A METHOD FOR ADAPTIVE ANISOTROPIC REFINEMENT AND COARSENING OF PRISMATIC POLYHEDRA

Sandeep Menon[1] Thomas Gessner[2]

[1]*Ansys Inc, Chicago, IL, U.S.A. sandeep.menon@ansys.com*
[2]*Ansys Inc, Lebanon, NH, U.S.A. thomas.gessner@ansys.com*

ABSTRACT

A method is proposed to enable the anisotropic refinement and coarsening of prismatic boundary-layer polyhedra within the context of unstructured polyhedral meshes for viscous flow simulations using computational fluid dynamics. The method is compatible with the Polyhedral Unstructured Mesh Adaptation (PUMA) algorithm for the isotropic refinement and subsequent coarsening of general polyhedral cells. This allows simulations to leverage the full flexibility of polyhedral meshes for adaptive mesh refinement, while retaining the benefits of an improved accuracy-to-computational cost ratio.

Keywords: mesh adaptation, refinement, coarsening, anisotropy, polyhedra

1. INTRODUCTION

Polyhedral meshes have been established over the past two decades and are now widely supported in commercial and academic CFD codes [1][2][3]. For the finite volume method [4], polyhedral meshes combine the ease-of-use of unstructured tetrahedral mesh generation with the superior numerical properties [5][6] of hexahedral or structured meshes that are more challenging to generate automatically. The finite volume method does not impose any restrictions on the number of faces bounding a control volume in the mesh and consequently the number of neighbors per cell. The larger number of neighboring cells enables an enriched stencil and therefore the better approximation of gradients. This results in improved solution accuracy and faster convergence with fewer cells when compared to tetrahedral meshes [5][6] and an overall gain in computational efficiency.

Adaptive mesh refinement for numerical simulations has a rich history spanning nearly four decades [7][8], with the intention of balancing numerical accuracy with reduced computational cost. Traditional methods for mesh refinement employ templates based on cell type, and in the case of isotropic refinement, split these cells equally along all directions. Anisotropic methods have also been explored at various times, as a means of reducing the computational cost further by preferring certain directions for refinement while ignoring others that are irrelevant to the physics being resolved. Metric-based anisotropic methods, introduced for three-dimensional meshes in [9], are most commonly used. These methods on triangle / tetrahedral cells have the attractive ability to align closely with significant flow features such as shocks and fluid interfaces, which significantly reduces computational cost while maintaining accuracy within the bulk of the domain [10] [11]. But since metric-based methods are generally tied to specific mesh cell types or grid hierarchies [12] they do not leverage the full flexibility of polyhedral meshes that are widely used for finite volume discretizations. Furthermore, these methods and are generally unable to recover the original mesh. And simplicial meshes also come with the drawback of excessively diffusive solutions (when discretized with a second-order finite-volume scheme).

Due to the increased popularity of polyhedral meshes and hybrid meshes that combine polyhedral boundary layer elements with size-field based hexahedra meshes away from the boundaries [13], it becomes imperative to accommodate these types of meshes for adaptive mesh refinement when possible. This was the impetus behind the PUMA algorithm [14] that was devised for Ansys Fluent and is widely used for that purpose today. The PUMA method can be regarded as a generalization of the hexahedral refinement template for arbitrary polyhedra and can therefore accommodate all traditional cell types such as tetrahedra, hexahedra, prisms and pyramids as well.

2. METHODOLOGY

Prior to the description for anisotropic mesh refinement, it is necessary to define terms used for isotropic refinement, since they become significant during the discussion of transitions from refined to un-refined areas of the mesh and the compatibility between isotropic and anisotropic mesh adaptation. Moreover, the definitions in this section are generalized for anisotropy at a later stage.

2.1 Isotropic PUMA Terminology

The concept of a "refinement level" is introduced at mesh elements (such as nodes, faces and cells), which is a numerical value denoting the hierarchy of refinement levels (see Figure 1). It is initialized to zero for an unrefined mesh and is incremented by one for each additional level of refinement. In general, the mesh adaptation is constrained such that adjacent cells do not differ by more than one level of refinement. The isotropic refinement algorithm begins with the faces of the original polyhedral cell, referred to in this context as "parent" faces or cells, which are subsequently split into "child" faces and cells after refinement. For each face of the polyhedral cell, a mid-node is introduced. The coordinate of this mid-node is typically at the face centroid, but this can be adjusted based on other conditions such as mesh quality. The quality metrics used for polyhedral cells is based on the alignment of the vector pointing from one adjacent cell to the other and the face normal as in [5]. Avoiding large angles between these vectors is crucial to obtain acceptable results for a finite-volume discretization.

For each edge of the face under consideration, a new mid-node is introduced, typically at the edge centroid. For anisotropic refinement within boundary layers, it is sometimes convenient to choose a mid-edge location closer to one of the nodes. This fraction of the length to the splitting point over the original edge-length is defined as the split-ratio. For the edge centroid, the split-ratio would be 0.5. The refinement level of the

