

23rd International Meshing Roundtable (IMR23)

# Automatic Unstructured Element-Sizing Specification Algorithm for Surface Mesh Generation

Zhoufang Xiao, Jianjun Chen\*, Yao Zheng, Lijuan Zeng, Jianjing Zheng

*Center for Engineering and Scientific Computation, and School of Aeronautics and Astronautics, Zhejiang University, Hangzhou 310027, China*

---

## Abstract

Automatic and accurate element-sizing specification is crucial for the efficient generation of surface meshes with a reduced number of elements. This study proposes a novel algorithm to fulfill this goal, which is based on the unstructured background mesh, and provides an alternative of existing Cartesian mesh based algorithms. The geometric factors are considered and combined with some user parameters to form an initial size map. This map is then smoothed to a well-graded one. The timing efficiency of the size-query routine is improved to a comparable level with that based on the Cartesian mesh. Finally, the size map produced by the proposed algorithm is used to control the scales of elements generated by an advancing front surface mesher. Meshing experiments of complex computational aerodynamics configurations show that the proposed algorithm only consumes minutes to prepare the size map for the surface mesher; however, a conventional grid sources based scheme may need many hours by a tedious manual interaction process.

© 2014 The Authors. Published by Elsevier Ltd.

Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23).

*Keywords:* Mesh generation; Surface mesh; Element size; Adaptive mesh; Background mesh

---

## 1. Introduction

In the community of computational aerodynamics, a great success of unstructured mesh technologies has been witnessed in the past decades due to its automatic and adaptive abilities for complex geometry configurations [1]. Nowadays, many commercial or in-house codes can generate unstructured meshes in a very reliable and computationally efficient manner. Owing to the rapid advance of parallel mesh generation algorithms, the time cost consumed by the pipeline of unstructured mesh generation can be further reduced to a very low level [2–5]. For instance, the authors have recently implemented a parallel pipeline where the three major steps of unstructured mesh generation (i.e., surface meshing, volume meshing and volume mesh quality improvement) are all parallelised [4,5]. Experiments show that this parallelised pipeline can employ about 100 computer cores to generate a high-quality mesh composed of hundreds of millions of tetrahedral elements in minutes.

---

\* Corresponding author. Tel.: +0-086-571-87951883 ; fax: +0-086-571-87953168.

*E-mail address:* [chenjj@zju.edu.cn](mailto:chenjj@zju.edu.cn)

Although the pipeline of unstructured mesh generation might not be a performance bottleneck any more in many cases, how to prepare a valid and suitable input for this meshing pipeline remains challengeable. In general, this input contains a geometry that defines the meshing domain and an element-sizing function that defines the distribution of element scales over the meshing domain. In the following discussions, our focus is only on the preparation of the element-sizing function for unstructured surface mesh generation. The preparation of a high-quality geometry input is not discussed in this study. However, it needs to be emphasised that this issue is a very active research topic in the community of mesh generation [6].

A good element-sizing function should define small element scales in the region where geometrical and physical characteristics exist and large scales elsewhere. Moreover, the gradation of the element scales must be limited so that the quality of elements in the gradation region is ensured. In the community of computational aerodynamics, grid sources are popular tools and employed by many meshing codes to define element-sizing functions [7–10]. For a simple configuration such as the DLR-F6 wing-body-nacelle-pylon aircraft (referred to as the F6 model hereafter), tens of grid sources are enough to help generate a good-quality surface mesh for typical aerodynamics simulations. The time cost of defining the sources of this magnitude is usually affordable if a user friendly graphic interface is available. However, if a more complicated aerodynamics model such as a full loaded fight aircraft is input, more than hundreds of grid sources might be required, and the interactive process that defines these sources is error-prone and time-consuming. In our experience, even days of wall-clock time might be consumed. By contrast, the subsequent automatic meshing pipeline consumes only minutes of wall-clock time. In this sense, if not considering the geometry preparation step, the real performance bottleneck of generating unstructured meshes for a complicated aerodynamics model lies in the phase of defining element-sizing functions rather than mesh generation itself.

For many years, solution-adaptive techniques have been expected to be able to remove the dependence of the computing accuracy on the initial mesh configuration. However, it was reported that an adapted solution might be invalid if it originated from a poor-quality mesh [8]. In fact, a good initial mesh can either eliminate the need for adaptive grid refinement for certain class of problems or enhance the performance of many adaptation methods. Therefore, even configured with an adaptive solver, the requirement that defines a suitable element-sizing map for the initial mesh generation is still indispensable.

Many automatic algorithms have been designed to define an element-sizing map in a less intensive user-interactive manner. A fundamental feature that can be used to classify these algorithms is the background mesh adopted, where the element-sizing map is stored by assigning element scales at mesh nodes and interpolating scales in background element interiors using the node values. The most prevailing background mesh adopted so far is the Cartesian mesh, which is interiorly organised as a quadtree or octree; thus, the time complexity of searching a background element that contains a point is proportional to the depth of the tree. This fast search property is vital to reduce the meshing time because a meshing process would call this search routine frequently. However, the disadvantage of using a Cartesian mesh is that the tree-level difference of neighboring cells is required to be less than or equal to one, and the refinement of a single cell has to be propagated into its neighboring cells. Therefore, the Cartesian mesh based scheme is expensive in terms of computational time and storage requirement [7–9]. This shortcoming even became one of the reasons that Pirzadeh gave up his early Cartesian mesh based element-sizing scheme [7] when he developed a new scheme for volume meshing problem [8].

In this study, a new automatic algorithm that initialises element-sizing maps for surface mesh generation is proposed by using unstructured background meshes. In contrast to the Cartesian background mesh, the unstructured mesh has a more flexible topological structure and its local update does not need to be propagated; hence, it can represent a size map with reduced memory and timing requirements. In addition, instead of using a volume Cartesian background mesh for surface mesh generation, the unstructured background mesh considered in this study is a surface mesh. This feature can further reduce the memory and time consumption of the proposed algorithm. To highlight the contributions of this study, some features of the proposed algorithm are listed as below:

- (1) Curvature and proximity features of the input geometry are calculated and element scales are adapted to these features automatically.
- (2) The requirement of a well-graded size map defined on the unstructured mesh is formularised, and an automatic procedure is suggested to limit the gradation of element scales.
- (3) A brute-force procedure that queries the size value of a point in an unstructured background mesh is very slow and its practical efficiency is unacceptable. An improved query procedure that does not depend on any auxiliary structure is suggested in this study.

