlying surface, and the native constructors are used for this through the CAD API. Reference 25 gives a modified point placement strategy for splitting encroached edges. Our implementation of Alg. M adopts this strategy without modification.

### 3.4. Triangulation Predicates

Algorithm M rests on two new triangulation predicates. The first tests a triangle for an obtuse angle,  $\tilde{\Phi}_{ob}(t)$  and the second,  $\tilde{\Phi}_{XN}(e)$ , tests if the edge,  $e$ , shared by any two triangles maximizes the minimum angle in both triangles. This is accomplished by comparison with a swapped edge,  $e'$ , which connects the opposite vertices of the two triangles. Our approach hinges on the hope of evaluating these predicates robustly in physical space.

Both of these predicates can be formulated with direct measurement of the angles in a mesh. Since a triangle has three points, it uniquely defines a plane in three dimensions. Angle measurement in a plane is unambiguous – despite the fact that the triangles form a discrete manifold which is of lower dimension than the surrounding space. Numerically, we recall that since all vertex locations returned by the CAD engine are considered exact, the error bound on the input is identically zero.

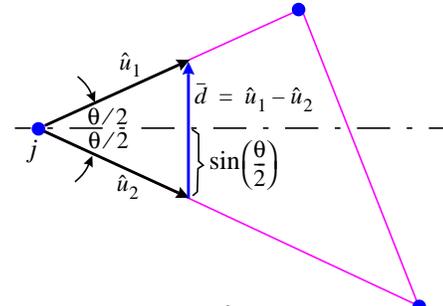
$\tilde{\Phi}_{ob}(t)$  is applied by a logical *or* accumulation of  $\tilde{\Phi}_{\angle_{j-1, j, j+1}}$ , where  $j$  is a cyclic index running over the vertices of  $t$ ,  $j \in \{1, 2, 3\}$ . This angle predicate is formed by comparison of the square of the edge opposite  $j$  to that of a hypothetical edge formed by assuming that the edges of  $t$  incident on  $j$  form a right angle in the plane of  $t$ .

$$\tilde{\Phi}_{ob}(t_{123}) = \tilde{\Phi}_{ob}(\angle_{123}) \vee \tilde{\Phi}_{ob}(\angle_{231}) \vee \tilde{\Phi}_{ob}(\angle_{312}) \quad (5)$$

where

$$\tilde{\Phi}_{\angle_{j-1, j, j+1}} = \begin{cases} T & \text{if } |\mathbf{v}_j - \mathbf{v}_{j-1}|^2 + |\mathbf{v}_{j+1} - \mathbf{v}_j|^2 > |\mathbf{v}_{j+1} - \mathbf{v}_{j-1}|^2 \\ F & \text{otherwise} \end{cases} \quad (6)$$

Notice that eq. (6) requires only subtraction and multiplication of data which is known exactly. Thus the computational



**Figure 6.** Construction for  $|d|^2$ , the square of the magnitude of the difference of unit vectors incident on vertex  $j$ .

requirements for exact evaluation are the same as for evaluation of the dot product for  $\tilde{\Phi}_{Encroached}(e)$  in §3.1.

Evaluation of  $\tilde{\Phi}_{XN}(e)$  requires a more direct method of angle measurement. Figure 6 shows a construction for this measurement. Recalling that  $\sin(\cdot)$  is monotone over the interval  $[0, \pi/2]$  the construction in the figure shows that the difference of the unit vectors of the edges incident upon any vertex,  $j$ , is sufficient to define a vector  $\bar{d}$  whose magnitude varies monotonically with the angle formed by the edges incident upon  $j$ . As in the preceding two predicates, computationally, it is sufficient to evaluate only the square of this magnitude.

The computational requirements for this predicate are somewhat less simple. The computation of unit vectors requires robust computation of the inverse of a vector magnitude  $1/|\bar{v}|$ . Thus, exact evaluation of  $\tilde{\Phi}_{XN}(e)$  requires software arithmetic, like the packages in references 26, 27 or 28. Although this need must be viewed as a drawback, we note that it is confined to a single predicate, and to a relatively simple expression. Moreover, since the input geometry is exact, exact computation remains a feasible strategy. In the preliminary results shown in §4, all computation was performed using only double-precision floating-point hardware, and the option of software arithmetic was not pursued.

