ORIGINAL ARTICLE
DOI: 10.5433/1679-0375.2021v42n1p29

Two-dimensional mesh generator and quality analysis of elements on the curvilinear coordinates system

Gerador de malhas bidimensionais e análise da qualidade dos elementos no sistema de coordenadas generalizadas

Gustavo Taiji Naozuka[1]; Neyva Maria Lopes Romeiro[2];
Alan Salvany Felinto[3]; Paulo Laerte Natti[4]; Eliandro Rodrigues Cirilo[5]

Abstract

Through mathematical models, it is possible to turn a problem of the physical domain to the computational domain. In this context, the paper presents a two-dimensional mesh generator in generalized coordinates, which uses the parametric linear spline method and partial differential equations. The generator is automated and able to treat real complex domains and, consequently, more realistic problems. However, there is a possibility that lower quality elements may be introduced into the computational mesh. Thus, metrics are investigated that identify elements considered to be of lower quality. Experiments are carried out to verify the efficiency of the adopted metrics, considering meshes with single block, and multi-blocks. According to the experiments, the work allowed the detection of elements of lower quality, contributing to the realization of an adequate modeling of geometries.

Keywords: Automated two-dimensional mesh generator. Parametric linear spline. Generalized coordinates. Python language.

Resumo

Por meio de modelos matemáticos é possível transformar um problema do domínio físico para o domínio computacional. Neste contexto, o presente trabalho apresenta um gerador de malhas bidimensionais em coordenadas generalizadas, que utiliza o método *Spline* linear parametrizado e equações diferenciais parciais. O gerador é automatizado e capaz de tratar domínios complexos e, consequentemente, mais realísticos. Entretanto, existe a possibilidade de elementos de menor qualidade serem introduzidos na malha computacional. Assim, investigam-se métricas que identificam elementos considerados de menor qualidade. Experimentos são efetuados para verificar a eficiência das métricas adotadas, considerando malhas com um bloco e multiblocos. De acordo com os experimentos, o trabalho permitiu a detecção de elementos de menor qualidade, contribuindo para a realização de uma modelagem adequada de geometrias.

Palavras-chave: Gerador de malhas bidimensionais automatizado. *Spline* linear parametrizado. Coordenadas generalizadas. Linguagem *Python*.

[1] PhD student, National Laboratory for Scientific Computing, Petrópolis, RJ, Brazil, E-mail: naozuka@lncc.br
[2] Prof. Dr., Dept. of Mathematics, CCE/UEL, Londrina, PR, Brazil, E-mail: nromeiro@uel.br
[3] Prof. Dr., Dept. of Computer Science, CCE/UEL, Londrina, PR, Brasil, E-mail: alan@uel.br
[4] Prof. Dr., Dept. of Mathematics, CCE/UEL, Londrina, PR, Brazil, E-mail: plnatti@uel.br
[5] Prof. Dept. of Mathematics, CCE/UEL, Londrina, PR, Brazil, E-mail: ercirilo@uel.br

Introduction

The modeling and simulation of natural phenomena using differential equations is an important tool for science. However, to represent physical structures, that is, the domain to be studied in a computational environment, the use of simpler data organization structures, such as matrices and vectors, tend not to be able to realistically represent the object of study.

Different techniques have been used for the representation of domains of irregular regions, such as structured and unstructured meshes, generalized or hybrid meshes, among other techniques, that have advantages and disadvantages in terms of flexibility and ability to represent objects under study (THOMPSON; WARSI; MASTIN, 1985; THOMPSON *et al*., 1997; THOMPSON *et al*., 1998; MALISKA, 2004; CIRILO; BORTOLI, 2006; KOOMULLIL; SONI; SINGHR, 2008; LAIPING *et al*., 2013; SAITA *et al*., 2017; BELINELLI *et al*., 2020; MAGANIN *et al*., 2020; ROMEIRO *et al*., 2021).

In this work, we looked for a simplified way to build a structured mesh for a complex domain, which was possible through a change in the coordinate system. This process allowed to convert a complex domain into a set of easily manipulated data MALISKA, 2004).

Mathematically, any geometry, described in Cartesian system, can be transformed into a generalized system, allowing better adaptation in computational modeling (CIRILO; BORTOLI, 2006; MALISKA, 2004; ROMEIRO *et al*., 2011; PARDO *et al*., 2012; SAITA *et al*., 2017; ROMEIRO *et al*., 2017; CIRILO *et al*., 2018).

In this context, this work describes the creation and implementation of a two-dimensional mesh generator using transformation metrics in generalized coordinates. This procedure generates a domain mapped for mathematical manipulations, which describes any physical object, from a finite set of points.