Furthermore, the size map produced by the proposed algorithm is used to control the scales of elements generated by an advancing front surface mesher. Two typical aerodynamics models are tested in this article. One is the F6 model, and the other is a full loaded F16 aircraft model with landing gear details. Experiment results show that the proposed algorithm can help generate high-quality surface meshes in a far more computationally efficient manner than the meshing process that controls element scales purely by grid sources.

## 2. Preliminary definitions

### 2.1. Unit mesh

As shown in Figure 1, assuming that the size function of this line is  $h(t)(t \in [0, 1])$ , the length of the line is  $l$ , the number of mesh segments in this line is:

$$n = l \int_0^1 \frac{1}{h(t)} dt. \quad (1a)$$

For each segment of this line,

$$1 = l \int_{t_i}^{t_{i+1}} \frac{1}{h(t)} dt \quad (i = 1, 2, \dots, n). \quad (1b)$$

Given a metric  $\mathbf{M} = [1/h^2(t)]$  defined in this line, in the space defined by  $\mathbf{M}$ , the length of this segment is

$$(\Delta l_i)_{\mathbf{M}} = \int_{t_i}^{t_{i+1}} \sqrt{l \cdot \frac{1}{h^2(t)} \cdot l} dt = l \int_{t_i}^{t_{i+1}} \frac{1}{h(t)} dt = 1 \quad (i = 1, 2, \dots, n),$$

which means that an ideal mesh edge in the space defined by  $\mathbf{M}$  has a unit length. Correspondingly, we call this mesh a *unit mesh* [11].

Moreover, if  $h(t)$  is a linear function, i.e.,  $h(t) = (1-t)h_0 + th_n$ , we can get a new equation from Equation 1a:

$$n = \begin{cases} l/h & h_0 = h_n = h \\ l \cdot \ln(h_n/h_0)/(h_n - h_0) & h_0 \neq h_n \end{cases}. \quad (2a)$$

Assuming that the length of the segment  $a_{i-1}a_i$  is  $\Delta l_i$ , for each segment of this line,

$$1 = \begin{cases} \Delta l_i/h & h_i = h_{i-1} = h \\ \Delta l_i \cdot \ln(h_i/h_{i-1})/(h_i - h_{i-1}) & h_i \neq h_{i-1} \end{cases} \quad (i = 1, 2, \dots, n). \quad (2b)$$

### 2.2. Geometric progressive mesh

For the mesh shown in Figure 1,  $\forall \beta \geq 1.0$ , if

$$\frac{\Delta l_{i+1}}{\Delta l_i} = \begin{cases} \beta & \Delta l_{i+1} \geq \Delta l_i \\ 1/\beta & \Delta l_{i+1} < \Delta l_i \end{cases} \quad (i = 1, 2, \dots, n-1),$$

this mesh is called a *geometric progressive mesh*, and  $\beta$  the progressive factor. Loosely, if

$$\frac{1}{\beta} \leq \frac{\Delta l_{i+1}}{\Delta l_i} \leq \beta \quad (i = 1, 2, \dots, n-1),$$

this mesh is called a *quasi geometric progressive mesh*.

### 2.3. H-variant for a line

It is supposed that the mesh shown in Figure 1 meets the following assumptions: (1) It is a unit mesh, i.e., the predefined size map is accurately respected; (2) It is a geometric progressive mesh with the progressive factor  $\beta$ ; (3) The size variation follows a linear function.

Without loss of generality, we assume  $h_n > h_0$  and

$$\Delta l_{i+1}/\Delta l_i = \beta \quad (i = 1, 2, \dots, n-1).$$

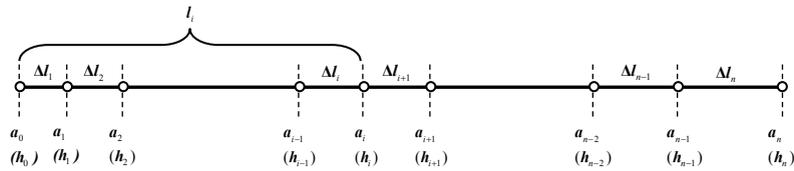


Fig. 1: A line mesh example.

According to Equation 2b, we can get

$$\Delta h_{i+1}/\Delta h_i = \beta \ln(h_{i+1}/h_i) / \ln(h_i/h_{i-1}), \quad (3)$$

where  $\Delta h_{i+1} = h_{i+1} - h_i$  and  $\Delta h_i = h_i - h_{i-1}$ . Because the size variation follows a linear functions, so

$$\Delta h_{i+1}/\Delta h_i = \Delta l_{i+1}/\Delta l_i = \beta. \quad (4)$$

According to Equations 3 and 4, we can get

$$h_{i+1}/h_i = h_i/h_{i-1}, \quad (5)$$

i.e., the size values of mesh vertices form a geometric sequence.

Assuming that  $\Delta h = h_n - h_0$ , the size values of the middle vertices are given by

$$h_i = (1 - w_i)h_0 + w_i h_n = h_0 + \Delta h w_i \quad (i = 1, 2, \dots, n), \quad (6)$$

where

$$w_i = l_i/l = \sum_{j=1}^i \Delta l_j/l = (\beta^i - 1)/(\beta^n - 1) \quad (i = 1, 2, \dots, n). \quad (7)$$

According to Equations 5 and 6, we can get

$$\Delta h = h_0(w_{i-1} + w_{i+1} - 2w_i)/(w_i^2 - w_{i-1}w_{i+1}). \quad (8)$$

Introducing Equation 7, Equation 8 can be simplified:

$$\Delta h = h_n - h_0 = (\beta^n - 1)h_0 \Rightarrow h_n = \beta^n h_0. \quad (9)$$

Therefore, the geometric factor of the sequence formed by the size values of mesh vertices is also  $\beta$ , similar to that of the sequence formed by the edge lengths.

Finally, combining Equations 2a and 9, the relation between the size difference of the end points of a line and its length is given by

$$\Delta h/l = h_n - h_0 = \ln \beta. \quad (10)$$

Note that the above deduction is based on the assumption of  $h_0 \neq h_n$ . However, it is also valid when  $h_0 = h_n$ , where a uniform mesh is generated and  $\beta = 1$ . In [11],  $\Delta h/l$  is called *H-variant*. When  $l$  approaches zero, it represents the derivative of the sizing function.

