THE GENERATION OF HEXAHEDRAL MESHES FOR ASSEMBLY GEOMETRIES: A SURVEY

Timothy J. Tautges

Sandia National Laboratories, Albuquerque, NM.

Address: 1500 Engineering Dr., Madison, WI 53706

Email: tjtautg@sandia.gov

Telephone/FAX: 608-263-8485; 608-263-4499

ABSTRACT

The finite element method is being used today to model component assemblies in a wide variety of application areas, including structural mechanics, fluid simulations, and others. Generating hexahedral meshes for these assemblies usually requires the use of geometry decomposition, with different meshing algorithms applied to different regions. While the primary motivation for this approach remains the lack of an automatic, reliable all-hexahedral meshing algorithm, requirements in mesh quality and mesh configuration for typical analyses are also factors. For these reasons, this approach is also sometimes required when producing other types of unstructured meshes. This paper will review progress to date in automating many parts of the hex meshing process, which has halved the time to produce all-hex meshes for large assemblies. Particular issues which have been exposed due to this progress will also be discussed, along with their applicability to the general unstructured meshing problem.

Keywords: mesh generation, hexahedral, quadrilateral, geometry decomposition

1 SANDIA IS A MULTIPROGRAM LABORATORY OPERATED BY SANDIA CORPORATION, A LOCKHEED MARTIN COMPANY, FOR THE UNITED STATES DEPARTMENT OF ENERGY UNDER CONTRACT DE-AC04-94AL85000.
1. INTRODUCTION

Once considered a tool for performing a-posteriori failure analysis, finite element analysis is becoming more common during the design phase; and indeed is just starting to be substituted for physical prototyping in the product development process [1]. This situation is the result of increases in computer speed over the last decade, and by advances in FEA capabilities during that time. One advantage of FEA over physical prototyping is that, in principle, it can respond almost immediately to incremental changes in product design. For this to be true in practice, however, the entire FEA process must be able to respond quickly to such changes.

One of the barriers to a true simulation-based design process is the time required to construct finite element models. This process dominates the analysis time, accounting for over 90% of that time in some cases [2]. There are many bottlenecks in the model construction process: geometry cleanup, geometry simplification or de-featureing, and generation of a finite element mesh. Improvements in all these areas are necessary to bring the model construction time down to reasonable levels. However, for hexahedral finite element models, the mesh generation process remains the most time-consuming task in this process.

Along with increasing numbers of design applications of FEA, analysis of assembly models is also becoming more common. The most obvious reason for this is that these analyses can now be completed in reasonable amounts of time, and thus the analysis models are becoming more representative of the components being analyzed: these components are often assemblies of parts rather than single parts. Another, indirect reason for assembly models becoming more common can be attributed to the particular approach to hex meshing used in practice; this will be discussed more later in the paper. Assembly models are distinguished from a collection of single parts by the fact that they are usually modeled using shared surfaces between the parts. These surfaces tend to couple the model between parts, which complicates the mesh generation process.

Although there is still no universal algorithm for generating all-hexahedral meshes for general geometries, there are other tools and algorithms developed recently which have substantially reduced the time to mesh assemblies. The purpose of this paper is to describe these algorithms, and quantify the reduction in time to mesh using real examples. This paper goes on to show why, even with an all-hexahedral meshing algorithm, these other tools will continue to play a role in all-hex meshing. Finally, specific needs in both mesh generation and FEA are described which will further reduce the time to mesh for assemblies.

This paper is arranged as follows. Section 2 describes in more detail the divide and conquer approach to hex meshing of assembly models, and gives some insight into why hex meshing these models is difficult. Section 3 describes recent progress automating the hex meshing of assembly models, not with progress in a single area, but across a range of algorithms and tools. Section 4 describes further opportunities for reducing the time to mesh for these models. Section 5 gives conclusions and future directions.
DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, make any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.
DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.
Hexahedral Mesh Generation

Hex meshing has been a very active research area in the meshing community for some time. Algorithms can be classified loosely into inside-out or outside-in, and decomposition-based varieties.

So-called inside-out algorithms attempt to mesh the inside of a part with a regular mesh and fit the outer layers of mesh to the boundary geometry and topology. Notable efforts in this area are by Schneiders [3] and Tagavi [4]. These algorithms, while typically fairly robust, result in a mesh with poor quality near the boundary, where typically high-quality mesh is required. Furthermore, some of these algorithms [4] resulted in very large meshes or suffered from unreasonable runtimes. More recently, Dhondt has extended the work of Schneiders and thereby increased its robustness[5]; however, in addition to the quality issues mentioned earlier, this approach still suffers from the difficulty of matching a pre-existing mesh on the boundary. It is shown later in this paper that even with a good all-hexahedral meshing algorithm, matching a pre-existing boundary mesh is still a critical requirement.

