

FULLY ANISOTROPIC UNSTRUCTURED GRID GENERATION FOR AERODYNAMIC SIMULATIONS

V. SELMIN AND E. PELIZZARI¹

¹Alenia Aeronautica S.p.A., Corso Marche 41, 10146 Torino, Italy

SOMMARIO

La relazione descrive un metodo per la generazione di griglie 3D anisotrope per domini di forma arbitraria. La spaziatura sulla griglia di contorno è calcolata tramite l'analisi delle curvature principali e delle direzioni principali di curvatura delle superfici di contorno. La spaziatura all'interno del dominio di calcolo è ottenuta come l'interpolazione della spaziatura sui contorni sfruttando una griglia di appoggio. Esempi che illustrano le prestazioni del metodo sono presentate.

ABSTRACT

This paper describes a fully automatic 3D anisotropic mesh generation method for domains of arbitrary shape. The spacing of the boundary mesh is computed by the analysis of the principal curvatures and directions of the boundary surfaces. The spacing in the domain is obtained by interpolation of the spacing at the boundaries on a suitably constructed background mesh. Examples which illustrate the performance of the proposed methodology are presented.

1. INTRODUCTION

The availability of ever increasing computational resources and of efficient and robust CFD solution algorithms allows the numerical simulation of very complex flow fields. The generation of the grid is probably the bottleneck in terms of human resources required to complete a CFD simulation for domains of complex shape such as, for example, an aircraft configuration. The construction of a grid which minimizes the number of nodes required to compute a flow field within a prescribed tolerance level, or even of a grid which approximates the geometry of the domain with sufficient accuracy to allow the computation of an initial coarse solution in an adaptive solution process, remains in fact a difficult and time consuming task with the commonly available software.

Among the many tasks which require substantial user input in grid generation, that of prescribing the mesh spacing for a complex domain is probably the most demanding in terms of human resources required. We here describe a three-dimensional anisotropic unstructured grid generation algorithm which instead requires a very limited user input even for very complex geometries since the mesh spacing is automatically defined as a function of the curvature of the boundary of the domain. We follow the widely adopted approach [3, 5] which consists in augmenting the domain to be meshed with a metric field $\mathbf{M}(x)$ (i.e. to define a Riemannian structure on the domain) and in requiring that all the mesh edges have the same length with respect to $\mathbf{M}(x)$ (typically unit length edges lengths in $\mathbf{M}(x)$ are chosen). This leads to an anisotropic control of the mesh spacing which means to control not only the mesh element size but also their aspect ratio. This approach decreases significantly the nodes number used when the flow field or the geometry of the domain are characterized by anisotropic features

Following the ideas introduced in [4], the metric field on the boundary is computed according to the principal curvatures and directions of the boundary surface. This approach is, however, not appropriate if the surface curvature

is very small, since it leads to excessive mesh spacings. The generally adopted approach to deal with “nearly-flat” surfaces is to limit the spacing with a threshold value prescribed in an isotropic manner. As a consequence all the informations regarding the anisotropy of the mesh are completely lost in regions of small surface curvature and much of the potential savings allowed by anisotropic unstructured meshes are not fully realized in practice. This situation is typical in many applications such as, for example, high aspect ratio wings and turbomachinery bladings. An original anisotropic treatment for “nearly-flat” surfaces has been developed in this work. After establishing the metric field distribution on the boundary, we proceed with the surface mesh generation using an approach, sometimes referred to as the “indirect approach”, which consists in generating the mesh in the parametric domain associated to the surface and in mapping the resulting mesh onto the real surface. Using the surface mesh as input data it is then possible to build a background grid, which is used to interpolate the value of the boundary metric field at any other points. The computational domain is then discretized by using a 3D anisotropic mesh generator based on Front Advancing method.

