Hybrid Grid Generation for Viscous Flow Simulations in Complex Geometries

Hongfei Ye\textsuperscript{a,b}, Yang Liu\textsuperscript{c}, Bo Chen\textsuperscript{c*}, Zhiwei Liu\textsuperscript{a,b}, Jianjing Zheng\textsuperscript{a,b}, Yufei Pang\textsuperscript{c}, Jianjun Chen\textsuperscript{a,b,d†}

\textsuperscript{a} Center for Engineering and Scientific Computation, Zhejiang University, Hangzhou 310027, China
\textsuperscript{b} School of Aeronautics and Astronautics, Zhejiang University, Hangzhou 310027, China
\textsuperscript{c} China Aerodynamics Research and Development Center, Mianyang, 621000, China.
\textsuperscript{d} State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310027, China

Abstract: In this paper, we present a hybrid grid generation approach for viscous flow simulations by marching a surface triangulation on viscous walls along certain directions. Focuses are on the computing strategies used to determine the marching directions and distances since these strategies determine the quality of the resulting elements and the reliability of the meshing procedure to a large extent. With respect to marching direction, three strategies featured with different levels of efficiencies and robustness performance are combined to compute the initial normals at front nodes to balance the trade-off between efficiency and robustness. A novel weighted strategy is used in the normal smoothing scheme, which evidently reduce the possibility of early stop of front generation at complex corners. With respect to marching distances, the distance settings at concave and/or convex corners are locally adjusted to smooth the front shape at first; a further adjustment is then conducted for front nodes in the neighbourhood of gaps between opposite viscous boundaries. These efforts, plus other special treatments such as multi-normal generation and fast detection of local/global intersection, as a whole enable the setup of a hybrid mesher that could generate qualitied viscous grids for geometries with industry-level complexities.

Key words: mesh generation; hybrid mesh; mesh quality; viscous flow

1 Introduction

For RANS computations involving complex geometries, a challenging task is the generation of high-quality RANS meshes. Among the different mesh types, prismatic hybrid meshes are preferred in many applications because they represent a good compromise between solution accuracy and ease of use [1-4]. In a prismatic hybrid mesh, the near field of viscous walls (referred to as a boundary layer hereafter) is configured with layered prismatic elements to resolve high flow gradients normal to the walls, whereas the remaining domain is usually filled with an unstructured mesh. Thus, the generation of a hybrid prismatic mesh usually consists of two individual meshing steps: boundary layer mesh generation and unstructured mesh generation.

* Corresponding author. E-mail: chenbo01010401@163.com
† Corresponding author. E-mail: chenjj@zju.edu.cn
In general, the generation of boundary layer elements starts from the surface triangulation on viscous walls. The initial front is defined on this triangulation, and a marching direction is computed at each mesh point of the front. Each front point is then propagated to a new position by adding a step value along the marching directions. As a result, a layer of prismatic elements can be formed by connecting all the front points with their new neighbours. The entire boundary layer mesh could then be created by repeating the above procedure a few times [2-7]. With respect to the generation of unstructured meshes, either the advancing front technique (AFT) based approach [8] or the Delaunay triangulation (DT) based approach [9-11] could be adopted. In some studies, it has been suggested to first fill an axis-aligned Cartesian mesh in the far field of viscous walls and then connect the boundary layer mesh and the Cartesian mesh with a few transition layers of unstructured elements [12-15].

Compared with the now mature unstructured mesh generation and Cartesian mesh generation, the generation of boundary layer meshes still gives rise to numerous difficulties and is therefore the main challenge in generating the entire prismatic hybrid mesh. Among those difficulties, those induced by the computations of marching directions and marching distances should be highlighted because both computations determine the quality of the resulting elements and the reliability of the meshing procedure to a large extent. A large portion of the efforts involved in the development of a prismatic hybrid mesher were invested in tackling these issues [17-23].

In principle, a practically useful hybrid mesh generation scheme should take the quality of the resulting elements and the reliability of the meshing procedure as the primary consideration. Following this principle, we proposed several novel computing strategies for marching directions and marching distances. With respect to the computation of marching directions, three strategies featured with different levels of efficiencies and robustness performance are combined to compute the initial normals at front nodes to balance the trade-off between efficiency and robustness. After that, a revised smoothing scheme is proposed for these normals to avoid the abrupt changes on lengths of neighbouring front lines. As a result, this smoothing could evidently reduce the possibility of early stop of front generation at complex corners. With respect to the computation of marching distances, the initial marching distances are computed by the user-specified parameters, followed by a two-step adjustment: the distance settings at concave and/or convex corners are locally adjusted to smooth the front shape at first; a further adjustment is then conducted for front nodes in the neighbourhood of gaps between opposite viscous boundaries. To support the second-step adjustment, an improved ray-casting algorithm is developed for the automatic identification of the gaps. In the meantime, the cost of this computation is reduced to a very low level at the aid of a background mesh.