Decomposition-based approaches to all-hex meshing attempt to decompose the part into simpler pieces, all of which are meshable using primitive hex meshing operations. Armstrong et. al used a medial surface as the basis of their decomposition [6][7], while Holmes attempted to infer internal features [8]. Both these methods suffered from a lack of robustness for general geometries and topologies. More recently, Sheffer et. al use a modified medial axis approach to improve the robustness [9][10]; while this approach shows more promise, there may still be issues with mesh size and there will still be complications with pre-existing boundary mesh.

Outside-in algorithms, in essence advancing front algorithms, attempt to mesh from the boundary inward. These algorithms attempt to generate high-quality mesh on the boundary, leaving irregular mesh on the inside of the part. Several approaches based on the dual of an all-hexahedral mesh have emerged [11][12][13]. However, the robustness of Muller-Hannemann’s method has not been demonstrated, while Folwell et. al’s approach suffers from poor mesh quality (due to highly unstructured concentrations of mesh) and unreasonable runtimes. Both approaches currently require modification of the bounding mesh, although those performed by Whisker Weaving are incremental in nature.

In summary, although there are several promising algorithms currently being investigated, no one algorithm has delivered the desired levels of mesh quality, boundary sensitivity, and reasonable runtimes. Therefore, the overall approach to all-hexahedral meshing of assembly geometries, based on a divide and conquer technique, has remained unchanged for some time. This is in contrast to the tetrahedral meshing area, where several good algorithms exist [14][15], and have been incorporated into
commercial tools[15][16]. Fortunately, recent advances in the tools and algorithms used for decomposition-based hex meshing have met the overall goal of reducing the time to mesh.

**Assembly Model Mesh Generation**

Before going into detail on recent advances in hex meshing, it is instructive to discuss the divide and conquer and general assembly meshing techniques in more detail.

In the divide and conquer approach to hex meshing, a part is decomposed into smaller pieces, each of which is presumably easier to mesh than the original part. Decomposition is continued until all pieces can be meshed with existing algorithms, e.g. mapping [17], submapping [18] and sweeping or mesh extrusion [19]. This process is depicted for a simple example in Figure 1, and results in an assembly of parts sharing common surfaces.

Although geometry decomposition would be unnecessary with a suitable automatic hex meshing algorithm, there are two reasons why this approach is still desirable. Decomposing a part into multiple pieces allows the use of different meshing algorithms on different pieces. Typically, the more structured the algorithm, the higher the resulting mesh quality. Therefore, the decomposition approach allows the use of more structured hex meshing algorithms over more of the geometry than would be possible in the absence of decomposition. This process is depicted in Figure 2. Furthermore, these structured algorithms tend to run faster; therefore, decomposition also improves meshing time, both by allowing more use of $O(n)$ meshing algorithms, and by reducing the magnitude of $n$ where $O(n \text{log} n)$ algorithms must still be used. Although these time savings are not significant in the current decomposition-based approach to hex meshing, they could become so with other advances in the approach, some of which are discussed later in this paper.

In essence, geometries resulting from the decomposition approach are no different than assemblies representing real components. Several recent component assemblies analyzed at Sandia are shown in Figure 3. So, even in the absence of a decomposition-based approach to hex meshing, component assemblies will result simply from the generation of higher-fidelity analysis models. Therefore, many of the techniques which reduce the time to mesh these models will apply both before and after the discovery of a suitable all-hex meshing algorithm.

Assembly models with shared surfaces complicate the process of meshing for several reasons. First, sharing a geometric surface between two volumes typically means the surface mesh must be shared between the volumes as well; this couples the meshes in neighboring volumes, and for hexahedral meshes this coupling can propagate across the entire assembly. The reason for this propagation in hex meshes is most easily visualized using the Spatial Twist Continuum definition of a "chord" [20]. A chord defines a column of hexes, with each hex sharing a face with the next (see Figure 4). Chords begin and end with faces on the surface mesh, or on the same internal face: moreover, they cannot end on an internal edge or pair of edges. Thus, for every mesh...
face on a particular surface, there is a corresponding chord, and that chord ends on another mesh face. on that surface or another one. These chords propagate across a volume, and in the presence of shared surfaces can propagate across the entire assembly. This means that changing the mesh on one side of a large assembly mesh may affect the mesh throughout the assembly.

Furthermore, in the case of hex meshes, some decomposition surfaces (which were inserted to make a volume meshable) can also constrain the mesh, making it difficult to generate high-quality mesh in those regions. Because at this time decomposition surfaces in CUBIT are treated as if they are part of the geometric model definition, these surfaces cannot be moved to improve mesh quality locally. Decomposition implemented with virtual surfaces would improve this situation.

So, we are faced with a situation where assembly models are becoming more prevalent, both for good reasons and for not-so-good reasons. Developing techniques for meshing assembly models will improve meshability in the short term, and in the long term will improve mesh quality, speed and fidelity to original designs.

