Extrusion of Complex Surface Meshes Utilizing Face Offsetting and Mean Curvature Smoothing

Peter L. McCloud[*]
ERC-Inc, Houston, TX 77058

Grids for three-dimensional Computational Fluid Dynamics problems frequently require a prismatic layer of cells, typically extruded in the off-body direction from a two-dimensional surface mesh to properly resolve boundary layers. When the surface geometry is complex, the extrusion process can fail, resulting in the physical boundary layer being thicker than the prismatic layer, leading to under-resolved boundary layers. To address the shortcomings of existing grid generation tools, a new tool has been developed as part of the *Mesh_Tools* suite, that is capable of extruding complex surface meshes. The new tool uses a face offsetting method to preserve surface curvature and a mean curvature smoothing algorithm to prevent the cells from self-intersecting in concave regions. Testing of the new tool found that not only were complex surface meshes able to be extruded to the desired thickness, but extrusion of simple surface meshes was also improved, due to the face offsetting method.

I. Introduction

As compute systems become more powerful and Computational Fluid Dynamics (CFD) solvers improve, simulations on geometries of ever-increasing complexity become possible. One example of evolving CFD requirements is a notional re-entry capsule, shown in figure 1, which would have been traditionally modeled with a simple structured grid representing the as-built shape before entry. However, this notional design has a Thermal Protection System (TPS) composed of blocks instead of a monolithic heatshield. The design of such a heatshield requires modeling the environments as the block seams recede or swell, creating gaps or fences, increasing the geometric complexity.

Figure 1. Notional re-entry capsule with a block heatshield layout. Including heatshield gaps or fences increases the geometric complexity considerably.

[*]Aerothermodynamics Engineer, JETS-EG3 NASA JSC, Lifetime Member

One way to handle the increased geometric complexity in a CFD simulation is to use unstructured grids, utilizing grid generation packages such as AFLR3 [1, 2] or Pointwise [3]. However, the fact that an unstructured grid can be generated around a complex geometry, does not necessarily mean that the grid will have sufficient quality or even be valid. As with all grid generation, care must be taken to ensure that the volume grid properly captures the various flow phenomena, such as boundary layers, shear layers, shocks, etc.

For boundary layers in unstructured grids, the strategy employed by the two grid generation packages mentioned is to create prismatic elements in the off-body direction by extruding the surface mesh. For smooth, convex geometries, the process is straightforward and growing the prismatic layer to a height greater than or equal to the physical boundary layer is achievable. An example of such a case, a smooth re-entry capsule heatshield, is shown in figure 2a. The process quickly becomes much more complicated when geometric details are included in the surface mesh, particularly concave corners. As the prismatic layer gets thicker, the cells can become highly skewed, leading to poor grid quality, or even become non-convex, leading to an invalid grid. Figure 2b shows what happens to the heatshield grid when a shear groove around a compression pad is included. The grid generation software, in this case AFLR3, can't grow prismatic layers in the concave corners very far and the region is later filled in with tetrahedrons.

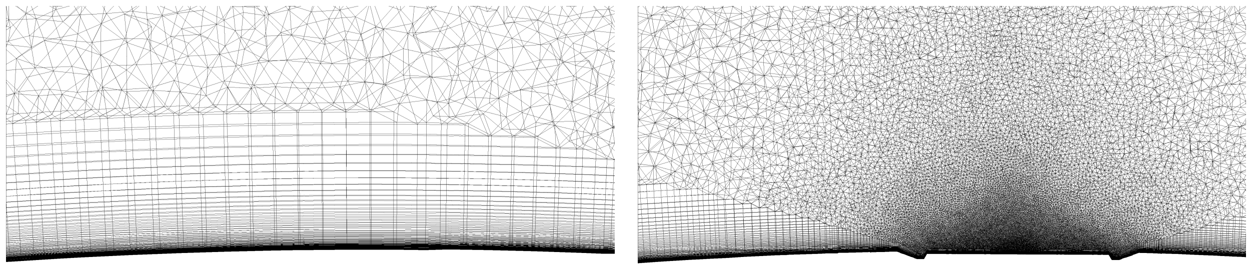

(a) Grid Slice of a smooth OML heatshield.   (b) Grid slice of the same heatshield with a shear groove included.

Figure 2. Prismatic layers generated on smooth and complex geometries.

In attempting to create a volume grid to simulate the flow field around the Orion block Avcoat heatshield with fences during re-entry, it became evident that none of the existing grid generation packages were up to the task of generating a volume grid of sufficient quality. The surface mesh representing the heatshield was a quad-dominant grid composed of approximately 7.8 million points and 8.0 million faces. While Pointwise could read in the surface grid, the memory required to extrude the mesh exceeded the available memory, even on high-end workstations. Attempts made with AFLR3 were more successful, resulting in a valid volume grid, but there were several shortcomings. The small surface faces in the concave corners led to the extrusion process ending early and tetrahedrons were used in the physical boundary layer, reducing solution accuracy. Additionally, the tetrahedrons in the concave corners tended to have high volume ratios, making it impossible to run time-accurate simulations. In order to move forward with simulating the heatshield with fences, a new grid generation tool was required.

II. Tool Development

To address the shortcomings of the existing tools, development of a new grid generation tool with the specific purpose of extruding surface grids was undertaken. Called *Extrude*, this new tool is part of the *Mesh_Tools* tool suite developed at the NASA Johnson Space Center (JSC) Aerosciences Branch. The software was development in C++ and leveraged the Visualization Tool Kit (VTK) [4] libraries, which provide efficient data structures and algorithms for handling much of the data and the required manipulations. The tool is intended to work with surface meshes, composed of triangles, quads, or a combination of both. *Extrude* is not meant to be a standalone tool, as it does not perform any tetrahedral grid generation. Once a prismatic layer has been extruded, any remaining grid regions can be created with other grid tools.