In order to build hybrid grids suited to solve Navier-Stokes equations, layers of prisms are introduced at the skin boundary, by using a spring analogy based node movement algorithm. Starting with layers of null thickness, a force proportional to the required displacement is applied to the skin nodes which result to push on the grid formed by tetrahedra, leaving then room to introduce prisms.

The anisotropic grid generation approach has been applied with success to the computation of inviscid and viscous flows around complex aeronautical configuration configurations. In addition to provide unstructured/hybrid grids of good quality formed by a relatively small number of elements, it allows to control some desirable properties of the grid. As an example, it is possible by using this method to maintain the chordwise number of nodes along the wing span almost constant, which is hard to obtain by using an isotropic unstructured mesh generation process.

2. RIEMANNIAN METRIC

An anisotropic control of the mesh spacing has been chosen for this work, which means that we will be able to control not only the mesh elements size but also their aspect ratio. This is realized by requiring that all the edges of the mesh are of unit length with respect to a suitable defined metric field $\mathbf{M}(x)$, where x is a point of \mathbb{R}^d ($d = 2, 3$). $\mathbf{M}(x)$ is a $(d \times d)$ symmetric positive definite matrix. For example, in two dimensions, we consider

$$\begin{pmatrix} a(x) & b(x) \\ b(x) & c(x) \end{pmatrix}, \quad (1)$$

such that $a > 0$, $c > 0$ and $ac - b^2 > 0$, for $a, b, c \in \mathbb{R}$. If the field of tensors thus defined is known, it induces a Riemannian structure over \mathbb{R}^d .

The length with respect to \mathbf{M} of an arbitrary mesh edge v is approximated as

$$\|v\|_{\mathbf{M}} = \sqrt{v^T \mathbf{M}(\hat{x}) v}, \quad (2)$$

where \hat{x} is the midpoint of edge v . When $\mathbf{M}(x)$ is independent of the position x , we again find the classical Euclidean case, otherwise we are in the Riemannian context.

The matrix $\mathbf{M}(x)$ can be factorized as

$$\mathbf{M}(x) = \mathbf{R}(x) \mathbf{\Lambda}(x) \mathbf{R}(x)^T, \quad (3)$$

where

$$\mathbf{R}(x) = [r_1(x), \dots, r_n(x)] \quad (4)$$

is the orthogonal matrix of the normalized (right) eigenvectors of $\mathbf{M}(x)$ and

$$\mathbf{\Lambda}(x) = \text{diag}(\gamma_1(x), \dots, \gamma_n(x)) \quad (5)$$

is the diagonal matrix of real eigenvalues $\gamma_i > 0$, $i = 1, \dots, n$.

We can associate to $\mathbf{M}(x)$ a transformation matrix $\mathbf{T}(x)$ defined as

$$\mathbf{T}(x) = \sqrt{\mathbf{M}(x)} = \mathbf{R}(x) \sqrt{\mathbf{\Lambda}(x)} \mathbf{R}(x)^T, \quad (6)$$

and we therefore can express $\mathbf{M}(x)$ in terms of $\mathbf{T}(x)$ as

$$\mathbf{M}(x) = \mathbf{T}(x)^T \mathbf{T}(x) = \mathbf{T}(x) \mathbf{T}(x)^T. \quad (7)$$

The transformation matrix \mathbf{T} maps an edge v of unit length with respect to \mathbf{M} into an edge $w = \mathbf{T}v$ of unit length in the usual Euclidean norm. We have in fact that

$$\|w\| = \sqrt{w^T w} = \sqrt{v^T \mathbf{T}^T \mathbf{T} v} = \sqrt{v^T \mathbf{M} v} = \|v\|_{\mathbf{M}}.$$

3. GEOMETRICAL CHARACTERISTICS OF AN ELEMENT

The geometrical characteristics of an element can be defined in terms of a set of n orthogonal directions α_i and n associated element sizes δ_i , where n is the number of dimensions. The transformation matrix \mathbf{T} can then be regarded as the result of combining n scaling operations, with factor $1/\delta_i$, in each direction α_i , i.e.