**Why All-Hexahedral Meshes?**

Currently, it is much more difficult to generate all-hexahedral meshes than it is to generate mixed-element or all-tetrahedral meshes. The natural question then becomes, why do we want all-hex meshes? While there have been no comprehensive studies on the subject, we offer several reasons for wanting hex meshes. First, the few studies that have been published on this subject have shown that hex meshes can be more accurate, for some problems and for a given number of degrees of freedom [22][23]. Sandia is sponsoring more comprehensive studies like these; preliminary results seem to reinforce this conclusion [24]. Second, for a given characteristic edge length and similar analysis accuracy, hex meshes are more computationally efficient. Various examples indicate factors of 4-10 improved performance of hex meshes in these situations. This is a significant difference, especially for sensitivity analyses or when computing resources are limited. Third, a hex mesh can mimic the structure of the geometry being modeled. This can be useful for special applications like viscous CFD. However, for the purposes of the CUBIT project in the context of the ASCI program, the fourth reason overrides the first three: analysts either prefer or require hex meshes, for many reasons (some good and others not so good). Therefore, we must find ways of generating all-hexahedral meshes with reasonable time and effort.

**3. PROGRESS ON HEXAHEDRAL MESH GENERATION FOR ASSEMBLY MODELS**

By its very nature, a decomposition-based approach requires the use of a collection of meshing tools and algorithms. Reducing the overall time to mesh requires improvements in not just a single tool but in all the tools used in this process. The degree of

---

2 Here, we assume single point-integrated, or constant-strain, hex elements. These elements have proven effective for non-linear structural mechanics applications, and are available in many commercial FEA codes.
improvement required in each tool depends on whether that tool is on the critical path in the meshing process, and if so on how much of that critical path time is spent in that tool. This is very analogous to the application of parallel processing to solve a graph of dependent tasks [25]. For this reason, the progress described in this section is not confined to a single area, but covers many different ones.

The meshing process has global components which propagate across assemblies; this is particularly true for hex meshes. Global problems occur in other areas of computational simulation, and are problematic because they increase computational complexity (i.e. cpu time). However, in the case of FE model construction, global problems are also more difficult because they increase the complexity of the problem seen by the user, and this complexity causes a much-increased time to mesh. Reducing the time to mesh for these models can be achieved in two ways: by reducing or eliminating the global coupling through the assembly, and by increasing the automation of both global and local components of the meshing process. In essence, the user must be removed from as much of the meshing process as possible (not a bad thing, according to most users). The recent progress described here can be attributed to these two strategies. Several specific accomplishments in these areas are described below; other opportunities for further automation that have not yet been resolved are described in the following section.

**Model Problem**

A simple example will be used to illustrate recent improvements in the assembly hex mesh generation process. This model is referred to as the Kim1 model, and is shown in Figure 5, left; this part was taken from the National Design Repository at Drexel University [26]. Note that this part starts out as a single part. The geometry is complicated by the existence of through-holes and blind-holes in the “arm” sections, as illustrated in Figure 5, right.

**Decomposition techniques**

The primary purpose of decomposition in hex meshing is to make a model meshable. However, decomposition is also used to improve mesh quality in specific regions. In fact, these two issues are inter-dependent, because the definition of “meshable” depends a certain amount on mesh quality. The decomposition process is very user-intensive, because it uses human reasoning to decide both where to perform decomposition and when to stop decomposing a given part.

Progress has been made in both automatic and manual decomposition. Lu et. al describe an automatic decomposition approach which uses surface topology and local geometric information to infer features in a part [27][28]. This technique, when applied to the model problem above, yields the decomposition shown in Figure 6, left. A key aspect of this approach is that the resulting decomposition is not required to be 100% meshable; remaining pieces are left for further manual decomposition or for an automatic algorithm. This strategy has also been embraced by Sheffer in her Embedded Voronoi Graph algorithm [10].
After automatic decomposition, many of the resulting volumes in the model problem are meshable, but some are not; unmeshable volumes are identified in Figure 6, right. Because we have no automatic all-hex meshing algorithm, we are forced to use manual decomposition on the remaining pieces. Although manual decomposition is essentially a geometric operation, support for this procedure does not exist in most CAD systems. Therefore, manual decomposition has been implemented in the CUBIT mesh generation toolkit; decomposition is accomplished by cutting with:

- Semi-infinite planar, cylindrical surfaces
- Extended existing CAD surfaces
- Existing CAD bodies

These capabilities have a low level of automation, but still decrease the amount of user time spent decomposing a solid model, thereby decreasing the time to mesh. The model problem required six additional cuts to make all the volumes meshable; the result is shown in Figure 7, left. Note that the planar cuts were extended through all the volumes, even though they were only necessary in the arm sections; this was to minimize user effort in specifying the volumes to cut, and probably degraded the mesh quality slightly.

An important goal when implementing decomposition capability is minimize the effort required to define the cutting surface or tool, since this reduces user effort. Attempts to implement a decomposition process inside the Pro/Mold portion of Pro/Engineer failed in this regard, rendering that capability too cumbersome to use in practice [29]. PTC has since begun implementing a decomposition capability in the Pro/Engineer product itself [30].