II.A. Extrusion Method

While extruding a surface mesh composed of 2D elements into a volume grid seems straightforward, there are a myriad of methods available to accomplish it. One of the more common methods is a simple translation of the vertices a prescribed distance along the point normals. For the present work, the extrusion method chosen was the face offsetting method (or FOM) described by Jiao [5]. This method starts by offsetting all the faces connected to a particular vertex by a prescribed distance and finding the intersection of the offset faces. Since the number of faces can vary, the intersection can be under- or over-prescribed and a least squares fit is used to obtain a valid solution. From Jiao's work, it can be shown that the least squares intersection is a linear system, shown in equations 1-3, where $\mathbf{N}$ is the matrix of face normals and $\mathbf{a}$ is the vector composed of the face offset distances.

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1}$$

$$\mathbf{A} = \mathbf{N}^T\mathbf{W}\mathbf{N} \tag{2}$$

$$\mathbf{b} = \mathbf{N}^T\mathbf{W}\mathbf{a} \tag{3}$$

Next, an eigenvalue analysis described by Jiao, is used so that the vertex translation is restricted to the primary space composed of the primary eigenvectors. The offset vector, d, is defined in equation 4, where k is the dimension of the primary space, $\mathbf{e}_i$ is a primary eigenvector of $\mathbf{A}$ and λ_i is the associated eigenvalue. This process preserves ridges and corners, where appropriate. The remaining eigenvectors constitute the null space, which is saved for subsequent smoothing steps. While more computationally expensive than the point offsetting method, Jiao's FOM has the advantage that it precisely offsets the faces and preserves the surface curvature. This is not the case for the point offsetting method, which will be shown in section III.

$$\mathbf{d} = \sum_{i=1}^{k} \mathbf{e}_i^T \mathbf{b} \mathbf{e}_i / \lambda_i \tag{4}$$

An example of the FOM applied to the shear groove geometry is shown in figure 3. There it can be seen that the method creates cells that maintain the planes of each face and the concave and convex ridges are also preserved. The basic FOM does not account for self-intersection, and it can be observed that some self-intersections occur in the concave corner of the shear groove.

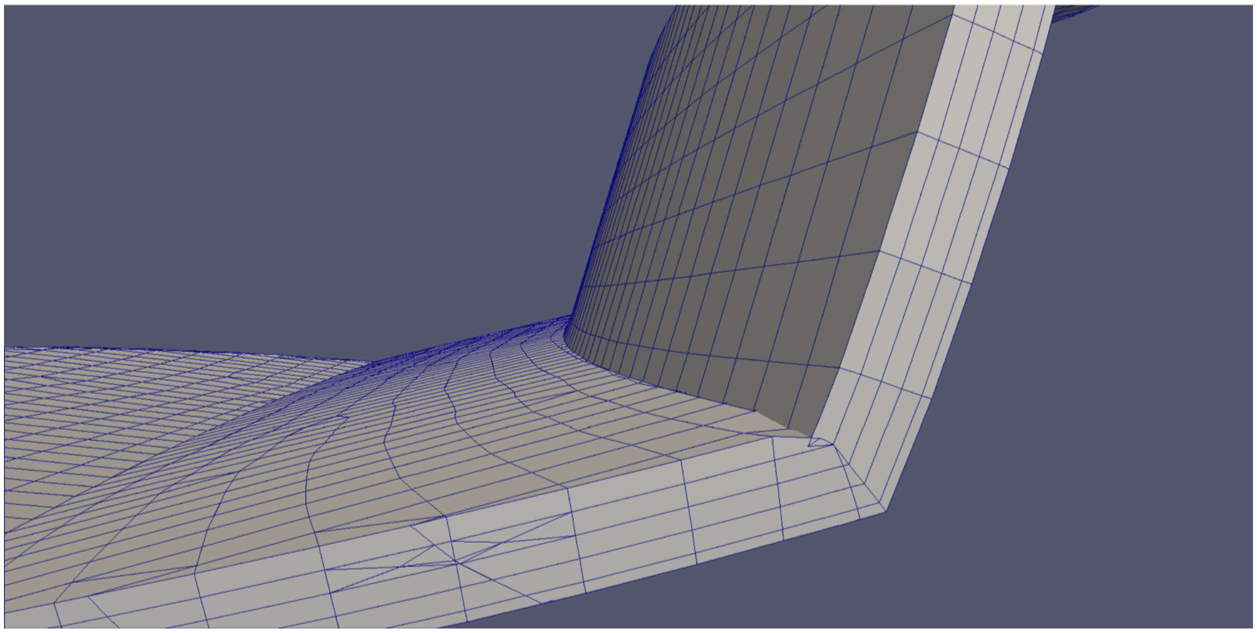


Figure 3. Results of the face offsetting method applied to a shear groove geometry. Each generated layer maintains the planes of each face, but the cells self-intersect in the concave corner.

II.B. Smoothing Methods

To address self-intersections in concave regions, the *Extrude* tool allows the user to apply various smoothing methods that spread the normals apart. Depending on the complexity of the surface mesh, some or all of the methods can be used to generate a valid mesh. Each of the methods are outlined below.

