Eleventh International Conference on Computational Fluid Dynamics (ICCFD11), Maui, Hawaii, July 11-15, 2022

ICCFD11-11-1303

Parametrically Uniform Mesh Adaptation for Unstructured Grids

M. Lawry* and M. Opgenorth*

* Sierra Space, 1722 Boxelder St., Louisville, CO, 80027.

Abstract: Grid tailoring and adaptation is essential for rapid aerothermal computational fluid dynamic (CFD) analysis to ensure simulation fidelity. Industry standards use this approach with structured CFD solvers such as NASA Ames' Data Parallel Line Relaxation[1] (DPLR) and NASA Langley's Langley Aerothermodynamic Upwind Relaxation Algorithm[2] (LAURA). For complex reentry vehicles such as the Dream Chaser spaceplane, these adaptation methods can be impractical. New development at Sierra Space has shown promise in using a combination of Non-Uniform-Rational-B-Spline (NURBS) surfaces and parametric coordinate definitions to tailor any unstructured grid to a bow shock produced by unstructured solvers like FUN3D[3] and US3D[4].

Keywords: Numerical Algorithms, Computational Fluid Dynamics, Mesh Adaptation, Aerothermal.

1 Introduction

The common and industry standard technique for solving hypersonic flows, with accurate surface heating, is to adapt point-matched structured grids. This adaptation will move points to both the bow shock locations and within the boundary layer to resolve accurate surface heating. Both DPLR[1] and LAURA[2] utilized these methods and are considered the standards.

In recent years, several unstructured solvers have continued to develop their hypersonic capabilities, such as FUN3D[3] and US3D[4]. The difficulty is resolving the bow shock rapidly for many different freestream conditions. While a structured grid and solver can march along it's grid and adapt to find the shock front, unstructured solvers take additional steps to refine and adapt its grid. This process tends to be a time consuming process requiring user intervention. McCoud[5] recognized this short-coming and proposed best practices for unstructured shock detection and shock-fitting. His method outlines an iterative loop of shock-detection, shock smoothing, and re-meshing the volume grid. A second approach is to use grid refinement. Nastac et. al.[6] investigated using NASA refine[7, 8] to adapt unstructured grids to the bow shock of a hemisphere cylinder. The benefit to adaptation is that the refinement will also add cells to secondary shocks, but at the expense of increasing grid size and computational cost.

Parametrically Uniform Mesh Adaptation (PUMA) utilizes an initial solution's bow shock iso-surface and characterizes it using a Non-Uniform-Rational-B-Spline (NURBS) surface. Using this surface, an unstructured grid can be represented by a three dimensional parametric space. The parametric coordinates can be altered to get the desired effect, such as collapsing to a bow shock. The final step is to map the coordinates back to physical space. This approach differs from McCoud[5] because it does not have to remesh a volume grid and can continue from the previous restart file. Furthermore, this approach does not increase the cell count, similar to DPLR and LAURA, utilizing the original cells more efficiently, whereas refine techniques can dramatically increase the volume grid and slow the time to get a solution.

This technique has been benchmarked against classical aerothermal CFD cases such as a cylinder and hemisphere in hypersonic flow. Initial testing has been performed on the Dream Chaser with promising results. PUMA combines the ease of unstructured grid generation with the convenience of automated grid tailoring used by structured codes, while minimizing excessive dissipation of the solution through the bow shock.

2 Methodology

For simple geometries like a capsule, a surface grid can be generated, then extruded outwards to some arbitrary far-field distance. This mesh generation technique results in a structured mesh due to its global sense of order concerning mesh connectivity. In this work, we refer to this type of mesh as a structured mesh. The inherent global order of structured meshes can be advantageous in many ways; to increase computational efficiency, reduce memory requirements, and leverage knowledge of cell adjacency. For complex geometries, it becomes intractable to generate a structured grid, and instead an unstructured grid is used. Unlike structured grids, unstructured grids explicitly define and store mesh connectivity. A common technique to generate unstructured grids is called Delauney Triangulation[9]. For structured grids, mesh adaptation techniques exploit the globally consistent order of cell connectivity to expand or contract the grid, effectively refining areas of interest in the solution space. This requires all nodes be assigned to a unique grid line and any surface node must have a direct line from the surface to the grid boundary (k-line).

For unstructured grids, development of a new mesh adaptation method does not alter mesh connectivity or grid-point count and is described below. With exception to proprietary grid file formats, this method of mesh adaptation can be used for most Finite Element or Finite Volume solvers. PUMA involves the following steps:

1. Characterize the shock-front surface as a Non-Uniform-Rational-B-Spline (NURBS) surface
2. Define a three-dimensional parametric space based on the NURBS surface
3. Map each grid point from the mesh into parametric space
4. Alter parametric coordinates to get the desired effect
5. Map back into physical space, export mesh file

With an analytical expression for the shock front itself, the space inboard and outboard of the shock front are defined as a third dimension in parametric space. Three methods have been developed: vanishing point, vanishing line, and surface normal method. Illustrated in Figure 1, the green line represents the NURBS surface for the shock front, $\hat{p}_k^d$ are example grid points within the mesh, V_i^d are method specific parameters, black lines represent the direction of mesh adaptation for the example grid points, and green dashed lines represent constant values of the third parametric dimension. Each of these methods require a non-linear solve to convert the Cartesian coordinates for each grid point to parametric space, and will be described in further detail in Section 5.

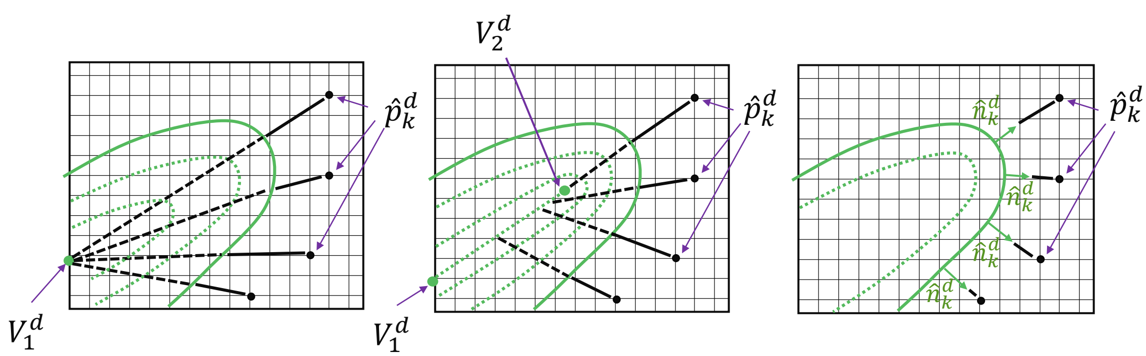


Figure 1: Parametric Space Definitions- Vanishing Point (Left), Vanishing Line (Center), Surface Normal (Right)