### 3.5. Edge refinement

Algorithm M drives small angles out of the evolving triangulation. In addition we wish to satisfy some edge-based criterion like a chord-height tolerance. After initially creating a triangulation free of small angles we apply an edge-refinement procedure to enforce such requirements. Algorithm E considers a generic edge-based scalar  $\gamma(e)$ . In our implementation *chord-height* is defined as the square of the distance from the middle of an edge to the corresponding location on the actual surface of the model (provided through the CAD API by the geometry engine).

**Algorithm E:** Edge refinement of Manifold Triangulation.

**Input:** Underlying CAD volume,  $P$ , with current triangulation,  $CXN(V)$ , and vertex set,  $V$ .  
Edge criteria  $\gamma$ .  
**Output:**  $CXN(V_{out})$  with all edges satisfying  $\gamma(e) < \gamma$ .  
**Initialize:** Build priority queue,  $PQ_\gamma$  with  $e_{PQ}$  denoting edge at head of queue, having  $\gamma(e_{PQ})$ .

1. Apply  $\tilde{\Phi}_{Encroached}$  to all constraint edges:
  - While(any constraining edge is encroached){
  - Split constrained edge. Update  $CXN(V)$ , Update  $PQ_\gamma$ .
  - }
2. While ( $\gamma(e_{PQ}) > \gamma$ ){
  - 2.a Let  $p$  be midpoint of  $e$ .
  - 2.b If ( $p$  encroaches any constraining edge,  $e$ )
  - 2.c Split constrained edge. Update  $CXN(V)$ , Update  $PQ_\gamma$ .
  - 2.d Else Split edge  $e$ :
    - Add  $p$  to  $e$ . Update  $CXN(V)$ , Update  $PQ_\gamma$ .

3. Output  $CXN(V)$ .

Assuming that  $\gamma(e)$  is a static criterion, like a chord-height tolerance, Alg. M can then be re-applied to  $CXN(V)$  to remove any small angles created during edge refinement and

swapping.

## 4. Results and Discussion

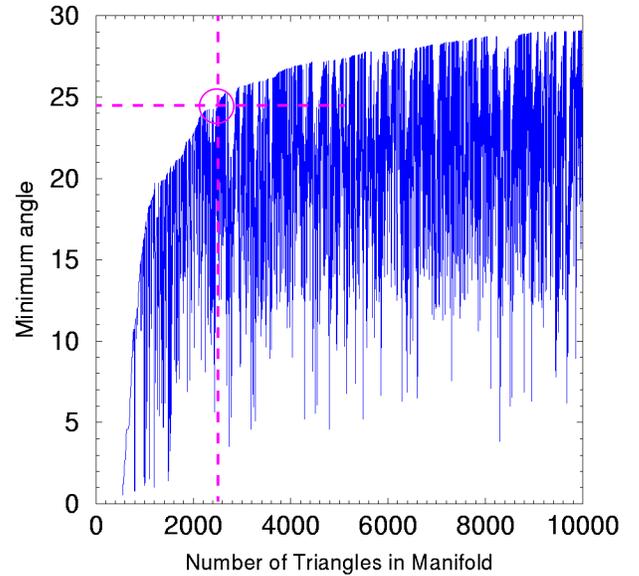
This section presents example meshes on several CAD parts from a variety of sources. All the example parts were read in their native CAD file format using the CAPRI API without special treatment. The investigations focus on examination of issues raised in the presentation of the triangulation algorithm in §3.3 and the edge refinement strategy from §3.5.