The mesh generator is encoded in *Python* and uses the libraries *numpy* and *matplotlib* for manipulations and operations on matrices and graphical plots, respectively (CAI; LANGTANGEN; MOE, 2005; SCIPY-NUMPY, 2020; SCIPY-MATPLOTLIB, 2020).

On the other hand, a mesh generator can introduce elements considered to be of lower quality, depending on the complexity of the geometry and the refinement employed.

It is known that in a lower quality element the resolution of differential equations, which describe any phenomenon, can produce unsatisfactory results, such as numerical instability and inadequacy to reality (PARK; SHONTZ, 2010). It can be stated that the greater regularity of the element's geometry, this implies a better quality of the mesh that contains it (BOROUCHAKI; FREY, 1998; JOHNEN; ERNST; GEUZAINE, 2015).

Since the elements belonging to the meshes in generalized coordinates are quadrilateral, it is important that the geometry of the element approaches a square. For this, after the generation process, it is essential to identify the elements of lower quality, so that, later, they can be improved.

The process of identifying lower quality elements can be carried out through the application of quality metrics, which assign numerical values to the elements, order them according to the calculated values and classify them in terms of quality.

In this work, three criteria for analyzing the quality of the mesh elements are proposed and applied: the ratios between the size of each side of the element with the others (BOROUCHAKI; FREY, 1998; JOHNEN; ERNST; GEUZAINE, 2015), the internal angles (COELHO; LOURENCO, 2001) and the compactness of the element (GOSE; JOHNSONBAUGH; JOST, 1996; GONZALEZ; WOODS, 2011). All criteria verify the similarity of the elements of the computational mesh to a square, the element considered ideal, thus identifying those elements considered to be of low quality.

The contribution of this paper consists in the use of the three criteria, individually or collectively, in the context of mesh quality analysis. We illustrate this contribution through a variety of applications.

The work is structured as shown below. Initially, transformation metrics are described in generalized coordinate theory. Then the mesh generator is developed. In the sequence, the metrics used to assess the quality of the elements are described. Finally, some examples of meshes obtained by the generator are presented, as well as an analysis of the quality of the elements obtained.

Generalized coordinates

For a computational methodology to be applied to a physical problem, it is necessary to discretize the domain of the problem, that is, to build a computational mesh that can represent the studied geometry and, thus, obtain the values of interest.

The discretization of the physical domain can be carried out according to a structured or unstructured mesh. Unstructured meshes are more adaptable than structured meshes, especially in problems with complex geometries (MALISKA, 2004; FORTUNA, 2012). However, the major disadvantage of unstructured meshes is the difficulty of ordering the elements, which implies a variation in the size of the diagonals of the coefficient matrix and the additional cost of using memory, making it difficult to apply numerical methods to obtain the solution of linear systems. Therefore, the domain discretization in this work will be performed using structured meshes.

As for the coordinate system, in general, the problem domain is discretized according to the Cartesian coordinate system because it is simpler. However, for problems with complex geometry, it is convenient to adopt another coordinate system, due to the fact that the Cartesian coordinate system leads to a poor adaptation of the border, since the physical domain does not always coincide with the domain of the Cartesian mesh. To solve the problem, the generalized coordinate system will be used.

In generalized coordinates, the computational mesh coincides with the geometry of the problem and computational treatment becomes more appropriate. Other reasons that justify the use of generalized coordinates in the discretization of computational meshes are the simplicity in programming computational codes to solve complex problems and the ease in developing generic methodologies.

In the following, some important concepts about generalized coordinate theory will be presented.

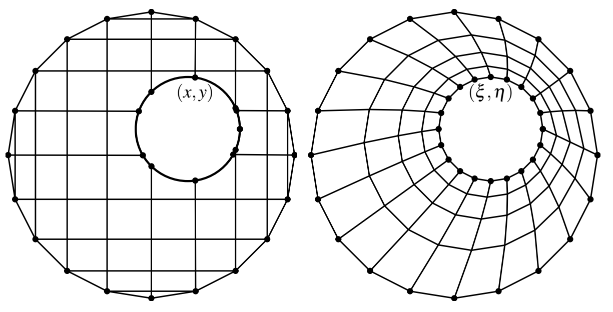
Transformation metrics

The Cartesian system (x,y) is called the physical domain and the generalized system (ξ,η) is called the transformed domain or computational domain.

The transformation from a non-trivial geometry described in a Cartesian coordinate system to a generalized coordinate system involves transformation metrics, or mathematical relations, that can accurately describe the transformed data.