In addition to the above novel strategies, other special treatments are developed to improve the robustness and efficiency of the hybrid meshing procedure, multi-normal generation, fast detection of intersections between front faces, remove of non-manifold fronts, to name a few. These efforts, as a whole, enable the setup of a hybrid mesher that could generate qualified viscous grids for geometries with industry-level complexities. Numerical experiments are conducted including comparison with results by state-of-the-art commercial tools to verify the effectiveness and efficiency of the mesher.

The remainder of this paper is organized as follows. Section 2 reviews the existing computing strategies of marching directions and distances briefly. Section 3 presents an outline of the hybrid meshing method. Section 4 introduces important implementation details involved in the
hybrid meshing method. Section 5 presents various numerical experiments. Section 6 concludes
with the outcomes of the study and points to some directions for future studies.

2 Literature review

2.1 On computation of marching directions

Presently, the most prevailing approaches for computing marching directions are those based
on the analysis of the manifold of a point [16-21]. The manifold of a point here refers to the set
of front faces adjacent to the point, and these front faces are thus named manifold faces of that
point. Intuitively, the marching direction at a point could be obtained by computing a weighted
average of the normal vectors of its front faces. However, this intuitive computation strategy
cannot ensure the resulting marching direction is always visible to all the manifold faces. As a
remedy, Kallinderis and Ward [16] presented the visibility cone concept, which refers to a subset
of the space depicted by the manifold of the point. To ensure the mesh validity, the computed
marching direction must be located within the visibility cone. Based on the visibility cone
concept, Aubry and Löhner [17, 18] recast the problem of computing a marching direction into
an optimization problem. The solution of that problem could result in the `best’ marching
direction at a point by providing an optimal angle property for the next layer of elements that
meet at that direction. Nevertheless, it was reported that if the marching direction at each point
of a front face was computed in a locally optimal fashion [20, 21], it still might not be optimal
for the prism carried by the face. Therefore, some kind of global smoothing must be performed
after the initial computation of the marching directions. To improve the effect, a front node
classification procedure is required before the execution of such smoothing techniques such
that the marching directions defined at different types of front nodes could be treated differently.

It is worth noting that the results of this node classification procedure are very sensitive to the
user-specified angle thresholds [20, 21]. There also exist some other approaches for computing
marching directions. For example, in the face offsetting method proposed by Jiao [22] faces are
directly propagated along their normals and the vertices are then reconstructed through an
eigenvalue analysis locally, and good resulting boundary layer meshes are presented [23] for
several biomedical models based on this method.

Recently, a few new techniques that rely on the solution of a partial differential equation
(PDE) have been investigated for boundary layer mesh generation [1, 14, 24-27]. Accordingly,
the computation of marching directions is defined in the solution space of the adopted governing
equation rather than in the geometric space. For example, the marching direction at a point
could be defined as the gradient vector of the solution at that point [25]. At present, a frequently
adopted governing equation is the Eikonal equation, and the adopted numerical schemes for the
solution include the fast-marching method [25, 28], fast sweeping method [29], the finite
element method and the finite difference method. To harness these numerical schemes, an
additional volume background mesh is always created [24-27]. Recently, the Laplacian
equation has been chosen as the governing PDE but changed from the scalar function to a vector
one [1]. Mathematically speaking, the solution of this vector form Laplacian equation could
smoothly propagate the marching vectors defined at initial fronts into the domain interior.
Meanwhile, the authors suggest the boundary element method (BEM) adopted as the numerical
scheme of the governing equation since two advantages of the BEM over other numerical
schemes. Firstly, the result of BEM is computed by boundary integration equations rather than
by interpolations; therefore, the computed directions are more accurate. Secondly, the BEM only needs a surface mesh input rather than a volume counterpart.

In comparison with conventional approaches based on local geometric computations, the PDE based approaches provide a new global angle to view the front propagation problem. However, the present PDE-based approaches have some common issues as well. For example, these approaches are usually much slower. Moreover, it remains a challenging issue on how to combine these approaches with multi-normal generation schemes.