#### 2.4. Gradation control for a triangular region

Assuming that  $T$  is a triangular background element,  $\Omega_T$  is the domain that  $T$  covers, and a linear size map is defined over  $\Omega_T$ ,

$$h(x, y) = \sum_{i=0}^2 w_i(x, y)h_i \quad (x, y) \in \Omega_T,$$

where  $h_i (i = 0, 1, 2)$  are the size values at three background vertices, and  $w_i(x, y) (i = 0, 1, 2)$  are the area coordinate functions. Given a progressive factor  $\beta (\beta \geq 1.0)$ , the basic requirement to define a well-graded size map over  $\Omega_T$  is:

$$h_v(p_1 p_2) = |h(p_2) - h(p_1)| / |p_1 p_2| \leq \ln \beta (\forall p_1, p_2 \in \Omega_T),$$

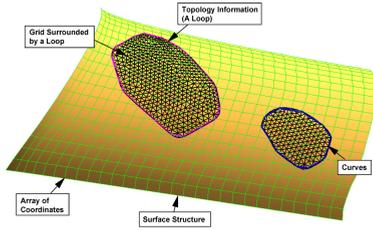


Fig. 2: Illustration for the surface B-rep.

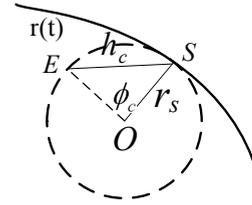


Fig. 3: Illustration for the curvature-oriented size value.

where  $h_v(p_1p_2)$  is the H-variant of the line  $p_1p_2$ . When  $p_2$  approaches  $p_1$ , it represents the direction derivative of the sizing function  $h(x, y)$  along  $p_1p_2$ .  $p_1p_2$  can be an arbitrary direction around  $p_1$ , therefore,

$$|\nabla h(p_1)| = \max_{\forall p_2 \in \Omega_T} \frac{\partial h}{\partial p_1 p_2} \leq \ln \beta \quad (\forall p_1 \in \Omega_T).$$

The gradient of a linear function  $h(x, y)$  over  $\Omega_T$  is constant:

$$\nabla h = const = \frac{1}{2A} \left( \sum_{i=0}^2 b_i h_i \quad \sum_{i=0}^2 c_i h_i \right)^T,$$

where  $A$  is the area of  $T$ ,  $b_i = y_j - y_m$ ,  $c_i = x_m - x_j$  ( $i = 0, 1, 2; j = (i + 1) \bmod 3; m = (j + 1) \bmod 3$ ), and  $(x_i, y_i) (i = 0, 1, 2)$  are the coordinates of the three vertices of the triangle  $T$ . Therefore, the size map over  $\Omega_T$  is identified as well-graded only when

$$|\nabla h|^2 = H^T K H \leq \ln^2(\beta), \tag{11}$$

where

$$H = (h_0, h_1, h_2)^T; K = [k_{ij}]_{0 \leq i, j \leq 2}; k_{ij} = (b_i b_j + c_i c_j) / (4A^2).$$

Obviously,  $K$  is a symmetric matrix, and the principal diagonal elements are positive:

$$k_{ii} = (b_i^2 + c_i^2) / (4A^2) = l_i^2 / (4A^2) > 0,$$

where  $l_i$  is the length of the edge opposite to the vertex  $i$  [12].

### 3. Initialisation and smoothing of the size map

The proposed algorithm is composed of two separated components. The first component inputs a geometric model and outputs a geometry-based adaptive size map. The second component provides an interpolation routine to return the size value of any position which is required by meshing algorithms. The major steps of the first component are investigated in this section, while the second component is to be discussed in the next section.

Furthermore, the flowchart of the first component can be organised into the following steps.

- (1) A discretised dual of the input continuous geometry is generated and used as the background mesh.
- (2) An initial size map is formed by calculating the curvature and proximity features defined at each background vertex. A size value is then determined for each vertex under the combined influence of the calculated geometry features and some user parameters.
- (3) Given a user parameter  $\beta$ , the initial size map is smoothed such that Inequality 11 is met for all background elements.

#### 3.1. Initialisation of the background mesh

The input geometry considered in this study is a composite parametric surface model. This type of model is composed of many continuous surface patches. Interiorly, the model is organised with a surface boundary representation (B-rep). The surface B-rep is a subset of the solid B-rep and includes three basic topology entities, i.e., *face*, *curve* and *point*. As illustrated in Fig. 2, a loop is a specific topology entity that limits the valid region of a face. Internally, a loop refers to a set of boundary curves and termed as a group entity to distinguish it with other topology entities.

Firstly, the curves and faces of the input geometry are meshed to linear segments and triangles, respectively. In this stage, the triangle shapes do not matter; however, the tessellated model must be geometrically close to the continuous model. Most CAD software or kernel products provide the tessellation function for continuous surface models.

For a discrete model, three topology entities of different dimensions are defined, called *facets*, *edges* and *vertices*, respectively. A connection between these entities and B-rep entities is necessary for the proposed algorithm. Here, this connection is represented by the following mappings:

- (1) The *face-facet mapping*. A face corresponds to a group of facets.
- (2) The *curve-edge mapping*. A curve corresponds to a group of edges.
- (3) The *point-vertex mapping*. A point corresponds to a vertex.

Other mappings can be defined as well, e.g., between a curve and all vertices that lie on the curve, or between a face and all edges that bound the face. These mappings can be derived from the above basic mappings; therefore, they are not explicitly represented in the extended B-rep.

A definition is presented to describe the above mappings:

**Definition 1 (Classification)** [13]. Given a  $d_i$ -dimensional topology entity of the discrete model,  $M^{d_i}$  is classified on a  $d_j$ -dimensional topology entity ( $d_i \leq d_j$ )  $G^{d_j}$  of the B-rep if  $M^{d_i}$  lies on  $G^{d_j}$ , denoted as  $M^{d_i} \subseteq G^{d_j}$ .

### 3.2. Initialisation of the size map

#### 3.2.1. Curvature-oriented size values

In the highly curved region, the element size needs to be very small to avoid the formation of a big gap between the mesh and the geometry. Figure 3 illustrates the geometric meaning of the curvature-oriented size value  $h_c$  of a point  $S$ , where  $r_s$  is the curvature radius of  $S$ ,  $\mathbf{r}(t)$  is the curve or the principal curvature line that  $S$  lies in, and  $O$  is a point along the normal vector of  $S$  with  $|OS| = r_s$ . With  $O$  as the center,  $r_s$  as the radius,  $S$  as the contact point, an inscribed circle of  $\mathbf{r}(t)$  is defined, and  $E$  is a point on the circle to make a central angle  $\varphi_c = \angle SOE$ . Select  $\varphi_c$  as the user parameter, then  $h_c$  equals the chord length between  $S$  and  $E$ , given by

$$h_c = |SE| = 2r_s \sin(\varphi_c/2).$$

For a background vertex that might be classified on more than one face, a pair of principal curvatures need be computed for each face. The final curvature at the vertex is the principal curvature that has a maximal absolute value.