3 NURBS Definitions

To afford smooth and continuous mesh adaptation, we wish to transform the Cartesian coordinates of mesh grid-points into parametric coordinates, defined by the shock front. These parametric coordinates must be continuous, differentiable, and uniquely represent any point within the region of mesh adaptation. To this end, NURBS surfaces were selected as the foundation of our parametric representation. Subsequently, we have adopted the widely used Cox-deBoor's Algorithm[10] for B-spline functions:

$$B_{i,0}(t) = \begin{cases} 1, & \text{if } t_k \le t < t_{k+1} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

$$B_{i,d}(t) = \frac{t - t_k}{t_{k+d} - t_k} B_{i,d-1}(t) + \frac{t_{k+d+1} - t}{t_{k+d+i} - t_{k+1}} B_{i+1,d-1}(t) \tag{2}$$

where B is the blending function for control point i, at parametric location t, between the knot interval $t \in [t_k, t_{k+1}]$, for the specified degree represented by subscript $, d$. This algorithm is extensible to any degree of continuity $(, d)$. The n^{th} spatial derivative of this b-spline is known as:

$$\frac{\mathrm{d}^{(n)} B_{i,d}(x)}{\mathrm{d}x^{(n)}} = (d-1) \left(\frac{-\mathrm{d}^{n-1} B_{i+1,d-1}(x)/\mathrm{d}x^{(n-1)}}{k_{i+d} - k_{i+1}} + \frac{\mathrm{d}^{(n-1)} B_{i,d-1}(x)/\mathrm{d}x^{(n-1)}}{k_{i+d-1} - k_i} \right) \tag{3}$$

Any position (p^d) along the B-spline is defined in parametric coordinates (t) as:

$$p^d(t) = \sum_{i=0}^{m} B_{i,d}(t) C_i^d \tag{4}$$

where C_i^d is the i^{th} control point location. The superscript d represents the dimension being defined, e.g. x, y, or z, and m is the number of control points. A Non-Uniform Rational B-Spline (NURBS) normalizes the blending functions such that at any point along parametric coordinate t, the blending functions sum to one:

$$p^d(t) = \sum_{i=0}^{m} R_{i,d}(t) C_i^d, \tag{5}$$

$$R_{i,d}(t) = \frac{w_i B_{i,d}(t)}{\sum_{i=0}^{m} w_i B_{i,d}(t)} \tag{6}$$

where $R_{i,d}$ is the NURBS blending function, and w_i is a weighting value for the i^{th} control point. This normalization allows the object to be invariant under rotation, scale, translation, and perspective transformations. Subsequently, a NURBS surface is defined as:

$$p^d(l,t) = \sum_{i=0}^{m} \sum_{j=0}^{n} R_{ij,d}(l,t) C_{ij}^d, \tag{7}$$

$$R_{ij,d}(l,t) = \frac{w_{ij} B_{i,d}(l) B_{j,d}(t)}{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{ij} B_{i,d}(l) B_{j,d}(t)} \tag{8}$$

where surface location p^d is defined by two parametric coordinates, l and t. For brevity, derivatives of any position within a NURBS object with respect to control points or spatial location are omitted, but can be derived via chain rule.

4 Surface Characterization

By itself, an iso-surface of the shock front provided from the physics simulation is C^0 continuous, and not sufficient to inform mesh adaptation throughout the entire domain. We wish to characterize the shock front as a NURBS surface with $n \times m$ control points, such that the shape of the NURBS surface conforms to the shock iso-surface as closely as possible. Consider the following optimization problem:

$$\min \; z(C_{ij}^d) \tag{9}$$

$$\text{s.t. } C_{ij}^d \in \mathbb{R} \tag{10}$$

$$z(C_{ij}^d) = \sum_{d=1}^{3}\sum_{k=1}^{s} \hat{w}_k \left(p_k^d(l(C_{ij}^d), t(C_{ij}^d), C_{ij}^d) - \hat{p}_k^d\right)^2 \tag{11}$$

where $\hat{p}_k^d$ are the Cartesian coordinates for the k^{th} point on the shock iso-surface, $\hat{w}_k$ are weighting factors associated with the k^{th} point on the shock iso-surface p_k^d are the corresponding locations on the NURBS surface, C_{ij}^d are the control points for the NURBS surface, and s is the number of points on the iso-surface. The weighting factors $\hat{w}_k$ are arbitrary, but beneficial to assign when mesh discretization is non-uniform. For efficiency, this work utilizes a gradient based optimization algorithm. Design sensitivities are defined as:

$$\frac{\mathrm{d}z}{\mathrm{d}C_{ij}^d} = \sum_{d=1}^{3}\sum_{k=1}^{s} 2\hat{w}_k \left(p_k^d - \hat{p}_k^d\right) \frac{\mathrm{d}p_k^d}{\mathrm{d}C_{ij}^d} \tag{12}$$

and the derivative of the k^{th} NURBS location with respect to control points are defined in Equation 13.

$$\frac{\mathrm{d}p_k^d}{\mathrm{d}C_{ij}^d} = \frac{\partial p_k^d}{\partial C_{ij}^d} + \frac{\partial p_k^d}{\partial l}\frac{\partial l}{\partial C_{ij}^d} + \frac{\partial p_k^d}{\partial t}\frac{\partial t}{\partial C_{ij}^d} \tag{13}$$

Note that the sensitivity of NURBS location with respect to the control points has both intrinsic and transportive terms. Coincident location between the k^{th} shock-front grid point and NURBS surface is defined by Equations 14 and 15, where parametric coordinates l and t are solved for via Newton-Raphson inner-iterations.

$$\sum_{d=1}^{3} 2\left(p_k^d - \hat{p}_k^d\right)\frac{\mathrm{d}p_k^d}{\mathrm{d}l} = 0 \tag{14}$$

$$\sum_{d=1}^{3} 2\left(p_k^d - \hat{p}_k^d\right)\frac{\mathrm{d}p_k^d}{\mathrm{d}t} = 0 \tag{15}$$

It is relatively inexpensive to optimize a NURBS surface to fit a desired shock front iso-surface, with an analytical expression for design sensitivities. Figure 2 illustrates this process for a 250,000 grid point iso-surface (semi-transparent red surface). For this example, 6x6 control points were selected to represent the NURBS surface, and was iterated 150 times to find a best-fit design. In Figure 2, the NURBS surface control points are represented in black wire-frame, and the NURBs surface itself is illustrated as the grey surface. The optimization process exhibits a steep descent to a plateau, indicative of the local best-fit solution. The entire process of reading the iso-surface, and arriving at the optimal NURBS surface profile took 216 seconds, in Matlab, on a 4 core Xeon E3-1505M (2.8GHz). The maximum discrepancy of the NURBS surface position compared to the iso-surface grid point locations was 1%, normalized by the shock front depth.