II.B.1. Normal Smoothing

The first smoothing step utilized in the *Extrude* tool is a normal smoothing method described by Tong et.al [6]. Each iteration of the normal smoothing process begins by computing new vertex normals, which are the average of the face normals connected to the vertex. Next, the face normals are re-computed by averaging the vertex normals connected to the face. The overall smoothing process is then repeated for the desired number of iterations (typically five to ten). Once all of the iterations are complete and the final vertex normals have been computed, the vertices are translated in the null space (determined in the face offsetting step) to match the smoothed normals. By limiting the vertex movement to the null space, the noise introduced to the mesh curvature is minimized.

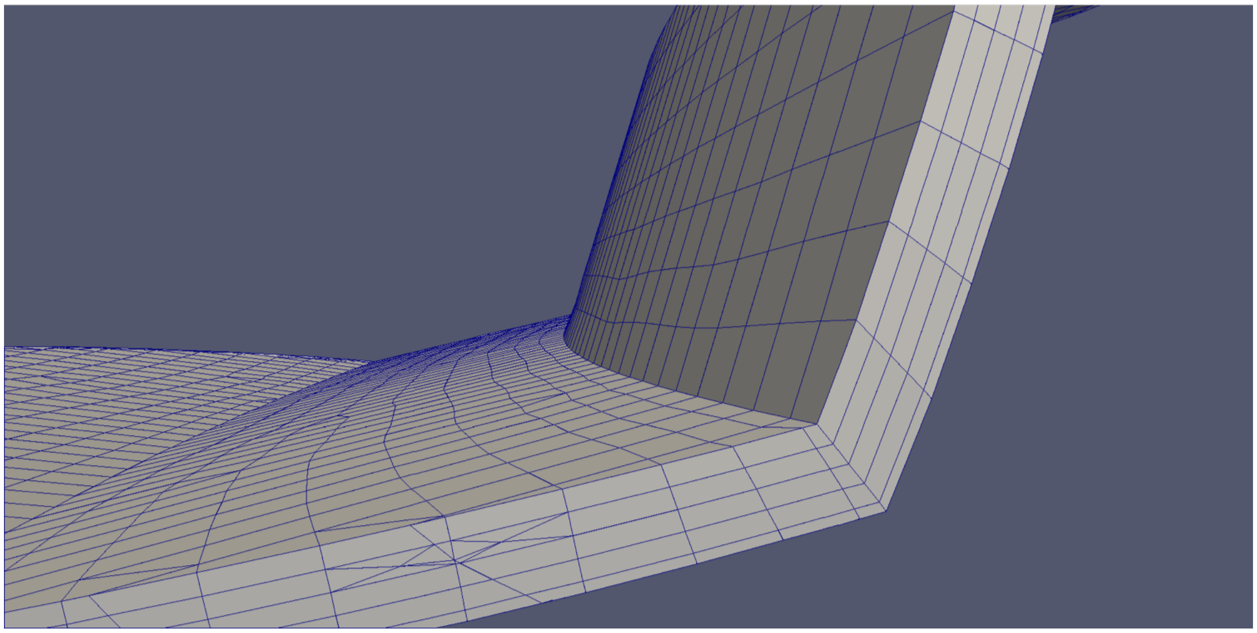


Figure 4. Face offsetting method with five normal smoothing iterations applied to the shear groove geometry.

II.B.2. Null Space Smoothing

The next smoothing step applied, if desired, is a null space smoothing method developed by Jiao [5]. The smoothing at each point is defined by equation 5, where $\mathbf{T}$ is a $3 \times (3 - k)$ matrix, the column vectors of which are the basis of the null space. Variable $\mathbf{c}_i$ is the vector from the vertex v to the centroid of face i, while w_i is the associated weight. For the *Extrude* tool, the weight chosen is the ratio of the current area to the initial area, recommended by Tong et.al [6]. This choice of weighting maintains the face size distribution as the surface is extruded. The advantage to using null space smoothing is that it preserves sharp features, and doesn't shrink the surface.

$$\mathbf{t} = \mathbf{T}\mathbf{T}^T \frac{\sum_{i=1}^{m} w_i \mathbf{c}_i}{\sum_{i=1}^{m} w_i} \tag{5}$$

While equation 5 doesn't implicitly depend on the face type, in practice it was found that the method created bad cells for certain topologies with mixed triangles and quads. To generalize the method for meshes with mixed faces, when a quad is connected to a specific vertex, only one-half of the connected quad is considered, with the portion of the quad defined by the two face edges that contain the vertex of interest.

When null space smoothing is chosen by the user, the number of iterations must also be specified. Depending on the severity of the concave corners, values between 10 and 150 iterations have been used. An example of using null space smoothing on the shear groove geometry is shown in figure 5. For this case, five normal smoothing iterations and 15 null space smoothing iterations were used. In the figure, it can be seen that the addition of the null space smoothing helped to further spread the cells away from the concave corner.