### 4.1. Minimum Angle Bound

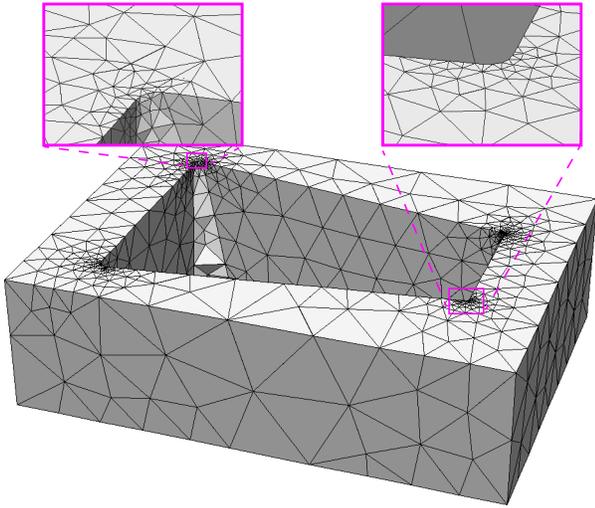
In §3.3, Algorithm M was presented without firm proof of termination. Moreover, the discussion noted that the modified site insertion strategy for obtuse triangles violates one of the assumptions that establishes a bound on the output size of the mesh in the PSLG method. It is therefore necessary to demonstrate the performance of Alg. M to show that it both terminates and produces meshes with an economy similar to that of the PSLG method upon which it is based.

Figure 7 contains a histogram of the evolution of the smallest angle in the mesh as Alg. M proceeds on the manufacturing die example problem used in earlier illustrations. While this curve is far from monotone, it clearly displays the steady improvement of the minimum angle in the mesh as the algorithm proceeds. The steep initial rise indicates rapid annihilation of extremely small angles in the mesh, and the mesh achieves a minimum angle of almost  $29^\circ$  by the end of the histogram.

The dashed line at  $25^\circ$  highlights the first time that all angles in the mesh exceeded this value. Tracing this value on the abscissa shows that setting the angle bound,  $\alpha$ , to  $25^\circ$  will cause Alg. M to terminate after generating 2606 triangles. Figure 8 shows the resulting triangulation. As discussed in



**Figure 7.** Histogram of minimum angle during mesh evolution using Alg. M (without the edge refinement of §3.5) for the manufacturing die example presented earlier. In this example, a mesh with a minimum angle of  $25^\circ$  would contain 2606 triangles.

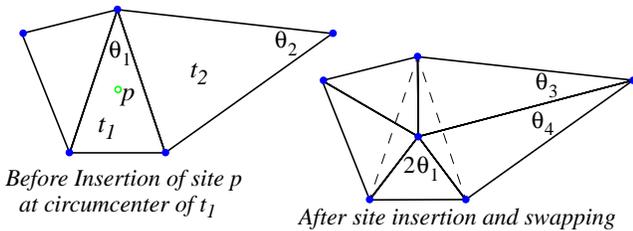


**Figure 8.** Quality manifold triangulation for manufacturing die example generated with Alg. M. in §3.3. Mesh improvement terminated after generating 2606 triangles when the minimum angle in the triangulation reached 25°. Chord-height

§3.1 and §3.3, the presence of an angle bound ensures that small features are surrounded by proportionally small triangles (see the inset frames in figure 8), and that the mesh length-scale varies smoothly over the part. The smallest angle in the mesh is 25.02°, which corresponds to a maximum aspect ratio of between 2.36 and 4.7 (by eq. (3)).

While the histogram in Figure 7 shows a steady increase in the minimum angle, there is an irregular array of downward spikes in the profile. In the presentation of the PSLG algorithm, reference 23 included a similar histogram and noted the same characteristic. Consider the two triangles  $t_1$  and  $t_2$  shown at the left in Figure 9, Assume that  $\theta_1$  is the smallest angle in the mesh lying in triangle  $t_1$ . Furthermore, assume that  $t_1$ 's face neighbor,  $t_2$ , has a small angle  $\theta_2$  opposite the shared edge which is only slightly larger than  $\theta_1$ . Site  $p$  will get added at  $t_1$ 's circumcenter, improving  $\theta_1$  to  $2\theta_1$ . After application of the maxmin predicate  $\Phi_{XN}$  on the shared edge, the swapped configuration at the right of Fig.9 may occur. This configuration includes two new triangles with minimum angles  $\theta_3$  and  $\theta_4$  either of which may now be the smallest angle in the mesh and may actually be smaller than the original angle  $\theta_1$ .