**Automatic scheme selection**

The decomposition-based approach to hex meshing results in a large number of volumes in the final model. For example, several recent meshing problems have resulted in factors of 4-10 increase in the number of volumes [31][32]. Typically these volumes are meshed with a variety of different meshing algorithms, depending on their geometry and topology. In the past, each of these volumes required manual specification of meshing technique or algorithm, along with data specific to the chosen algorithm. For example, if mesh sweeping were chosen, the source and target surfaces of the sweep also needed to be specified. This process was error-prone, and also required the user to "touch" each and every volume, which increased the complexity of the overall task.

Recently, a technique for selecting meshing algorithms based on surface geometry and topology has been developed [33]. This "automatic sweep detect" algorithm is used to separate a large group of volumes into groups of meshable and unmeshable volumes; this has served as a powerful filtering tool to guide the user to which volumes need further attention. For example, on a recent meshing problem, this tool filtered out 90% of the volumes in a decomposed model that could be meshed with mapping.
submapping and sweeping algorithms, leaving only 10% of the volumes for further inspection by the user [32]. This tool was used to determine the unmeshable volumes in Figure 7, left. The automatic scheme selection tool could also be used to determine stopping criteria for automatic decomposition, and is already being used in the Volume Interval Matching algorithm described in the next section.

**Interval assignment**

Because of the way quadrilateral and hexahedral mesh propagates across a volume, there are certain constraints that must be met before a surface or volume can be meshed with these elements. The simplest of these constraints is the even-interval constraint. Consider a surface meshed with quadrilaterals. Starting with a mesh edge on a boundary, draw the dual edge, a "surface chord", proceeding from edge to opposite edge on the quadrilaterals. Since the mesh is composed entirely of quads, chords starting on the boundary must end somewhere else on the boundary (there are also chords that begin and end on the same internal edge). The result is that in order for a surface to be meshable with quadrilaterals, it must be bounded by an even number of mesh edges. The number of mesh edges on a geometric curve is referred to as the number of "intervals" on that curve. The evenness constraint for a surface places a constraint on the intervals on each of the bounding curves, referred to as a sum-even constraint.

In addition to the sum-even constraint, certain quadrilateral meshing algorithms impose other constraints on the intervals of bounding curves. For example, the mapping algorithm requires that curves on opposite "sides" of the map have the same number of intervals (see Figure 8); there are similar constraints for the triangle primitive and other surface quadrilateral meshing algorithms [34]. These constraints can be relatively light (sum-even) or rather heavy (opposite side equal). Since geometric curves can be shared by two or more geometric surfaces, the number of intervals on a given curve must satisfy constraints on all surfaces to which it is connected. These constraints propagate across the assembly because of the shared surfaces. In order to ensure that all constraints are met, the "interval assignment" problem must be solved globally.

One method used to avoid interval constraints altogether has been to only allow one geometric curve per "side" of the map [35]: this reduces the interval assignment problem to specifying an interval count for each independent loop of geometric curves in the model. However, this also results in the propagation of decompositions across the entire model, which increases the complexity of the overall geometry.

A linear programming technique has been used to solve the global interval matching problem [34]. More recent advances have increased both the quality of the interval assignment and the runtime of that solution, still based on a linear programming method: this method has been used to mesh thousands of surfaces in a large assembly mesh [32].

In addition to surface meshing algorithm constraints, recent work has shown there to be constraints imposed by certain volume meshing algorithms. Specifically, sweeping a solid containing through-holes requires that the number of intervals along one path
be equal to those on all other paths. Automating the solution of this interval assignment problem can greatly simplify the meshing of a part such as that shown in Figure 9, where there are many curves spanning each sweep path [36]. Other volume meshing algorithms may add constraints as well; for example, it has been shown that there are additional surface mesh constraints in order for volumes with genus greater than zero to be meshable [37]. The implementation described in [34] allows new constraints to be added to the interval matching solution with little effort.

The solution to the interval matching problem is instructive since it demonstrates the separation of a difficult meshing problem into local and global components. The constraints on individual curves in the model are based on characteristics local to the surfaces and possibly volumes to which they are attached. Once the meshing schemes have been assigned (automatically), enumeration of the constraints can also be performed automatically, and can be adjusted by the user locally if desired. The global part of the problem is left for a global solution step, which is efficient in part because it solves nothing but the global aspects of the problem. Many of the techniques being used to reduce the time to mesh consist of separating the problem into local and global components, and solving each separately.

**Robust, high-quality sweeping**

One of the most important algorithms for hex meshing of assembly models is sweeping, where a surface mesh is extruded into 3D elements. This algorithm often is used on more than 50% of the volumes resulting from geometry decomposition, and the volumes being swept are of increasing difficult (i.e. both the sweep direction and cross-section are varying, sometimes simultaneously). This makes the robustness and mesh quality of the sweeping algorithm of utmost importance. An example of how important this tool is is given in the next section.