#### 3.2.2. Proximity-oriented size values

Finer meshes are required near narrow regions to avoid the generation of elements with high aspect ratios. However, the computation of proximity distances in narrow regions is non-trivial. Cunha *et al.* [14] and Zhu *et al.* [15] proposed to compute distances between all combinations of sampled entities. This kind of scheme involves a large number of unstable geometry computations. Based on the Cartesian mesh, several more efficient algorithms were presented later, such as the wave propagation algorithm by Quadros *et al.* [16], the geometry rasterisation algorithm by Deister *et al.* [9] and Voronoi cell based algorithm by Luo *et al.* [17]. However, because unstructured meshes are preferred in this study, an alternative scheme has to be developed.

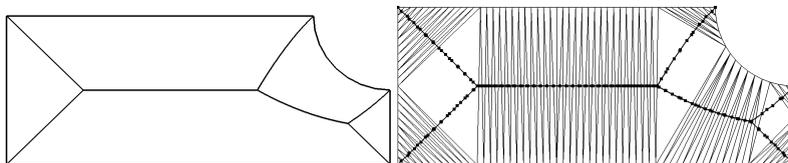


Fig. 4: (a) The medial axis of a domain, and (b) its discrete representation by connecting the circumcenters of neighbouring elements of a Delaunay triangulation.

In [18], we developed a general algorithm that can calculate surface proximity distances by using an unstructured Delaunay background mesh. As shown in Fig. 4b, the lines that connect the circumcenters of neighboring elements of a Delaunay triangulation are the discrete representation of the medial axis. This dual relation has a theoretical proof if the boundary edges of the model are all contained in the Delaunay triangulation [19].

However, in our experiments for the surface mesh generation problems of typical aerodynamic models, it is observed that a simpler algorithm is usually more practical than the general one investigated in [18]. This algorithm

computes a global proximity distance for each face:

$$d_s = 2.0 \times A_s / l_s,$$

where  $A_s$  is the area of the face and  $l_s$  is the circumference of the face. For background vertices classified on this face, their proximity-oriented size values are

$$h_{d1} = d_s / \mu_d,$$

where the user parameter  $\mu_d$  defines the expected number of elements within the proximity distance. Apart from the proximity distance between boundary curves, another proximity feature considered in this study is the curve length, i.e., the distance between the end points of a curve. Because at least one mesh segment is required to be generated in a curve, the size values of a background vertex classified on a curve must be less than the length of the curve. Assuming that  $h_{d0}$  is the smallest length of all curves containing a background vertex, the final proximity-oriented size value of this vertex is

$$h_d = \min(h_{d1}, h_{d0}).$$

### 3.3. Smoothing of the size map

The initial size map usually disobeys Inequality 11; hence, a smoothing procedure is required to guarantee that Inequality 11 is met for all background triangles. In one of the first solutions to this problem, Borouchaki *et al.* [11] corrected the size map to ensure that the H-variants of background edges are less than  $\ln\beta$ . For a triangle, the H-variants defined on its three edges are only three directional derivatives of the linear sizing function. The fact that their values are not greater than  $\ln\beta$  is only a necessary condition that requires  $|\nabla h|$  not being greater than  $\ln\beta$ . Later, Pippa *et al.* [12] suggested correcting the size map according to Inequality 11 so that the gradation of the size map is limited everywhere. However, because the smoothing algorithm proposed by Pippa *et al.* does not attempt to minimise the change of the size map, its result may deviate a lot from the original size map. Persson *et al.* [20] formalised the problem of smoothing a size map to be a Hamilton-Jacobi equation, and many solutions were suggested for different types of background meshes. However, it is doubtful whether this algorithm can be applied in this study because background elements considered here are mainly badly-shaped triangles.

The suggested smoothing algorithm makes a compromise between the algorithm proposed by Borouchaki *et al.* [11] and that by Persson *et al.* [12]. Firstly, the background edges are looped to ensure that their H-variants meet the requirement. Assuming that the sets of background mesh vertices and edges are  $P$  and  $E$ , respectively, the procedure that smoothes the H-variants of background edges attempts to solve an optimisation problem:

$$\begin{aligned} & \min \sum_{p \in P} (h'(p) - h(p))^2 \\ & \text{s.t. } h'_v(e) \leq \ln\beta (\forall e \in E), \\ & \quad h'(p) \leq h(p) (\forall p \in P) \end{aligned} \quad (12)$$

where  $h'_v(e)$  is the H-variant of a mesh edge  $e$  after smoothing. Note that the one-way adjustment only reduces the size values of background vertices. By solving this optimisation problem, the change to the original size map is minimised by a least square method. There are many standard solutions for this optimisation problem. In our implementation, the open-sourced code LPSolve [21] is employed. In practice, only a very small fraction of background elements disobey Inequality 11 after the first stage of smoothing. The background elements that disobey Inequality 11 are put into an array, and for each of them, a correction scheme is executed [12]. Note that the background triangles are defined in three dimensions. Each triangle must be transformed to a new coordinate system that uses the normal vector of the triangle as the Z axis before conducting the correction scheme depicted as below:

(1) Assuming that  $h_0 \leq h_1 \leq h_2$ , the value of  $h_2$  is altered such that  $|\nabla h|^2 = \ln^2(\beta)$ :

$$h'_2 = (-r_b + \sqrt{r_b^2 - 4r_a r_c}) / (2r_a),$$

where

$$r_a = k_{22}$$

$$r_b = (k_{02} + k_{20})h_0 + (k_{21} + k_{12})h_1 = 2k_{02}h_0 + 2k_{12}h_1$$

$$r_c = k_{00}h_0^2 + (k_{01} + k_{10})h_0h_1 + k_{11}h_1^2 - \ln^2(\beta) = k_{00}h_0^2 + 2k_{01}h_0h_1 + k_{11}h_1^2 - \ln^2(\beta)$$