2.2 On computation of marching distances

The default marching distance at a front node could be computed by user-specified parameters. This default value leads to viscous elements with same lateral edge lengths at each layer. For viscous walls having complex corners and/or small gaps, local adjustment is necessary to avoid intersections of front lines and generation of low-quality elements. Most existing algorithms support increase marching distances slightly at concave corners and vice versa at convex corners in order to smear concave and convex corners and facilitate the marching process. The computation of local curvatures or angle values, based on either the discrete manifold or the original CAD model, is commonly used to determine local marching distances [1, 3, 21]. Nevertheless, for some PDE-based approaches [1], the solution of PDEs could support a correct adjustment of local marching distance as above. In this case, no local geometric computation is needed any more.

In the neighbourhood of small gaps, reducing marching distances appropriately is an option to avoid global intersection of viscous elements propagated from opposite viscous walls. Here, the main issue is efficient computation of gap distances. Normally, an extra data structure (e.g. quadtree in two-dimensional and octree in three dimensional) is required. In [21], an approach relying on constrained DT is suggested.

The local adjustment of marching distances may lead to an abrupt change of marching distances at neighbouring front nodes. If this issue happens, Laplacian-type smoothing strategies are usually suggested to resolve it.

The more challenging issue is to adapt the mesh to flow solutions or boundary movements. Since this issue is not involved in this study, the discussion is beyond the scope of this paper. Interested readers are referred to [30, 31].

3 Outline of the hybrid meshing method

Figure 1 presents the main steps included in our hybrid meshing method. Given a valid CAD model, the proposed method mainly takes the following steps to output a hybrid mesh.

Step 1. Apply the approach proposed in [32] to compute a sizing function for surface mesh generation and define boundary conditions on surface patches of the model.

Step 2. Given the input model and the sizing function, create a surface triangulation by an in-house advancing front mesher [33].

Step 3. The viscous layer meshing step needs three user parameters that indicate the height of the first layer $h_0$, the expansion ratio of neighbouring layers $\mu$ and the allowed maximum number of layers $n_l$, respectively. According to these parameters, we can compute the marching distance at each front node. In addition, the marching
direction at each front node can be computed by analyzing the manifold of the node. Once the marching directions and marching distances are determined at all the front nodes, a layer of prismatic elements can then be created by connecting the front nodes and their duals after propagating the front. Repeating this front propagation procedure for at most $n_i$ times, we can then create semi-structured prismatic elements in the vicinity of the viscous walls.

**Step 4.** If a symmetry plane is defined on the domain boundary, layered quadrilateral elements should be created in the vicinity of the common curves of the symmetry plane and viscous walls after Step 3. Therefore, the surface mesh of the symmetry plane, which is initially composed of triangular elements only, need be updated to accommodate these quadrilateral elements.

**Step 5.** We can then collect the surface triangles that depict the remaining unmeshed volume region. These triangles include those located at the boundaries with the non-viscous wall types and those depicting the outmost boundary of the boundary layer elements. With these surface triangles as the input, we finally employ an in-house mesher to fill the unstructured tetrahedra in the domain enclosed by the input surface triangles [10, 11]. A feature of the employed mesher is its robust capability to create a boundary constrained tetrahedral mesh. This feature is a key for the success of this step, where a point-to-point conformity is required between the unstructured tetrahedra and boundary layer elements.

![Flowchart of hybrid mesh generation.](image)

The above discussion only sketches the main steps in our method. Nevertheless, to be concise, this discussion does not include a few non-trivial techniques incorporated in our method. These techniques are necessary to improve our method for application to real problems. In Sections 4 to 6, we will discuss the important technical details involved in the three steps, respectively, with a particular focus on Step 3.

4 Boundary layer mesh generation
4.1 Outline of the method

The right part of Figure 1 presents the workflow of our boundary layer mesh generation method. The inputs include the surface triangulation of the domain boundary and some user parameters (such as \( h_0 \), \( \mu \) and \( n_x \)). A list of front faces \( L_f \) and a list of front nodes \( L_n \) are maintained during the entire workflow. Accordingly, flags are attached to the active front nodes and faces to distinguish them from others.

Initially, \( L_f \) and \( L_n \) are filled in with those input surface elements and surface nodes located on the viscous walls, respectively. After that, four steps are consecutively followed to create the boundary layer mesh: (1) computing marching directions, (2) computing marching distances, (3) creating a layer of elements and (4) updating \( L_f \) and \( L_n \). To ensure the reliability of the algorithm and the validity and usability of the output mesh, the intermediate outputs of the former three steps are checked carefully.

In the following subsections, we will present the algorithmic details of the four main steps.