Sweeping techniques have been used for some time to generate all-hexahedral meshes [38] [41]. However, several recent developments have increased the demands placed on sweeping algorithms. First, while in the past users defined geometry and mesh at the same time, often using linear or rotated sweeps [38], many meshing tools are now based on solid model geometry, and this geometry allows arbitrary combinations of surface types to define volumes [16][39]. An example of a complicated swept volume is shown in Figure 10. Second, experience has shown that as the capabilities of automatic decomposition and automatic sweep detection improve, the complexity of the volumes assigned the sweep scheme also increases. This may be due to removing the user from the scheme selection process, who in the past has limited the volumes sent to the sweeping algorithm to relatively simple ones.

Several improved sweeping algorithms have appeared recently which address these increased demands. The Cooper Tool uses a least-squares averaging technique to handle varying sweep cross-sections and sweep directions [40], while the BMSweep algorithm uses barycentric coordinates in a background mesh for projecting source nodes through the sweep [41]. CUBIT’s
sweep tool uses a weighted Winslow projection technique to account for variations in both the source surface and cross section geometries [19]. These sweep tools each possess strengths and weaknesses that are problem-dependent or are relatively undocumented. In addition, meshing approaches based on automated decomposition are beginning to target sweeping as an available meshing technique [10][28].

Examples of parts swept using CUBIT’s sweep tool are shown in Figure 10: these parts are clearly more challenging than simple translate or rotate extrusions.

**Toolkits**

As stated earlier, a decomposition-based hex meshing approach relies on many different tools. Although not obvious, a large benefit is derived from having these tools available in a single meshing package. Experience shows that meshing complex assemblies is an interactive and iterative process, in which the users apply both automated tools and manual selection of tools to obtain a mesh of suitable quality. This process is reflected by the appearance of more comprehensive meshing tools in the commercial marketplace, e.g. ANSYS [16], GAMBIT [39], and ABAQUS/CAE [42], as well as the continued development of the CUBIT mesh generation toolkit [44].

Constructing such mesh generation toolkits also allows program-level coupling between meshing tools. Examples of this type of coupling include:

- Volume Interval Matching: uses sweep loops, which are part of the automatic scheme detection algorithm in CUBIT [33][36].

- Multisweep: implemented in CUBIT using virtual geometry, in order to take advantage of automatic scheme selection, interval matching, and other geometry-based tools[45].

Coupling at this level would be impossible without a toolkit-based implementation.

**Model Problem Summary**

Using the advances discussed in this section, and due to other improvements in robustness and user friendliness, the Kim1 problem was meshed in approximately one hour of user effort; the resulting mesh is shown in Figure 7, right. The total number of volumes after automatic and manual decomposition were 17 and 59, respectively. There were 2396 elements in the final mesh.
4. EXAMPLES

Using the techniques described above, the time required to mesh complex assemblies has been reduced significantly. This can be demonstrated using time to mesh data for two recent mesh generation efforts, both based on meshing a Neutron Generator (NG) component.

The first effort, referred to as the “Power Supply”, required meshing the lower half of the Neutron Generator [31]. This was a challenging meshing problem for two reasons: first, there was a large variation in part sizes across the model, approximately 1:200; and second, all the parts in this model were surrounded by an insulating material, which coupled the mesh across the assembly. Figure 11 shows the outer portion of this model, consisting mostly of the insulating material (further details of the geometry are classified). The Power Supply mesh generation effort took sixteen months of man-effort distributed over six months of wall clock time.

The second effort, referred to as the “Tube”, required meshing the upper half of the NG [32]. This section was similar to the Power Supply, in that it was surrounded by an insulating material; in addition, the Tube contained more small parts than the Power Supply. Figure 12 shows an upper section of this model: just above this section is a cylindrical resistor, oriented transversely, and surrounding it all but not shown is the insulating material. The Tube mesh was completed in six months of effort distributed over two month’s time.

Various metrics of the two problems are shown in Table 1 which help characterize the complexity of both meshing tasks. Clearly, the tasks are of similar overall complexity.

The time spent in various portions of the meshing effort for each problem are shown in Table 2. Several conclusions can be made from this data. First, the effort is distributed fairly evenly across many tasks. This implies that in order to decrease the overall time to mesh, many tasks must be optimized. Also, it is clear that the distribution of time did not change much between the Power Supply and Tube efforts. This is a result of careful planning of the CUBIT project, where effort was distributed based on the expected reduction in overall time to mesh of each task.

It is clear from Table 1 and Table 2 that, over a course of one year, the time to mesh was reduced by a factor of 2.5, as measured by meshing two problems of similar complexity. This improvement was due to the new techniques described in the previous section, as well as improvements in the overall robustness and usability of CUBIT as a whole. Note that the two problems were meshed by two teams of people, with no overlap in team members; therefore, increased experience of individual personnel was not a factor in decreasing the time to mesh. To be fair, there was however a general improvement in the state of knowledge about meshing real applications that did factor into this improvement.
Finally, an important point is that the origin of new ideas and the prioritization of work on the CUBIT project over this period and into the near future can be attributed largely to these two meshing efforts. These efforts not only provided a means for measuring progress in time to mesh, but also served as a genesis for new ideas for reducing the time to mesh.