(2) Obviously,  $h'_2$  does not have a real value when  $r_b^2 - 4r_ar_c < 0$ . In this case, we attempt to alter the values of  $h_1$  and  $h_2$  simultaneously such that

$$h'_1 = h'_2 = (-z_b + \sqrt{z_b^2 - 4z_az_c}) / (2z_a),$$

where

$$z_a = k_{11} + k_{12} + k_{21} + k_{22} = k_{11} + 2k_{12} + k_{22} = -(k_{01} + k_{02}) = k_{00}$$

$$z_b = (k_{01} + k_{10} + k_{02} + k_{20})h_0 = 2(k_{01} + k_{02})h_0 = 2k_{00}h_0$$

$$z_c = k_{00}h_0^2 - \ln^2(\beta)$$

$h'_1$  and  $h'_2$  always have a real value since

$$z_b^2 - 4z_az_c = 4k_{00}^2h_0^2 - 4k_{00}(k_{00}h_0^2 - \ln^2(\beta)) = 4k_{00}\ln^2(\beta) \geq 0.$$

### 3.4. User options

An element-sizing specification algorithm attempts to reduce the intensity of user interactions because manual specification of the sizing information is tedious and time-consuming. However, the user priori expertise for the simulation problem is usually very important for preparing a suitable mesh model. Its role is currently irreplaceable in many meshing tasks. Therefore, the proposed algorithm provides the user with options to influence the element-sizing results via the following parameters so that a variety of meshes can be generated to satisfy different simulation needs: (1) Global user parameters  $\varphi_c$ ,  $\mu_d$  and  $\beta$ . (2) A local  $\varphi_c$  value for any curve or face. (3) Local  $\mu_d$  and  $\beta$  values for any face. (4) A predefined size value or function for any geometry point, curve or face.

Only the first group of parameters is mandatory and the other parameters are optional. In most meshing tasks, the user need only set the mandatory parameters. For each background vertex, its initial size value is computed under the combined influence of the calculated geometry features and user parameters:

$$h_a = \max(h_{\min}, \min(h_d, h_c, h_u, h_{\max})),$$

where  $h_c$  and  $h_d$  are curvature and proximity oriented size values, respectively;  $h_u$  is the predefined size value by the user (if any); and  $h_{\min}$  and  $h_{\max}$  are the user parameters which limit the minimal and maximal size values.

## 4. Integration scheme with a surface mesher

Once the size map defined on the background mesh is smoothed, it is ready to provide the interpolation service for mesh generation algorithms. In its simplest form, the interpolation routine inputs the coordinates of a point, then returns the size value defined on that point. Interiorly, this routine takes two steps:

- (1) Search for the background element that contains the point (referred to as a *base element* hereafter);
- (2) Interpolate the size value of this point with the size values of the forming vertices of the base element.

The efficiency of the first step heavily impacts the efficiency of the meshing procedure. Taking a sequential advancing front surface mesher as the example, we will introduce how the proposed algorithm can serve a typical surface mesher efficiently.

### 4.1. The sequential surface mesher

Like most of prevailing surface meshers, our mesher exploits the parametric representation of input surfaces, and can be classified as mapping based. In contrast, some authors suggested generating the mesh in the original physical space [22–24], and the algorithms they adopt are referred to as direct algorithms. Mapping based algorithms can adopt the advancing front technique (AFT) [22–25], Delaunay triangulation [26,27] or a combination of two techniques [28] to mesh the parametric region of a surface. To handle the mapping distortion, the parametric region is covered by a Riemannian metric [26,27] or a normalised matrix [25], where the length of an ideal mesh edge is always one. In the CFD community, the advancing front surface mesher is favored because it provides better control on the element shapes, in particular in the neighborhood of surface boundaries. However, with respect to the timing performance, a Delaunay mesher may run faster by several times than an advancing front mesher.

The surface models used in the real world may be badly parameterized. If no cautions are taken, an advancing front mesher may fail to handle these surfaces, or even if succeed, provide invalid or badly shaped elements. To enhance the robustness, Tremel *et al.* suggested two procedures [29]:

- (1) Calculate the ideal point of a given front in the physical space instead of the parametric space.
- (2) Check the validity of a new element in the physical space instead of the parametric space.

As demonstrated by Tremel *et al.*, these two procedures improve the robustness of their mesher up to a production level. However, it is observed that the two procedures defined in the physical space are time consuming and do slow down their surface mesher [29].

The advancing front mesher adopted in this study is enhanced with the above two procedures as well. In our tests, the procedure that calculates ideal points in the physical space mainly accounts for the slowdown of the surface mesher. To improve the timing performance but not to sacrifice the robustness, we choose to employ this procedure adaptively. In our implementation, the more efficient procedure that calculates ideal points in the parametric space is employed in default, and the improved procedure is employed only when the outcome of the default procedure is not satisfactory.

#### 4.2. The base-element search routine

For a Cartesian background mesh, the time complexity of the base-element search routine is proportional to the depth of the tree that stores the mesh. However, for an unstructured background mesh, a brute-force search procedure consumes  $O(n)$  time in its worst case, where  $n$  is the number of background elements required to be searched. In a mesh generation process, the number of callings for this search routine is several times of the number of mesh nodes to be generated. Therefore, a fast base-element search routine is mandatory.

Because the surface mesher deals with faces individually in parametric spaces, the search routine can be implemented in the two-dimensional parametric space of a face as well. Apart from physical coordinates, each background vertex stores the parametric coordinates in its classification faces. When the surface mesher queries the size value of a point, it inputs the face ID and the parametric coordinates of the point. The base-element search procedure only visits the background elements classified on the input face, and moreover, the search occurs in the two-dimensional parametric space of the input face instead of the three-dimensional physical space.

The timing performance of the search routine can be further improved if we make the most of the spatial locality of the searching callings, i.e., the base elements for two close positions are usually similar or adjacent. A smarter search procedure can be developed by inputting one more parameter that refers to a good enough guess for the base element. In our algorithm, the procedure suggested by Shan *et al.* is adopted [30], which requires no auxiliary structures but the initial guess and neighboring indices of background elements.

For an advancing front algorithm, a front is first selected, and a new node is then created. Obviously, when querying the base element of the new node, either of the base elements of the front nodes can be chosen as the initial guess.

## 5. Results

The proposed algorithm has been integrated into an in-house preprocessor HEDP/Pre developed at our lab [31]. To demonstrate the capabilities of the proposed algorithm, we select two aircraft models: the F6 model [32] and the F16 fighting aircraft (referred to as F16 hereafter). All the tests presented here were conducted on a PC (CPU: 3.5GHz, Memory: 24GB).