The mapping of irregular or regular geometries written in Cartesian coordinates (x,y), Figure 1a, is performed numerically for regular geometries written in generalized coordinate system (ξ,η), Figure 1b.

Figure 1 – Coordinate systems



(a) Cartesian coordinate **(b)** Generalized coordinate

Source: The authors.

Since the transformed domain is regular, for convenience, unitary normalization of elementary volumes is assumed, that is, $\Delta\xi = \Delta\eta = 1$. In this way, even if in the physical plane the coordinated lines assume arbitrary spacing, in the computational plane the dimensions are fixed to the unit.

To obtain the (ξ,η) system, the following generating equations are used

$$\xi = \xi(x,y), \tag{1}$$

$$\eta = \eta(x,y). \tag{2}$$

The transformation metrics, based on differentials of the equations (1) and (2), are

$$d\xi = \xi_x dx + \xi_y dy, \tag{3}$$

$$d\eta = \eta_x dx + \eta_y dy, \tag{4}$$

or, in the matrix form

$$\begin{pmatrix} d\xi \\ d\eta \end{pmatrix} = \begin{pmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{pmatrix} \begin{pmatrix} dx \\ dy \end{pmatrix}, \tag{5}$$

which can also be written as

$$d^t = Ad^f, \tag{6}$$

where ξ_x, η_x, ξ_y, and η_y denoting partial derivatives, d^t and d^f represent, respectively, the differentials in the transformed and physical domain, while A is the transformation matrix between the domains.

Starting from the assumption that it is possible to find a representation in a Cartesian coordinate system for a model described in generalized coordinates, then we admit the existence of the inverse of the equations (1) and (2), so

$$x = x(\xi, \eta), \tag{7}$$

$$y = y(\xi, \eta), \tag{8}$$

and from differentials of equations (7) and (8) we have

$$\begin{pmatrix} dx \\ dy \end{pmatrix} = \begin{pmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{pmatrix} \begin{pmatrix} d\xi \\ d\eta \end{pmatrix}, \tag{9}$$

which can also be written as

$$d^f = Bd^t, \tag{10}$$

where x_ξ, x_η, y_ξ, and y_η denoting partial derivatives and B is the transformation matrix between the physical and transformed domains.

Replacing equation (6) in equation (10), we get $d^f = BAd^f$, so that $BA = I$, or equivalent $A = B^{-1}$. Thus, the A matrix becomes

$$\begin{pmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{pmatrix} = \begin{pmatrix} \dfrac{y_\eta}{x_\xi y_\eta - x_\eta y_\xi} & \dfrac{-x_\eta}{x_\xi y_\eta - x_\eta y_\xi} \\ \dfrac{-y_\xi}{x_\xi y_\eta - x_\eta y_\xi} & \dfrac{x_\xi}{x_\xi y_\eta - x_\eta y_\xi} \end{pmatrix}, \tag{11}$$

where $J = \det(A) = (x_\xi y_\eta - x_\eta y_\xi)^{-1}$ is called Jacobian of transformation (MALISKA, 2004).

Two-dimensional mesh generation

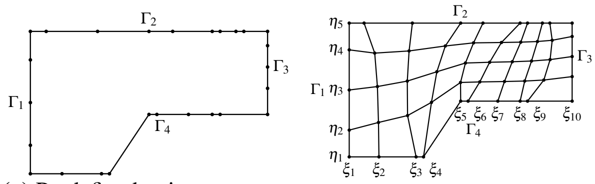
In this work, we opted for the use of elliptical partial differential equations (EPDE) as a method of generating two-dimensional meshes, since their solutions do not generate null Jacobian, and the lines ξ or η never intersect (THOMPSON; WARSI; MASTIN, 1985; MALISKA, 2004). Thus, the governing equations for the generation of two-dimensional meshes in a domain, for example the domain illustrated in Figure 2(a), are

$$\nabla^2 \xi = P(\xi, \eta), \tag{12}$$

$$\nabla^2 \eta = Q(\xi, \eta), \tag{13}$$

whose boundary conditions of the type *Dirichlet* are expressed by $\xi = \xi_1$ in Γ_1 (left border), $\xi = \xi_N$ in Γ_3 (right border); $\eta = \eta_1$ in Γ_4 (lower border) and $\eta = \eta_M$ in Γ_2 (upper border). In particular, in Figure 2a were used $N = 10$ and $M = 5$, where N is the number of lines from ξ and M is the number of lines from η.

Figure 2 – Example of mesh in generalized coordinates



(a) Predefined point

(b) Mesh in generalized coordinates

Source: The authors.