5. NEEDS

While the process of generating all-hex meshes for complex geometries has become significantly easier in recent years, a large amount of progress still needs to be made to bring model construction costs in line with analysis costs. Improvements in several specific areas are needed.

Mesh Skew Control

Because sweeping is a workhorse algorithm for meshing large assembly models, these models typically have many mapped and submapped surfaces, coupled through the assembly. Since multiple curves per side of the map or submap are allowed in CUBIT, this can challenge the interval assignment algorithm. For example, in Figure 13 left, mapping the three surfaces with a contiguous mesh at a coarse resolution results in some skewing of the mesh on all three surfaces. While in this case the skew can be removed by adding more intervals (see Figure 13, right), this process can be difficult with larger assemblies because of the added coupling. Resolving mesh quality problems introduced by mesh skew required a substantial effort in the NG Tube meshing effort described earlier.

Solutions to the mesh skew problem are being explored at BYU on behalf of the CUBIT project. While a workable solution has not yet been completed, it will likely include adding virtual vertices, new interval assignment constraints, and/or modifying the interval assignment optimization process [46].

Sweep, Multisweep Algorithms

Use of the automatic decomposition and scheme selection algorithms in CUBIT tends to increase the average complexity of swept volumes, to the point where some volumes are sweepable in principle, but not in practice (using the current sweep algorithm in CUBIT). For example, using the model problem from before, the volume shown in Figure 14, left, is sweepable in principle; that is, one can see that a swept mesh could be placed on this volume that would be of sufficient quality. However, CUBIT's current sweeping algorithm generates a mesh shown in Figure 14, right (it is unlikely that the sweep algorithms described in [40] or [41] would perform as well or better). Therefore, we conclude that in order to take full advantage of more recent automation capabilities, the current sweeping technology will need to be improved, particularly in the geometric placement...
of nodes through the sweep. Recent work on sweeping and on mesh quality optimization has shown much promise in solving this problem [42].

There has also been recent work to improve the topological capabilities of sweeping, so that volumes can be swept from and to multiple source and target surfaces, respectively [40][45]. This requires the topological imprinting of source and target surfaces, and can either be done explicitly using geometry [45] or implicitly by special assignment of nodes to source and target surface topology [40]. Experience meshing large assembly models indicates that the following are key parts of a multisweep capability:

- Geometry-based: the imprinting process should be geometry-based, that is it should result in the addition of geometric entities to the source and target surfaces. This allows the use of interval matching, automatic surface scheme selection, and other geometry-based tools, and also simplifies the remeshing of multisweep volumes.

- Assembly multisweep: because there are sweep dependencies across volumes in an assembly [33], there may also be dependencies in the imprinting of source/target surfaces. In order to guarantee meshability after automatic scheme assignment and interval matching, the sweep imprinting process will likely need to be performed across multiple dependent swept volumes.

Geometric and topological improvements in sweep algorithms will result in less decomposition for difficult geometries, which in turn will decrease the model complexity seen by the user. This results in an overall decreased time to mesh.

**Automatic Scheme Adjustment**

The current automatic sweep detect algorithm uses a rather conservative method for determining submappable and mappable surfaces, based on local geometric angles at vertices bounding the surface; this choice was intentional, since it errs on the side of mesh quality rather than increased automation. Since mappable and submappable surfaces are required for a volume to be sweepable [33], this reduces the number of volumes which are automatically assigned the sweep scheme.

In practice, it is often necessary to make local adjustments of vertex type assignments in order to force the sweeping of difficult volumes, at the expense of mesh quality in those volumes. Figure 15 shows an example where adjusting n vertex types makes the center volume sweepable, greatly simplifying the meshing of the entire assembly. Although quality near the adjusted vertices is degraded, this does not degrade analysis results in the region of interest. However, adjusting individual vertex types can be a tedious and error-prone process [47].

This problem could be solved in part by automatic, volume-based adjustments of local vertex types to make volumes sweepable. Preliminary work in this area has reduced by 80% the number of individual vertex types needing adjustment [47]; however, this
still requires user intervention to specify which volumes need adjustment. Further automation of this process would probably require the user to provide general guidance on the desired mesh quality bounds.

**Automatic Hexahedral Meshing**

The most obvious long-term need is for an automatic all-hexahedral meshing algorithm for general geometries. Such an algorithm would eliminate the need to decompose for meshability. However, the more unstructured the algorithm, in general the poorer the resulting mesh quality. A more appropriate use for such an algorithm would be to mesh a subset of volumes in a decomposed model. Judicious application of this tool would decouple volumes across an assembly, also drastically reducing the time to mesh. The decomposition could be accomplished using automatic tools, or a limited amount of user interaction in areas of particular interest. Mixed-element meshing algorithms would also accomplish this purpose [48], at the expense of moving to a hex-dominant instead of an all-hex mesh.

Using an automatic hex meshing algorithm as a decoupling tool, in conjunction with other meshing tools, would strike a balance between mesh quality and time to mesh; experience will show whether or where this is an acceptable tradeoff.