With this behavior understood, Figures 10 and 11 present angle histograms and example meshes for a more compli-

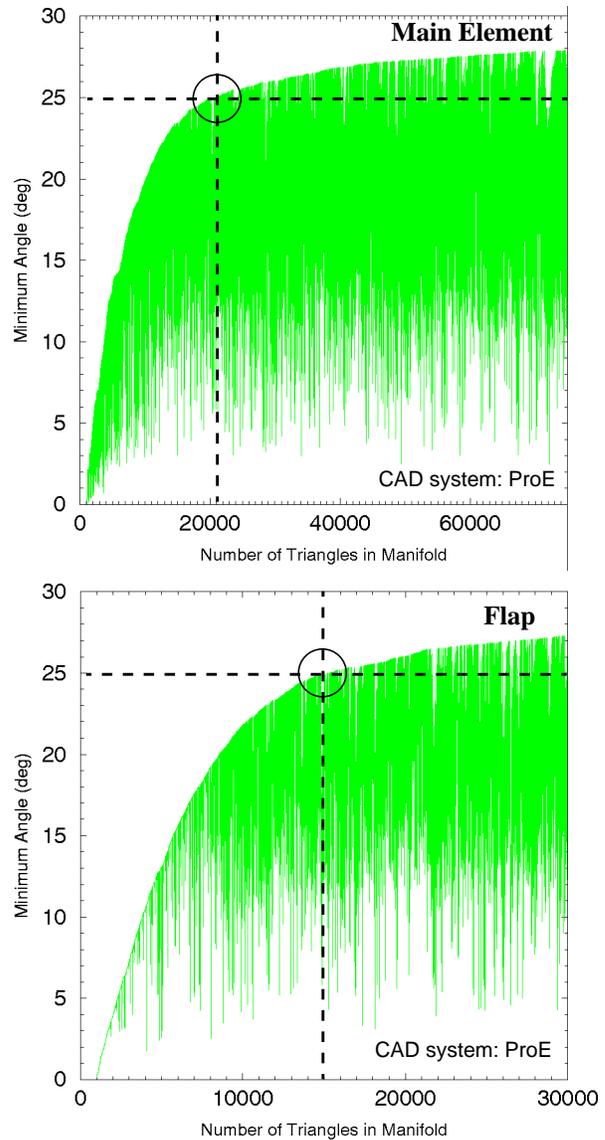


**Figure 9.** Mechanism responsible for downward spikes in histogram of minimum angle as Alg. M proceeds. If  $\theta_1$  is initially the smallest angle in the mesh, angles  $\theta_3$  and  $\theta_4$  may be smaller after insertion of  $p$  and enforcement of  $\Phi_{XN}$  by edge-swapping.

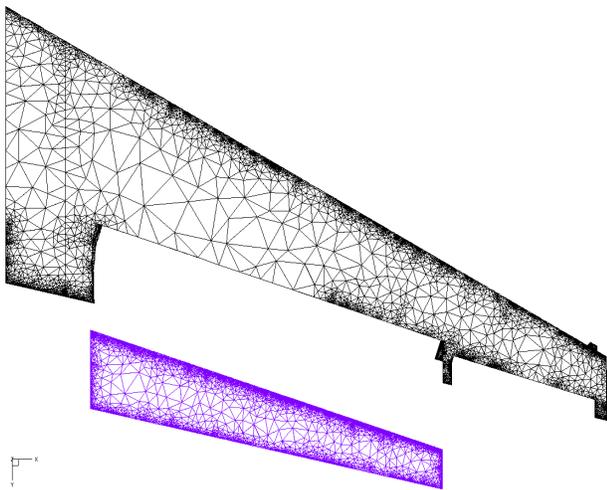
cated assembly of parts. Alg. M was run on CAD parts for the main element of a transport wing, and a flap element for the same wing. The main element consisted of 224 rational B-spline curves and 36 trimmed NURBS surfaces. The flap contained 31 rational B-spline curves and 10 trimmed NURBS surfaces.