4.2 Computing marching directions

The computation of marching directions is based on the classification of front nodes. A front node is labelled as flat if the maximal angle is smaller than 5 degrees between any two normals of the faces connected to the node. For those unlabelled nodes, a further classification is conducted by computing the average of angles between neighbour face normals. Here, the average angle is denoted by \( \beta \), and approximately, front nodes are classified as convex or concave ones by their \( \beta \) values.

For flat nodes, its marching direction is computed by a simple average of all neighbouring normals. For other nodes, three strategies are combined to set up a cost-effective scheme for marching direction computation:

**Strategy I.** Compute the normals of faces connected to a given front node and classify them into groups such that the number of groups is as small as possible under the condition that the maximal angle is smaller than 25 degrees between any two normals belonging to the same group. For each group, a representative normal is computed by averaging all normals belonging to the group. The normal at the front node is exactly the average of all representative normals.

**Strategy II.** Compute the marching vector lying on the bisection plane of the two faces on the manifold forming the wedge with the smallest angle. The location of the marching vector on that plane is evaluated by bisecting the visibility region on that plane [6]. As is shown in Figure 2, the visibility region is represented by a polyhedral cone extending outward from the point, and it can be simplified into a visibility cone with the circular cross section and half-cone angle, which can also be called the visibility angle \( \beta_v \).

**Strategy III.** This is an iterative algorithm aimed at finding the ‘most normal’ normal, i.e., the normal that minimizes the maximal angle with the given set of normals [17]. Weights are given to each face normal depending on the angle created with the current normal. If the angle is high, more weight is given to the normal. See [17] for a pseudo code of this implementation.
To balance the trade-off between efficiency and robustness, the above strategies are conducted in the order listed above. The quality of the normal at the front node is evaluated by the maximal angle between the normal and normals of manifold faces. A hill-climbing scheme is used to ensure the optimal normal is kept always. Meanwhile, the next level of strategy gets no opportunity in order to save computing time when the quality of the present ‘optimal’ normal is less than 30 degrees.

![Figure 2. The visibility region and its subset (a cone region) at a front node](image)

After computing the initial marching directions, a further smooth is executed to ensure a desirable variation across the front and facilitate the following marching process. Here, the smooth is performed by a weighted Laplacian approach, i.e.,

\[
N_i^n = (1 - \omega)N_i^{n-1} + \sum_j \omega \sum_j (w_{ij})N_{ij}^{n-1},
\]

where \(N_i^{n-1}\) and \(N_i^n\) are normals at front node \(p_i\) after \(n\) and \(n-1\) iterations, \(N_{ij}^{n-1}\) is the normal at neighbouring front node \(p_j\) after \(n-1\) iterations, and \(w_{ij}\) is the weight defined at \(p_j\).

Note that it is beneficial to let normals at convex corners be closer to their neighbours, and vice versa for concave corners. To achieve this, \(n_{ij}\) is defined as below,

\[
w_{ij} = \begin{cases} 
k_{ij}^2 & \text{if } p_i \text{ is a convex point} \\
1/k_{ij}^2 & \text{if } p_i \text{ is a concave point} \\
1 & \text{otherwise}
\end{cases}
\]

where

\[
k_{ij} = n \frac{d_{ij}}{\sum_{j \neq i} d_{ij}}
\]

and \(d_{ij}\) are the distance between \(p_i\) and \(p_j\).

There are cases where one single normal could not ensure the validity of visibility cone. A multi-normal strategy is adopted in these cases. Interested readers are referred to [34] for more details.

To ensure the validity of the computed marching directions, for each marching direction, we check whether it is visible to all the front faces adjacent to the front nodes. If no valid marching direction can be defined at a front node, we stop propagating the front node and clear the ‘front’ flag attached to that node.

Figure 3 presents the computed marching directions in the neighbourhood of two typical
corners. It can be seen that the computed directions are reasonable and no abrupt changes occur between neighbouring marching directions.

![Marching directions in the neighbourhood of two different types of corners.](image)

(a) A concave corner. (b) A convex corner.

### 4.3 Computing marching distances

Suppose $p_i$ is a front node, $u_i$ is the marching direction at $p_i$, and $m$ is the present layer number. The marching distance at $p_i$ can be computed by

$$h_i = \mu^{m-1} h_0.$$

If simply applying the above Equation in all front nodes, the computed marching sizes would be the same. As a result, the front will conduct a so-called *advective motion* at expansion. As illustrated in Figure 4, *wave-frontal motion* may be more desirable than advective motion because it would smooth out convex or concave corners, whereas the advective motion would preserve them. To implement the effect of a wave-frontal motion, a coefficient $\alpha$ can be added such that the distances at the convex corners need to be shortened and the marching distances at the concave corners need to be lengthened.