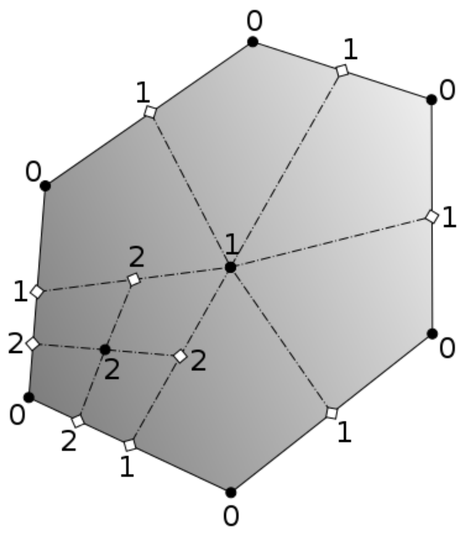


Figure 1: Isotropic refinement levels at nodes

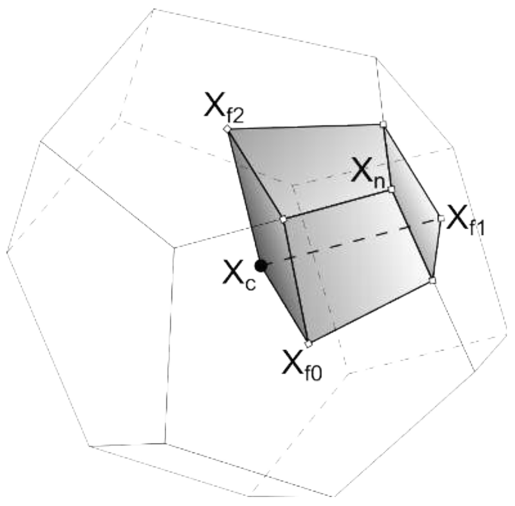


Figure 2: Subtending a child to cell mid-point

new mid-edge node is designated as an increment to the current face refinement level. For each node in the parent face, a new quadrilateral child face is created by connecting the mid-edge nodes along with the mid-face node and the original face node. A node is introduced at an appropriate location within the polyhedral cell. The cell centroid is a convenient choice, but this can be altered based on the requirements of mesh quality after refinement. For each node in the parent cell, a new child cell is created by connecting the mid-edge nodes along with the mid-face nodes ($X_{f0,1,2}$), mid-cell node (X_c) and the original cell node (X_n) (see Figure 2).

It is worthwhile to note that the isotropic refinement of arbitrary polyhedra typically results in a mesh that is largely hexahedral (see Figure 3), and subsequent refinements can be optimized to deal with hexahedral cells. At the transition between refined and non-refined cells, the connectivity of the non-refined cells must then be updated to account for additional nodes and faces arising from refinement. This can be done quite efficiently by tagging adjacent cells as they are being refined, and then processing tagged cells for updates after the refinement step is complete.

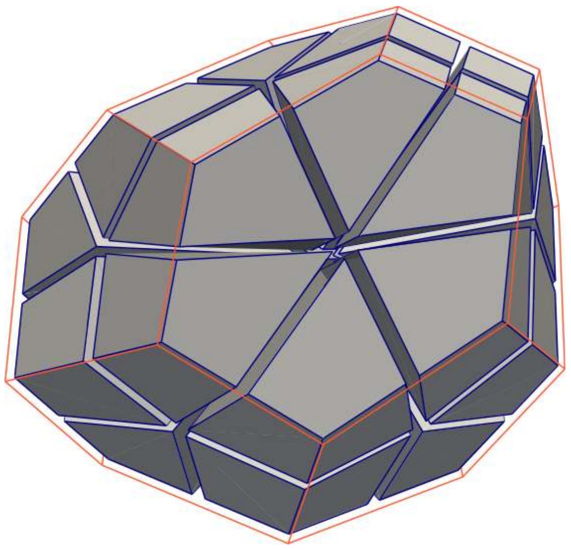


Figure 3: Exploded view of a refined polyhedral cell

2.2 Anisotropic PUMA Terminology

The case for anisotropic PUMA is a specialization of the isotropic method for prismatic polyhedral cells. These cells are typically encountered at the boundary layers of viscous polyhedral meshes and consist of polygonal lower / upper faces with an equal number of nodes that are connected by quadrilateral faces on the side. Once these cells have been identified within the original polyhedral mesh, it is now possible to define two modes of anisotropic mesh adaptivity, namely, tangent (Figure 4) and normal (Figure 5) refinement.

It is immediately apparent that both modes of anisotropic refinement split a prismatic polyhedral cell in a specific direction while avoiding the other. This allows mesh refinement to be guided towards flow features that have a directional bias, while avoiding increased mesh resolution in directions that do not require it. Tangent refinement is well suited for turbulent flows with specific y^+ requirements, because it increases mesh resolution in the wall-normal direction, while keeping the span-wise resolution intact. Normal refinement is typically used to reduce the aspect ratio of prismatic cells in a boundary layer. An example for which high aspect ratio prisms can be challenging are overlapping overset meshes [15] where local normal refinement can reduce cell size jumps between meshes resulting in a robust mesh intersection without orphan cells [16].

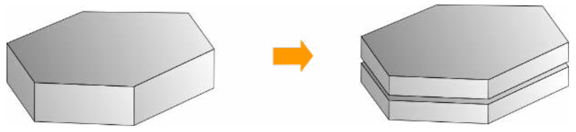


Figure 4: Tangent refinement

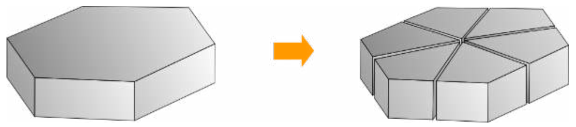


Figure 5: Normal refinement

Both modes can also be applied in a sequential manner to a prismatic parent cell in order to achieve isotropic refinement within the boundary layer, as shown in Figure 6. In the upper transition, normal refinement is applied to the parent cell, followed by tangent refinement on each of the child cells. For the lower transition, tangent refinement is first applied, followed by normal refinement on the two child cells. Both transition modes yield the same isotropic result.

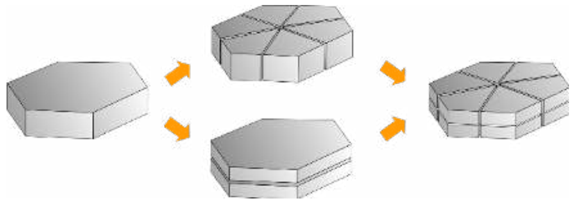


Figure 6: Sequential application of anisotropic modes for isotropic refinement

The first step while performing tangent or normal refinement is to identity the prismatic cells in a given mesh. Using the cell type makes this trivial for wedge elements, but hexahedral and polyhedral prisms require additional flagging of top and bottom faces to define the normal or tangent direction. This is typically done by visiting the boundary faces of the mesh and checking whether adjacent cells are prismatic i.e., they possess unique top and bottom faces with an equal number of nodes and they have quadrilateral side faces. Thereafter, each subsequent layer of prismatic cells is detected and flagged by a face-cell walk through top and bottom faces discovered in the previous sweep, until cells are no longer prismatic. In situations involving hexahedral cells adjacent to multiple mesh boundaries, the top and bottom faces are no longer unique and therefore such cells are ineligible for anisotropic refinement.