$$\mathbf{T} = \sum_{i=1}^n \frac{1}{\delta_i} \alpha_i \alpha_i^T. \quad (8)$$

By rewriting Eq. (6) as

$$\mathbf{T} = \sum_{i=1}^n \sqrt{\gamma_i} r_i r_i^T \quad (9)$$

and by comparing with Eq. (8) we therefore have that

$$\delta_i = \frac{1}{\sqrt{\gamma_i}}, \quad \alpha_i = r_i. \quad (10)$$

The spacing $s(\beta)$ in the generic direction defined by the unit vector β is given by

$$s(\beta) = \|\mathbf{T}\beta\|^{-1}.$$

4. MESH SPACING CONTROL ON THE BOUNDARY

The objective of surface mesh generation is to construct a triangulation in which the maximum distance from the real surface is below a prescribed threshold value — this is a useful and practical way to arrive at meshes which approximate the real surface as accurately as required. In practice, we have however to construct a metric field \mathbf{M}_3 on the real surface Σ so that the above criterion is satisfied by unitary (in \mathbf{M}_3) mesh edges, and a corresponding metric \mathbf{M}_2 in the parametric space Ω which will be used as input data by the surface mesh generator. The manner in which \mathbf{M}_3 and \mathbf{M}_2 are constructed is described in the following.

4.1. Mesh spacing based on local curvature

Let Σ be a surface defined by the parametrization $r(u, v) \in \mathbb{R}^3$, P a point of Σ of coordinates $r_P(u_P, v_P)$, n the normal to Σ at P , and t the unit tangent vector along some curves on Σ passing at P . By locally approximating in the neighborhood of P the curve by its osculating circle, the ratio ε between δ , the maximal gap between the osculating circle and its chord of arc, and the osculating circle radius ρ , is given by

$$\varepsilon \equiv \frac{\delta}{\rho} = 1 - \sqrt{1 - (\lambda/2)^2}, \quad (11)$$

where $\lambda = h_t/\rho$, and $h_t = \lambda\rho$ is the chord length. The parameter λ is therefore given as a function of ε according to

$$\lambda \equiv \frac{h_t}{\rho} = 2\sqrt{\varepsilon(2 - \varepsilon)}. \quad (12)$$

The above equation is employed to compute the chord length h_t for a prescribed value of ε — a parameter which measures how closely the chord approximates the real curve.

The geometrical characteristics of a surface element may be obtained by applying the above analysis in the surface principal directions, see e.g. [1], i.e. the directions in which the surface curvature assumes its minimum and maximum values. The two principle directions, $d_i \in \mathbb{R}^3$, can be defined as

$$d_i = \frac{\mathbf{J}(r)\dot{v}_i}{\sqrt{\dot{v}_i^T \mathbf{G} \dot{v}_i}} \quad i = 1, 2, \quad (13)$$

where $\mathbf{J}(r)$ is the Jacobian matrix of the mapping $r = r(u, v)$, i.e.

$$\mathbf{J}(r) = \left[\frac{\partial r}{\partial u}, \frac{\partial r}{\partial v} \right], \quad (14)$$

$\mathbf{G} = \mathbf{J}^T \mathbf{J}$ is the first fundamental matrix of the surface Σ , and $v_i \in \mathbb{R}^2$ the principal directions in the parametric space $u - v$. Notice that, by construction, the direction d_1 and d_2 are orthogonal. The element sizes h_i associated to the directions d_i are, then, computed as

$$h_i = 2\rho_i \sqrt{\varepsilon(2 - \varepsilon)} = \frac{\sqrt{\varepsilon(2 - \varepsilon)}}{\kappa_i}, \quad (15)$$

where ε is a user prescribed parameter, ρ_i the principal radii of curvature and κ_i the principal curvature. Finally, the metric tensor $\mathbf{M}_3(P)$ associated to the surface is computed according to

$$\mathbf{M}_3(P) = \sum_{i=1}^3 \frac{1}{h_i^2} d_i d_i^T, \quad (16)$$

where $d_3 = n$ and $h_3 = \min(h_1, h_2)$. The metric $\mathbf{M}_2(P)$ can be simply expressed as

$$\mathbf{M}_2(P) = \mathbf{J}(r)^T \mathbf{M}_3(P) \mathbf{J}(r), \quad (17)$$

where \mathbf{J} is defined in Eq. (14).