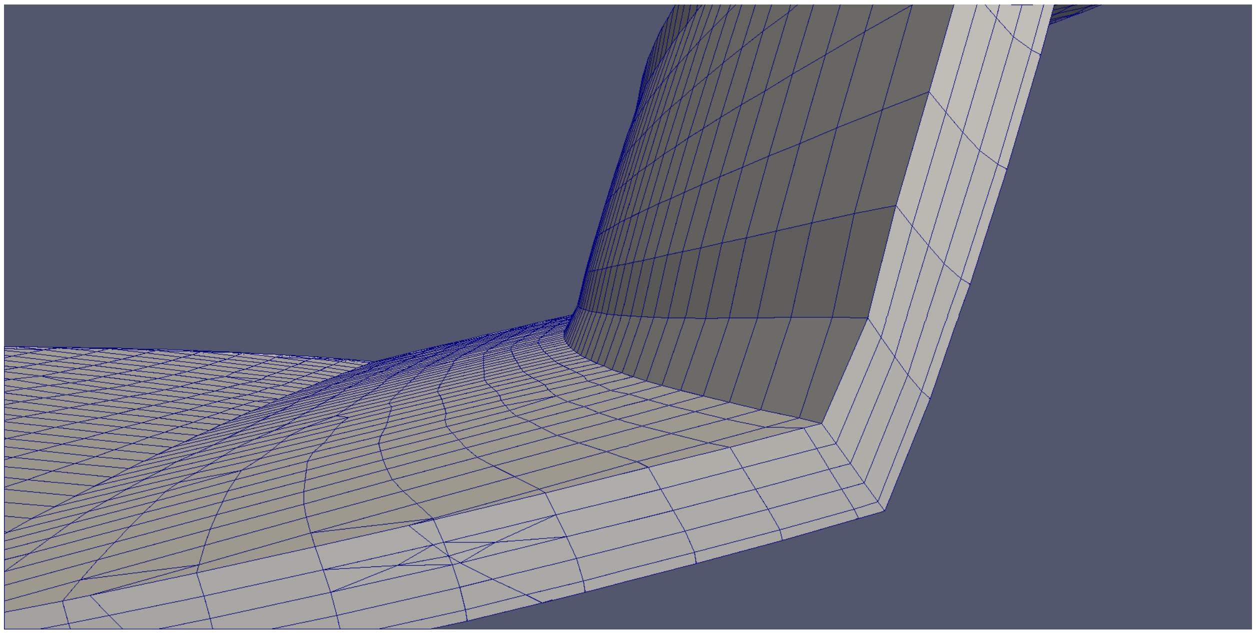


Figure 5. Face offsetting method with five normal smoothing iterations and 15 null space smoothing iterations applied to the shear groove geometry

II.B.3. Mean Curvature Smoothing

For difficult geometries, both normal smoothing and null space smoothing may not be enough to prevent cells from self-intersecting in concave corners. Mean curvature flow can be utilized to modify the layer thickness in high curvature areas, making the cells in concave regions thicker and the cells in convex region thinner. Mean curvature, $\bar{\kappa}$, is defined in equation 6 as the average of the principal curvatures κ_1 and κ_2.

$$\bar{\kappa} = \frac{1}{2}(\kappa_1 + \kappa_2) \tag{6}$$

Mean curvature flow is a smoothing method whereby each point is moved along its normal, $\mathbf{n}$, with a velocity $\bar{\kappa}$ as shown in equation 7.

$$\frac{\partial \mathbf{x}_i}{\partial t} = -\kappa_i \mathbf{n_i} \tag{7}$$

When equation 7 is applied in an explicit manner, small time steps are required to keep the method stable, making it inefficient. Therefore, the implicit scheme of Desbrun et.al [7], was used in the *Extrude* tool. This implicit scheme allows the smoothing to be solved in a single step, but requires more memory since the movement of all of the points is solved as a single linear system.

When the user enables mean curvature smoothing, a scaling factor must also be specified, which limits the change in the layer thickness. A scaling factor of 1.5 means that the layer thickness will be altered up to a maximum +/- 50% and is scaled on the curvature magnitude. This is demonstrated in figure 6 where the highest curvature magnitude in the grid is negative (concave) and the cells are 50% thicker than cells where the curvature is zero (flat).

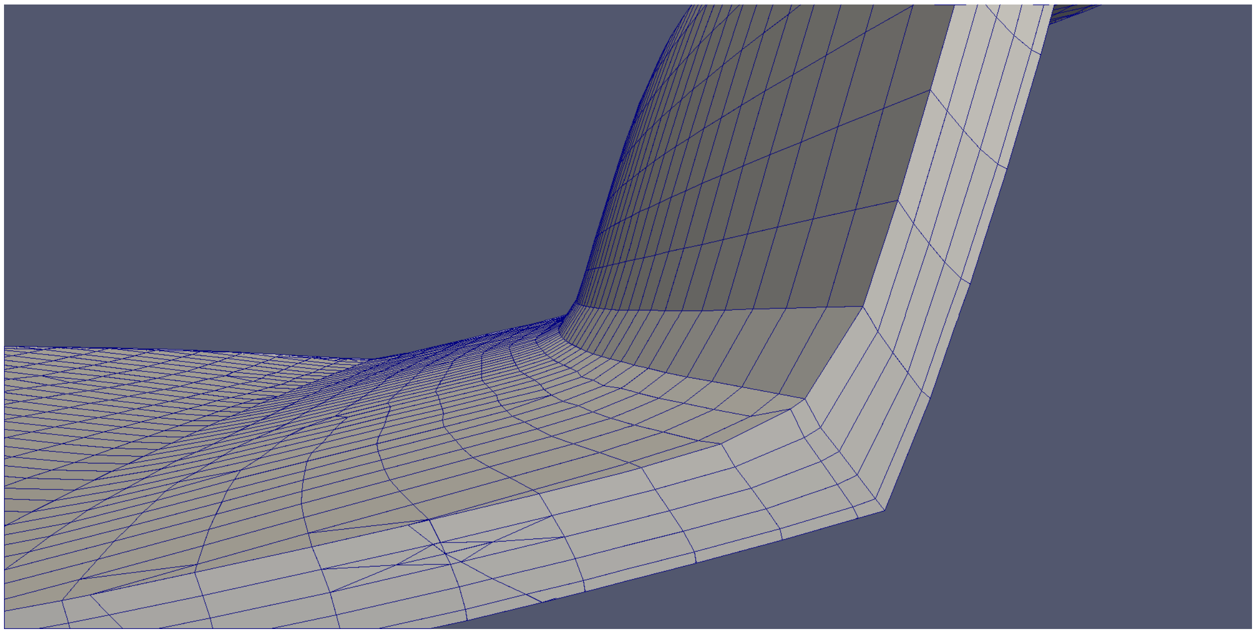


Figure 6. Face offsetting method with five normal smoothing iterations, 15 null space smoothing iterations and a curvature scaling factor of 1.5 applied to the shear groove geometry.