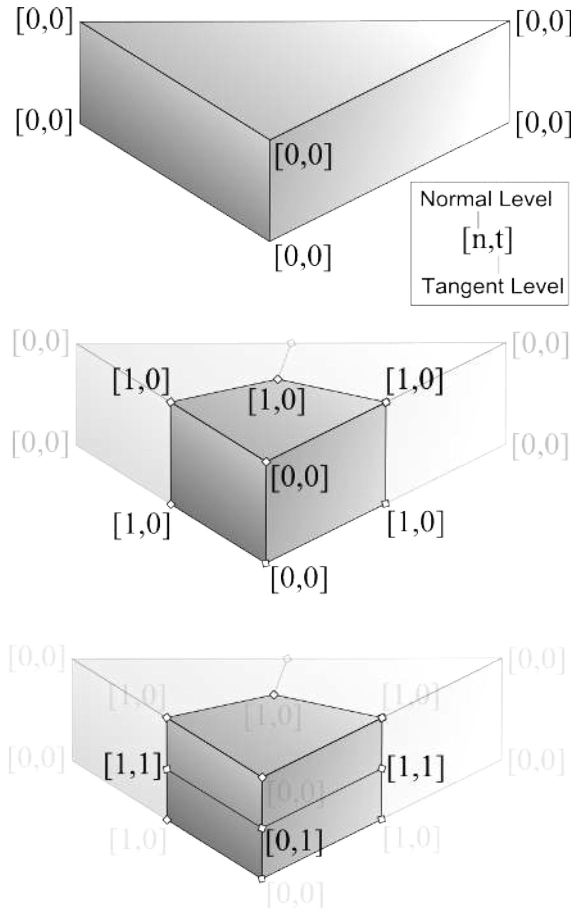


Figure 7: Separation of anisotropic refinement levels

While a scalar refinement level is sufficient to describe isotropic refinement levels, it is required to distinguish between levels for each refinement mode in the anisotropic situation. Every node, face and cell in the prismatic boundary layer is now described with a normal and tangent refinement level, with the isotropic refinement level set to the maximum of both. Taking the example of tangent refinement, both child cells will possess a tangent refinement level that is one level higher than the parent, while the normal refinement level remains the same. The converse of this situation occurs for normal refinement. It follows naturally that the mid-point of a face matches the isotropic refinement level, and this characteristic can be used to seamlessly transition between isotropic and anisotropic regions of the mesh for refinement.

This scheme is depicted in Figure 7 for anisotropic refinement of a wedge cell, but it can be extended to any arbitrary prismatic polyhedral shape. The first image shows the original wedge cell with both normal and tangent refinement levels at nodes initialized to zero. The second image shows one level of normal refinement into three prismatic child cells, where the mid-nodes possess a normal refinement level of one, but the tangent refinement levels remain at zero. The final image shows the tangent refinement of one of the child cells, where the tangent refinement level is incremented by one, while the normal levels remain unchanged.

Tangent refinement is typically the preferred mode in the boundary layer, as it aligns with the flow direction and captures viscous effects quite well. However, a common scenario is the introduction of isotropic refinement via the cell capping the prism layer. In this situation, it is important to refine the entire prism stack with normal refinement to maintain the one-level balance constraint and to avoid a local degradation of the cell quality as introduced above. Naturally, when cells in the boundary layer are already marked for tangent refinement, the introduction of normal refinement results in those cells being refined isotropically. This is depicted in Figure 8.

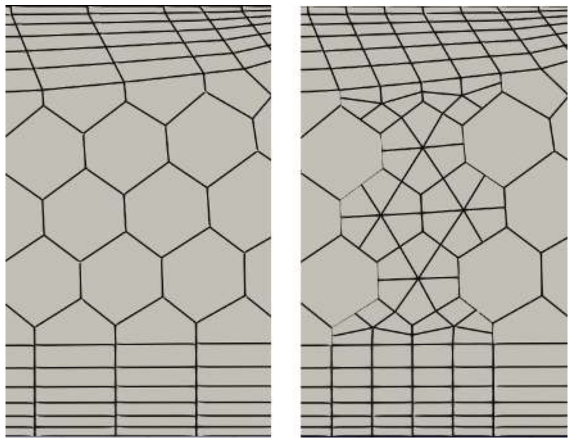
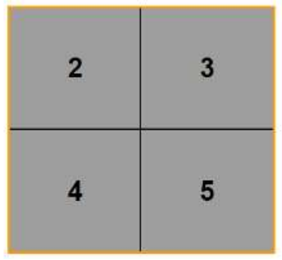


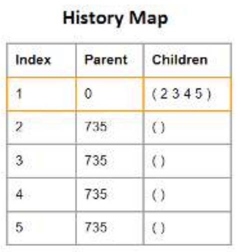
Figure 8: Normal refinement through the stack

2.3 Coarsening with PUMA

Coarsening is the process of reverting changes due to refinement in order to recover the original mesh. This typically requires the maintenance of some form of refinement history, which describes the relationship between parent and child entities. One possible approach is to retain all parent entities after refinement and reinstate them during coarsening after discarding their children. However, this leads to a significant increase in memory usage and is only applicable to the hanging-node style of adaptive mesh refinement [8][17], where the cell type typically does not change. The coarsening step is also constrained such that the one-level balance between adjacent cells is maintained. The PUMA approach for maintaining refinement history described in this section is very lean in terms of memory usage and is applicable to both isotropic and anisotropic coarsening.

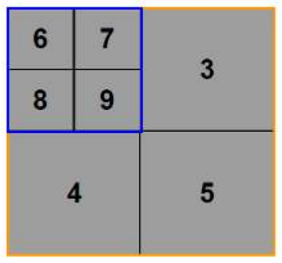
The first step is to recognize that parent faces and cells are not stored during the refinement process and must therefore be recovered by agglomerating children. This requires the identification of all children that belong to a given parent. A lean way to achieve this is by defining a unique "parent index" that will be assigned to each child face or cell of the parent that is refined. This can be any arbitrary integer, with the only constraint that all children of a parent must possess the same unique value.