5 3D Parametric Definitions

With a parametric representation of the shock surface defined, there are many ways that one could specify a third parameter for grid points above or below. This decision can heavily impact mesh quality during adaptation, and is problem-specific. This third parameter is analogous to the k-lines of structured mesh adaptation. Three methods were developed for characterizing the out-of-plane dimension; vanishing

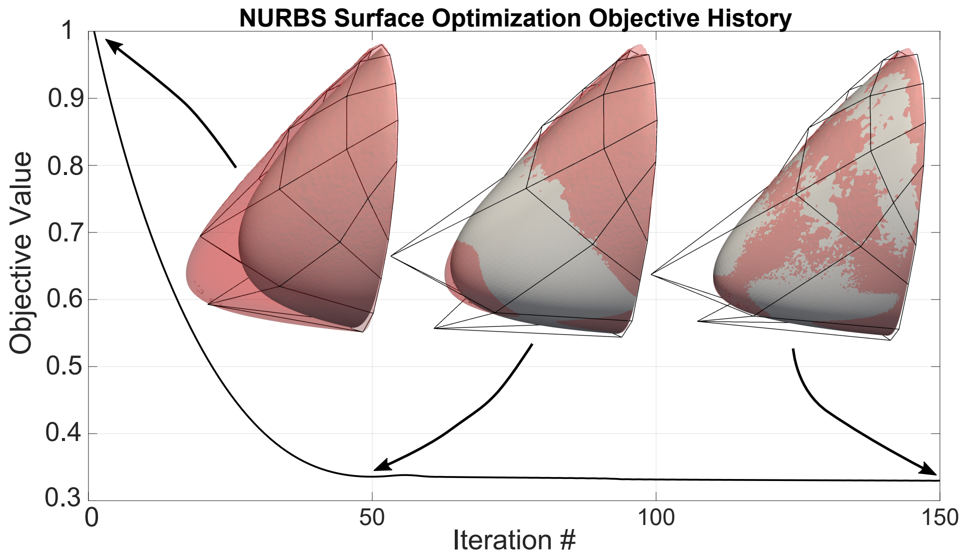


Figure 2: Shock front optimization, objective history example.

point method, vanishing line method, and surface normal method.

5.1 Vanishing Point Method

For the vanishing point method, mesh adaptation is prescribed by assuming the NURBS surface collapses to a singular point in space, V^d, which is user defined. To transform Cartesian coordinate grid-points to vanishing point parametric space, the following system of equations must be solved for variables l, t, and k:

$$(k+1)\sum_{i=0}^{m}\sum_{j=0}^{n} R_{ij,d}(l,t)\left(C_{ij}^{d} - V^{d}\right) - \hat{p}_{k}^{d} + V^{d} = \mathbf{0} \tag{16}$$

It is recommended to set the vanishing point to be somewhere in the wake region of the vehicle, centered in the shock front. As long as any line drawn from the vanishing point outwards only intersects the shock front surface once, convergence is guaranteed. This method also exhibits quadratic convergence when solving for variables l, t, and k. This method is ideal for near-spherical shock fronts. The left illustration in Figure 1 depicts the pseudo k-lines of adaptation in black dashed lines for grid-points $\hat{p}_k^d$.

5.2 Vanishing Line Method

The vanishing line method prescribes mesh adaptation by assuming the NURBS surface collapses to a line in space, defined by V_1^p and V_2^p. The vanishing point for the k^{th} NURBS surface location (p_k^d) exists somewhere between V_1^d and V_2^d. This relationship is defined in Equation 17.

$$V_k^d = \frac{\sum_{d=1}^{3}\left(\left(p_k^d - V_1^d\right)\left(V_2^d - V_1^d\right)\right)}{\sqrt{\sum_{d=1}^{3}\left(V_2^d - V_1^d\right)^2}}\left(V_2^d - V_1^d\right) + V_1^d \tag{17}$$

The following system of equations must be solved for variables l, t, and k.

$$(k+1)\sum_{i=0}^{m}\sum_{j=0}^{n} R_{ij,d}(l,t)\left(C_{ij}^{d} - V_k^{d}\right) - \hat{p}_{k}^{d} + V_k^{d} = \mathbf{0} \tag{18}$$

Similar to the vanishing point method, as long as any line drawn from the vanishing line outwards intersects the shock front only once, convergence is guaranteed. This method is better suited for elliptical shock fronts, however, the convergence rate for this method is linear. It can take some effort to find the optimal locations for V_1^d and V_2^d.

5.3 Surface Normal Method

The surface normal method assumes the third parametric coordinate (k) is aligned with the NURBS surface normal direction. The NURBS surface normal at the parametric coordinates l, t, is defined as:

$$n_k^d = \frac{\partial p_k^d}{\partial l} \otimes \frac{\partial p_k^d}{\partial t}, \qquad \hat{n}_k^d = \frac{n_k^d}{|n_k^d|} \tag{19}$$

where $\hat{n}_k^d$ is the unit direction out-of-plane on the NURBS surface. To convert coordinates into surface-normal oriented parametric coordinates, the following system of equations must be solved for l, t, and k:

$$\sum_{i=0}^{m}\sum_{j=0}^{n} R_{ij,d}(l,t)C_{ij}^d + k\hat{n}_k^d(l,t) - \hat{p}_k^d = \mathbf{0} \tag{20}$$

Mesh adaptation for this method is defined by the surface alone, without additional parameters. Convergence of the parametric solve is only guaranteed if the shock front is a convex hull shape. However, as long as the pseudo 'k-lines' do not intersect within the local region of mesh adaptation, the parametric solve is guaranteed locally. This method requires a good initial guess at the parametric location, and exhibits a linear convergence rate.

6 Mesh Adaptation

With parametric coordinates l, t, and k known, the focus of this study is on modifying parameter k to condense the mesh near the shock front. Figure 3 illustrates lines of constant l and t for each method, radiating outwards from the shock front. These lines are the prescribed direction of mesh adaptation for each method. The CFD grid is illustrated as a grey surface.