Solving the equations (12) and (13), in relation to the (x, y) coordinates, using the transformation metrics of the (ξ, η) system, the coordinated lines can be generated in the directions ξ and η, inside the computational mesh, Figure 2(b), as in any other geometry, through the equations

$$\alpha x_{\xi\xi} + \gamma x_{\eta\eta} - 2\beta x_{\xi\eta} + \frac{1}{J^2}(Px_\xi + Qx_\eta) = 0, \tag{14}$$

$$\alpha y_{\xi\xi} + \gamma y_{\eta\eta} - 2\beta y_{\xi\eta} + \frac{1}{J^2}(Py_\xi + Qy_\eta) = 0, \tag{15}$$

where x and y are the Cartesian coordinates of the physical domain, ξ and η are the generalized coordinates of the computational domain, P and Q are the source functions, and

$$\alpha = x_\eta^2 + y_\eta^2, \tag{16}$$

$$\beta = x_\xi x_\eta + y_\xi y_\eta, \tag{17}$$

$$\gamma = x_\xi^2 + y_\xi^2. \tag{18}$$

The numerical solution of elliptical PDEs (14) and (15), subject to initial and boundary conditions, provide the lines ξ and η, which generate the computational mesh (MALISKA, 2004; CIRILO; BORTOLI, 2006; FORTUNA, 2012; DE BORTOLI, 2000). Note that initial conditions are null due to the elliptical characteristic of the equations.

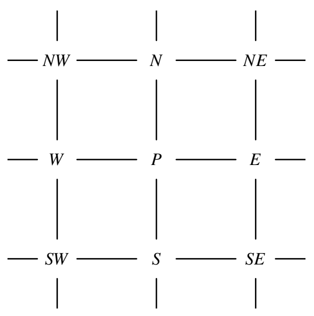
For convenience, equations (14) and (15) can be written using a generic ϕ variable as

$$\alpha\phi_{\xi\xi} + \gamma\phi_{\eta\eta} - 2\beta\phi_{\xi\eta} + \frac{1}{J^2}(P\phi_\xi + Q\phi_\eta) = 0. \tag{19}$$

To approximate the derivatives in equation (19), the finite difference method is used. The mesh nodes are labeled by the cardinal points P (center), E (east), W (west), N (north), S (south), NW (northwest), SW (southwest), SE (southeast) and NE (northeast), as in Figure 3.

Using second-order finite differences by means of central differences to approximate the derivative terms of equation (19), around P, we obtain

Figure 3 – Index labeling



Source: The authors.

$$\alpha(\frac{\phi_E - 2\phi_P + \phi_W}{\Delta\xi^2}) + \gamma(\frac{\phi_N - 2\phi_P + \phi_S}{\Delta\eta^2})$$
$$-2\beta(\frac{\phi_{NE} - \phi_{NW} + \phi_{SW} - \phi_{SE}}{4\Delta\xi\Delta\eta}) +$$
$$\frac{1}{J^2}(P\frac{\phi_E - \phi_W}{2\Delta\xi} + Q\frac{\phi_N - \phi_S}{2\Delta\eta}) = 0, \qquad (20)$$

and regrouping the terms

$$-(2\alpha - 2\gamma)\phi_P + (\alpha + \frac{P}{2J^2})\phi_E + (\alpha - \frac{P}{2J^2})\phi_W +$$
$$(\gamma + \frac{Q}{2J^2})\phi_N + (\gamma - \frac{Q}{2J^2})\phi_S - \frac{\beta}{2}\phi_{NE} + \frac{\beta}{2}\phi_{NW} -$$
$$\frac{\beta}{2}\phi_{SW} + \frac{\beta}{2}\phi_{SE} = 0. \qquad (21)$$

Thus, the numerical solution of the two-dimensional mesh generation equations, equations (14) and (15), written according to the generic ϕ variable, equation (21), is given by

$$\phi_P = \frac{1}{A_P}(A_E\phi_E + A_W\phi_W + A_N\phi_N + A_S\phi_S +$$
$$A_{NE}\phi_{NE} + A_{SE}\phi_{SE} + A_{NW}\phi_{NW} + A_{SW}\phi_{SW}), \qquad (22)$$

where

$$A_P = 2\alpha + 2\gamma, \qquad A_N = \gamma + \frac{Q}{2J^2}, \qquad A_{SE} = \frac{\beta}{2},$$
$$A_E = \alpha + \tfrac{P}{2J^2}, \qquad A_S = \gamma - d\tfrac{Q}{2J^2}, \qquad A_{NW} = \frac{\beta}{2}, \qquad (23)$$
$$A_W = \alpha - \frac{P}{2J^2}, \qquad A_{NE} = -\frac{\beta}{2}, \qquad A_{SW} = -\frac{\beta}{2},$$

with derivatives present in J approximated by central differences.