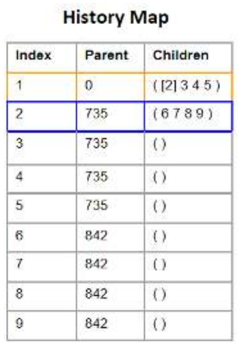



History Map

Index	Parent	Children
1	0	(2 3 4 5)
2	735	( )
3	735	( )
4	735	( )
5	735	( )

Figure 9: Refinement history for first level




History Map

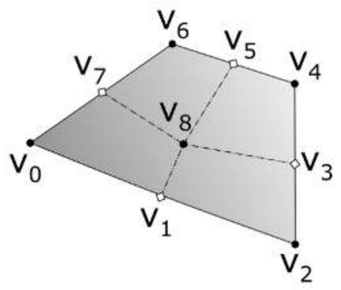
Index	Parent	Children
1	0	([2] 3 4 5)
2	735	(6 7 8 9)
3	735	( )
4	735	( )
5	735	( )
6	842	( )
7	842	( )
8	842	( )
9	842	( )

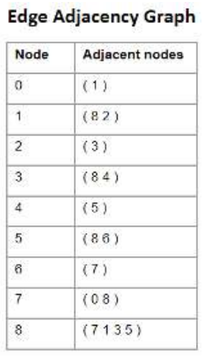
Figure 10: Refinement history for second level

This is depicted in Figure 9 for one level of refinement of an element (face or cell) into 4 children. During the refinement step, a unique parent index is generated (in this example, the integer 735) and assigned to each child. It is assumed that elements with a parent index of zero denote the coarsest level of the mesh. For a second level of refinement in this example, element 2 is refined into 4 children, while other elements are left unrefined. In this case, another unique index is generated (integer 842 in this example) and assigned to each child, as shown in Figure 10.

At this point, however, element 2 no longer exists in the mesh (since parent elements are not stored) and therefore, a separate "history map" is introduced with a single entry which maps 842 to 735. During the coarsening stage for all elements that share index 842, the parent is first created by agglomerating all children, followed by a lookup in the history map to determine the parent index for the new element (namely, index 735). This approach can be repeated for each additional level of refinement, and the only storage cost incurred for each new element is a single integer along with a history map entry for each parent.

Once all children of a parent have been determined, the actual process of coarsening first involves the detection of common interior entities (i.e., interior edges for child faces and interior faces for child cells) which are subsequently marked for removal from the mesh. For groups of child cells, discovering common faces among them is a trivial step. Once these interior faces have been identified, they are removed from the mesh, leaving the bounding child faces on the parent cell. The next step is to identify whether child faces sharing the same parent index point to the same parent cells on each side (or just one side for boundary faces), which indicates that these child faces are candidates for coarsening into a parent face. Child faces that point to different cells are left in the refined state.




Edge Adjacency Graph

Node	Adjacent nodes
0	(1)
1	(8 2)
2	(3)
3	(8 4)
4	(5)
5	(8 6)
6	(7)
7	(0 8)
8	(7 1 3 5)

Figure 11: Edge adjacency while coarsening faces

While coarsening child faces, there is the additional constraint that resulting edges must form a counterclockwise chain around the parent face centroid / normal, according to the right-hand rule. This can be achieved by adding directed edges of all child faces into an edge adjacency graph, as shown in Figure 11. The next step is to loop through each node, check its adjacency list and remove duplicate edges for each node on that list. For example, while testing node 0, the only entry in its adjacency list is node 1, but node 1 only has nodes 8 and 2 on its list but not node 0 and so, the adjacency list is left unmodified. While checking node 1, it is seen that node 8 is on its adjacency list, and node 8 also has node 1 on its adjacency list, indicating that it is a duplicate interior edge that can be removed. This process is repeated for subsequent nodes and a single pass through the list is sufficient to remove duplicates. The final graph contains a list of nodes with a single entry in each adjacency list, indicating the next node in the chain. At this point, constructing the parent face is as simple as picking a node and following its adjacent node, adding each one to a list until the first node is reached, thereby completing the chain.

The size associated with this history storage approach for a sample polyhedral mesh that has been uniformly refined several times is shown in Table 1. It can be noted that the memory consumption is a single integer for each face and cell (for the parent index) and a tuple of integers for each entry in the face and cell history maps. The history maps are only required after the first level of refinement. For most simulations, the default of two levels of refinement is sufficient as it provides a significantly improved spatial resolution while constraining the total cell count over time for efficiency.

Level	Mesh Size		History Map Size (Tuple Count)	
	Face	Cell	Face	Cell
Initial	660	119	0	0
1	5,944	1,851	0	0
2	46,088	14,858	3,155	1,851
3	362,848	118,964	26,931	16,709
4	2,879,360	951,912	211,283	135,673

Table 1: History storage cost for multiple levels of uniform refinement on a sample polyhedral mesh

2.4 Distributed Parallel PUMA

Maintaining the scalability of Ansys Fluent [18], a distributed parallel flow solver, was a requirement for the PUMA algorithm. This was achieved by embedding load-balancing and migration into the mesh adaptation algorithm and by avoiding any constraints on the distribution of cells across partitions. The latter can easily be achieved for cell refinement which is entirely local to a partition and can proceed normally. However, cell coarsening can span several child cells distributed across multiple partitions, as shown in Figure 12, which typically occurs as a result of a load-balancing step that maintains an equal distribution of cells.

A convenient choice is to encapsulate all children of a parent cell on the same partition, which effectively restricts all partitioning methods to use the coarsest level of the mesh while distributing cells. While this simplifies coarsening behavior, it can significantly affect flow solver performance, particularly for simulations that involve higher refinement levels.