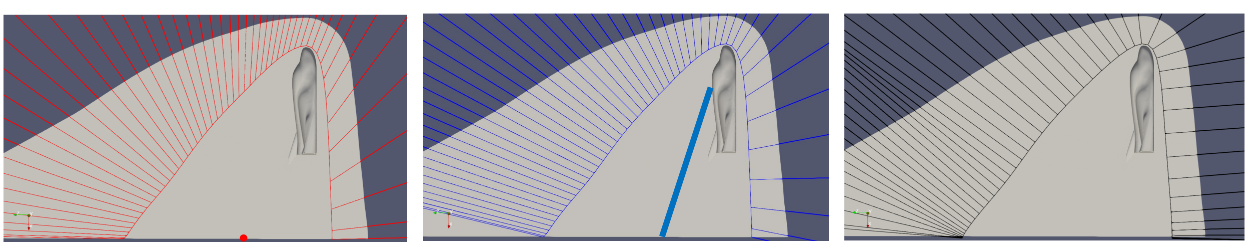


Figure 3: Adaptation lines for vanishing point method (left), vanishing line method (center), and surface normal method (right).

For all three methods, parameter k is conveniently zero at the shock front, positive upstream from the shock front, and negative downstream from the shock front. To condense the mesh outboard of the shock front:

$$k(k \geq 0) = \epsilon k(k \geq 0) \tag{21}$$

where ϵ is a compression factor, $0 < \epsilon < 1$. One may define ϵ as a scalar value, however for mesh quality considerations, it is beneficial to gradually taper this compression factor to the un-adapted portion of the mesh:

$$\epsilon = \begin{cases} 0, & \text{for } k < a \\ \frac{C}{2}\left(\sin\left(\frac{\pi(k-a)}{b-a} - \frac{\pi}{2}\right) + 1\right), & \text{for } a \leq k \leq b \\ C, & \text{for } k > b \end{cases} \tag{22}$$

Mesh compression as a function of parameter k is illustrated in Figure 4, where a is the parametric location where mesh adaptation begins, b is the parametric location where compression attains its maximum value, and C is the maximum compression value. With the new parameter k defined, one may map back to physical coordinates by solving Equations 16, 18, or 20 for $\hat{p}_k^d$.

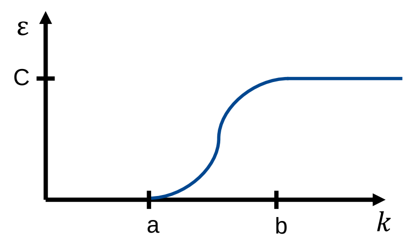


Figure 4: Compression vs. k parameter.

7 Benchmarks

To understand how these techniques work in practice, several test cases and benchmarks were performed. The first case presented was a quarter hemisphere in hypersonic flow at Mach 17.6. A quarter hemisphere was chosen to investigate PUMA's performance with a structured grid and two symmetry planes. The second case presented examines a more complicated geometry of the Dream Chaser. A mixed-element grid of prisms and tetrahedrals was adapted to a Mach 15.6 hypersonic flowfield. NASA's FUN3D[3] was used to solve all cases.

7.1 Hemisphere in Hypersonic Flow

A quarter hemisphere in hypersonic flow was examined for several reasons. Firstly, a hemisphere in hypersonic flow is a classic test case where comparisons can easily be found and reproduced. Secondly, a fully structured grid along with two planes of symmetry were created to compare PUMA to common hypersonic codes such as DPLR[1] and LAURA[2]. The quarter hemisphere consisted of an O-H style grid with two blocks covering a diameter of 1 meter. The first block on the axis of the hemisphere consisted of 17 points in both the i and j directions. The radial block was 21 by 33 in the i and j directions respectively. There are 129 points from the hemisphere wall to the farfield. Figure 5 shows the quarter hemisphere grid.

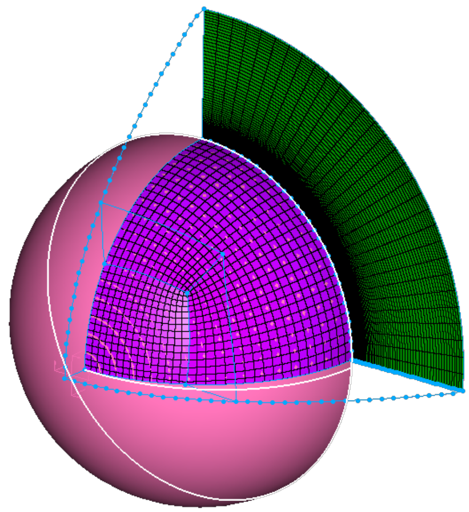


Figure 5: Quarter hemisphere grid

The Mach 17.6 flow field was produced with density of 0.001 kg/m^3, temperature of 200 K, and velocity of 5000 m/s. The wall temperature was set to 500 K with reaction cured glass (RCG) catalytic model. After an initial solution, PUMA's vanishing point method adapted the grid to the bow shock three times using 95% of the free stream Mach number as the metric. PUMA's vanishing point's parameters are provided in Table 1. The results and adaptation to the shock front from this case can be seen in Figure 6.