### 5.1. F6 aircraft model

The F6 model contains 35 surfaces and 97 curves. Figure 5 presents its continuous geometry, tessellated geometry (i.e., the background mesh), initial size map and smoothed size map, respectively. The background mesh contains 14,406 elements and 7,312 vertices. The size map is initialised and smoothed by using the default user parameters:  $\varphi_c = 5^\circ$ ;  $\mu_d = 2$ ;  $\beta = 1.2$ . Figure 6 presents the surface triangulation generated using the smoothed size map. The mesh includes 113,385 elements and 56,968 nodes. Figure 6b shows the mesh details of the engine intake lip in a strongly magnified resolution. A very fine mesh resolution is observed in the engine intake lip because high curvature features exist there. Moreover, the variation of element scales is controlled very well owing to the proposed smoothing procedure for the size map. We used to configure grid sources to define the size map of this model before. One configuration we adopted frequently for exterior flow simulations includes 31 line sources and 3 point sources.

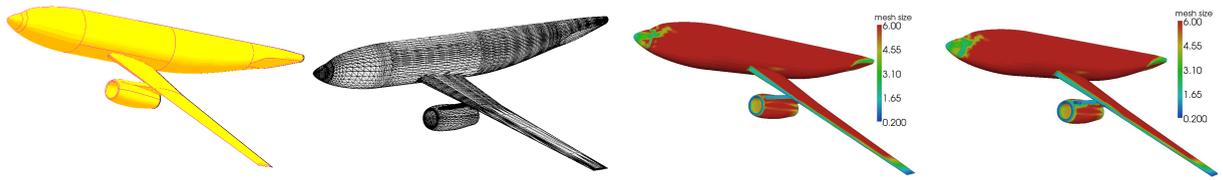


Fig. 5: (a) The continuous geometry, (b) tessellated geometry (i.e., the background mesh), (c) initial size map and (d) smoothed size map of the F6 model.

Although HEDP/Pre provides rather friendly user interface to configure these sources, an expert user still needs at least one hour of interaction time. In contrast, the proposed algorithm only consumed 21.4 seconds to generate the size map, of which the solver of Equation 12 and the computation of parametric coordinates of background vertices dominated, consuming 10.6 and 10.7 seconds, respectively. The surface meshing process consumes 49.9 second based on the computed size map, and the average search depth of querying the size value of a point during the meshing process is 8. Figure 7 draws the distribution of the minimum interior angles ( $\alpha$ ) of surface elements. The focus is on

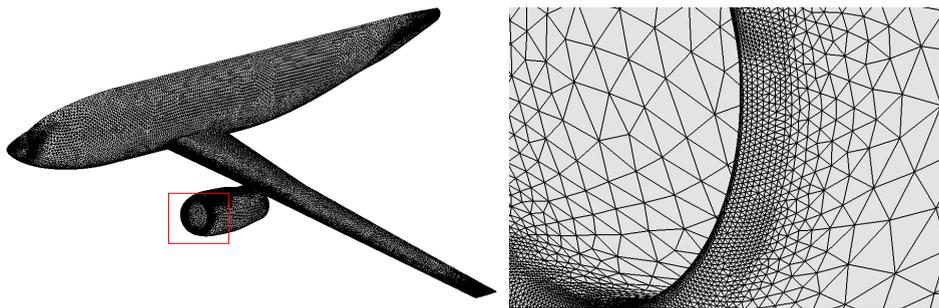


Fig. 6: (a) The surface mesh of the F6 model. (b) The mesh details of the engine intake lip.

the worst elements in numerical simulations. Here, a triangle is classified as a low-quality element if its  $\alpha$  value is smaller than 24 degrees or as a bad element if its  $\alpha$  value is smaller than 12 degrees. For the surface mesh shown in Figure 6a, the numbers of bad elements and low-quality elements are 59 and 345, respectively.

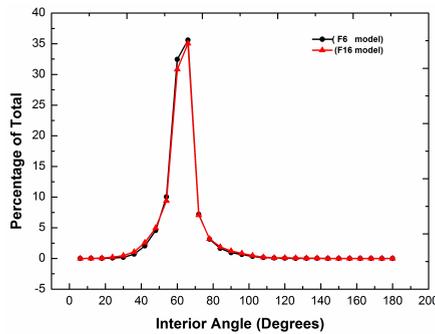


Fig. 7: Distribution of the minimum interior angles ( $\alpha$ ) of surface elements.

### 5.2. F16 aircraft model

The second model is one half of the full loaded F16 fighting aircraft model. A landing gear is configured and introduces more geometry details. Totally 604 surfaces and 1,488 curves are included in this model and abundant curvature and proximity features exist. Figure 8 presents the continuous geometry, tessellated geometry, initial size map and smoothed size map of the F16 aircraft model, respectively. The background mesh contains 47,127 elements and 23,806 vertices. Like the F6 model, the size map is initialised and smoothed under a default configuration of the user parameters. Figure 9a presents the surface triangulation of the F16 aircraft generated using the smoothed

size map. The mesh includes 331,377 elements and 166,536 nodes. In Figure 9a, the mesh details near a proximity feature is enlarged. In Figures 9b and 9c, the closeup views of the mesh details near a missile and the landing gear are presented, respectively.

One grid sources configuration we adopted for this model contains 9 point sources and 122 line sources. In our experience, many hours of manual labours is needed using the graphical interface of HEDP/Pre to configure these grid sources. In contrast, the proposed automatic algorithm consumed only 134.5 seconds to generate the size map, of which the solver of Equation 13 dominated, consuming 107 seconds. Besides, the computation of parametric coordinates of background vertices consumed 26.9 seconds. The surface meshing process consumes 130 second based on this computed size map, and the average search depth of querying the size value of a point during the meshing process is 4. It is evident that the proposed algorithm substantially enhanced the automation of element-sizing specification over the method of using grid sources by reducing the wall-clock time remarkably. Fig. 8 draws



Fig. 8: (a) The continuous geometry, (b) tessellated geometry (i.e., the background mesh), (c) initial size map and (d) smoothed size map of the F6 aircraft model.

the distribution of values of the surface mesh of the F16 aircraft model. 320 and 2,866 bad elements and low-quality elements are included in this mesh, respectively.