**Long-Term Meshing Needs**

In the interest of brevity, this paper has not discussed in more detail some of the long-term needs in meshing large assemblies; these needs are mentioned below, with a small amount of discussion for each.

- **Geometry fix-up**: a large portion of time is typically spent cleaning up a design model before it can be transformed to a model suitable for analysis. The fix-ups involve making the solid watertight, eliminating small features resulting from loose tolerances, etc. Transformation from a design model to an analysis model involves eliminating features of no significance to the analysis. Table 2 shows that geometry fixup accounts for a significant part of the overall time to mesh. See [50] for further discussion of these issues.

- **Parameterized meshing**: most high-end CAD systems now support parametric definition of solid models [30][52]. Coupling the definition of the meshing input to these parameters would simplify propagating incremental design changes to the analysis model. While solutions to this problem exist for tetrahedral-based meshes [53][54], only one is available for hex-based analysis [42].

- **Adaptive conformal hex refinement**: adaptive FEA has been demonstrated using both tetrahedral and hexahedral meshes [55][56]; however, hex-based adaptive FEA requires the use of “hanging” or constrained nodes, which limit their applicability to analysis codes which can handle those types of meshes. Adaptive conformal hex refinement would refine a
hex mesh without the use of hanging nodes; these meshes could be used with any hex-based FEA code. Conformal hex refinement may be possible using concepts from the dual of the hex mesh [57].

**Analysis-Side Needs**

In recent years, the mesh generation R&D community has devoted a great deal of effort towards developing meshing tools to overcome what could be interpreted as shortcomings in FEA capability. Some of those needed capabilities are discussed below.

- **Accurate tetrahedral mesh-based analyses:** although the issue of hex versus tet accuracy for FEA is a controversial subject, it is still widely believed that hexahedra are preferable for some analyses, and there are limited data which show this to be true. Promising research is being done to develop more accurate tetrahedral elements for non-linear structural mechanics; these elements have eight nodes (four vertex nodes and four mid-face nodes), and are more efficient than the traditional 10-node tetrahedron [58]. It is not clear whether these are efficient enough to justify their use in place of hexahedra.

- **Tied contact:** tied contact, also known as node collocation, involves writing constraints to force nodes on one side of an interface to be bound to elements on the other side, using methods similar to those used for contact enforcement in transient dynamics. This technology has been proposed at Sandia as a means of constructing non-conforming interfaces, which would reduce coupling across assembly meshes. While this technology would greatly simplify the construction of hex meshes, some recent studies have shown that tied contacts may introduce errors in the local neighborhood of the non-conforming interface; they may also slow convergence in some analyses [60]. However, this method may be useful in regions where solution accuracy is not needed.

- **Element-based support for non-conforming interfaces:** some recent work by Dohrmann et. al [60] have attempted to write a new type of hex-like element which conforms to the geometry of an interface; in this approach, material is added to or subtracted from an element so that it closes any gap or overlap with a “host” mesh. This work has shown promising results in terms of accuracy, and would require little extra work on the meshing side; however, it does require adding a new element formulation to the analysis code.

- **Mixed-element meshes:** it has been postulated for some time that it may be possible to obtain sufficient analysis accuracy using mixed hex-pyramid-tet meshes, with the hexes occupying the majority of the volume and the tets restricted to volume interiors[59]. While the capability to generate these types of meshes has been demonstrated [48], there is a lack of experience in performing real analyses on them, partly due to the lack of a pyramid element in many analysis codes. Mixed-element meshes would reduce coupling across assemblies, thereby simplifying the overall hex meshing process.
While the mesh generation research community has grown and matured during the past five years, in many cases it has also specialized in that area, and away from the FEA side. In order to make progress in the analysis-side needs mentioned above, closer links between the mesh generation and analysis communities will have to be re-established. Fortunately, the need for adaptive mesh refinement is providing an impetus for the same, and some groups exist which have not lost those close links [61][62].

6. CONCLUSIONS

Hexahedral mesh generation is heavily dependent on decomposing geometry into simpler, meshable pieces. Also, the trend in modeling and simulation is towards more complex geometries, often with multiple parts and materials. These trends all point toward an increased need for assembly meshing. Assembly meshing is difficult because of the propagation of mesh across shared surfaces. This is particularly difficult in hex meshing, but also affects tetrahedral meshing. This propagation tends to couple the meshing problem across the assembly, making purely local solutions insufficient for these problems.

While an all-hexahedral meshing algorithm capable of meshing arbitrary geometries would certainly solve many of the time-to-mesh problems, there are also other opportunities for automating the decomposition-based process have been explored. Specific areas of progress include automatic and simplified manual geometry decomposition; automatic mesh scheme selection; fast, robust surface- and volume-based interval assignment; and robust mesh sweeping algorithms. There are also several all-hexahedral meshing algorithms that are showing some promise. Used together, these tools not only speed the process of meshing assembly models, but also improve the quality of the resulting mesh. For these reasons, the decomposition-based hex meshing approach will be useful even after an all-hex meshing algorithm is available.