4.2. Treatment of “nearly-flat” surfaces

The values of h_i have to be occasionally limited, in order to avoid too large or too small elements. If h_1 and h_2 denote the smallest and the largest spacings, respectively, the limited spacings \tilde{h}_i are defined as

$$\begin{aligned} \tilde{h}_1 &= \min(\max(h_1, h_{1,m}), h_{1,M}) \\ \tilde{h}_2 &= \min(\max(h_2, h_{2,m}), h_{2,M}, s_M h_1), \end{aligned} \quad (18)$$

where $h_{i,m}$ and $h_{i,M}$ are the minimum and maximum value that h_i can assume. The parameter s_M represents the maximum value of element stretching. By choosing $s_M = 1$, an isotropic element will be formed.

The quantities $h_{i,m}$ and $h_{i,M}$ are defined as follows. Let denote by l_u and by l_v , the length of the curves represented parametrically by equations $u = [u_P, v(s)]^T$ and $u = [u(s), v_P]^T$, respectively. The spacing h_u and h_v are computed as a fractional part of the lengths l_u and l_v , i.e.

$$h_u = \frac{l_u}{n_u}, \quad h_v = \frac{l_v}{n_v}, \quad (19)$$

where the parameters n_u and n_v ($n_u, n_v \geq 1$) are constants (integer) defined on Σ . The quantities $h_{i,M}$ are computed by using the Euler theorem

$$\begin{pmatrix} \frac{1}{h_u^2} \\ \frac{1}{h_v^2} \end{pmatrix} = \sum_{i=1}^2 \frac{1}{h_{i,M}^2} \cos^2 \begin{pmatrix} \varphi_{i,u} \\ \varphi_{i,v} \end{pmatrix}, \quad (20)$$

where $\varphi_{i,u}$ and $\varphi_{i,v}$ are the angles formed by the principal direction d_i with the tangent vectors to the curves $\mathbf{u} = [u_P, v(t)]^T$ and $\mathbf{u} = [u(t), v_P]^T$, denoted by t_u and t_v , respectively. In addition, $h_{i,m}$ are defined as a fraction of $h_{i,M}$ or by taking a constant value on the surface.

When the geometric model is too complex, the computation of $h_{i,M}$ is performed by using the concept of background surface spacing characteristics, which are built by the knowledge of the characteristic dimensions of the object to be discretized and of the tangent vectors to the surface.

When the surface is a plane ($\rho_i = \infty$) or when the principal curvatures are equal, the principal directions of normal curvature are undefined. However the directions t_u and t_v still exist and can be chosen as directions d_i , if $t_u \cdot t_v = 0$. In this case the spacings \tilde{h}_i associated to the directions d_i are, respectively, h_u and h_v . if $t_u \cdot t_v \neq 0$, d_1 is chosen coincident with the direction which specifies the minimum spacing and $d_2 = d_1 \times n$. In this case $\tilde{h}_1 = \min(h_u, h_v)$ and \tilde{h}_2 is computed by using the equation (20).

4.3. Metric intersection

A prerequisite for the grid generation algorithm is the specification of only one metric at each point. However, the metrics are multi-defined on the boundary curves and at each corner. Thus, in these points, a metric intersection is requested.