The crosshairs on the curves in figure10 show that with a 25° angle bound, Alg. M will produce 20846 triangles on the main element and 15334 triangles on the flap. Figure 11 displays these triangulations.

The histograms in Figure 10 bear close resemblance to the one presented for the simple die example shown in figure 7. All of these profiles are characterized by a sharp initial angle improvement and then a rolling-off as the minimum angle climbs above about 20°. All three profiles exceed 27°, but



**Figure 10.** Angle histograms for Alg. M on the main element of a transport wing(upper), and on the main flap (lower). An angle bound of 25° would produce triangulations with 20846 and 15334 triangles on the main element and flap respectively.



**Figure 11.** Bounded angle triangulations of main element of a transport aircraft wing and flap generated by Alg. M in §3.3. Minimum angle  $25^\circ$ , 20846 triangles on wing, 15334 triangles on flap.

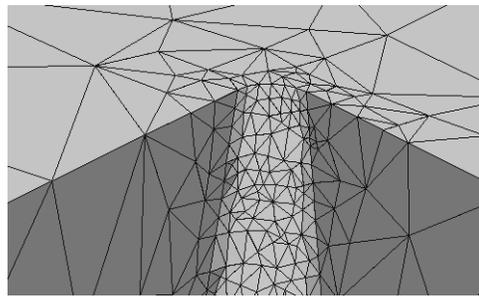
reaching  $30^\circ$  seems unlikely.

The abscissa values in figure 10 indicate relatively large triangulations as compared with the die example. The size-optimality property cited in §3.1 and §3.3 implies that triangle size must vary smoothly between different sized constraints. In both of these examples the smallest constraint in the CAPRI triangulation was a factor of  $10^4$  smaller than the largest constraining edge. In addition, practical experience with the algorithm indicates that larger initial triangulations,  $S_C$ , (eq. (2)) have a proportionally slower initial rise in their angle profiles. This seems reasonable since if the CAPRI triangulation is complex there may initially be many bad angles which need improvement. Often CAD parts are unnecessarily complex for reasons dating to the specific events in their creation. Our experience with CAD repair software indicates that CAPRI generally produces less complex (sometimes by an order of magnitude) initial triangulations if the parts have been processed by CAD repair tools. Since tolerance to poor CAD parts is one type of robustness that we seek in this research, neither part shown in Fig.11 underwent such repair prior to creation of these triangulations.

With the minimum mesh angle restricted to  $25^\circ$ , triangles are again restricted to aspect ratios less than 4.7. Such an isotropic mesh is very inefficient at meshing features with curvature in only one dimension. Therefore, if one intends to produce meshes for viscous computation, a substantially smaller angle bound may be appropriate.

#### 4.2. Chord-Height Refinement

Enforcement of the angle bound does not guarantee that the edges are refined when they are sufficiently far from the underlying surface. Alg. E in §3.5 presented an edge refinement strategy for automatically breaking edges whose midpoints are far from the geometry. Figure 12 shows an interior corner of the die example after the imposition of a chord-height tolerance of  $10^{-4}$  times  $L$ , where  $L$  is the maximum



**Figure 12.** Interior corner of manufacturing die example triangulation after imposition of a chord-height tolerance of  $1 \times 10^{-4}$  normalized by the maximum outer dimension of the part. Edge refinement was performed using Alg. E. in §3.5

outer dimension of the part. Away from the curved interior corners, the triangulation remains unchanged since the faces of the die are planar.

## 5. Conclusions and Future Work

### 5.1. Conclusions

This paper examined the direct use of CAD geometry in surface meshing. It argued in favor of accessing CAD parts and assemblies in their native format, without translation and for the use of CAD-native predicates and constructors in mesh generation. The discussion also contended that since physical-space mesh generation techniques permit exact numerical computation, they can be more robust than their mapped-space counterparts.