Next, the mesh generator in generalized coordinates is presented.

Mesh generator in generalized coordinates

For the creation of an automated procedure for mapping a geometry in cartesian coordinates, through a generalized coordinate system, we chose to use the programming language Python.

The reason for this choice is due to the fact that Python is a general, free, open, and multiplatform language, which can allow the creation of extensions of the developed application, such as, for example, a graphical interface module, among others.

Regarding performance, it is necessary to use extra libraries, specific to numerical computing applications, to achieve satisfactory results at run time. Because it is a scripting language, that is, interpreted, the performance of an application written in Python tends to be inferior when compared to a compiled language, as is the case of the Fortran language. In Python, the tools available for matrix manipulation and linear algebra are limited and not optimized for the type of application that was intended to be developed. However, because the adopted language is modular, it is possible to use routines and libraries in compiled languages, in order to optimize more complex operations, such as those used for domain transformations (CAI; LANGTANGEN; MOE, 2005).

For the creation of the mesh generator, the following libraries were used:

- *Numpy* - library dedicated to matrix and algebraic operations in general (SCIPY-NUMPY, 2020)
- *Matplotlib* - library used to generate graphs and manipulate data in graphical form (SCIPY-MATPLOTLIB, 2020).

The additional modules employed are also free for use.

Description of the algorithm developed

For the definition and creation of a geometry mapped in generalized coordinates, it starts from an initial set of points that describe the border, that is, the physical contour. In this work, the parametric linear spline method was chosen to interpolate the set of border points under study. Parametric linear spline equations (BURDEN; FAIRES; BURDEN, 2015) are

$$s_i^x(t) = x_{i-1}\frac{t_i - t}{t_i - t_{i-1}} + x_i\frac{t - t_{i-1}}{t_i - t_{i-1}}, \qquad (24)$$

$$s_i^y(t) = y_{i-1}\frac{t_i - t}{t_i - t_{i-1}} + y_i\frac{t - t_{i-1}}{t_i - t_{i-1}}, \qquad (25)$$

$\forall t \in [t_{i-1}, t_i]$, where s is the interpolating spline curve and t is the interpolated variable.

Having obtained the interpolating lines of the border, equations (24) and (25), the number of partitions desired for each border must be defined.

Then, points of the splines curves are selected, which define the new border of the approximate geometry. To do this, an approach is used to position the points by weighted average of the components of the border, i.e.,

$$x_{ij} = p_x^{\Gamma_1\Gamma_3}(x_{0j} + i\Delta_{j-1}^{x\xi}) + p_x^{\Gamma_2\Gamma_4}(x_{i0} + j\Delta_{i-1}^{x\eta}), \qquad (26)$$

$$y_{ij} = p_y^{\Gamma_1\Gamma_3}(y_{0j} + i\Delta_{j-1}^{y\xi}) + p_y^{\Gamma_2\Gamma_4}(y_{i0} + j\Delta_{i-1}^{y\eta}), \qquad (27)$$

where $\Delta_j^{x\xi} = \dfrac{x_{\xi j} - x_{0j}}{\xi}$, $\Delta_i^{x\eta} = \dfrac{x_{i\eta} - x_{i0}}{\eta}$, $\Delta_j^{y\xi} = \dfrac{y_{\xi j} - y_{0j}}{\xi}$, $\Delta_i^{y\eta} = \dfrac{y_{i\eta} - y_{i0}}{\eta}$, $i = 1, ..., \xi$ and $j = 1, ..., \eta$.

The values of $p_x^{\Gamma_1\Gamma_3}$, $p_x^{\Gamma_2\Gamma_4}$, $p_y^{\Gamma_1\Gamma_3}$, and $p_y^{\Gamma_2\Gamma_4}$ indicate the weights on the left/right borders in x, top/bottom in x, left/right in y and top/bottom in y , respectively. The percentage weights in equations (26) and (27) can better adjust the distribution of points in the domain, avoiding concentrations of points. In this way, the arrangement of the internal points is influenced by borders. Finally, on the set of ordered pairs, the resolution of the equation (22) is applied.

In summary, the Algorithm 1 describes the proposed and implemented mesh generator code.

The Algorithm 1 is a two-dimensional mesh generator developed for geometries consisting of a single block. For complex geometries, to maintain the quality of the elements, it is convenient to use the multi-block technique (DE BORTOLI, 2000; CIRILO; BORTOLI, 2006; PARDO *et al*., 2012; SAITA *et al*., 2017; ALMEIDA *et al*., 2018), which uses the mesh generator defined in Algorithm 1 to generate each mesh block.