Table 1: Vanishing point parameters

Adaptation	C	a	b
1	0.60	-0.05	-0.02
2	0.75	-0.05	-0.02
3	0.80	-0.04	-0.01

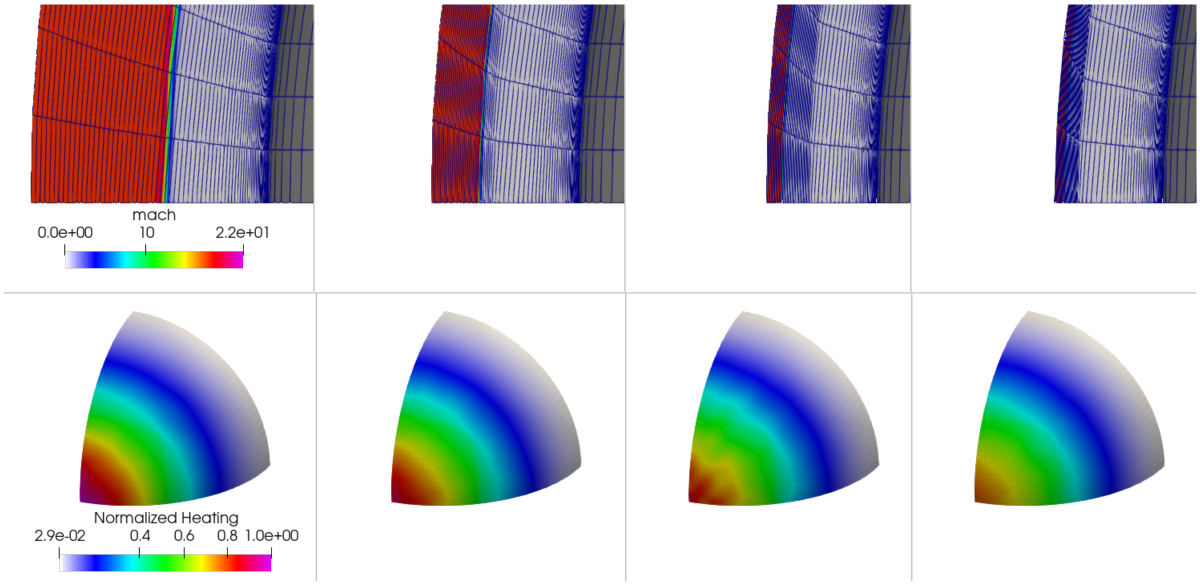
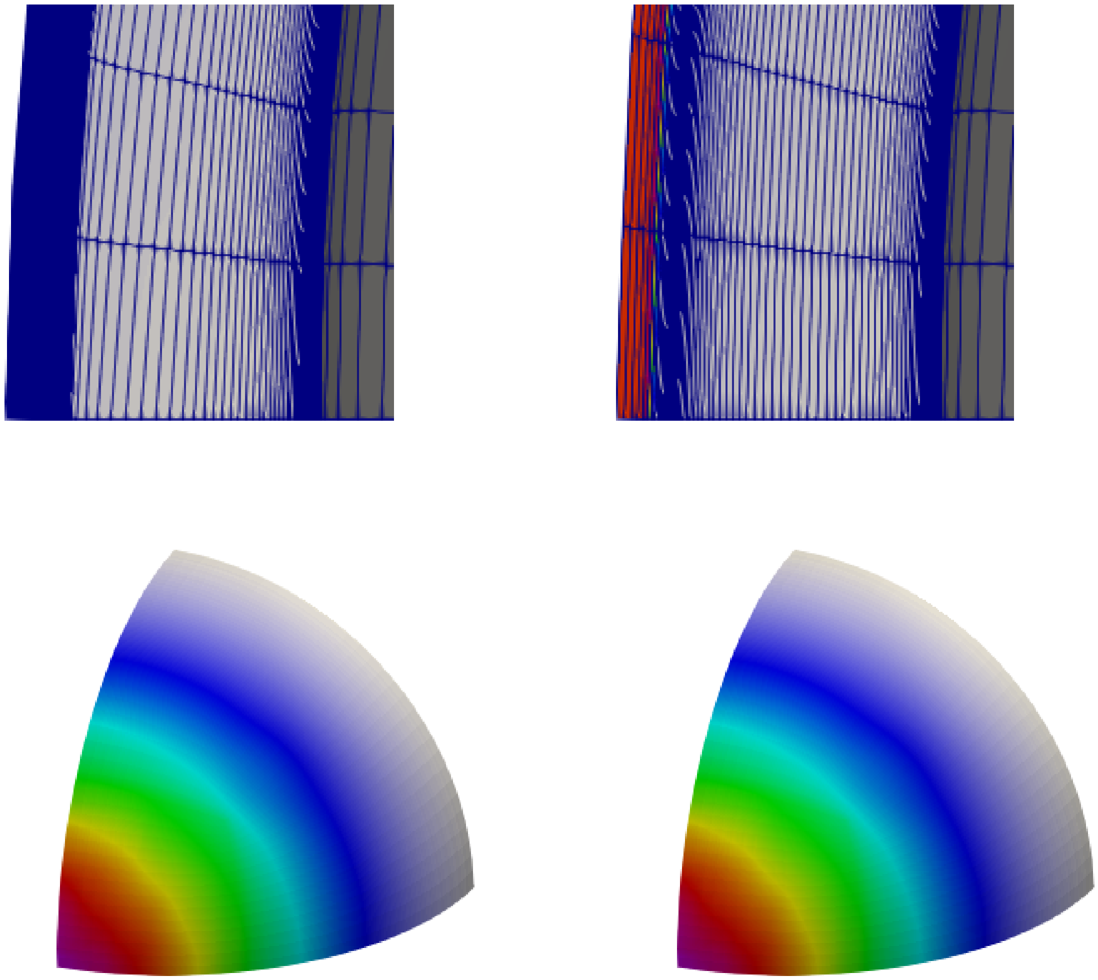


Figure 6: The progression of adaptations for the quarter hemisphere test case using PUMA. The leftmost picture is the initial grid and the rightmost is the third adaptation.

In order to compare PUMA's mesh adaptation results, comparisons to FUN3D's implementation of LAURA's mesh adaptation scheme, LADAPT, was executed. In similar fashion, three adaptations were performed after an initial solution. The LADAPT targeted temperature and the parameters used for all adaptations were fsh of 0.9, fctrjmp of 1.05, and ep0grd of 4. These represent the percentage of grid points inside of the shock, ratio of the freestream to detect the shock, and grid clustering to the shock, respectively. Figure 7 shows the final adaptation comparison of PUMA and LADAPT. For the given settings, PUMA was able to cluster the grid tighter to the bow shock compared to LADAPT. The level of PUMA's clustering is not needed for this case and at the time of writing, PUMA is not able to modify the grid inside of the bow shock metric.



(a) PUMA (b) LADAPT

Figure 7: Comparison of PUMA and LADAPT, focusing on the mesh near the bow shock after adaptation. Top images colored by Mach, bottom images colored by heating.

Figure 8 shows the centerline heating for each adaptation of PUMA. After the second adaptation, a reduced number of iterations were run to speed up the analysis. As a consequence, the surface heating was not converged and some oscillatory behavior can be observed. However, the bow shock converges much quicker than the surface heating and was satisfactory to perform the next adaptation. It can be seen in Figure 9 that the heating results from PUMA agree really well with LADAPT's method.

