Conformal Assembly Meshing with Tolerant Imprinting

Brett W. Clark[1], Byron W. Hanks[2], and Corey D. Ernst[3]

[1] Sandia National Laboratories*, P.O. Box 5800, MS 0376
Albuquerque, New Mexico, 87185-0376
bwclark@sandia.gov
[2] Sandia National Laboratories, P.O. Box 5800, MS 0376
Albuquerque, New Mexico, 87185-0376
bwhanks@sandia.gov
[3] Elemental Technologies, Inc., 17 North Merchant Street, Suite 3
American Fork, Utah, 84003
corey@elemtech.com

Abstract. Solid assembly meshing has all of the same "dirty geometry" induced issues as single part meshing but also has the difficulty associated with generating a conformal mesh between solids where solid-solid interfaces are not obvious. Mesh generators usually don't have CAD assembly constraint information to identify interfacing solids and must therefore rely on geometric proximity to deduce these interactions. "Slop" in the positioning and alignment of parts in the assembly makes automatically discovering the interfaces and generating a conformal mesh at the interfaces very difficult. Most of the efforts in this area resort to some sort of discrete representation to deal with these issues losing the capability to do further solid modeling engine operations often necessary for all-hexahedral meshing. This paper presents a method for defining the non-manifold interfaces between volumes in an assembly required for generating a conformal mesh while maintaining the original solid modeling engine format.

1 Introduction

Conformal assembly meshing is a common requirement for finite element analysis of complex assemblies of solid parts. However, the solid model assemblies provided as input for mesh generation often contain inaccuracies or sloppiness resulting in a lack of clean or clear interfaces between adjacent volumes. Without clear definitions of the interfaces between adjacent volumes it is very difficult to generate a conformal assembly mesh.

In some cases small modifications can be made to individual volumes in the assembly to improve the mating interfaces between volumes but in general this will not work because of the global nature of modifying B-rep models composed of non-facet based surfaces. As a result, most efforts to solve this problem have

* Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energys National Nuclear Security Administration under contract DE-AC04-94AL85000.

been made in some kind of a faceted representation whether it is a faceted representation of the input solid model or in the generated mesh itself (if the mesh is tetrahedral). In either case, very local modifications can easily be made to the facets for the desired effect. The drawback, however, of going to a faceted representation is the loss of the ability to do subsequent Boolean-type solid modeling operations on the volumes of the assembly. For tetrahedral meshes this may not be necessary, but for all-hexahedral meshing there is often decomposition of the volumes that is required prior to applying meshing algorithms. Therefore, it is desirable to be able to define the interfaces between adjacent volumes required for a conformal mesh and still be able to do solid modeling operations on the volumes for decomposition.

This work describes a "tolerant imprinting" capability and supporting tools that allow the definition of a clean interface between adjacent volumes in an assembly without having to go to a faceted representation. The result is the ability to stay within the solid modeling engine representation throughout allowing downstream solid modeling operations such as volume decomposition.

Section 2 will discuss related work in this area. Section 3 will describe conformal assembly meshing. Section 4 will discuss the difficulties in generating conformal assembly meshes. Section 5 will describe imprinting and its role in facilitating conformal assembly meshes. Section 6 will describe the tolerant imprinting algorithm. Section 7 describes some tools for determining the appropriate tolerance to use when tolerant imprinting. Section 8 will give some examples.

2 Related Work

White et al. [1] presented an algorithm that facets the bounding curves of the original surfaces at the interface of two volumes and generates an intersection graph between the two sets of facets using a user-specified tolerance. The resulting intersection graph defines the topology of the interface surface(s). "Virtual geometry" is used to represent the resulting interface surfaces and appropriate imprints on those surfaces. One advantage that White's approach has in common with the authors' approach is that by modifying the geometry to be meshed (as opposed to meshing the geometry and then modifying the mesh to make it conformal) the modifications are done only once and then multiple mesh instances can be generated for the modified geometry as opposed to doing the modifications each time a mesh is generated. A drawback of White's approach is the inability to do solid modeling engine operations on the geometry after the "virtual geometry" is applied.