An in-depth comparison between the proposed algorithm and existing Cartesian mesh based schemes is currently not available because we have no corresponding codes at hands. However, a simple comparison is possible by borrowing the performance data of two existing Cartesian mesh based schemes from the literature. In [9], another F16

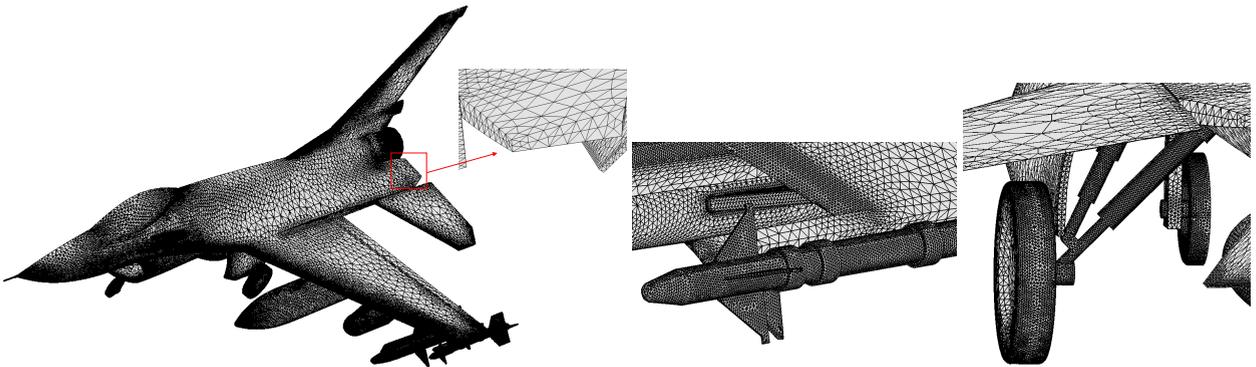


Fig. 9: The surface mesh of the F6 model and the closeup views of some local mesh details.

aircraft model (full loaded but without the landing gear, containing 500 surfaces) with closed geometry complexity to our F16 aircraft model was tested using a Cartesian mesh based scheme. Depending on different user parameters, the number of the background elements (volume Cartesian cells) varies from 349,613 and 1,987,477, and the typical configuration contains about 800,000 cells. In [7], a high-lift configuration containing 138 surfaces was tested to verify another Cartesian mesh based element-sizing specification scheme. Compared with the F16 aircraft model investigated in this study, this configuration is simpler. Even though, a Cartesian background mesh that contains 354,257 cells is generated to define a suitable size map for the surface mesher. In contrast, the unstructured background mesh adopted in this test includes only 47,127 surface triangles. This comparison reveals that, to define a suitable size map for models with similar geometric complexity, the unstructured background mesh required can be coarser than the Cartesian mesh required by one order.

In [9], the sequential algorithm consumed about 300 seconds (CPU: 2.67 GHz) to generate a background mesh that contains nearly one million Cartesian cells. Furthermore, a parallel algorithm was developed in [9], which reduced the time to 75 seconds by using 16 processors. In [7], their algorithm consumed 67 minutes (CPU: SGI Octane

with a 195MHz R10000 processor) to generate the Cartesian background mesh for the high lift configuration. In contrast, the timing performance of our sequential algorithm has no evident advantage. The main timing bottleneck of our algorithm is the solution process of Equation 12. We are presently using LPSolve as a black-box tool to solve Equation 12. It deserves further investigation on the implementation of LPSolve so that we can find ways to speed up our algorithm. Besides, parallelisation is another possible way to reduce the CPU time of our algorithm because the main time-consuming steps of our algorithm, i.e., the solution of Equation 12 and the computation of parametric coordinates of background vertices, are all parallelisable.

To query the size value at a point within the Cartesian mesh based size field [16], the hexahedral cell that contains the point is located first. This cell is a leaf node of the tree structure that represents the Cartesian mesh, and the timing performance of searching this cell is proportional to the depth of the tree, i.e., averagely  $O(\log n/3)$ , where  $n$  is the number of cells. Because the boundary faces and edges of a hexahedral cell may be subdivided by hanging nodes, a classic way to get a continuous interpolation scheme within a hexahedral cell is to subdivide the cell into many tetrahedral elements so that no hanging nodes exist after refinement. Afterwards, the size-query routine need further search one tetrahedral element that exactly contains the point and then interpolate the size value at this point using the size values at the forming points of the tetrahedral element. In its best case (the hexahedral cell has no hanging nodes), a hexahedral cell can be subdivided into 5 or 6 tetrahedral elements; however, in its worst case (hanging nodes exist in the center positions of all boundary edges and faces), this number increases up to several tens (nearly eight times of the number for the hexahedral having no hanging nodes). It is reasonable to guess that more than ten tetrahedral elements perhaps need be searched on average. This search procedure involves many geometric computations and thus dominates the size-query routine in terms of the timing performance.

In our implementation of the size-query routine, two techniques are employed to improve the timing performance. Firstly, because the unstructured background mesh is boundary conforming and the B-rep topology entities of the input surface is mapped to the background mesh, the size-query routine is defined in the 2D parametric space of each individual surface patch. Secondly, because the base elements of two geometrically close points are usually topologically close with each other, a good initial guess of the base element can be given in the size-query routine. Combined with a fast walk-through scheme, each size-query calling usually visits several background triangles on average. For instance, for the F6 and F16 models tested in this article, about 8 and 4 background triangles are visited for each size-query calling on average. Nevertheless, the Cartesian mesh based size-query routine may need visit more background tetrahedra to interpolate the size value at a point.

## 6. Conclusions

Surface mesh generation is a time-consuming step when complex aerodynamics models are considered. One main performance bottleneck lies in the element-sizing specification procedure. The conventional grid sources based scheme involves intensive manual labours. In this study, an automatic scheme is proposed for the element-sizing specification of unstructured surface mesh generation. Different with existing Cartesian mesh based schemes, the proposed algorithm adopts unstructured triangular mesh as the background mesh. Experiments of complicated aerodynamics configurations show that our algorithm can automatically produce a suitable size map in minutes. In contrast, a conventional grid sources based scheme may need many hours by a tedious manual interaction process.

Due to the topological flexibility of unstructured meshes, the proposed algorithm generates a far coarser background mesh than existing Cartesian mesh based schemes for the problem of unstructured surface mesh generation. Nevertheless, if volume meshing is considered, the surface background mesh is not suitable any more. A possible way is to use the surface background mesh as inputs to construct a Delaunay volume mesh. Whether the Delaunay background mesh is more flexible than the Cartesian mesh needs further analysis.

## Acknowledgements

The authors appreciate the joint support for this project by the National Natural Science Foundation of China (Grant No. 11172267, 10872182 and 61100160), Zhejiang Provincial Natural Science Foundation (Grant No. Y1110038). The second author acknowledges the joint support from Zhejiang University and China Scholarship Council for his visiting research at Swansea University, UK.