The feasibility of this approach was demonstrated by application to a novel, physical-space, curved surface meshing algorithm to several CAD parts. Model data was accessed in its native format using the CAPRI API. This triangulation algorithm maintained a locally maxmin triangulation and used an approximate circumcenter site insertion strategy. All triangulation predicates were formulated for evaluation in physical-space, and examples demonstrated that the method produced a bounded aspect ratio manifold triangulation. The computational requirements for exact evaluation of all triangulation predicates were discussed. The algorithm was demonstrated on CAD parts of varying complexity and reliably produced triangulations with minimum angles in excess of  $27^\circ$ .

### 5.2. Future Research

- Although Alg. M. produces meshes with generally smooth length-scale variation, there is occasionally a discernible irregularity in the triangulations. This behavior becomes more pronounced when higher angle bounds are specified. Similar behavior has been noted in the PSLG algorithm 23, although in two dimensions, the behavior seems less pronounced. One possible source for this is the abrupt change in insertion location from circumcenter to centroid (of  $t_{opp}$ ) if an obtuse triangle is encountered. Alternative strategies should be investigated since this behavior can degrade the efficiency of the triangulations.

- The initial triangulation returned by CAPRI depends

strongly upon the “history” of the CAD part. Such effects are evident in the triangulation of the main wing element in figure 11, for example, where the outboard portion of the flap cut-out is excessively resolved due to relics of the part’s creation process. The only effective strategy we’ve found for avoiding this is to “clean” the geometry using CAD repair software. Alternative approaches should be investigated.

## 6. References

- 1 Cosner, R.: Issues in Aerospace Applications of CFD Analysis. *AIAA Paper 94-0464*, Jan. 1994.
- 2 Aftosmis, M.J.; Berger, M.J.; Melton, J.E.: Robust and Efficient Cartesian Mesh Generation for Component-Based Geometry.” *AIAA Paper 97-0196*, Jan. 1997.
- 3 Löhner, R.: Finite Element Methods in CFD: Grid Generation, Adaptivity and Parallelization. *AGARD Special course on unstructured Grid methods for Advection dominated Flows*, AGARD-R-787, May, 1992.
- 4 Marcum, D. L.; and Weatherhill, N. P.: Unstructured Grid Generation Using Iterative Point Insertion and Local Reconstruction.” *AIAA J.* vol. 33, no. 9, Sept. 1995.
- 5 Pirzadeh, S.: Unstructured Viscous Grid Generation by the Advancing Layers Method. *AIAA Paper 93-3453-CP*. Jun. 1993.
- 6 Wang, Z. J.; Przekas, A.; and Hufford, G.: Adaptive Cartesian/ Adaptive Prism Grid Generation for Complex Geometry. *AIAA Paper 97-0860*, Jan. 1997.
- 7 ICEM CFD/CAE V.3.1.3: User Manual, Volume 2, Unstructured Grid Generation. ICEM Systems Inc. Berkeley CA, Nov. 1994.
- 8 Bohn J. W.; and Zozny, M. J.: Automatic CAD-model repair: Shell-closure. *Proc. Symp. On Freeform Fabrication*. Dept. of Mech. Eng.; Univ. of Texas at Austin, 1992.
- 9 Guéziec, A.; Taubin, G.; Lazarus, F.; and Horn, W.: Cutting and Stitching: Efficient Conversion of a Non-manifold Polygonal Surface to a Manifold.”, IBM-RC-20935. IBM Research Division, Yorktown Heights, NY, Jul. 1997.
- 10 VGRID: An Unstructured Tetrahedral Grid Generator Based on the Advancing Front Method. Vigyan Inc. Hampton VA. <http://www.vigyan.com>, 1998.
- 11 CFD-GEOM User’s Manual. CFD Research Corp., Huntsville AL, 1997.
- 12 Reed, K.: The Initial Graphics Exchange Specification (IGES) Version 5.1. Sept. 1991.
- 13 Industrial Automation Systems and Integration -- Product Data Representation and Exchange -- Part 1: Overview and Fundamental Principles. *ISO/TR 10303-1*. International Standards Org. Genève, Switzerland, 1994.
- 14 3D Systems Inc.: Stereolithography Interface Format Specification. 1988.
- 15 Haimes, R.; and Follen, G.: Computational Analysis PRogramming Interface. *Proceedings of the 6th International Conference on Numerical Grid Generation in Computational Field Simulations*, Eds. Cross, Eiseman, Hauser, Soni and Thompson, July 1998.
- 16 Yap, C.: Robust Geometric Computation. *CRC Handbook of Discrete and Computational Geometry* Eds. Goodman J.E.; and O’Rourke, J., CRC Press, Boca Raton Fl.; 1997.
- 17 Shewchuk, J.R.: Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates. *CMU-CS-96-140*. Carnegie Mellon Univ. School of Computer Science, May, 1996.
- 18 Farin, G.: Curves and Surfaces for Computer Aided Geometry Design, A Practical Guide. Academic Press, 1990.
- 19 Michelucci, D.: The Robustness Issue. Internal Report, Laboratoire d’Image de Synthèse de St Etienne, France. See <http://www.emse.fr/~micheluc/english/michelucci.html>
- 20 Goldberg, D.: What Every Computer Scientist Should Know About Floating-Point Arithmetic. *ACM Comput. Surveys*, vol 23, no. 1, pp. 5-48, Mar. 1991.
- 21 Bern, M.; Eppstein, D.; and Gilbert, J.R.: Provably Good Mesh Generation. in *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*. pp. 231-241, IEEE, St. Louis, Missouri, Oct. 1990.
- 22 Chew, L.P.: Guaranteed-Quality Triangular Meshes. Technical Report, *TR-89-983*. Computer Science Dept, Cornell Univ. Apr. 1989.
- 23 Ruppert, J.: A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation. *J. of Algorithms*, vol. 18, no. 3, pp. 548-585, May 1995.
- 24 Chew, L.P.: “Guaranteed-quality mesh generation for curved surfaces.” *Proc. of the Ninth Annual Symposium on Computational Geometry*, 274-280. ACM, 1993.
- 25 Green, P.J.; and Sibson, R.: Computing the Dirichlet Tessellation in the Plane. *The Computer J.* vol. 2, no. 21, pp. 168-173, 1977.
- 26 Real/Expr homepage: Source Code, Documentation, Examples and Literature. <http://simulation.nyu.edu/projects/exact>, 1996.
- 27 Ouchi, K.: Real/Expr: Implementation of an exact computation package. Master’s Thesis, New York Univ., Dept. of Comp. Sci., Courant Inst. NY. Jan. 1997.
- 28 Trimaran homepage: <http://react-ilp.nyu.edu>. 1998.