Figure 4 shows a rectangular geometry with an obstacle. It makes a comparison between the meshes generated using Algorithms 1 e 2. It is observed that the elements of Figure 4(a), close to the obstacle, have lost their square shape, thus generating elements of low quality. On the other hand, due to the shape of the geometry and the structure of division in blocks, 3 blocks, the mesh in curvilinear multi-block coordinates, Figure 4(b), resulted in square elements, maintaining the quality of the elements. In order to maintain the same number of nodes and elements in Figure 4(a), the elements of block 2 in Figure 4(b) have been refined.

Algorithm 1: Two-dimensional mesh generator in generalized coordinates: single block

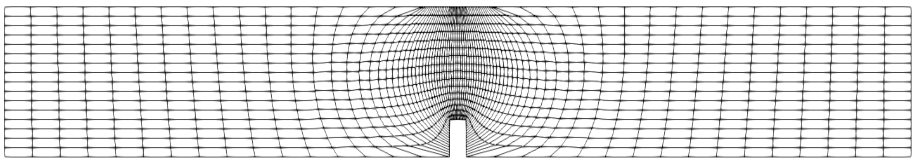
```
Input  : Number of partitions from the physical and transformed planes, border weights
         and contour points.
Output : Coordinates of the computational mesh nodes.
1 begin
2     Reading the input data;
3     Border interpolation: determine the border interpolating curves using the
        parameterized linear spline method;
4     Calculation of points of the spline lines: find the new border points using a
        weighted average of the interpolated borders;
5     Solving coordinate transformation differential equations: numerically solve the
        meshing equation in generalized coordinates (14) and (15), applying the
        Gauss-Seidel Iterative method;
6 end
```

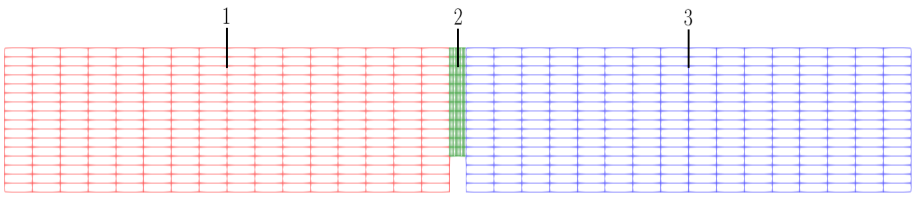
Algorithm 2: Two-dimensional mesh generator in generalized coordinates: multi-blocks

```
Input  : Blocks containing number of partitions from the physical and transformed
         planes, border weights and contour points.
Output : Graph and coordinates of the nodes of each block of the computational mesh.
1 begin
2     foreach input file - block do
3         Mesh generation: generate the computational mesh of the block by executing
            the algorithm 1;
4         Plot the grid: plot the computational grid of the block using the results;
5         Write of the results;
6     end
7     Graph display: show the computational meshes generated for all blocks in one graph;
8 end
```

It should be noted that another advantage of the multiblock technique is that it allows different refinements in the blocks, which can, in some situations, generate improvement of the elements.

Figure 4 – Comparison between meshes generated with a single block and multi-block techniques, both with the same number of partitions


(a) Mesh generated with single block



(b) Mesh generated with multiblock technique (3 blocks)

Source: The authors.

Quality analysis of mesh elements

In order to analyze the quality of the elements belonging to the mesh generated in generalized coordinates, it was considered that a higher quality element is a quadrilateral that comes closest to a square. Thus, three criteria for quality analysis are proposed and applied, namely:

- ratios between the size of each side of the element with the others (BOROUCHAKI; FREY, 1998; JOHNEN; ERNST; GEUZAINE, 2015);
- internal angles (COELHO; LOURENCO, 2001);
- compactness coefficient of the element (GOSE; JOHNSONBAUGH; JOST, 1996; GONZALEZ; WOODS, 2011).

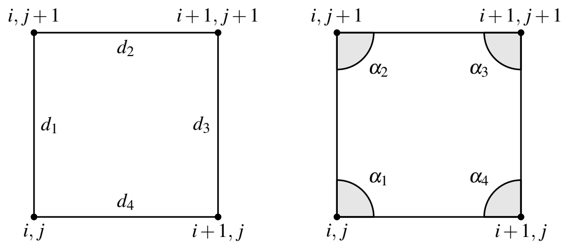
It is observed that the criteria verify the similarity of the elements of the computational mesh to a square, an element considered ideal, thus allowing to identify the elements of low quality.