$$h_i = \alpha \mu^{m-1} h_0,$$

where

$$\alpha = \begin{cases} 1 + |\cos \beta| & \beta < 180^\circ \\ 1 - |\cos \beta| & \beta \geq 180^\circ. \end{cases}$$

Another geometric factor that impacts the computation of marching distances is the gap between opposite viscous boundaries. Given a front node $p_i$ and a direction starting from $p_i$ to domain interiors, a minimal distance between $p_i$ with viscous walls is computed first, denoted by $d_i^G$. Denote the total height of viscous boundary at $p_i$ by

$$l_i = \sum_{m=1}^{n_i} \alpha \mu^{m-1} h_0.$$

If $d_i^G > 2l_i$, no further modification is required on the marching distance at $p_i$; otherwise, the height of the first viscous layer at $p_i$ is computed by
where $\varepsilon_i \in [0, 0.5]$ is a user parameter that determines how large the space is left for unstructured elements (less is more).

![Diagram](image_url)

**Figure 4.** Advection vs. wave-frontal propagation at expansion at a (a) convex corner and (b) concave corner.

Apparentally, the computation of $d_i^G$ should be speeded up at the aid of a spatial data structure. Here, we reuse the octree grid for speeding up front intersection checks to compute $d_i^G$. Algorithm 1 presents the setup procedure of this octree grid, and Algorithm 2 presents the computation of $d_i^G$ based on the grid.

Finally, a smoothing procedure is executed to avoid the abrupt changes of neighbouring marching directions. A simple Laplacian smoothing is presently employed.

---

**Algorithm 1** Setup procedure of this octree grid

**Input:** Octree root node $ocT$, Maximum depth $d_{\text{max}}$, Maximum number of triangles in one cube $T_{\text{max}}$, Boundary triangle Mesh topology

**Output:** Octree $ocT$

1: for triangles $T_i$ in boundary mesh topology do
2:     Find all the leaf node $l_{\text{node}}$ in $ocT$ with $T_i$ may intersect it by recursive
3:     for node $n$ in $l_{\text{node}}$ do
4:       if current depth $d_n < d_{\text{max}}$ and the number of triangles in $n$ $T_n >= T_{\text{max}}$ then
5:         Split $n$ into 8 cube
6:         move the triangles in $n$ to the subnode of $n$
7:       else
8:         Insert $T_i$ to $n$
9:     end if
10:   end for
11: end for
Creating a layer of elements

For each front node qualified for propagation, we can compute its dual by marching it along the marching direction. After that, we can create a layer of prismatic elements by connecting all the front nodes qualified for propagation and their duals.

Low-quality elements may be created in this step, in particular in the vicinity of concave corners. In this study, we selected scaled aspect ratio [23] to evaluate the quality of a prism. This quality measure in effect combines the measures of triangle shapes and edge orthogonality.

For a given prism \( \tau \), denote the scaled aspect ratio of this element by \( \rho(\tau) \) \([23]\) is computed, and the propagation will stop when \( \rho(\tau)<0 \);

(3) The new front faces starting from the front node are involved in global intersections;

(4) If one of the neighbouring nodes of the current node in the previous layer is set to stop propagating, then the propagation of that node is stopped.

Criterion 1 is straightforward and ensures the termination of the boundary layer mesh generation procedure. Criterion 2 avoids the generation of elements with negative signed
volumes. Figure 5a presents a node \( p_1 \) with \( \rho(p_1) < 0 \), which leads to an inverted element. Criterion 3 is used to avoid global intersection. Criterion 4 is only executed for 3D cases and ensures the difference in the layer numbers of the neighbouring nodes does not exceed one. In 2D problems, the exposed segments after boundary layer mesh generation are sent to a triangular mesh generator, and the difference in the layer numbers of the neighbouring nodes will not affect the resulting mesh quality. However, in 3D problems, the exposed faces include triangles and quadrilaterals, and some transition elements should be added to hide the quadrilateral faces before the exposed faces are sent to a tetrahedra mesh generator. If the difference in the layer numbers of the neighbouring nodes is allowed to be larger than one, stretched pyramids will be added as the transition elements. Figure 5b presents the added transition elements for two different cases.

Let \( F \) be one of the current propagating fronts and \( p_i (i=0, 1, 2) \) be the forming nodes of that front. If \( p_i (i=0, 1, 2) \) are all propagated to new positions, i.e., \( p'_i (i=0, 1, 2) \), then \( F \) is propagated to \( F' \), with \( p'_i \) as its forming nodes. In addition, \( F' \) will replace \( F \) as a new front in the next layer. However, if at least one node of \( p_i (i=0, 1, 2) \) is not allowed to propagate to the next layer, \( F \) will also be allowed to propagate to the next layer.