While this method can help prevent cells from self-intersecting, the user must take care not to get too aggressive with the scaling factor as the cell sizes in convex corners can be made so large that the boundary layer is no longer resolved properly.

II.C. Memory Management

When initially conceived, the *Extrude* tool was intended to create boundary layer grids for surface meshes on the order of 10 million faces with boundary layers approaching 1 billion cells. Performing operations on grids of this size requires more RAM than with which most workstations are equipped. To keep memory requirements reasonable, *Extrude* only keeps the the current layer in RAM and the previous layers are stored on the file system, with the nodes and each cell type assigned their own temporary data file. At the end of the grid extrusion process, the temporary data files are all appended together to create the final volume grid. During testing of a surface mesh with 7.8 million faces, extruded out 75 layers, the peak memory usage was 20 GB.

II.D. Parallelization

Existing grid generation tools are serial codes, leading to long processing times for large grids. For the *Extrude* tool, the FOM and the normal smoothing method have been implemented in a threaded manner using OpenMP [8]. Testing was conducted using both serial and parallel methods to ensure that the results were identical and that an appreciable speed up was achieved.

For just the FOM, the parallel process was tested on a surface mesh containing a quad dominant mixed element mesh containing 7.8 million nodes, to ascertain the speed up for an increasing number of threads. The test was performed on a Supermicro workstation equipped with Intel Xeon E5-2697V3 processors for a total of 28 cores and 128 GB of RAM. Results of the testing, shown in figure 7, show that the single core process takes 5.38 seconds to process a single layer, and as the number of cores are increased, the time required decreases to 0.41 seconds for 20 cores. Due to diminishing returns, the testing was stopped at 20 cores, which was sped up by a factor of 13.

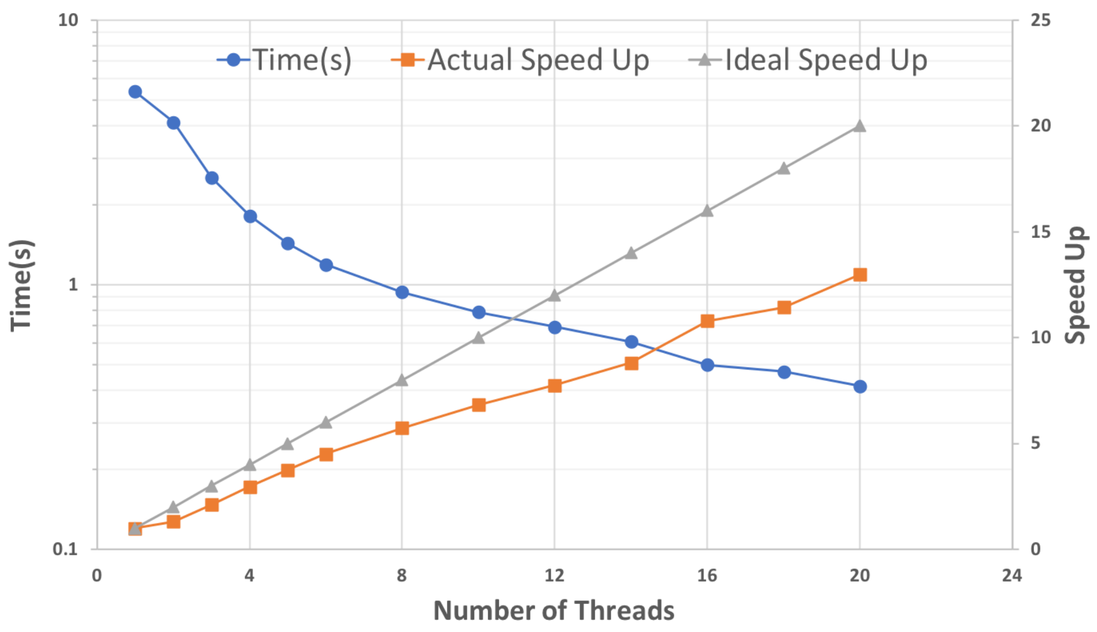


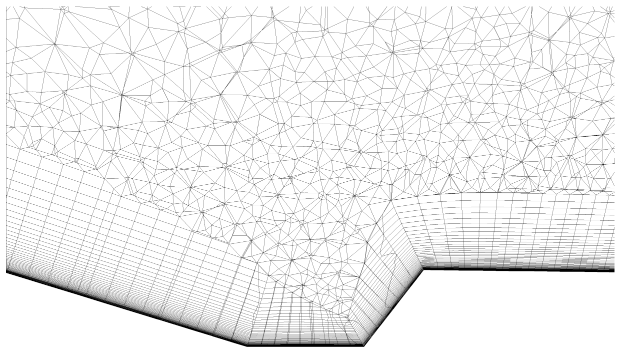
Figure 7. Parallel speed up of the face offsetting method using OpenMP. Tests were run on a workstation with 28 cores and 128 GB of RAM.