Quality metrics

The nomenclatures adopted to define the sides and angles of each element are presented in Figures 5a and 5b, respectively. It was also considered that the numbering starts at node i,j, rotating the element clockwise.

In this way, the evaluation of the generated mesh is performed, in which, for each element, the ratios $r_{i,j} = \dfrac{d_i}{d_j}$ between the size of sides d_i and d_j are calculated, for $i = 1,\ldots,4$, $j = 1,\ldots,4$ and $i \neq j$. Then, it is checked whether the ratios are close to the unit, $r_{i,j} = 1$, equivalent to the ratio between two sides of a square.

Figure 5 – Nomenclatures adopted to define the sides and angles of an element



(a) Sides **(b)** Angles

Source: The authors.

In addition to the ratio between the sizes of the sides of the quadrilateral, verifying the property that a higher quality element is a quadrilateral similar to a square, it is necessary to calculate for each element if the internal angles approach $\alpha_i = \dfrac{\pi}{2}$rad, using

$$\alpha_i = \arccos\left(\frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\cdot\|\vec{v}\|}\right), \quad (28)$$

with $\vec{u}$ and $\vec{v}$ vectors that represent, respectively, two sides of the element.

The third quality metric, used to verify the similarity between the elements of the computational mesh and a square, concerns the compactness coefficient c. This metric has the characteristic of being invariant to translation, rotation and scale change, given by:

$$c = \frac{P^2}{S}, \tag{29}$$

where P is the perimeter and $S = \frac{1}{J}$ is the area of the element.

Note that if d_i is the size of the side of a square, then the compactness coefficient c_i of the square is given by

$$c_i = \frac{(4d_i)^2}{d_i^2} = \frac{16d_i^2}{d_i^2} = 16, \tag{30}$$

so that the element loses the similarity of the square when $c \neq 16$, and consequently the metric evaluates the element as being a lower quality element. Finally, the importance of applying quality metrics in the analysis processes and, subsequently, improving computational meshes, lies in the fact that the regularity of the elements of geometry makes it possible to improve the stability and convergence of the solutions of the differential equations, in addition to reduce the execution time of the numerical algorithm (PARK; SHONTZ, 2010).

Description of the developed algorithms

The process of developing the algorithm for quality analysis of meshes, in generalized coordinates, was divided into two modules. In the first, the algorithm calculates the quality metrics, being implemented in the *Python* programming language (RUSSUM, 1995).

In the second, implemented in the programming language R (R CORE TEAM, 2020), the algorithm performs the identification of low quality elements, according to the metrics employed and calculated by the first module.

Briefly, the Algorithm 3 lists the sequence of steps used to identify low quality elements in meshes described by generalized coordinates. Due to its extension and in order to present it in a more didactic way, this algorithm was divided into two parts, the Algorithms 4 and 5. Further details can be obtained in (NAOZUKA, 2018).

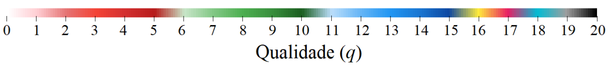
For the simulations (results), the parameters presented and defined in Algorithms 3–5 will be considered, as well as parameters that are not explicitly defined, such as the color legend that defines the quality metrics q. In this context, to the $U' - \overline{E_\cup}$ set of higher quality elements, $q = 0$ is assigned. The $\overline{E_r}$, $\overline{E_\alpha}$, and $\overline{E_c}$ sets of low quality elements, relative to one of the evaluation metrics, they are categorized, respectively, in the

Algorithm 3: Identification of low quality elements in meshes described by generalized coordinates

```
Input  : Total set of elements and their quality metrics, containing block number in
         which the element is allocated, node coordinates, side ratios $r$, internal angles $\alpha$
         and compactness coefficient $c$; number of partitions of the transformed plan for
         each block; and desired percentage $\overline{p}$ of low quality elements.
Output : Number of points in the directions $\xi$ and $\eta$ for each block, coordinates of the
         nodes and values of $q$ associated with each element of the block. The value $q$,
         defined between 0 and 20, describes the sets of quality metrics.
1 begin
2     Execution of the Algorithm 4;
3     Execution of the Algorithm 5;
4 end
```

value ranges $1 \leq q \leq 5$, $6 \leq q \leq 10$, and $11 \leq q \leq 15$. Also, $q = 16$, 17, 18, and 20 are assigned to the sets formed by the intersections $\overline{E_{r,\alpha}}$, $\overline{E_{r,c}}$, $\overline{E_{\alpha,c}}$, and $\overline{E_\cap}$, respectively. Finally, the elements where $r = +\infty$ and $\alpha = \nexists$, which belong to the set $U - U'$, are labeled with $q = 19$. The color legend can be seen in Figure 6, which will be the same for all experiments to be presented.