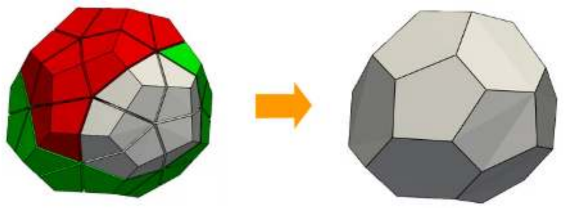


Figure 12: Coarsening of cells distributed across partitions

The chosen approach is to accept non-encapsulated cells in the coarsening algorithm by extending the parallel communication layer to include an additional layer of node-connected cells. The resulting parent cell after coarsening is assigned to one of the partitions after discarding all children. This removes all restrictions for mesh partitioning and significantly improves solver scalability, with the minor cost of ensuring that all parent indices are unique across all partitions, which is typically achieved with a few global communication calls during the mesh manipulation step.

3. EXAMPLES

This section will demonstrate the use of the isotropic and anisotropic adaptation algorithms described in Section 2 with a few examples.

3.1 Refinement of a Tetrahedral Mesh with Boundary Layers

The first example depicts the refinement of a mixed tetrahedral mesh with layers extruded from one half of the bottom boundary that contains quadrilateral faces as shown in Figure 13. The first layer of hexahedral cells adjacent to the bottom boundary is subsequently refined in the tangent direction with a split ratio of 0.3, while two tetrahedral cells adjacent to the boundary layers are refined isotropically.

The isotropic refinement of the tetrahedral cell at the top of the boundary layer forces normal refinement through the stack. The split ratio of 0.3 is also respected throughout the layer, except at the transition to the isotropic tetrahedral region at the side, where a split ratio of 0.5 is maintained. This refinement scheme also shows the applicability of the method to situations involving stair-stepping within the bound-

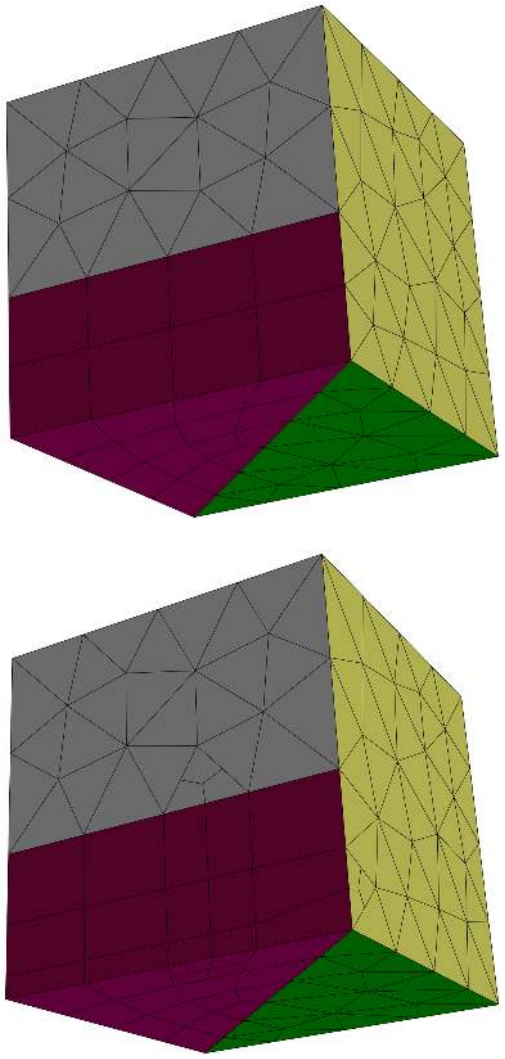


Figure 13: Refinement example of a tetrahedral mesh with boundary layers

ary layer, where a non-uniform number of layers may be present adjacent to any mesh boundary.

3.2 Isotropic PUMA for the Dam Break Problem

This example is a computational fluid dynamics case depicting the dam break problem with an obstacle placed within the domain, where the gas-liquid interface is modeled using a Volume-of-Fluid approach, along with adaptive time-stepping. The fluid is allowed to evolve over time under the influence of gravity. Various stages of the simulation are shown in Figure 14.

The mesh has an initial count of 111276 cells and is adaptively refined and coarsened at every other time-

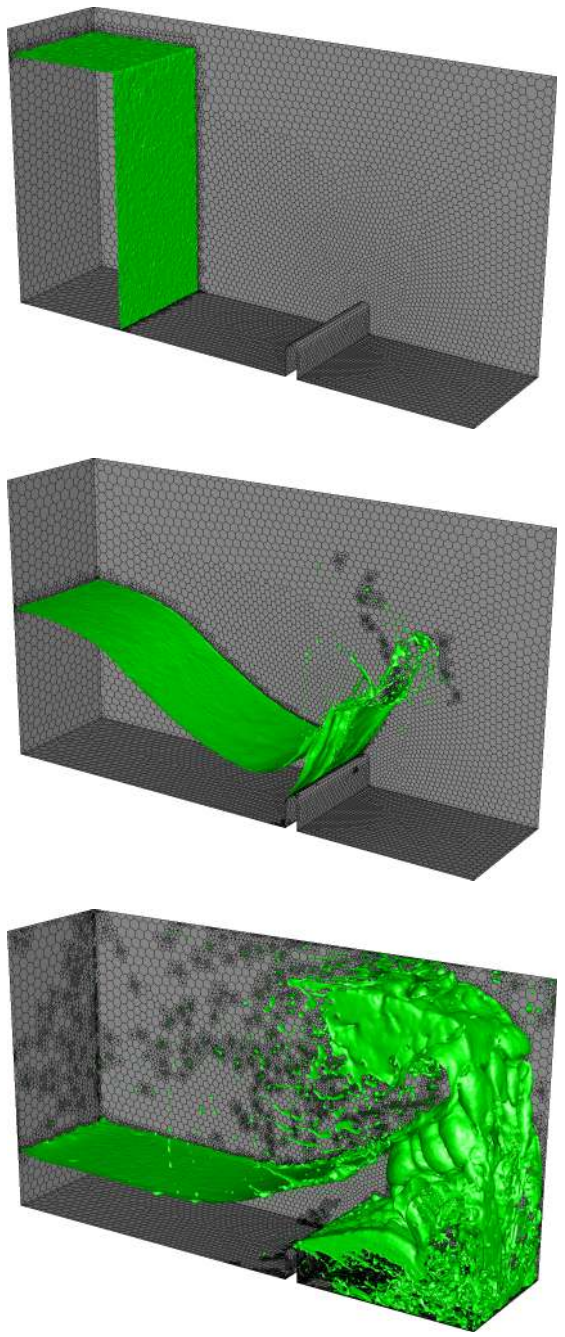


Figure 14: Evolution of dam break problem with adaptive mesh refinement at time-steps: 0, 5000, 15000