![Figure 5](image-url)

**Figure 5.** (a) An invalid element with \( \rho(p_1) < 0 \). (b) Transition elements in two different cases.

### 5 Unstructured mesh generation

If a symmetry plane is defined on the domain boundary, layered quadrilateral elements should have been created in the vicinity of the common curves of the symmetry plane and viscous walls after boundary layer mesh generation. To accommodate those quadrilateral elements, we first remove the original surface mesh of the symmetry plane, then obtain the boundary description of the unmeshed region of the symmetry plane, and finally mesh this region using an advancing front surface mesher [34].

We next employ an in-house DT mesher to fill the unstructured tetrahedra in the domain enclosed by the input surface. Note that the DT criterion provides a reasonable method to link a given point set; however, it cannot ensure the existence of boundary constraints in the resulting tetrahedralisation. A boundary recovery procedure is thus required to ensure the boundary integrity of the resulting mesh. For the hybrid meshing problem focused on in this study, one part of the surface input to the DT mesher is composed of the exposed faces of the boundary layer elements. Some of those faces could be rather stretched and thus lead to a more challenging task for the boundary recovery procedure. By incorporating a few novel techniques for boundary recovery [10, 11], a feature of our in-house mesher is its capability to robustly create a boundary constrained tetrahedral mesh. This feature is a key for the success of the
unstructured mesh generation step because point-to-point conformity is required between the unstructured tetrahedra and boundary layer elements.

6 Numerical experiments

6.1 DLR F6 aircraft model

The surface mesh of the F6 model is presented in Figure 6a; it contains 14,866 nodes and 29,732 triangle elements. Many complex concave regions and corner nodes are involved in this model, e.g., a complex corner node at the tail of the engine and several concave regions near the joint of the engine and wing (see close-up views in Figure 6a). In the process of boundary layer mesh generation, the surface mesh of the F6 model was set as the viscous wall boundary condition. Figure 10b presents a cut view of the hybrid mesh for exterior flow simulations, in which 689,281 prisms, 160,452 tetrahedron and 9,441 pyramids are consisted. A cut-out view of the boundary layer mesh and local mesh details around two complex corners are presented in Figure 10b. As can be seen, a valid normal vector was obtained by the proposed method in both corners. If using the average normal vectors of the neighbouring surfaces, it was difficult to determine suitable normals here because the normal vectors of several faces around both corners are nearly in opposite directions.

Figure 6. The hybrid mesh of the F6 model: (a) The surface mesh; (b) Cut-out views of the hybrid mesh.
6.2 Rocket model

To demonstrate that the proposed method could avoid global intersections by identifying small gaps and reducing marching distances locally, a rocket model is selected in which a few volume proximities exist between the main body and 9 boosters. This model contains 445 surface patches, on which 73,759 surface nodes and 147,470 triangles are generated. The resulting hybrid mesh contained 3,891,188 prisms, 520,245 tetrahedrons and 42,862 pyramids. Figure 7 presents the surface mesh and the close-up views for the color map of gap distances, in which regions with small gap distances are rendered in blue. As can be seen, the volume proximities between the main body and boosters are all correctly identified. Figure 8 presents two cut-out views of the hybrid mesh. Narrow gap can be clearly observed and the global intersection there was effectively avoided with the proposed method.

Figure 7. The hybrid mesh of a rocket model: (a) The surface mesh; (b) Close-up views of the color map of gap distances ($d^{ij}$).

6.3 Space shuttle model

To demonstrate the proposed method for configurations with industry-level complexity, we chose to generate the hybrid mesh of a space shuttle model. This model contains 595 surface patches, on which 170,933 surface nodes and 341,846 triangles are generated. Note that this model contains abundant geometric details near the joints of different parts, the volume proximities between the main bodies of boosters and fuel tanks, etc. The resulting hybrid mesh contained 6,176,455 prisms, 2,111,679 tetrahedrons and 114,465 pyramids. Figure 9 presents two cut-out views of the hybrid mesh. Figure 10 presents close-up views of three local details of the hybrid mesh.
Figure 9. Cut-out views of the hybrid mesh of the space shuttle model: (a) Front view; (b) Side view.

Figure 10. Local details of boundary layer meshes of the space shuttle aircraft. (a) Boundary layer meshes around the support between the plane and rocket; (b) Boundary layer meshes around the top view of rocket; (c) Boundary layer meshes around space shuttle tail.