Chouadria and Veron [2] showed an approach that works on polyhedral assemblies. The interfaces between adjacent polyhedral volumes were automatically discovered given a user-specified tolerance and then appropriate interfaces were generated by imprinting the interface faces onto one another. The adjacent polyhedral volumes were finally stitched up to the generated interface to create a non-manifold assembly. Although a related work, it does not help solve the

problem of obtaining a non-manifold assembly while maintaining the original solid modeling format.

Various commercial codes advertise the ability to generate conformal solid assembly meshes [3, 4, 5]. However, the details of how this is accomplished are proprietary. Despite the apparent advancements in the area of commercial conformal tetrahedral assembly meshing there does not appear to be a solution for hexahedral meshing.

3 Conformal Meshes

In a conformal finite element mesh of an assembly, the solid-solid interface between two volumes is represented by a set of faces such that each face on the interface is a sub-element of the two adjacent solid elements corresponding respectively to the two volumes. Figure 1 A shows a conformal mesh and figure 1 B shows a non-conformal mesh of two volumes. Having a conformal mesh avoids the need for contact elements or additional constraint equations that must be solved with the analysis. This generally reduces the compute time when running an analysis. For some types of analysis, the lack of appropriate contact element algorithms may prevent achieving a solution without a contiguous mesh.

One approach to achieve a conformal assembly mesh is to define a non-manifold representation of the assembly model in which adjacencies between parts in the assembly are explicitly represented as shared or "merged" topology. The B-rep solid for each part contains a topological face representing the coincident boundary region of each part. To merge these two parts, a non-manifold representation of the combined parts is created and the two faces are replaced with a single shared face. Child entities of the two faces are also merged so that edges and vertices that bound the face are shared between the two parts.

During mesh generation, the shared face will be meshed before the volume meshes are created. Since the face is referenced by both volumes, the mesh on the shared face will be used in each volume mesh, resulting in a contiguous or conformal mesh. Figure 2 shows the shared mesh on one of the assembly volumes.

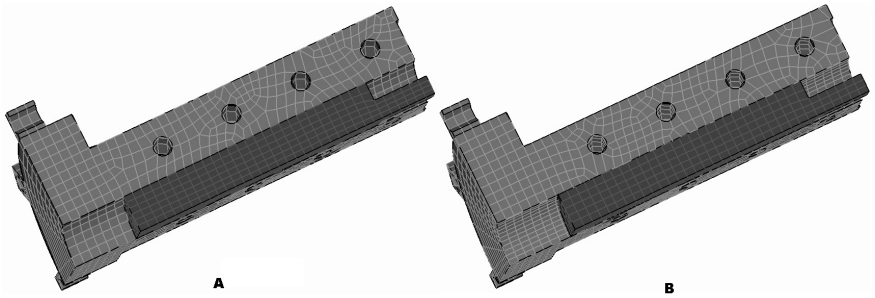

Fig. 1. Conformal and non-conformal assembly meshes

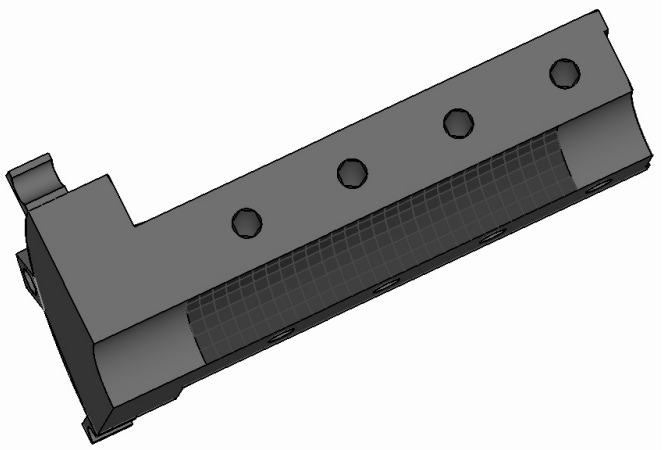

Fig. 2. Shared mesh between two volumes

It will be helpful here to discuss the requirements for merging to occur. At a minimum, the two topological subtrees to be merged must match exactly. Faces to be merged must have the same number of loops; corresponding loops must have the same number of edges; and corresponding edges must have the same number of vertices. Given such a topological match a merge can be forced, regardless of the geometric shape if desired. However, merging is usually only performed when the corresponding geometrical entities - the surface for each face, the curve for each edge, and the coordinates for each vertex - lie within a specified tolerance. This tolerance is referred to as the merge tolerance.