# REPORT DOCUMENTATION PAGE

Form Approved  
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE August 1999	3. REPORT TYPE AND DATES COVERED Technical Memorandum	
4. TITLE AND SUBTITLE On the Use of CAD-Native Predicates and Geometry in Surface Meshing			5. FUNDING NUMBERS  522-31-12	
6. AUTHOR(S) M. J. Aftosmis				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Ames Research Center Moffett Field, CA 94035-1000			8. PERFORMING ORGANIZATION REPORT NUMBER  A-99V0022	
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-1999-208782	
11. SUPPLEMENTARY NOTES Point of Contact: M. J. Aftosmis, Ames Research Center, MS T27B-2, Moffett Field, CA 94035-1000 (650) 604-4499				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified — Unlimited Subject Category 61 Availability: NASA CASI (301) 621-0390			12b. DISTRIBUTION CODE  Distribution: Standard	
13. ABSTRACT (Maximum 200 words) Several paradigms for accessing computer-aided design (CAD) geometry during surface meshing for computational fluid dynamics are discussed. File translation, inconsistent geometry engines, and nonnative point construction are all identified as sources of nonrobustness. The paper argues in favor of accessing CAD parts and assemblies in their native format, without translation, and for the use of CAD-native predicates and constructors in surface mesh generation. The discussion also emphasizes the importance of examining the computational requirements for exact evaluation of triangulation predicates during surface meshing.				
14. SUBJECT TERMS Triangulation algorithm, CAD, Curved surface			15. NUMBER OF PAGES 15	
			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	