6.4 Mesh quality

Quality of prismatic elements is our major concern. To evaluate the quality of the generated prismatic elements, the scaled aspect ratio quality measure was first adopted in this study. In this study, inverted elements are not allowed, and we refer to elements with $\rho(r)<0.2$ as low-quality elements. The distributions of scaled aspect ratios of prismatic elements for the F6, rocket and space shuttle models are presented in Figure 11. The ratio of low-quality elements accounts for 0.4%, 0.7%, 0.03% of the total numbers of prism elements for the three models, respectively.

The equiangle skewness is another commonly used quality measure for various types of elements. For a prism, it is represented as the maximum ratio of the element faces’ included angles to the angles of equilateral faces. Its value varies between 0 (good) and 1 (bad). It is recommended this skewness measure be kept below 0.8 for a good grid; values below 0.9 are acceptable, depending on the solver. Therefore, we refer to elements with skewness values larger than 0.9 as low-quality elements. Under this new standard, the ratio of low-quality elements accounts for 0.17%, 0.18%, 0.026% of the total numbers of prism elements for the three models, respectively.

Overall, the above data reveals that the boundary layer elements created by the proposed
method have acceptable shape and quality.

![Figure 11](image)

(a) Quality histogram of prisms. (a) Scaled aspect ratio; (b) Equiangle skewness.

### 6.5 Comparison with commercial tools

In most tests we conducted, the proposed method achieved the similar level of reliability and element quality as commercial mesh tools, such as Pointwise. However, it was also observed in some complex corners, the proposed method creates boundary layer elements with more desirable quality. Figure 12a presented the surface input used for comparison. With the same surface input and same user settings, we create two hybrid meshes by using the proposed method and Pointwise, respectively. Figures 12b and 12c enlarge the details of two meshers near the tail of the aircraft, in which a convex corner with a very small angle exist. Although the multi-normal technique was employed by Pointwise, the resulting boundary mesh by Pointwise stop its propagation much earlier than its counterpart by our method.

![Figure 12](image)

(a) Comparison with Pointwise. (a) Surface input. (b) Mesh by our method. (c) Mesh by Pointwise.

### 7 Concluding remarks

A prismatic hybrid mesh configured with layered prismatic elements in the near field of viscous walls and an unstructured mesh in the rest of the domain is preferred in many applications because it represents a good compromise between solution accuracy and ease of use. Several novel computing strategies for marching directions and marching distances are implemented by taking the quality of the resulting elements and the reliability of the meshing procedure as the primary consideration. These efforts enable the setup of a hybrid mesher that could generate...
qualified viscous grids for geometries with industry-level complexities. Numerical experiments including academic cases, benchmark cases and cases from industry-level simulations are presented to verify its effectiveness and efficiency.

Declarations

Acknowledgements

This research is funded by the National Numerical Wind Tunnel Project of China (Grant No. NNW2018-ZT5B10).

Authors’ contribution

The research output is coming from joint effort. All authors read and approved the final manuscript.

Funding

National Numerical Wind Tunnel Project of China (Grant No. NNW2018-ZT5B10).

Availability of data and materials

The datasets used and/or analysed during the current study are available from the corresponding author upon reasonable requests.

Competing interests

The authors declare that they have no competing interests.

Author detail

See the first page of the manuscript.