In general, the matching topology does not exist in an assembly where merging should occur between adjacent parts. To generate this topological match, a boolean imprint operation is used. The imprint operation is discussed further in Section 5.

The imprinting and merging approach described here is used in the CUBIT mesh generation software [6] to achieve conformal meshes. Other approaches to achieve a conformal mesh are mesh merging and mesh mirroring. Mesh merging can be manual, where the user is responsible to make the meshes match and merge the required nodes, or automatic, where an algorithm determines edge swaps and node insertions to make the meshes conformal. Mesh mirroring requires an imprint similar to mesh merging, but instead of a non-manifold data structure underlying the mesh, the mesh is mirrored to the corresponding geometry of an adjacent part such that the resulting mesh is conformal (see [1]).

4 Problem Definition

4.1 Assembly Models

Assembly models from a CAD system are typically represented with a B-rep solid model, or a reference to a solid model, for each part in the assembly. To position

these parts within the assembly, a transformation matrix is kept for each part to maintain the transformation from the local coordinate system of the part to the global coordinate system of the assembly. The transformation matrix is often calculated from a set of mating relationships that define the position of a part with respect to other parts in the assembly. In calculating the transformation from the given mating relationships, each part is treated as a rigid body and the relationships are used to constrain the six degrees of freedom of the body.

In the general case, assembly constraint information is insufficient to capture the adjacencies within an assembly model. For example, if one face of part A should mate with two separate but coplanar faces of an adjacent part B, only one mating constraint between the face of part A and one of the coplanar faces of part B will be defined. The other mating constraint would be redundant and is generally not created by the user. Determining adjacencies therefore requires a calculation using geometric proximity to find all solid-solid interfaces in the assembly.

4.2 Problems with Solid-Solid Interfaces

Since assembly modelers focus on rigid-body positioning of parts and not on modeling the adjacencies between parts, it can be challenging to produce a model where the adjacencies are captured cleanly. Slight overlaps or gaps can easily occur between parts, or adjacent parts can be slightly misaligned. Gaps between parts may also be the result of excluding welds or adhesives from the assembly model. These situations can cause problems in the proximity calculation.

When an assembly model is translated from one format to another, different tolerances used by different modeling software can also cause problems. If an assembly is modeled in software that uses a large tolerance, corresponding geometry may lie within tolerance and be considered coincident. However, if this same model is translated into another modeler that uses very tight tolerances, geometry considered coincident by the first modeler may now lie outside of tolerance in the second modeler.

Model simplification for analysis purposes can also lead to problems in the solid-solid interfaces of the assembly. The assembly model may contain details which are geometrically accurate, but represent too much detail for the desired analysis. The analyst may simplify the model, which again can introduce inaccuracies or approximations in the solid-solid interfaces.

5 Imprinting

Merging has been described as a way in CUBIT to create a non-manifold representation of an assembly model for conformal meshing. The merging of two surfaces requires that the surfaces and their children entities (curves and vertices) match both topologically and geometrically. Imprinting is a common process for introducing topology (vertices and curves) from one entity onto another and is provided by most solid modeling engines. Figure 3 shows two adjacent volumes

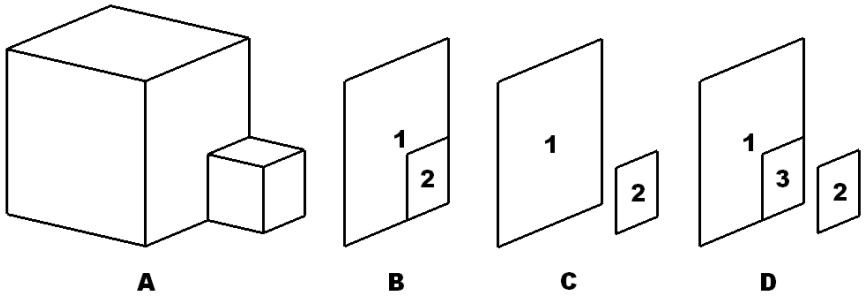

Fig. 3. Imprint example: A) two adjacent blocks, B) adjacent surfaces on blocks, C) exploded view of adjacent surfaces, D) exploded view of surfaces after imprint

and their overlapping surfaces. The surfaces lie right on top of each other but the boundaries of the surfaces don't match. By imprinting the surfaces (or volumes) onto one another curves are introduced into the larger surface creating a geometric and topological match between two of the surfaces as seen in Figure 3 D. After the imprint the two matching surfaces can be merged into one surface that is shared between the two volumes.