step with a maximum of two refinement levels imposed throughout the course of the simulation. The criteria for refinement and coarsening are defined by the normalized gradient of the gas-liquid volume-fraction. Any cell is refined if the magnitude of this gradient is larger than a specified threshold value and coarsened if the gradient falls below a second (lower) threshold value. Two refinement levels applied globally correspond to a mesh with a cell count of about 18.2 million. The dam break results match the fidelity obtained on a mesh of this size with a significantly lower cell count and computational cost. The mesh is automatically load-balanced during an adaptation step when the difference between maximum and minimum cell count per core exceeds 5% of the total cell count. The total wall-clock time of the flow-solver and mesh adaptation for 500 time-steps are shown in Figure 15 for various core counts.

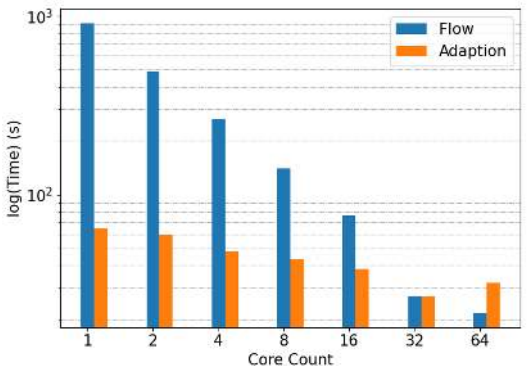


Figure 15: Comparison of flow solver and mesh adaptation at various core counts

In this case, the count after 500 time-steps is roughly 4000 cells per core on 64 cores and far from ideal. Nevertheless, the adaptive mesh refinement step along with load-balancing, maintains the scalability of the flow solver at these low cell counts per core as shown by the blue bars. Since any form of mesh manipulation comes with a certain fixed cost related to migration, garbage collection and establishing a parallel communication layer, irrespective of the number of cells involved in the operation, the performance of the mesh adaptation shown in the orange bars only improves up to 32 cores where its cost is comparable to the flow-solve. It should also be noted that most simulations can proceed with a lower frequency of an adaptation every 5 or 10 time-steps, as opposed to 2 in this case. The relative cost of each of these operations over 15000 time-steps for this simulation using two different configurations of computational cores and adaptation frequencies is shown in Table 2.

The flow solver dominates the simulation time, as anticipated, with the adaptation and load balancing steps consuming a relatively small fraction. The preparation phase involves the estimation of cell quality after refinement / coarsening. The cleanup phase encompasses steps related to garbage collection, solver

Operation	16 cores frequency = 2		24 cores frequency = 10	
	Time (s)	%	Time (s)	%
Flow	111989	75.8	47709	87.9
Adaptation	35050	23.7	5799	10.7
- Prepare	14846	10.1	2038	3.75
- Refinement	3233	2.18	687	1.26
- Coarsening	2789	1.89	460	0.85
- Cleanup	12680	8.58	2304	4.24
Balance	680	0.46	771	1.42
Total	147719	100	54279	100

Table 2: Relative cost of individual operations on 16 cores (frequency 2) and 24 cores (frequency 10)

array compaction and parallel communication layer setup. These phases of the adaptation process consume the bulk of the computation involved, while the actual refinement / coarsening steps are relatively cheap. The load balancing step is an expensive step but it is called infrequently and so, it consumes a very small percentage of the overall simulation time.

3.3 Anisotropic PUMA for Fuselage, Wing Configuration

The next example is an external aerodynamics simulation of a hypothetical aircraft with a wing and fuselage. The initial mesh consists of 340223 cells and a single boundary layer defined throughout the body of the aircraft as shown in Figure 16. The inlet flow is defined as Mach 0.6 with a gauge pressure of 35606Pa and assumed to be steady state. The pressure-based flow solver is used with SIMPLE for pressure-velocity coupling and the $k-\omega$ SST turbulence model.

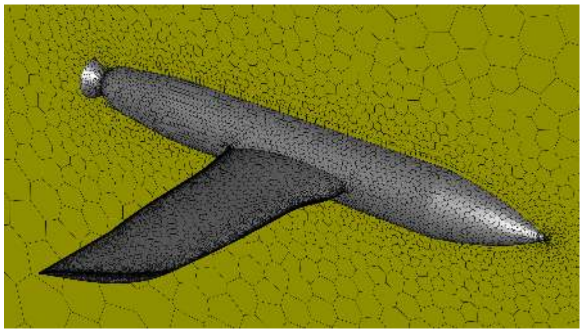


Figure 16: Initial mesh of aircraft

Eight successive levels of anisotropic tangent refinement are applied on all surfaces and the minimum / maximum y^+ values are computed at each step to determine whether sufficient mesh resolution is achieved to compute a reasonably accurate solution within the boundary layer (see Table 3), demonstrating the desired $y^+ \approx 1$ being achieved with the addition of only 201809 cells. Achieving the same y^+ goal with isotropic refinement would result in a significantly higher increase in the number of cells.