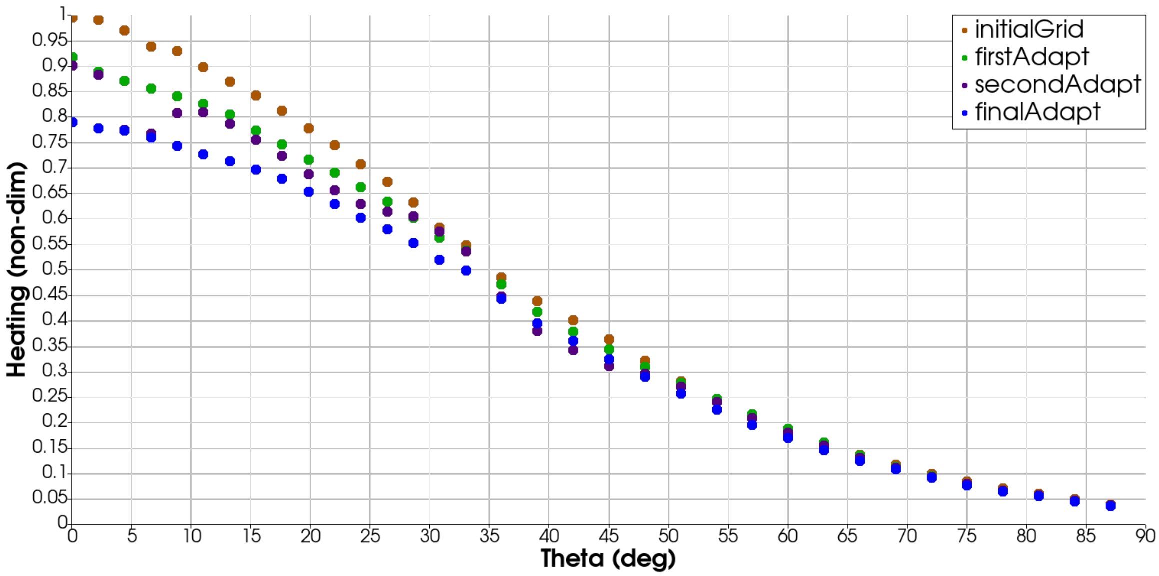


Figure 8: Heating along the centerline of the hemisphere for all adaptations using PUMA.

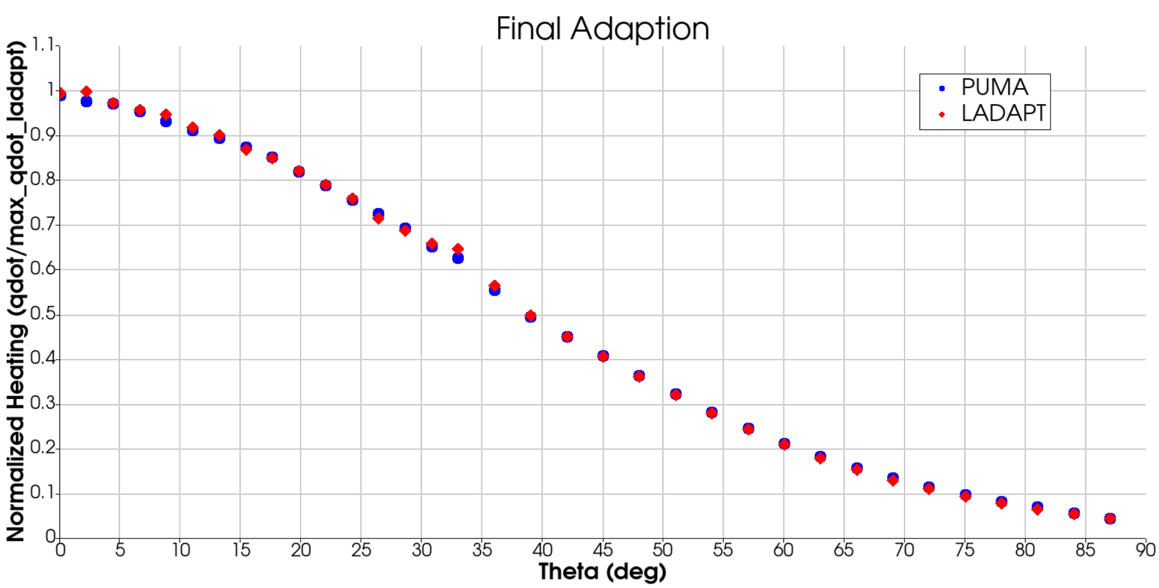


Figure 9: Heating along the centerline of the hemisphere showing the final adaptation results for both PUMA and LADAPT.

7.2 Dream Chaser in Hypersonic Flow

To further understand PUMA's capabilities with real-world geometry, the Dream Chaser was selected as a benchmark case. Furthermore, the grid generated for Dream Chaser only consisted of prisms and tetrahedral cells. The goal is to achieve accurate results with an easily generated grid. The grid was generated with 35 prism layers from a surface grid of only triangular elements. The volume had a reasonable cell width growth rate to the far field. The total node count was roughly 6.7 million. In order to get the correct location of the bow shock, PUMA was executed four times utilizing the surface normal method. Table 2 provides the parameters used for each adaptation.

Table 2: Surface normal parameters

Adaptation	C	a	b
1	0.97	-0.15	0
2	0.97	-0.08	0
3	0.97	-0.05	0
4	0.97	-0.05	0

After the first solution PUMA was able to adapt and compress the farfield to the shock front. Subsequent adaptations is mainly adjusting the grid to the bow shocks final location and converge on an accurate nose heating value. Figure 10 shows the initial and final adaptation. Figure 11 focuses on the nose and shows the reduction in bow shock thickness after four adaptations. Furthermore, it can be seen in Figure 12 that after second adaptation, PUMA has reasonably converged on the final shock location and is close to the final value. With some more exploration, it may be determined that the fourth adaptation is unnecessary in this case, ultimately saving some computation time.

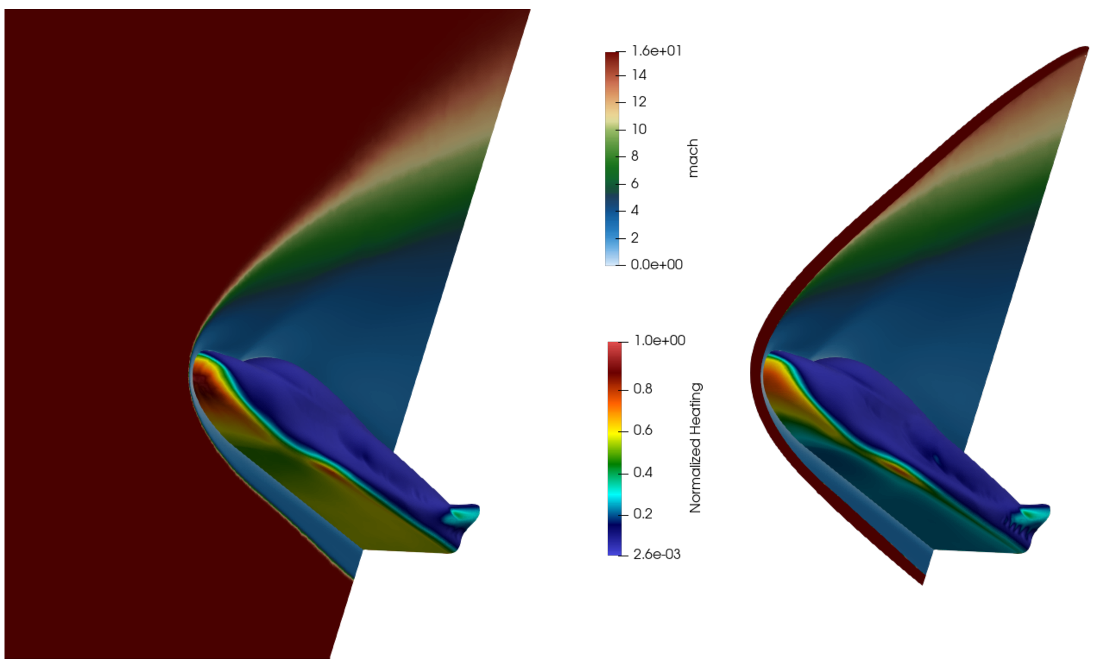


Figure 10: Symmetry plane Mach and surface heating, for the original mesh and 4^{th} PUMA adaptation.