Imprinting is a solid modeling operation that uses a tolerance provided by the solid modeling engine. CUBIT's default solid modeling engine uses a tolerance of 1e-6 during imprinting. This tolerance is used to determine what should or should not be imprinted during an imprint operation. If the two volumes in Figure 3 had a gap between them that was larger than 1e-6 no imprinting would have taken place. Similarly, consider the two volumes in Figure 4 A that are misaligned by a very small amount. The adjacent surfaces between the volumes have a distance of zero between them so they are candidates for imprinting but depending on the size of the misalignment the results of the imprint operation will differ (see Figure 4).

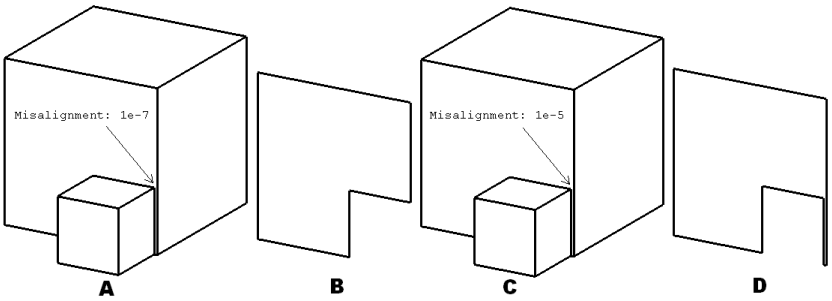

Fig. 4. Misalignment example: A) misalignment less than 1e-6, B) imprint result, C) misalignment greater than 1e-6, D) imprint result

The tolerance used when imprinting guarantees two things: 1) that entities will only be imprinted onto one another if they are within the imprint tolerance and 2) no features smaller than the imprint tolerance will be introduced into the model during the imprint operation. When considering an assembly in CUBIT (using the default solid modeling engine) this means that volumes must be within 1e-6 of one another to be imprinted and volume misalignments greater than 1e-6 will result in imprints and possibly sliver entities with size equal to the misalignment. Assembly constraints in CAD packages are often much looser than 1e-6 which can result in imprinting nightmares. For this reason "tolerant imprinting" has been developed.

6 Tolerant Imprinting

Tolerant imprinting is a combination of functionality provided by CUBIT and imprinting functionality provided by the solid modeling engine it uses. From the user's perspective tolerant imprinting is identical to regular imprinting in behavior except that it provides the user with a modifiable tolerance to be used during the imprinting process. This allows the user to set the imprint tolerance to be greater than the 1e-6 value provided by the solid modeling engine. As a result the user can set the imprint tolerance to match more closely the gap/misalignment sizes in his assembly and get the desired imprinting results.

The tolerance used in tolerant imprinting is the user-modifiable "merge tolerance" in CUBIT. Analogous to the way the solid modeling engine uses 1e-6 to determine what should or should not be imprinted CUBIT uses its merge tolerance value set by the user to decide what should or should not be imprinted. The tolerant imprinting algorithm has three main parts: 1) imprint vertices onto curves, 2) imprint curves onto surfaces, and 3) imprint vertices onto curves again.

6.1 Imprint Vertices onto Curves

The first step in the algorithm is to imprint curves with vertices. The algorithm will first search for all pairs of curves that overlap by at least merge tolerance.