Let P be a point in the space at which two different metrics exist, say $\mathbf{M}_a(P)$ and $\mathbf{M}_b(P)$, defined as

$$\mathbf{M}_a(P) = \sum_{i=1}^3 \frac{1}{\delta_i^2} \alpha_i \alpha_i^T \quad \mathbf{M}_b(P) = \sum_{i=1}^3 \frac{1}{\gamma_i^2} \beta_i \beta_i^T. \quad (21)$$

The directions ξ_i of the intersection metric are chosen coincident with the directions associated to the metric matrix which specifies the minimum spacing, i.e. $\xi_i = \alpha_i$. The new metric $\mathbf{M}_{new}(P)$ is defined as

$$\mathbf{M}_{new}(P) = \sum_{i=1}^3 \frac{1}{h_i^2} \xi_i \xi_i^T, \quad (22)$$

where

$$h_i = \min \left(\delta_i, \frac{1}{\|\mathbf{T}_b \xi_i\|^{-1}} \right). \quad (23)$$

4.4. Metric smoothing

Due to the fact that the metrics on the boundary curves and at each corner are, in general, multi-defined, discontinuity or large gradient may appear in the metric field, which can lead to the generation of bad quality elements.

A spring analogy like algorithm is used to propagate and regularize the metric field. It is first assumed that each node i is connected to each adjacent node j by a fictitious spring under the generalized force \mathbf{F}_{ij} defined by

$$\mathbf{F}_{ij} = K_{ij}(\mathbf{T}_j^{-1} - \mathbf{T}_i^{-1}), \quad (24)$$

where the scalar K_{ij} is the spring constant, \mathbf{T}_i and \mathbf{T}_j are the transformation matrices associated to nodes i and j . The spring constant can be expressed as

$$K_{ij} = K_{ij}^{sm} K_{ij}^{lg}, \quad (25)$$

where

$$K_{ij}^{sm} = \frac{A_{e1}^{ij} + A_{e2}^{ij}}{A_i} \quad (26)$$

is the spring constant associated to a Laplacian smoothing algorithm and

$$K_{ij}^{lg} = \frac{1}{\alpha} \sqrt{\frac{(x_j - x_i)^T \mathbf{M}_{ij} (x_j - x_i)}{(x_j - x_i)^T (x_j - x_i)}} \quad (27)$$

is related to the locally required grid spacing. A_{e1}^{ij} and A_{e2}^{ij} are the areas of the two elements adjacent to side ij , A_i is the area of the elements surrounding node i , α a scaling factor, which is taken equal to the maximum value of K_{ij}^{lg} (with $\alpha = 1$) over the patch of elements surrounding node i , \mathbf{M}_{ij} the metric matrix along side ij .

The resulting metric field is the solution of the equilibrium system for each vertex i :

$$\sum_{j \in \mathcal{K}_i} \mathbf{F}_{ij} = 0, \quad (28)$$

where \mathcal{K}_i is the set of nodes surrounding i . In this work, a few iterations of the following pointwise scheme is actually used to smooth the metric field:

$$\begin{aligned} \mathbf{T}_{i,(p)}^{-1} &= \mathbf{T}_{i,(p-1)}^{-1} + \sum_{j \in \mathcal{K}_i} \mathbf{F}_{ij,(p-1)} \\ &= \mathbf{T}_{i,(p-1)}^{-1} + \sum_{j \in \mathcal{K}_i} K_{ij} (\mathbf{T}_{j,(p-1)}^{-1} - \mathbf{T}_{i,(p-1)}^{-1}). \end{aligned} \quad (29)$$

5. MESH SPACING CONTROL IN THE VOLUME

The mesh spacing in the volume is controlled through a background grid, which is used to interpolate the boundary value of the metric field at any other point of the domain to be discretized. The procedure in order to build the background grid is described in the following.