For large grids that require null space or mean curvature flow smoothing, the parallel FOM and normal smoothing methods provided only a moderate speed up to the overall time. Work is on-going to implement parallel versions of the the null space and mean curvature flow smoothing methods.

III. Tool Comparisons

After the initial tool development was completed, testing was conducted to compare the resulting prismatic grids to other grid generation packages. The first comparison made between AFLR3 and *Extrude* was on the shear groove geometry and is shown in figure 8. When the prismatic layer is generated by AFLR3 (figure 8a), all of the faces are initially extruded. After several layers, AFLR3 stops extruding the faces in the concave corner before they intersect. Later, AFLR3 fills the remaining region with tetrahedrons, but this can lead to highly skewed cells, or cells with high volume ratio cells.

In figure 8b, the same surface grid is extruded by the new tool. The various smoothing algorithms are able to prevent the cell in the concave corner from self-intersecting and the prismatic layer can be grown as thick as desired with the limitation that the layer spacing isn't overly large.


(a) Grid with AFLR.

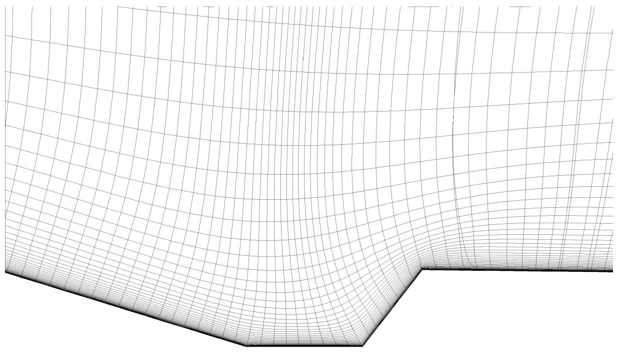

(b) Grid with Mesh Tools/Extrude.

Figure 8. Prismatic layers generated with different tools.

The second case used for comparison purposes was the extrusion of a prismatic layer around a shock for a hypersonic shock fit unstructured CFD solution. Prior work by the author [9] has shown that for unstructured hypersonic CFD cases, including a prismatic layer on both sides of the shock improves the surface heat flux predictions. For this comparison, two shock fit grids were generated, one using Pointwise and the other using the *Extrude* tool. For both grids, the shock surface was extruded outwards 10 total layers and inwards 14 layers. For the grid built with the *Extrude* tool, no smoothing was used. A CFD solution was then obtained using the LociCHEM [10, 11] CFD solver on both grids.

Figure 9 shows the initial shock surface, the grid metrics of the resulting outermost extrusion layer and the resulting heat flux obtained on both grids. In step 1, the original shock surface is shown, colored by the mean curvature. The mean curvature for the entire surface is positive (convex), with small variations in the magnitude. Step 2a shows the outermost layer of the extruded grid using Pointwise and the corresponding CFD solution can be seen in step 3a. The mean curvature of the Pointwise-generated surface varies from positive (convex) to negative (concave). This variation in mean curvature is likely a result of extruding the surface using a point offsetting method, which does not maintain the face planes.

Steb 2b in figure 9 displays the results for the same process when using the *Extrude* tool. The outermost extruded layer is colored by mean curvature. Because the face planes have been preserved, the mean curvature of the surface is also preserved. The resulting CFD solution can be seen in step 3b. The heat flux shows significantly less variation than the results from the grid generated using Pointwise (step 3a) and can be attributed to the decrease in noise generated by the grid generation process.