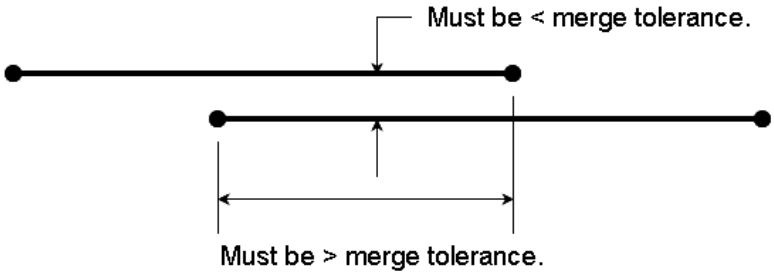

Fig. 5. Overlapping curve criteria

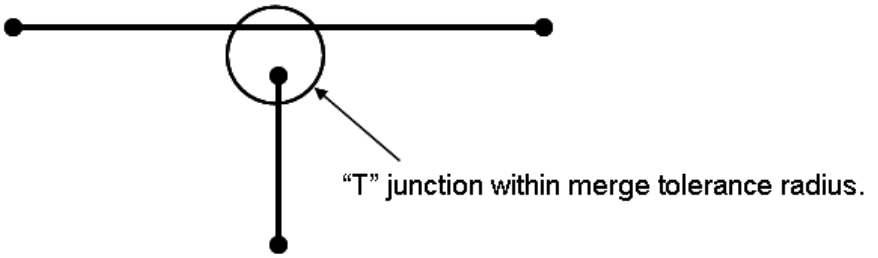

Fig. 6. T-junction criteria

Overlapping in this case means that the curves lay on top of one another spatially and have an overlapping region at least merge tolerance long (see Figure 5). Once these pairs are found each curve is imprinted by the vertices of the other curve where applicable. A common split curve operation provided by the solid modeling engine is used. "T" junctions (see Figure 6) between curves are also found and appropriate imprints are made there as well.

6.2 Imprint Curves onto Surfaces

The second part of the algorithm imprints curves onto surfaces. The algorithm will search for all pairs of surfaces that overlap within merge tolerance. Overlapping surfaces are defined as surface pairs that lay on top of one another spatially within merge tolerance and which have an overlapping region with area of at least merge tolerance squared. Once the overlapping surface pairs are found each curve from one surface is imprinted onto the other surface and vice versa. This part of the algorithm takes advantage of an api from the solid modeling engine that "tolerantly" embeds a curve into a surface. The advantage of using this api is that it will automatically lengthen/shorten the curve (within a specified tolerance) to match up with existing topology in the surface. In actuality it doesn't change the length of the curve but treats it as tolerant so that it can attach to existing topology in the surface. The default solid modeling engine in CUBIT provides this type of functionality but it would need to be implemented in CUBIT if the tolerant imprinting algorithm were ported to another solid modeling engine that didn't have this functionality.

It should be noted that when considering the curves to be imprinted onto a surface curves from one surface that are "overlapping" with curves in the other surface are excluded from the imprint operation as they have already been imprinted in the first step. One potential implementation would be to not do the first imprint overlapping curves step described above and just rely on the tolerant embed functionality to imprint the overlapping curves during the imprint overlapping surfaces step. However, the authors found that the tolerant embed functionality provided by the solid modeling engine was not robust in doing this and as a result the first step of imprinting overlapping curves is used.

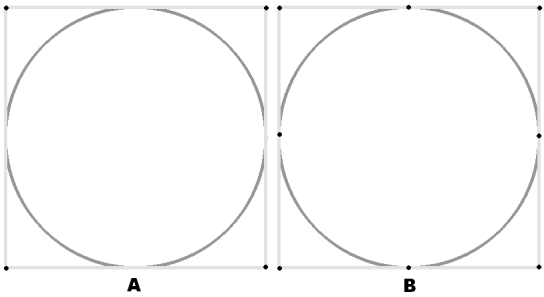

Fig. 7. Circular surface fitting tangentially inside a square surface

6.3 Imprint Vertices onto Curves Again

Finally, overlapping curves are once again imprinted in order to catch any remaining vertex-curve imprints that need to take place because of imprinting curves onto surfaces. One might expect that this final step is unnecessary. However, there are some geometry configurations that cause new vertices to be introduced during the "Imprint Curves Onto Surfaces" step that were not introduced during the first "Imprint Vertices Onto Curves" step and that subsequently need to be imprinted onto curves. One such configuration is a circular surface that fits tangentially into a square surface (see Figure 7 A). Vertices at the tangent locations are not introduced during the first "Imprint Vertices Onto Curves" step because the curves are not overlapping within tolerance. During the "Imprint Curves Onto Surfaces" step the curve of the circular surface is imprinted onto the square surface creating breaks in the curves of the square surface at the tangent locations. However, because the curves on the square surface are outside the circular surface they are not imprinted onto the circular surface so the circular curve does not get split. This requires the final "Imprint Vertices Onto Curves" step. The result is shown in Figure 7 B.