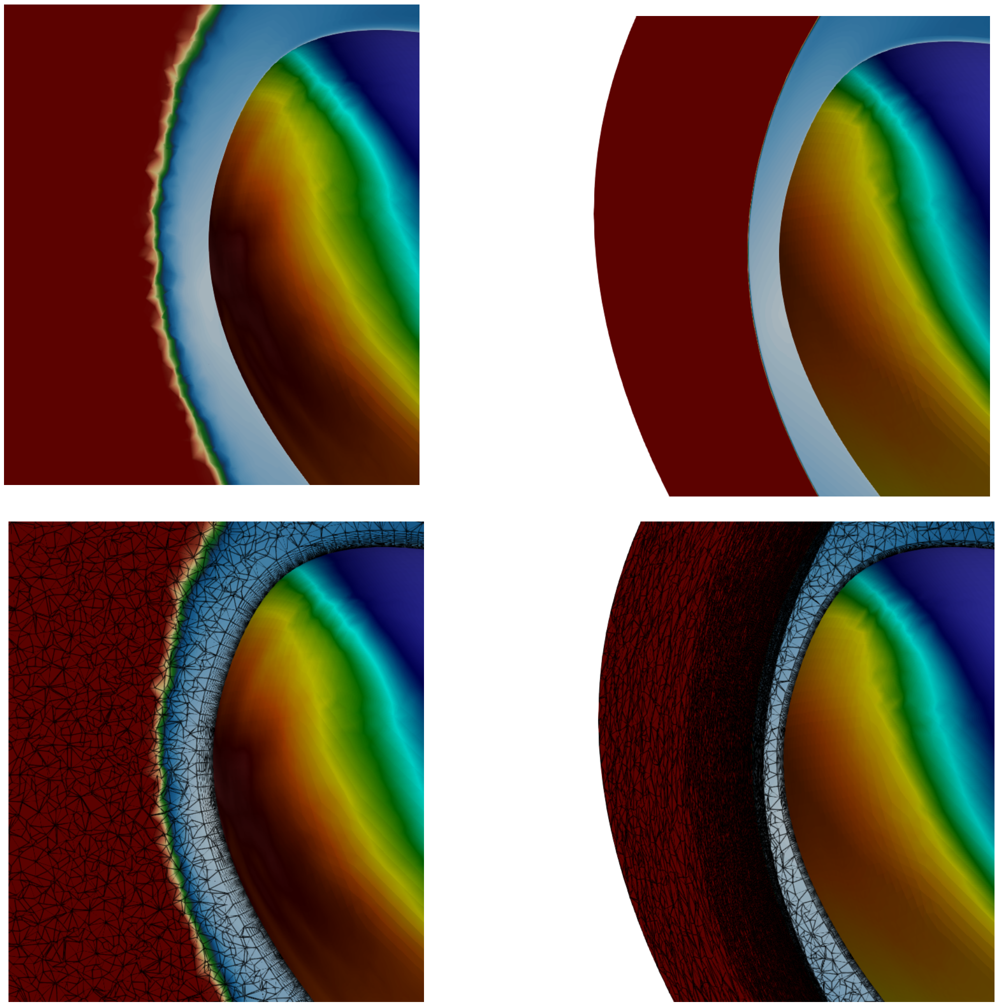


Figure 11: Comparison of the initial to final PUMA adapted grid and solution.

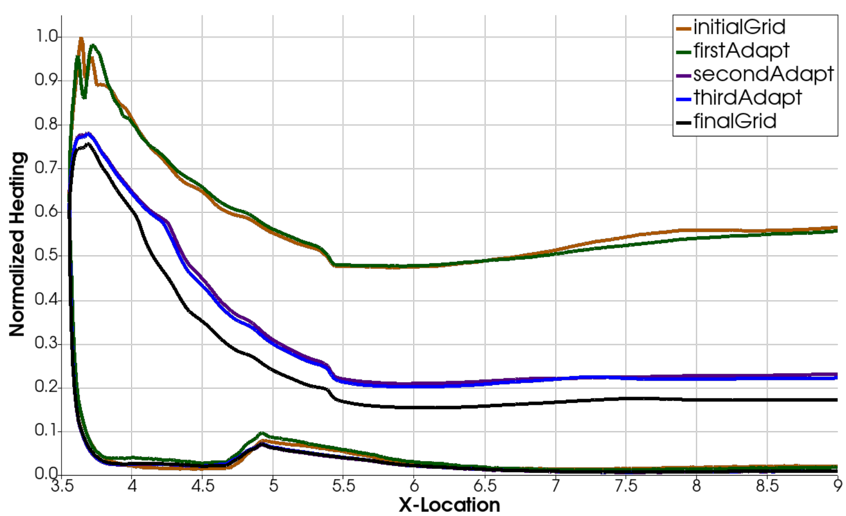


Figure 12: Heating along the centerline of Dream Chaser for all PUMA adaptations.

To further explore the quality of the surface heating results the same conditions was run using DPLR. Figure 13 shows PUMA's final adaptation compared to the final adapted DPLR solution. PUMA's final grid results in max nose heating 2.4% higher than DPLR. The lower aeroshell difference grows to around 50%. The primary reason for these differences is that PUMA has not been extended to adapt the region inside of the bow shock. For this Mach number, there are only a few cells between the adapted grid and the outermost prism layer near the nose. Additionally, there are not enough cells between the bow shock and the lower aeroshell further down the vehicle. Future versions will explore this region and increase fidelity.