The process starts constructing the Delaunay triangulation [7, 8] of the points which discretize the skin surface and the external boundaries. The circumcenters of the tetrahedra that have at least one node on the skin and that are not inside the body are computed. An “average” direction $\bar{\mathbf{n}}$ is obtained by solving the minimization problem

$$\min \sum_{j=1}^{n_j} \left(1 - \bar{\mathbf{n}} \cdot \frac{\mathbf{v}_{ij}}{\|\mathbf{v}_{ij}\|} \right)^2, \quad (30)$$

where \mathbf{v}_{ij} are the vectors that connect the node i on the skin with the set of circumcenters j associated to the elements which have node i as vertex, and n_j is the number of circumcenters belonging to the set. Then, the vector \mathbf{v}_{il} is constructed connecting the node i with the local node l which is located at the intersection of the direction $\bar{\mathbf{n}}$ with the surface formed by the n_j circumcenters. The procedure is repeated by taking the new layer of nodes l as external boundary until a maximum of iterations is reached. This results in the construction of a succession of nodes layers which become closer and closer to the skin surface.

The background grid is defined by building the Delaunay triangulation of the nodes on the skin, on the external boundary and on additional layers. The transformation matrix \mathbf{T}_l at the node l is computed as

$$\mathbf{T}_l = ((1 - \omega)\mathbf{T}_i^{-1} + \omega\mathbf{T}_\infty^{-1})^{-1} \quad \omega \in [0, 1], \quad (31)$$

where \mathbf{T}_∞ is the transformation matrix based on the element characteristics at the external boundary, \mathbf{T}_i the transformation matrix associated to the skin of nodes i and ω a parameter defined as

$$\omega = f(\|\mathbf{v}_{il}\|, r_\infty, b) = (1 - b) \left(\frac{\|\mathbf{v}_{il}\|}{r_\infty} \right)^2 + b \frac{\|\mathbf{v}_{il}\|}{r_\infty}, \quad (32)$$

where r_∞ is the distance from the skin to the external boundary and the coefficient b the slope of the function at its origin.

6. HYBRID MESH GENERATION

In the case of the simulation of high Reynolds number viscous flows, it is appropriate, knowing the physics of boundary layers, to consider a form of a priori adaptation to reflect the difference in gradients in flowfield variables across a boundary layer as compared with those along the boundary layer in the direction of the flow. If such an approach is followed then elements with high aspect ratios are required. This leads to build quadrilaterals in 2D and prisma in 3D within some regions of the computational domain, in particular close to solid boundaries, by using the method of advancing layers (or advancing normals). The basic approach amounts to growing layers of elements by advancing along lines which are approximatively normal to the boundary. These layers of elements are grown until a specified distance from the boundary is reached. After this, the domain is filled with triangles in 2D and tetrahedra in 3D by using standard approaches.

6.1. Basic algorithm

Let denote by \mathbf{x}_i^k the position of node i on the layer k , its new position on layer $k + 1$ may be computed according to the formula

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + d_i^k \mathbf{n}_i^k \quad (33)$$

where \mathbf{n}_i^k and d_i^k represent the normal vector and the displacement at node i with respect to layer k , respectively. The vector \mathbf{n}_i^k is obtained from the knowledge of the normal vector \mathbf{n}_e^k related to the set \mathcal{E}_i^k of elements belonging to the patch \mathcal{P}_i^k of elements surrounding node i by solving the minimisation problem

$$\min \sum_{e=1}^{n_e} \left(1 - \mathbf{n}_i^k \cdot \frac{\mathbf{n}_e^k}{\|\mathbf{n}_e^k\|} \right)^2 \quad (34)$$

where n_e is the number of elements that belong to the set. In general, the normal vector is subsequently smoothed by using a few iterations of a Laplacian smoothing procedure

$$\mathbf{n}_i^{k,l+1} = \mathbf{n}_i^{k,l} + 0.5 \omega_i^k \sum_{j \in \mathcal{N}_i^k} K_{ij} (\mathbf{n}_j^{k,l} - \mathbf{n}_i^{k,l}) \quad (35)$$