	Pressure [Pa]		y^+		Cell Count
	Min	Max	Min	Max	
0	12473.3	45153.3	0.52	240.8	340,223
1	13110.6	45310.3	0.34	134.6	382,556
2	15432.4	45481.3	0.12	73.8	424,479
3	17580.6	46318.6	0.03	37.9	464,816
4	17149.7	46927.4	0.06	18.5	495,199
5	17819.9	46948.7	0.06	9.4	517,993
6	17783.7	46888.0	0.07	4.6	532,775
7	17775.8	46887.0	0.09	2.3	538,871
8	17756.6	46887.5	0.08	1.1	542,032

Table 3: Comparison of min / max y^+ vs. cell count

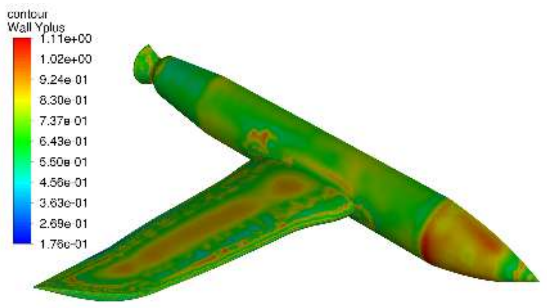


Figure 17: Contour plot of y^+ distribution across aircraft body

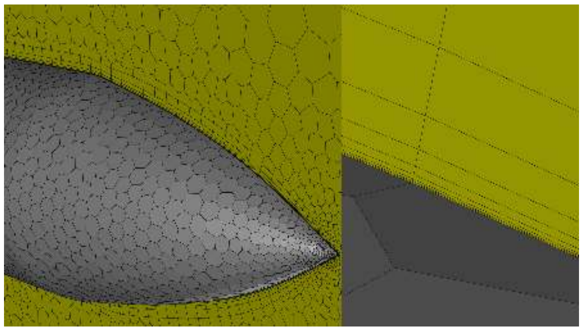


Figure 18: Tangent refinement detail at Level 8

A contour plot of the y^+ distribution across the surface of the wing / fuselage after 8 levels of refinement is shown in Figure 17. A uniform splitting ratio of 0.5 is used in this case, and the distribution is largely dictated by the single boundary layer on the initial mesh, but it is also possible to locally adjust the refinement ratio account for a variable layer height at each cell. Details of tangent refinements at Level 8 near the front of the aircraft are shown in Figure 18.

3.4 Combined Isotropic and Anisotropic PUMA for Space Capsule Re-entry

The final example is the simulation of a space capsule under hypersonic re-entry conditions with an angle-of-attack of -25^o. The trajectory, velocity and ambient fluid conditions represent the vehicle passing through the earth's atmosphere at an altitude of 50km. The initial mesh consists of 104581 polyhedral cells including 15 prismatic polyhedral boundary layers defined around the body of the capsule as shown in Figure 19. The inlet flow is defined as Mach 17 with a gauge pressure of 25Pa and assumed to be steady state. The fluid is modeled as an ideal gas using the two-temperature model to account for compressibility and thermophysical variations. The steady-state density-based flow solver is used along with the high-speed numerics. Turbulence is modelled with the $k - \omega$ turbulence model.

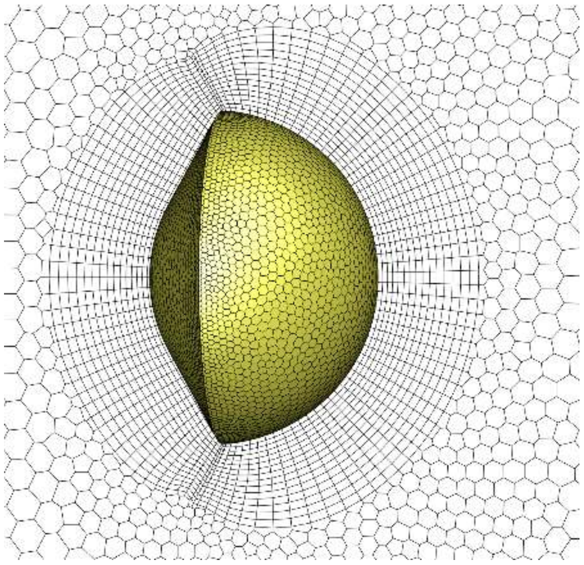


Figure 19: Initial mesh of the space capsule

The simulation is initially run for 500 iterations, after which the mesh is adapted periodically every 250 iterations. The error-based Hessian criterion [19] is used to identify cells in the domain for refinement and coarsening, along with anisotropic refinement in the boundary layers as needed. Snapshots of the adaptively refined mesh after 750 and 1500 iterations are shown in Figure 20 and Figure 21 respectively. The effect of tangent refinement in the prismatic bound-

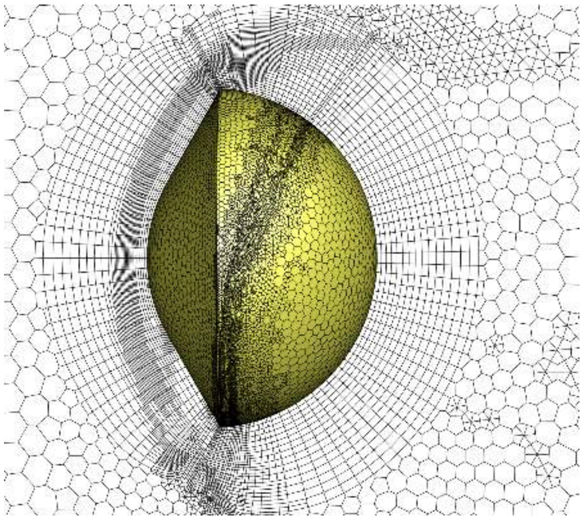


Figure 20: Refined mesh after 750 iterations

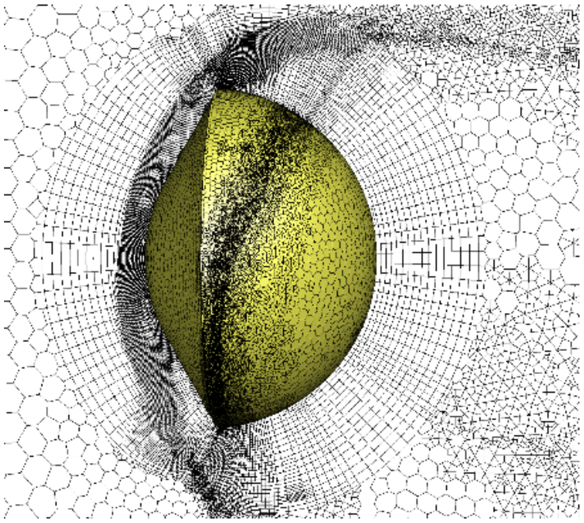


Figure 21: Refined mesh after 1500 iterations

ary layers is immediately apparent. Additionally, normal refinement through the prism stack occurs due to isotropic refinement at the transition from boundary layers to regular polyhedral cells, resulting in the surface refinement of the capsule. A contour plot for the Mach number is shown in Figure 22, showing details of the bow-shock captured by the local anisotropic refinement in the boundary layer.