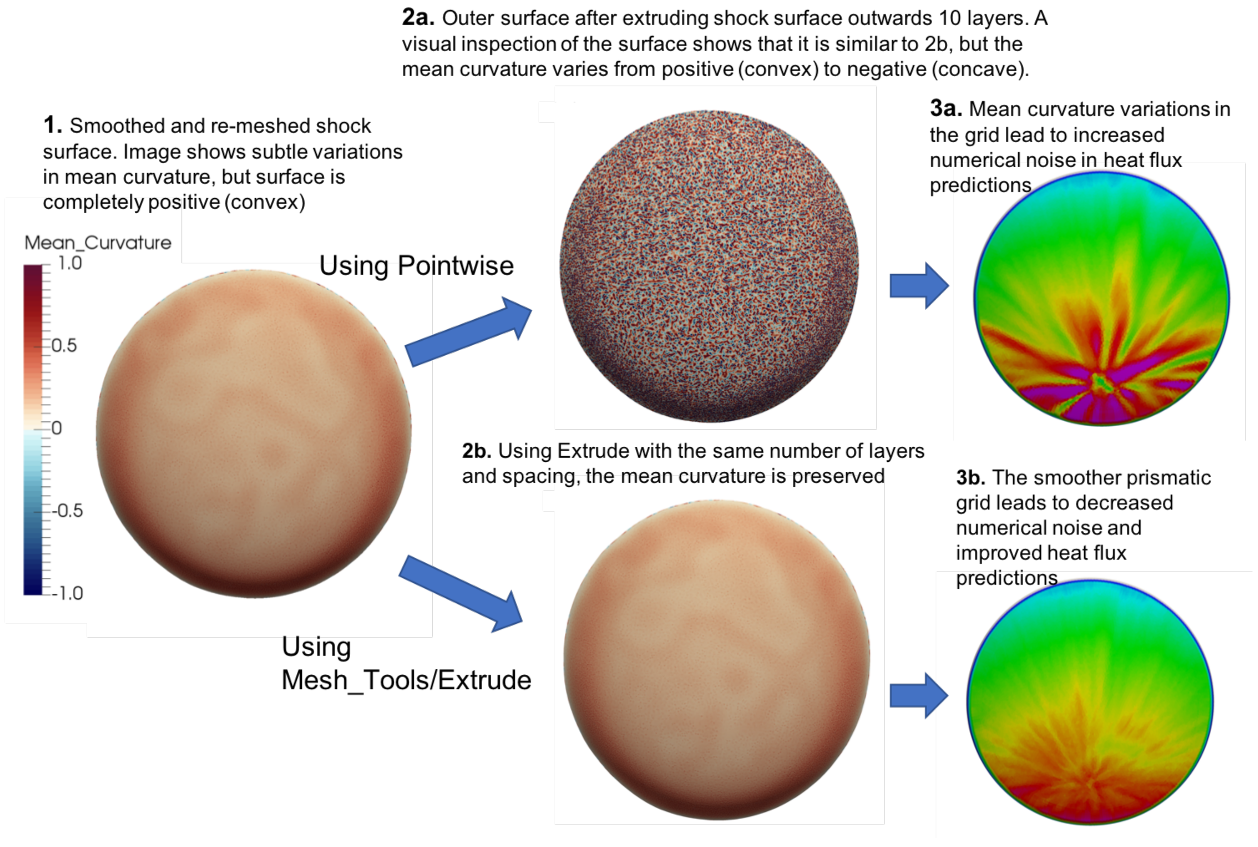
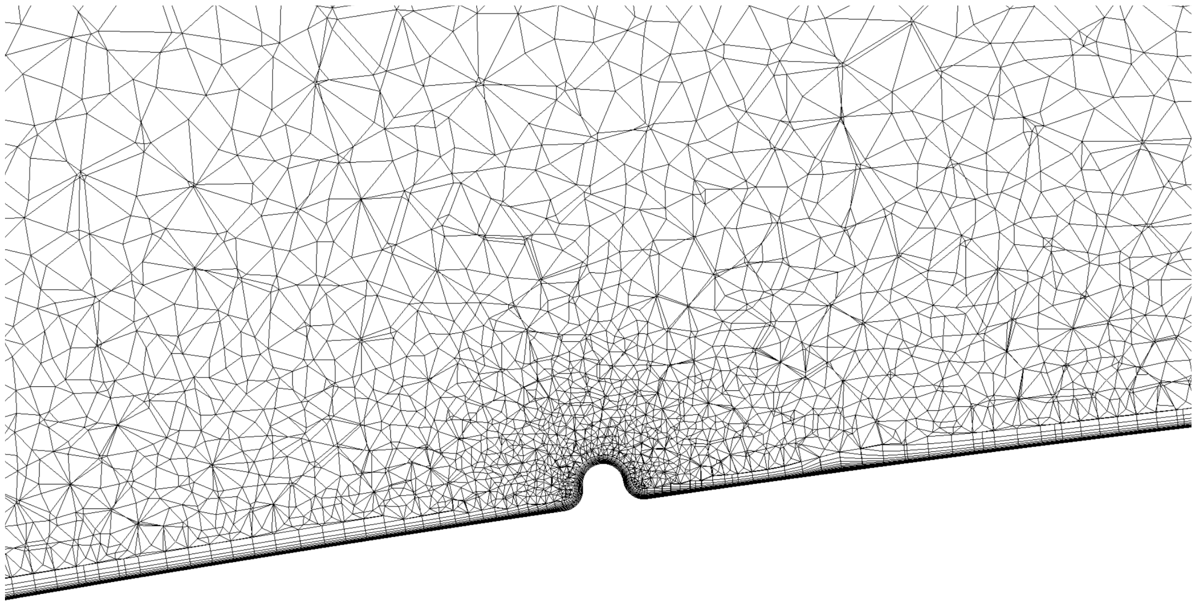

2a. Outer surface after extruding shock surface outwards 10 layers. A visual inspection of the surface shows that it is similar to 2b, but the mean curvature varies from positive (convex) to negative (concave).
1. Smoothed and re-meshed shock surface. Image shows subtle variations in mean curvature, but surface is completely positive (convex)
3a. Mean curvature variations in the grid lead to increased numerical noise in heat flux predictions
Mean_Curvature
1.0
0.5
0
-0.5
-1.0
Using Pointwise
2b. Using Extrude with the same number of layers and spacing, the mean curvature is preserved
3b. The smoother prismatic grid leads to decreased numerical noise and improved heat flux predictions
Using Mesh_Tools/Extrude