Just as regular imprinting will not introduce features smaller than 1e-6, tolerant imprinting will not introduce features smaller than merge tolerance.

7 Determining a Good Merge Tolerance

Determining an appropriate merge tolerance is the main key to effectively using tolerant imprinting. Too small a merge tolerance will result in missed imprints and too large a merge tolerance will result in unintended imprints. Unfortunately, automatically determining a merge tolerance for a given assembly is difficult without a user in the loop. Because of the variations in gap and misalignment sizes within a given assembly it is difficult to automatically determine what volumes should or should not be "touching." Therefore, in most cases the user needs to specify the merge tolerance after an examination of the assembly. Below are some important characteristics to consider when examining an assembly and

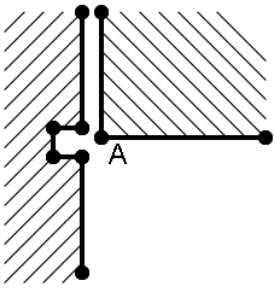

Fig. 8. Ambiguous merge example

a proposed process implemented in CUBIT to aid the user in determining a suitable merge tolerance.

7.1 Smallest Feature Size

The smallest feature size in the model helps to limit how large the merge tolerance can be. If the merge tolerance is larger than or on the order of the smallest feature size there can be ambiguities as to what should be merged between two volumes. For example consider the geometry configuration in Figure 8. The volume on the right contains vertex A. The volume on the left has a small feature near vertex A. Depending on the size of merge tolerance it will be unclear which vertices on the left volume could possibly be merged with vertex A. Merging will only allow one of the vertices from the volume on the left to merge with vertex A but there may be more than one candidate vertex based on the value of merge tolerance. To avoid these types of ambiguities merge tolerance should not be as large as the smallest feature the user wants resolved in the mesh. If gaps/misalignments in the model require a merge tolerance larger than the smallest feature the model really needs to be cleaned up with tighter assembly constraints or small features need to be removed.

Because of the potential ambiguities described above it is important to first know the size of the smallest feature you want to represent in the assembly. CUBIT provides tools for either finding the n smallest features in the model or for finding all of the features with a size less than some user-specified feature. When first becoming familiar with the model the user does not usually know a small size to specify so finding the n smallest features is very useful. This tool looks for proximities between different topological entities (vertex, curve, surface) within a given volume instead of just looking for "small curves" or "small surfaces." This was deemed a more general search. It is also important to note that this search is done on a per-volume basis; it is not a tool for finding proximities between two volumes. Figure 9 shows an example of the tool. In this example Vertex 42 and Curve 57 were found to be 0.254 units apart and are highlighted in orange in the graphics window. The user can select results from the search and zoom to