References

1. Zheng Y, Xiao Z, Chen J, Zhang J. Novel Methodology for Viscous-Layer Meshing by the Boundary-Element Method. AIAA Journal 2018; 56: 209-221.
2. Ito Y, Murayama M, Yamamoto K, et al. Efficient hybrid surface and volume mesh generation for viscous flow simulations. AIAA paper, 2011, 3539: 2011.
3. Sharov D, Nakahashi K. Hybrid prismatic/tetrahedral grid generation for viscous flow applications. AIAA journal, 1998, 36(2): 157-162.
4. Hassan O, Morgan K, Probert E J, et al. Unstructured tetrahedral mesh generation for three-dimensional viscous flows. International Journal for Numerical Methods in Engineering, 1996, 39(4): 549-567.
5. Marcum DL. Efficient generation of high-quality unstructured surface and volume grids. Engineering with Computers, 2001, 17(3): 211-233.
6. Pirzadeh S. Three-dimensional unstructured viscous grids by the advancing-layers method. AIAA journal, 1996, 34(1): 43-49.
7. Garimella R V, Shephard M S. Boundary layer mesh generation for viscous flow simulations. International Journal for Numerical Methods in Engineering, 2000, 49(1): 193-218.
8. Löhner R, Parikh P. Generation of three-dimensional unstructured grids by the advancing front method. International Journal for Numerical Methods in Fluids, 1988, 8(10): 1135-1149.
9. Bowyer A. Computing dirichlet tessellations. The Computer Journal, 1981, 24(2): 162-166.
10. Chen J, Zhao D, Huang Z, Zheng Y, Gao S. Three-Dimensional Constrained Boundary Recovery with an Enhanced Steiner Point Suppression Procedure. Computers and Structures 2011; 89: 455-466.
11. Chen J, Zheng J, Zheng Y, Si H, Hassan O, Morgan K. Improved Boundary Constrained Tetrahedral Mesh Generation by Shell Transformation. Applied Mathematical Modelling 2017; 51: 764-790.
12. Zheng Y, Liou MS. A Novel Approach of Three-Dimensional Hybrid Grid Methodology: Part 1. Grid Generation. Computer Methods in Applied Mechanics and Engineering 2003; 192:4147-4171.
13. Liou MS, Zheng Y. A Novel Approach of Three-Dimensional Hybrid Grid Methodology: Part 2. Flow Solution. Computer Methods in Applied Mechanics and Engineering 2003; 192: 4173-4193.
14. Park S, Jeong B, Lee J G, et al. Hybrid grid generation for viscous flow analysis. International Journal for Numerical Methods in Fluids, 2013, 71(7): 891-909.

15. Zhang L, Zhao Z, Chang X, He X. A 3D Hybrid Grid Generation Technique and a Multigrid/Parallel Algorithm Based on Anisotropic Agglomeration approach. Chinese Journal of Aeronautics 2013; 26:47-62.

16. Kallinderis Y, Ward S. Prismatic grid generation for three-dimensional complex geometries. AIAA Journal, 1993, 31(10): 1850-1856.

17. Aubry R, Löhner R. On the ‘most normal’normal. Communications in Numerical Methods in Engineering, 2008, 24(12): 1641-1652.

18. Aubry R, Mestreau E L, Dey S, et al. On the ‘most normal’normal—Part 2. Finite Elements in Analysis and Design, 2015, 97: 54-63.

19. Loseille A, Löhner R. Robust Boundary Layer Mesh Generation. In: 21st International Meshing Roundtable, Orlando, Florida, USA; 2013. p.457-474.

20. Ito Y, Shih A M, Soni B K, et al. Multiple marching direction approach to generate high quality hybrid meshes. AIAA Journal, 2007, 45(1): 162-167.

21. Wang F, di Mare L. Hybrid Meshing Using Constrained Delaunay Triangulation for Viscous Flow Simulations. International Journal for Numerical Methods in Engineering 2016; 108:1667-1685.

22. Aubry R, Löhner R. On the ‘most normal’normal. Communications in Numerical Methods in Engineering, 2008, 24(12): 1641-1652.

23. Dyedov V, Einstein D R, Jiao X, et al. Variational generation of prismatic boundary layer meshes for biomedical computing. International Journal for Numerical Methods in Engineering, 2009, 79(8): 907-945.

24. Wang Y. Eikonal equation based front propagation technique and its applications. AIAA Paper, 2009, 375.

25. Wang Y, Guibault F, Camarero R. Eikonal equation based front propagation for arbitrary complex configurations. International Journal for Numerical Methods in Engineering, 2008, 73(2): 226-247.

26. Xia H, Tucker P G, Dawes W N. Level sets for CFD in aerospace engineering. Progress in Aerospace Sciences, 2010, 46(7): 274-283.

27. Dawes W N, Harvey S A, Fellows S, et al. Viscous layer meshes from level sets on cartesian meshes[C]/45th AIAA Aerospace Sciences Meeting & Exhibit. 2007: 8-11.

28. Sethian JA. Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science. Cambridge University Press: Cambridge, 1999.

29. Zhao HK. A fast sweeping method for Eikonal equations. Mathematics of Computation 2005; 74(250):603–627.

30. Zheng J, Chen J, Zheng Y, et al. An improved local remeshing algorithm for moving boundary problems. Engineering Applications of Computational Fluid Mechanics, 2016, 10(1): 405-428.

31. Hassan O, Sørensen KA, Morgan, K, Weatherill NP. A method for time accurate turbulent compressible fluid flow simulation with moving boundary components employing local remeshing. International Journal for Numerical Methods in Fluids, 2007, 53(8), 1243-1266.

32. Chen J, Xiao Z, Zheng Y, et al. Automatic Sizing Functions for Unstructured Surface Mesh Generation. International Journal for Numerical Methods in Engineering, 2009, 77(9): 1247.