While the time required to generate an all-hexahedral mesh for assembly geometries has been reduced significantly in recent years, further improvements are needed elsewhere on the critical path of this process. Specific needs described in this paper include mesh skew across submappable and mappable surfaces, fault-tolerant volume-based vertex type assignment, better sweeping algorithms, and all-hexahedral meshing algorithms. Efforts are underway at Sandia to address these needs.

On a broader horizon, the computational simulation world is clearly headed toward higher-level uses of analysis, for example in performing adaptive analysis and optimization. Conformal adaptive hex refinement will be needed to support these analyses. In addition, for analysis to be truly integrated with the design process, techniques will have to be developed for constructing parameterized models, which allow small modifications of the underlying geometric models without requiring a completely new effort to generate a corresponding mesh. This will be enabled by parametric modeling capabilities in the CAD world, several of which exist already.
Recent research has shown that there is still a well-founded need for hexahedral meshes in many areas of computational simulation. While the generation of all-hex meshes has been a tedious and time-consuming process in the past, recent progress is reducing that time, and current efforts should reduce that time even further. The goal is to make the process of generating high-quality, all-hex meshes for assembly geometries simple enough for non-experts to use in the everyday design process.

ACKNOWLEDGEMENTS

This paper is the result of the author's experience leading the CUBIT mesh generation toolkit project over several years. The accomplishments during that time, many of which are described in this paper, would not have been possible without the full dedication of the CUBIT team, including Pat Knupp, Darryl Melander, Scott Mitchell, Jason Shepherd, David White, and others. In addition, the CUBIT project would not have survived long enough to make this progress without key management support; thanks to Grant Heffelfinger and Robert Leland, we received that support and more.

REFERENCES


Figure 1: Decomposition of a part (top) into two parts each meshable with simple algorithms (bottom).
Figure 2: Decomposition of part on left into parts on right allows use of more structured algorithms.
Figure 3: Assemblies meshed recently at Sandia National Labs; braze joint vacuum seal (upper left); detonator cap (upper right); sub-component in Neutron Generator (bottom).
Figure 4: Chords defined in the hex mesh[12].
Figure 5: Kim1 model (left); arm section, which contains two through-holes and two blind holes (right)[26].
Figure 6: Kiml model after automatic decomposition (left); unmeshable volumes, highlighted (right).
Figure 7: Kim1 model after manual decomposition (left); final mesh, after automatic scheme selection, interval assignment, and meshing (right).
Figure 8: Interval constraints for mapped surface: sum of intervals on edges 1 and 2 (side 1) must equal sum of intervals on edges 4 and 5 (side 2).
Figure 9: Part with volume meshing interval constraints; intervals along inside sweep path must be equal to intervals along outside path.
Figure 10: Complicated parts meshed using CUBIT's advanced sweep algorithm.
Figure 11: NG Power Supply outer skin, showing insulating material and a few small components.
Figure 12: NG Tube mesh, upper section (surrounding insulating material not shown).
Figure 13: Skew control, caused by multiple vertices per mapping side.
Figure 14: Volume sweepable in principle but not in practice. Mesh quality is sufficient on source surfaces (left), but compression into target surface results in mesh tangling (right).
Figure 15: Volume for which adjusting 1 vertex type greatly simplifies overall meshing process.
Table 1: Various metrics from the Neutron Generator Power Supply and Tube meshing efforts.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Power Supply</th>
<th>Tube</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Geometry (# volumes)</td>
<td>21</td>
<td>75</td>
</tr>
<tr>
<td>Decomposed Geometry (# volumes)</td>
<td>184</td>
<td>418</td>
</tr>
<tr>
<td># Elements</td>
<td>6.7x10⁶</td>
<td>7.0x10⁶</td>
</tr>
<tr>
<td># Nodes</td>
<td>6.8x10⁶</td>
<td>7.4x10⁶</td>
</tr>
<tr>
<td>Mesh Completion Date</td>
<td>Sep 1997</td>
<td>Jul 1998</td>
</tr>
<tr>
<td>Time to Mesh (cumulative effort)</td>
<td>16 mo.</td>
<td>6 mo.</td>
</tr>
<tr>
<td>Time to Mesh (calendar time)</td>
<td>6 mo.</td>
<td>2 mo.</td>
</tr>
</tbody>
</table>
Table 2: Time distribution for Neutron Generator Power Supply and Tube meshing efforts.

<table>
<thead>
<tr>
<th>Task</th>
<th>Time Fraction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Power Supply</td>
</tr>
<tr>
<td>Geometry cleanup &amp; simplification</td>
<td>25%</td>
</tr>
<tr>
<td>Geometry modifications for meshing</td>
<td>30%</td>
</tr>
<tr>
<td>Pre-mesh &amp; mesh</td>
<td>20%</td>
</tr>
<tr>
<td>Mesh quality improvement</td>
<td>15%</td>
</tr>
<tr>
<td>Mesh output &amp; joining</td>
<td>10%</td>
</tr>
<tr>
<td>Code modifications</td>
<td>(not measured)</td>
</tr>
</tbody>
</table>