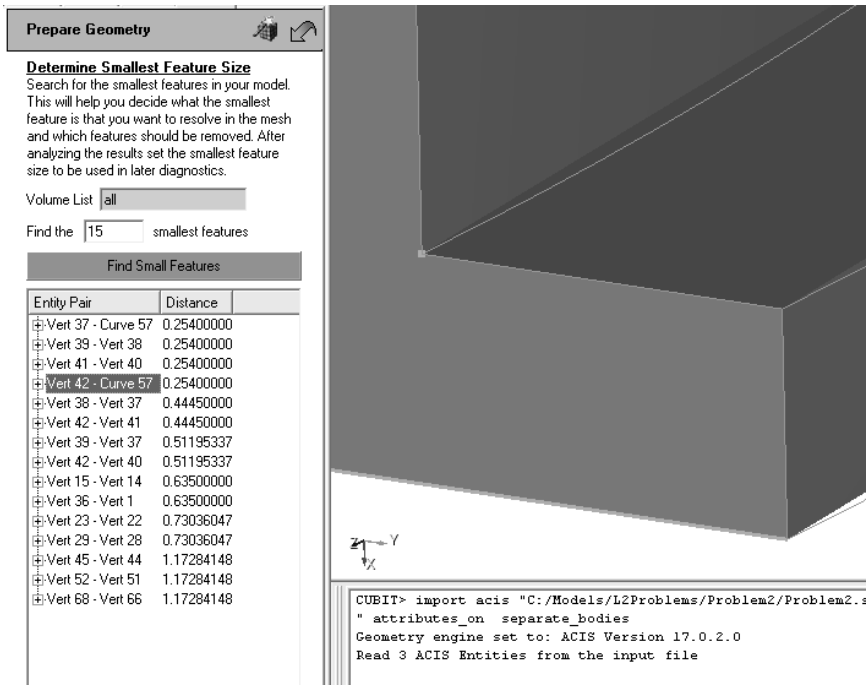

Fig. 9. Find smallest features tool

them quickly to determine which small features are intentional in the model and which ones need to be removed (sliver entities, etc.). Looking ahead to tolerant imprinting it is important to remove all features that are smaller than merge tolerance to eliminate any ambiguities when merging. Obviously, the user won't know the merge tolerance at this point but he can look at the small features that are in the model and remove any that are considered unnecessary for the analysis.

7.2 Merge Tolerance Estimator

Once the smallest feature size is established CUBIT provides a tool for automatically estimating the merge tolerance given the smallest feature size. This is useful in giving the user a starting point for the merge tolerance that can then be fine tuned. The estimator relies on there being a fairly consistent average gap/misalignment between volumes in the assembly. Obviously, this does not always hold true under which circumstances the estimation is not very good but in most cases there is a natural value at which most of the merging can take place in the model and after which not many new merges will take place. This can be

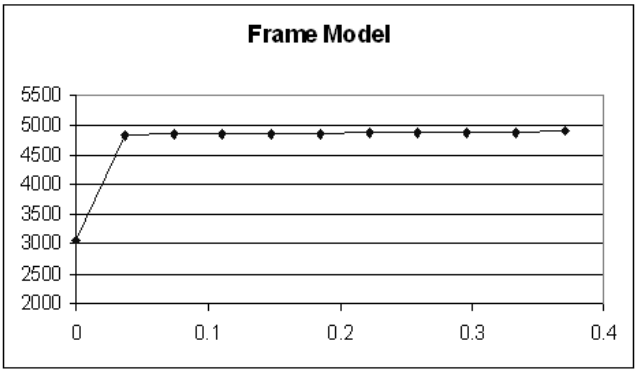

Fig. 10. Plot of number of merges vs. merge tolerance

seen in the plot in Figure 10 of values calculated from a real assembly containing about 900 volumes. The horizontal axis plots increasing merge tolerance values and the vertical axis plots the number of merges that would take place at that merge tolerance. Where the plot flattens out is usually a good guess of where the merge tolerance should be. The automatic estimator generates data similar to that in Figure 10 for the assembly and picks off a value where the plot starts to flatten out. To reiterate, this is only a rough estimation but usually gives a good starting point for the user.

7.3 Proximate Features

After an approximate merge tolerance has been decided upon either by the user or by the automatic estimator it is prudent to examine the assembly to see if the merge tolerance will be suitable to capture all of the desired merges. CUBIT provides a tool for doing this examination. The user can define a range around merge tolerance and search for entity-entity proximities in the assembly that fall within the range. For example, if the estimated merge tolerance was 0.005 the user could look at a range above merge tolerance, 0.005 to 0.05, to see if there are any entity pairs in that range that he would want to merge but that wouldn't because they are out of merge tolerance. Similarly, he could then look at the entity-entity proximities below merge tolerance, in the range 0.0005 to 0.005 for example, to see if there are some entities that would merge that really he wouldn't want to merge. In this way the user can fine tune what merge tolerance should be to capture the desired merges.

Figure 11 shows an example of the tool used to search a range for entity-entity proximities. Again, the tool allows the user to quickly zoom in on entity pairs to determine if merge tolerance should be adjusted.

Once the merge tolerance has been fine-tuned the user can run tolerant imprinting and then merging.