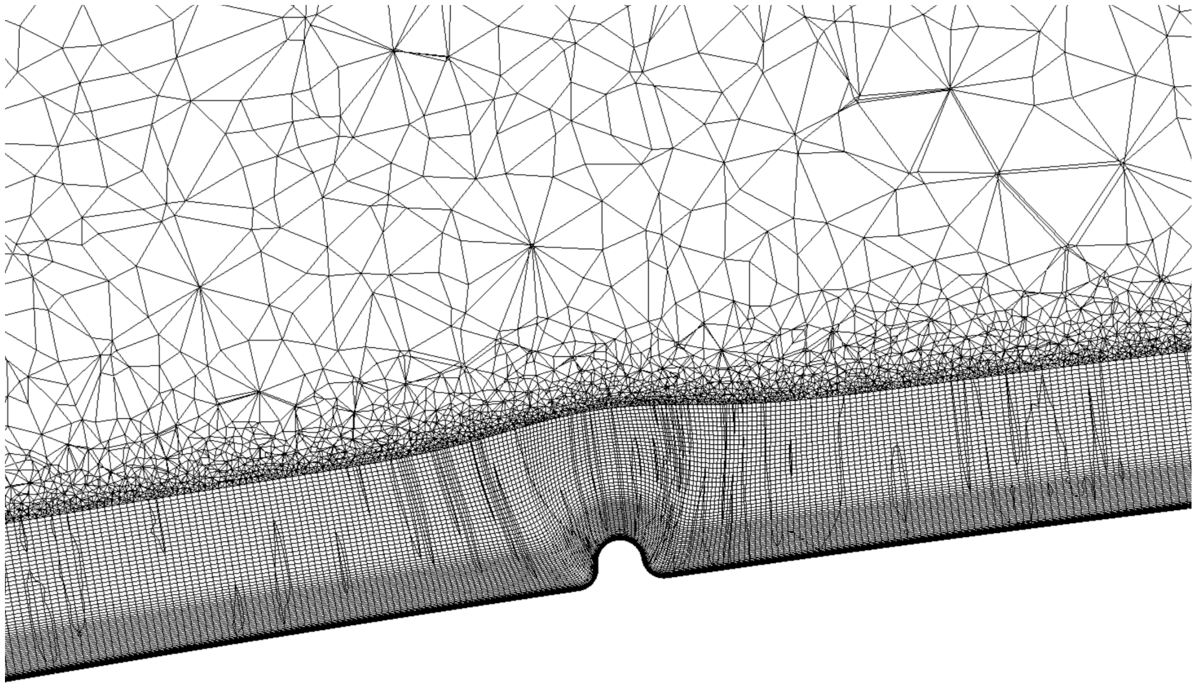
Figure 9. Grid metrics and CFD results for the shock layer extrusion case.

The final comparison was made by generating grids for the Orion block Avcoat heatshield, similar in nature to figure 1, where the blocks have receded to create fences at the seams. The fence dimensions are several orders of magnitude smaller than the heatshield itself, requiring large variations in face sizes to properly resolve all the features. A volume grid was generated for an early version of the surface grid using AFLR3, shown in figure 10a. Issues with the mesh are similar to the ones described for the shear groove geometry, with the extrusion process terminating locally when either the cells reached isotropic shape or to prevent self-intersection in the concave corners. CFD solutions were obtained on this grid, but the poor grid quality led to poor correlation with test data and prevented the solution being run in a time-accurate manner.

The latest version of the fence volume grid, generated using the *Extrude* tool is shown in figure 10b. The prismatic layer is much thicker, which helps the CFD solver resolve the boundary layer properly. The improved grid quality allowed the solution to be run in a time-accurate manner, which was found to be necessary for capturing the unsteady flow on the edges of the heatshield.



(a) Grid slice of a fence grid generated using AFLR3.



(b) Grid slice of a fence grid generated using Extrude for the prismatic layer and AFLR3 for the tetrahedrons.

Figure 10. Prismatic layers generated on a heatshield with fences.

IV. Conclusion

A new tool, *Extrude*, has been developed that addresses some of the shortcomings found in existing grid generation packages when extruding complex surface grids. This new tool uses a FOM, preserving surface curvature, and smoothing methods to grow thicker prismatic layers on complex grids. Testing has shown that the new tool behaves as expected on complex meshes and that the quality of simple shapes is improved with this method.

References

[1]Marcum, D., *Unstructured Grid Generation Using Automatic Point Insertion and Local Reconnection*, The Handbook of Grid Generation, Edited by J.F. Thompson, B. Soni, and N.P. Weatherill, Chapter 18, CRC Press, 1998, pp. 1-31.

[2]Marcum, D. and Weatherill, N., *Unstructured Grid Generation Using Iterative Point Insertion and Local Reconnection*, AIAA Journal, Vol. 33, No. 9, 1995, pp. 1619-1625.

[3]*Pointwise*, http://www.pointwise.com/pw/, Accessed: 2016-05-13.

[4]Schroeder, W., Martin, K., and Lorensen, B., *The Visualization Toolkit (4th ed.)*, Kitware, ISBN 978-1-930934-19-1, 2006.

[5]Jiao, X., "Face Offseting: A Unified Approach for Explicit Moving Interfaces," *Journal of Computational Physics*, Vol. 220, 2007, pp. 612-625.

[6]Tong, X., Thompson, D., Arnoldus, Q., Collins, E., and Luke, E.,"Three-Dimensional Surface Evolution and Mesh Deformation for Aircraft Icing Applications," *Journal of Aircraft*, Vol. 54, No. 3, May-June 2017.

[7]Desbrun, M., Meyer, M., Schroder, P., and Barr, A., *Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow*, Proceedings of ACM SIGGRAPH 99, 1999, pp. 317-324.

[8]Dagum, L. and Menon, R., *OpenMP: an industry standard API for shared-memory programming*, Computational Science & Engineering, IEEE, 1998, pp. 46-55.

[9]McCloud, P., *Best Practices for Unstructured Grid Shock Fitting*, AIAA-2017-1149, 55th AIAA Aerospace Sciences Meeting, AIAA SciTech Forum, 2017.

[10]Luke, E. and George, T.,"Loci: A Rule-Based Framework for Parallel Multidisciplinary Simulation Synthesis," *Journal of Functional Programming*, Vol. 15, No. 3, 2005, pp. 477-502.

[11]Luke, E., *On Robust and Accurate Arbitrary Polytope CFD Solvers (Invited)*, 18th AIAA Computational Fluid Dynamics Conference, Fluid Dynamics and Co-located Conferences, AIAA-2007-3956, 2007.