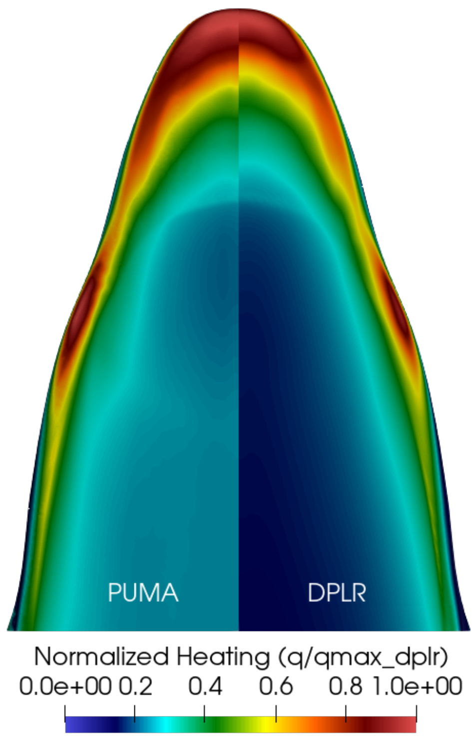


Figure 13: Side-by-side comparison of PUMA and DPLR surface heating.

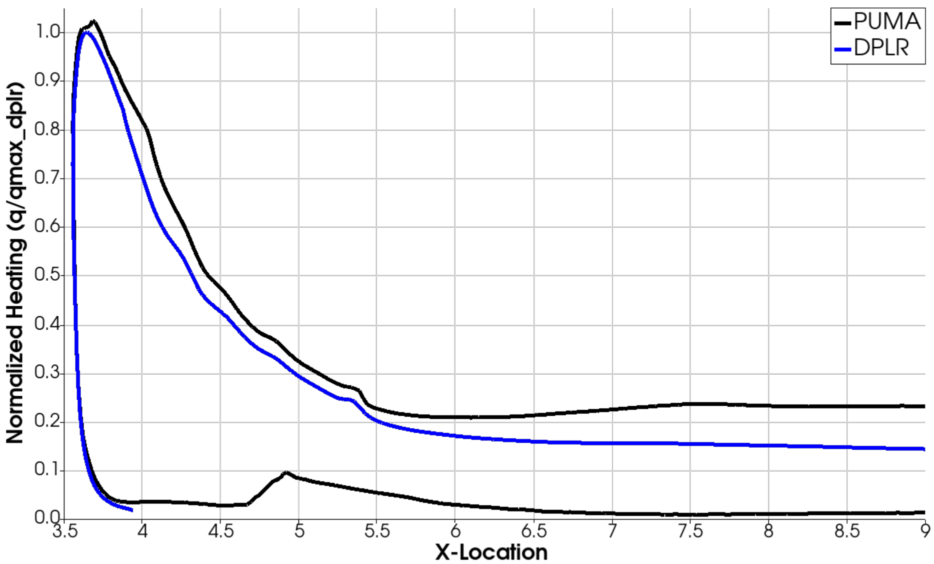


Figure 14: Centerline heating results from PUMA and DPLR.

8 Conclusion and Future Work

PUMA has shown to be robust and effective method to compress both structured and unstructured grids to the bow shock. For shock-front NURBS characterization, Section 4 introduced gradient based optimization of NURBS control point locations to quickly converge on a locally optimal geometry. The authors recognize that the quality of fit is subject to the number of control points defined, and the initial solution provided. Regularization penalties on control point spacing, or on NURBS surface curvature, could reduce the dependency of optimal geometry on the initial solution. Three methods of defining 3D parametric space were introduced in Section 5, informing the direction of mesh adaptation throughout the solution space. These methods have no knowledge of the discretized domain boundaries or mesh characteristics, requiring the analyst to determine the best approach. A NURBS surface fit of the far-field boundary could inform 3D parametric space definitions, and improve uniformity of the mesh upstream of the shock front.

In Section 7.1, the classical quarter hemisphere in hypersonic flow benchmark was solved with PUMA, and compared to results produced by LADAPT using the same discretized mesh. Heating along the centerline of the hemisphere was shown to be in good agreement. In Section 7.2, the Dream Chaser hypersonic benchmark was explored. This benchmark was discretized with an unstructured mesh and solved with PUMA, and compared to the solution determined by DPLR with a structured mesh. While nose heating was within 2.4% agreement, the discrepancy increases to 50% further downstream. This discrepancy is thought to be related to blending and number of cells between the bow shock and the prism layers. DPLR is able to adapt the mesh in this region, whereas PUMA is not currently equipped to do so. For future work, the CAD geometry of the surface could be used in conjunction with the methods described in Section 5 to adapt the region inside of the bow shock, as well as the boundary layer mesh.

For complex surface geometries where structured meshes are unfeasible, PUMA could be a promising alternative to Adaptive Mesh Refinement (AMR) techniques. Future work must be performed to benchmark PUMA against NASA refine[7, 8] or other AMR methods. Several improvements and additional capabilities are currently underway. Improvements can be made to the mesh adaptation using the distance from the shock to the vehicle surface. This will allow further control and adaptation into the region between the bow shock and the prism layers. Finally, isolation and freezing of the prism layers will be implemented.

References

[1] M. J. Wright, G. V. Candler, and D. Bose. Data-Parallel Line Relaxation Method for the Navier-Stokes Equations. *AIAA Journal*, 36(9):1603–1609, 1998.

[2] P. A. Gnoffo. An Upwind-Biased, Point-Implicit Algorithm for Viscous, Compressible Perfect-Gas Flows. NASA TP-2953, Feb. 1990, 1990.

[3] Carlson J.-R. Derlaga J.M. Gnoffo P.A. Hammond D.P. Jones W.T. Kleb B. Lee-Rausch E.M. Nielsen E.J. Park M.A. et al. Biedron, R.T. FUN3D Manual: 13.7. NASA/TM-2020-5010139, 2020, 2020.

[4] Graham V. Candler, Pramod K. Subbareddy, and Joseph M. Brock. Advances in computational fluid dynamics methods for hypersonic flows. *Journal of Spacecraft and Rockets*, 52(1):17–28, 2015.

[5] *Best Practices for Unstructured Grid Shock-Fitting*, 2017.

[6] Gabriel Nastac, Robert W. Tramel, and Eric J. Nielsen. *Improved Heat Transfer Prediction for High-Speed Flows over Blunt Bodies using Adaptive Mixed-Element Unstructured Grids*.

[7] M. A. Park. *Anisotropic Output-Based Adaptation with Tetrahedral Cut Cells for Compressible Flows,*. PhD thesis, Massachusetts Institute of Technology, Sep. 2008.

[8] Michael Park and David Darmofal. *Parallel Anisotropic Tetrahedral Adaptation*.

[9] B. Delaunay. Sur la sphère vide. *Bulletin de l'AcadeÂt́mie des Sciences de l'URSS. Classe des sciences matheÂt́matiques et na*, 1934(6):793–800, 1934.

[10] C. de Boor. Package for Calculating with B-Splines. *SIAM Journal on Numerical Analysis*, 14(10):57, 1973.