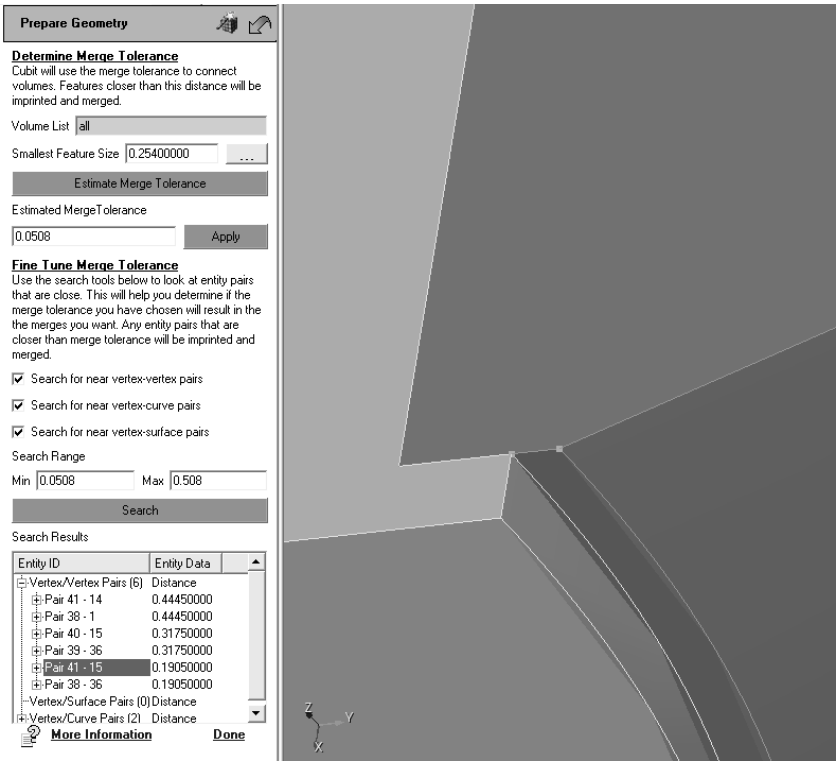

Fig. 11. Tool for determining merge tolerance

8 Example

The real benefit of the tolerant imprinting capability is the time saved in preparing a non-manifold assembly. Depending on the sloppiness and size of the assembly it can easily take days and even weeks to manually modify the assembly to have clean interfaces between adjacent volumes. Following is an example of the impact of having tolerant imprinting.

This example is of an AF&F (arming fusing and firing) assembly consisting of 240 volumes. This example is an Official Use Only model so pictures of the assembly are not included. The assembly is very compact in nature with a high number of volume-volume interfaces. This model is also considered quite "sloppy" in that many of the interfaces are out of tolerance when using the default imprinting tolerance of 1e-6. Two runs were made, first with regular imprinting and merging and then with tolerant imprinting and merging. Table 1 shows the results. The results show the time required during the imprinting step and the resulting number of overlapping surface pairs after imprinting and merging. An overlapping surface pair indicates two surfaces that did not merge correctly during the merge process. Overlapping surface pairs are usually resolved by the user

Table 1. AF&F data

	Time (min)	# Overlapping Surface Pairs
Regular Imprinting	3.5	169
Tolerant Imprinting	56	2

manually fixing the model so that the merge can take place. As can be seen in the results tolerant imprinting takes much longer to run. However, after running tolerant imprinting there are only 2 overlapping surface pairs to fix manually. Regular imprinting and merging resulted in 169 overlapping surface pairs that would each have to be analyzed and resolved manually. The time required to do this would easily outweigh the extra time required to run tolerant imprinting.

9 Conclusion

This paper has described the tolerant imprinting capability and other tools that can be used to generate a non-manifold representation of a volume assembly while maintaining the native solid modeling format of the assembly. This has the advantage of allowing the user to do subsequent solid modeling operations on the assembly (such as decomposition) to facilitate meshing. The real benefit of the tolerant imprinting capability is the time saved when the user does not have to modify the individual volumes in the assembly to define clean interfaces between adjacent volumes. An example of using tolerant imprinting on a complex assembly was given showing this potential time saving.

References

1. White, D.R.: An imprint and merge algorithm incoporating geometric tolerances for conformal meshing of misaligned assemblies. International Journal for Numerical Methods in Engineering 59, 1839–1860 (2004)
2. Chouadria, R., Veron, P.: Identifying and re-meshing contact interfaces in a polyhedral assembly for digital mock-up. Engineering with Computers 22, 47–58 (2006)
3. ALGOR, `http://www.algor.com`
4. Patran, `http://www.mscsoftware.com`
5. COSMOSWorks, `http://www.solidworks.com/pages/products/cosmos/cosmosworks.html`
6. CUBIT, `http://cubit.sandia.gov`