The simulation was repeated using identical parameters without anisotropic boundary layer refinement, and the comparison of cell counts at each adaptation cycle is shown in Table 4. To capture the details of the shock region, a significant amount of cells in the boundary layer are marked for refinement, and the cost savings of directional anisotropic refinement is immediately apparent.

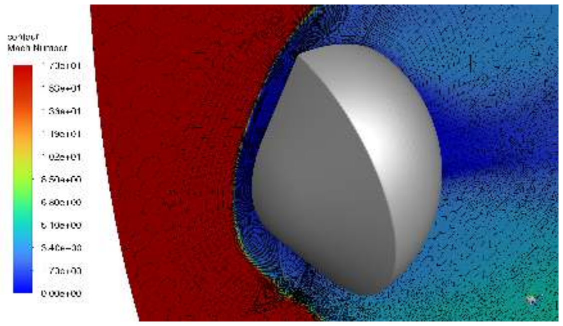


Figure 22: Contour plot for Mach number

Iteration	Cell Count	
	Isotropic	Anisotropic
Initial	104,581	104,581
500	447,834	350,392
750	1,385,658	857,307
1000	4,478,563	2,227,375
1250	6,964,051	3,649,348
1500	8,963,725	5,335,096

Table 4: Comparison of cell count at each adaptation cycle

4. CONCLUSIONS

This paper demonstrates a new procedure to adaptively refine arbitrary polyhedral meshes, including the anisotropic refinement and coarsening of prismatic polyhedral cells in boundary layers. The refinement scheme defines a conformal template that seamlessly transitions between isotropic and anisotropic regions of the mesh. The implementation is designed for a distributed parallel environment where it maintains the scalability of the flow solver via load-balancing. The applicability of this adaptive refinement scheme is demonstrated using several computational fluid dynamics tests that involve polyhedral meshes, with the conclusion that reliably accurate solutions can be achieved with a modest increase in calculation cost. The introduced mesh adaptation method can be used with any criterion that provides information where refinement and coarsening take place. Heuristic criteria, error indicators or estimators can all be applied without the need for the respective criterion to provide the direction of anisotropic refinement.

References

[1] *Simcenter STAR-CCM+*. Siemens Industries Digital Software, 2022

[2] *Ansys Fluent*. Ansys Inc, 2022

[3] Weller H.G., Tabor G.R., Jasak H., Fureby C. "A tensorial approach to computational continuum mechanics using object-oriented techniques." *Computers in Physics*, vol. 12, 620–631, 1998

[4] Hirsch C. *Numerical Computation of Internal and External Flows (Second Edition)*. Butterworth-Heinemann, Oxford, 2007

[5] Peric M. "Flow simulation using control volumes of arbitrary polyhedral shape." *ERCOFTAC Bulletin*, vol. 62, 25–29, 2004

[6] Spiegel M., Redel T., Zhang J., Struffert T., Hornegger J., Grossman R.G., Doerfler A., Karmonik" C. "Tetrahedral vs. polyhedral mesh size evaluation on flow velocity and wall shear stress for cerebral hemodynamic simulation." *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 14, no. 1, 9–22, 2011

[7] Berger M.J., Oliger J. "Adaptive mesh refinement for hyperbolic partial differential equations." *Journal of Computational Physics*, vol. 53, no. 3, 484–512, 1984

[8] Rivara M.C. "Algorithms for refining triangular grids suitable for adaptive and multigrid techniques." *International Journal for Numerical Methods in Engineering*, vol. 20, no. 4, 745–756, 1984

[9] Tam A., Ait-Ali-Yahia D., Robichaud M., Moore M., Kozel V., Habashi W. "Anisotropic mesh adaptation for 3D flows on structured and unstructured grids." *Comput. Methods Appl. Mech. Engrg.*, vol. 189, 1205–1230, 2000

[10] Alauzet F., Loseille A. "A Decade of Progress on Anisotropic Mesh Adaptation for Computational Fluid Dynamics." *Computer-Aided Design*, vol. 72, 13–39, 2016

[11] Davies D.R., Wilson C.R., Kramer S.C. "Fluidity: A fully unstructured anisotropic adaptive mesh computational modeling framework for geodynamics." *Geochemistry, Geophysics, Geosystems*, vol. 12, no. 6, 2011

[12] Freret L., Williamschen M., Groth C.P.T. "Enhanced anisotropic block-based adaptive mesh refinement for three-dimensional inviscid and viscous compressible flows." *Journal of Computational Physics*, vol. 458, 2022

[13] Zore K., Sasanapuri B., Parkhi G., Varghese A.J. "Ansys MOSAIC Poly-Hexcore mesh for high-lift aircraft configuration." *21st Annual CFD Symposium Conference*. 2019

[14] Menon S., Gessner T. "PUMA (Polyhedra Unstructured Mesh Adaption): A Novel Method to Refine and Coarsen Convex Polyhedra." *14th U.S. National Congress on Computational Mechanics, Montreal, Canada. July 17-20*, 2017

[15] Meakin R.L. *Composite Overset Structured Grids, Chapter 11. Handbook of Grid Generation*. CRC Press, 1999

[16] Parks S., Buning P., Chan W., Steger J. "Collar grids for intersecting geometric components within the Chimera overlapped grid scheme." *10th Computational Fluid Dynamics Conference*. 1991

[17] Verfürth R. "A posteriori error estimation and adaptive mesh-refinement techniques." *Journal of Computational and Applied Mathematics*, vol. 50, no. 1, 67–83, 1994

[18] Wasserman S. "Ansys Fluent sets record with 129,000 cores." `http://engineering.com/story/ansys-fluent-sets-record-with-129000-cores`, 2015

[19] Norman A., Viti V., MacLean K., Chitta V. "Improved CFD methodology for compressible and hypersonic flows using a Hessian-based adaption criteria." *AIAA SCITECH 2022 Forum*. 2022