## References

- [1] T. J. Baker, Mesh generation: Art or science?, *Progress in Aerospace Sciences* 41 (2005) 29 – 63.
- [2] N. Weatherill, O. Hassan, K. Morgan, J. Jones, B. Larwood, K. Sorenson, Aerospace simulations on parallel computers using unstructured grids, *International Journal for Numerical Methods in Fluids* 40 (2002) 171–187.
- [3] R. Löhner, A 2nd generation parallel advancing front grid generator, in: *Proceedings of the 21st International Meshing Roundtable*, 2013, pp. 457–474.
- [4] J. Chen, D. Zhao, Z. Huang, Y. Zheng, D. Wang, Improvements in the reliability and element quality of parallel tetrahedral mesh generation, *International Journal for Numerical Methods in Engineering* 92 (2012) 671–693.
- [5] J. Chen, D. Zhao, Y. Zheng, Z. Huang, J. Zheng, Fine-grained parallel algorithm for unstructured surface mesh generation, in: *Proceedings of the 22nd International Meshing Roundtable*, 2014, pp. 559–578.
- [6] K. Shimada, Current issues and trends in meshing and geometric processing for computational engineering analyses, *Journal of Computing and Information Science in Engineering* 11 (2011) 1–13.
- [7] L. A. Kania, S. Z. Pirzadeh, A geometrically-derived background function for automated unstructured mesh generation, in: *Proceedings of the 17th AIAA Computational Fluid Dynamics Conference*, 2005, pp. 2005–5240.
- [8] S. Z. Pirzadeh, Advanced unstructured grid generation for complex aerodynamic applications, *AIAA journal* 48 (2010) 904–915.
- [9] F. Deister, U. Tremel, O. Hassan, N. P. Weatherill, Fully automatic and fast mesh size specification for unstructured mesh generation, *Engineering with Computers* 20 (2004) 237–248.
- [10] Y. Zheng, N. P. Weatherill, E. A. Turner-Smith, An interactive geometry utility environment for multi-disciplinary computational engineering, *International Journal for Numerical Methods in Engineering* 53 (2002) 1277–1299.
- [11] H. Borouchaki, F. Hecht, P. J. Frey, Mesh gradation control, *International Journal for Numerical Methods in Engineering* 43 (1998) 1143–1165.
- [12] S. Pippa, G. Caligiana, Gradh-correction: guaranteed sizing gradation in multi-patch parametric surface meshing, *International Journal for Numerical Methods in Engineering* 62 (2005) 495–515.
- [13] S. Dey, M. S. Shephard, J. E. Flaherty, Geometry representation issues associated with p-version finite element computations, *Computer Methods in Applied Mechanics and Engineering* 150 (1997) 39–55.
- [14] A. Cunha, S. Canann, S. Saigal, Automatic boundary sizing for 2d and 3d meshes, in: *Proceedings of the AMD Trends in Unstructured Mesh Generation*, ASME, 1997, pp. 65–72.
- [15] J. Zhu, T. D. Blacker, R. Smith, Background overlay grid size functions., in: *Proceedings of the 11th International Meshing Roundtable*, 2002, pp. 65–73.
- [16] W. R. Quadros, V. Vyas, M. Brewer, S. J. Owen, K. Shimada, A computational framework for automating generation of sizing function in assembly meshing via disconnected skeletons, *Engineering with Computers* 26 (2010) 231–247.
- [17] X.-J. Luo, M. S. Shephard, L.-Z. Yin, R. M. OBara, R. Nastasi, M. W. Beall, Construction of near optimal meshes for 3d curved domains with thin sections and singularities for p-version method, *Engineering with Computers* 26 (2010) 215–229.
- [18] L. Xie, J. Chen, Y. Liang, Y. Zheng, Geometry-based adaptive mesh generation for continuous and discrete parametric surfaces, *Journal of Information & Computational Science* 9 (2012) 2327–2344.
- [19] P. Frey, P. George, *Mesh Generation: Application to Finite Elements*, HERMES Science Publishing, 2000.
- [20] P.-O. Persson, Mesh size functions for implicit geometries and pde-based gradient limiting, *Engineering with Computers* 22 (2006) 95–109.
- [21] M. Berkelaar, K. Eikland, P. Notebaert, Package 'Ipsolve' (2014).
- [22] K. Nakahashi, D. Sharov, Direct surface triangulation using the advancing front method, in: *Proceedings of the 12th AIAA Computational Fluid Dynamics Conference*, 1995, pp. 95–1686.
- [23] T. Lan, S. Lo, Finite element mesh generation over analytical curved surfaces, *Computers & Structures* 59 (1996) 301–309.
- [24] G. Foucault, J.-C. Cuillière, V. François, J.-C. Léon, R. Maranzana, An extension of the advancing front method to composite geometry, in: *Proceedings of the 16th International Meshing Roundtable*, 2008, pp. 287–314.
- [25] P. J. Surface grid generation. In: Thompson JF, Soni BK, Weatherill NP (eds) *Handbook of Grid Generation*, CRC Press, Inc., Boca Raton, FL, USA, Chapter 19, 1999.
- [26] H. Borouchaki, P. L. George, F. Hecht, P. Laug, E. Saltel, Delaunay mesh generation governed by metric specifications. part i. algorithms, *Finite Elements in Analysis and Design* 25 (1997) 61–83.
- [27] H. Borouchaki, P. L. George, B. Mohammadi, Delaunay mesh generation governed by metric specifications part ii. applications, *Finite Elements in Analysis and Design* 25 (1997) 85–109.
- [28] P. J. Frey, H. Borouchaki, P.-L. George, 3d delaunay mesh generation coupled with an advancing-front approach, *Computer Methods in Applied Mechanics and Engineering* 157 (1998) 115–131.
- [29] U. Tremel, F. Deister, O. Hassan, N. P. Weatherill, Automatic unstructured surface mesh generation for complex configurations, *International Journal for Numerical Methods in Fluids* 45 (2004) 341–364.
- [30] J. Shan, Y. Li, Y. Guo, Z. Guan, A robust backward search method based on walk-through for point location on a 3d surface mesh, *International Journal for Numerical Methods in Engineering* 73 (2008) 1061–1076.
- [31] L. Xie, Y. Zheng, J. Chen, J. Zou, Enabling technologies in the problem solving environment hdp, *Communications in Computational Physics* 4 (2008) 1170–1193.
- [32] Nasa dlr-f6 geometry.jun-08-2013, URL: <http://aaac.larc.nasa.gov/tsab/cfdlarc/aiaa-dpw/Workshop2/DLR-F6-geom.html> (2013).