Figure 6 – Color legend associated with the quality of the elements of a computational mesh



Source: The authors.

Results and discussion

To verify the efficiency of the developed algorithms, and the methodological procedures used, computational meshes generated by one block, Algorithm 1, and by multi-blocks, Algorithm 2, are presented.

In all experiments to be presented, the quantity of elements in the sets is obtained using quality metrics (q) of elements. Thus, information is obtained about the quantity of low quality elements, considering the desired percentages and tolerances, $\overline{p}$ e $\overline{\varepsilon}$, in relation to the quantity of elements in the sets: $U' - \overline{E_\cup}$, $\overline{E_r}$, $\overline{E_\alpha}$, $\overline{E_c}$, $\overline{E_{r,\alpha}}$, $\overline{E_{r,c}}$, $\overline{E_{\alpha,c}}$, $U - U'$, and $\overline{E_\cap}$.

Algorithm 4: Identification of low quality elements in meshes described by generalized coordinates - Part 1

```
1  begin
2      Reading of input data: Read the block number, the coordinates of the nodes, the side
       ratio $r$, the internal angle $\alpha$ and the compactness coefficient $c$ of each element;
3      Obtaining the $U'$ set of elements: Remove the elements where $r = +\infty$ and $\alpha = \nexists$
       from the total set of elements $U$;
4      Calculation of variances: Calculate the variance of the side ratios $s^2_R$, between internal
       angles $s^2_A$ and between compactness coefficients $s^2_C$, according to the equation
       $s^2_M = \frac{\sum_{k=1}^{|M|}(m_k-\overline{m})^2}{|M|-1}$, $k = 1,\dots,|M|$, where $M$ indicates
       the element of the set of sides ($R$), or of angles ($A$) or of compactness ($C$);
5      if $s^2_R = 0$ and $s^2_A = 0$ and $s^2_C = 0$ then
6          if $r_k = r_\square$ and $\alpha_k = \alpha_\square$ and $c_k = c_\square$ then
7              All mesh elements are identical and equal to a square;
8          else
9              All mesh elements are identical, but different from a square ;
10         end
11         break
12     end
13     Calculation of the minimum and maximum values: Calculate the minimum values $m_{\min}$ and
       maximum values $m_{\max}$ of the quality metrics, using the equations
       $m_{\min} = \min\{m_k, m_\square\}$, $m_{\max} = \max\{m_k, m_\square\}$;
14     Normalization: Normalize the values of the quality metrics, as well as the value of the
       metrics evaluated on a square, using the equations
15     $m_{\text{norm}_k} = (b-a)\frac{m_k - m_{\min}}{m_{\max} - m_{\min}} + a;$
16     $m_{\text{norm}_\square} = (b-a)\frac{m_\square - m_{\min}}{m_{\max} - m_{\min}} + a.$
17     Standardization of normalization: Take $a = 0$ and $b = 10$ for the interval $[a,b]$ ;
18     Calculation of variances: Calculate $s^2_{M_{\text{norm}}}$ variances from normalized quality metrics;
19 end
```

Algorithm 5: Identification of low quality elements in meshes described by generalized coordinates - Part 2

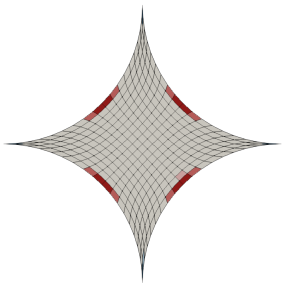
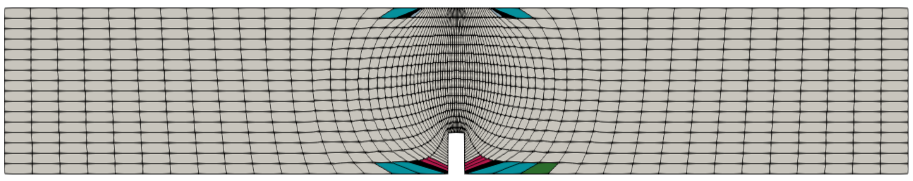
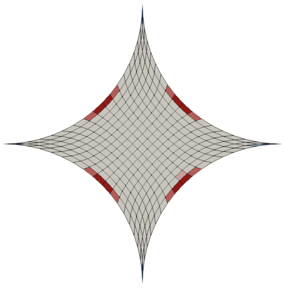
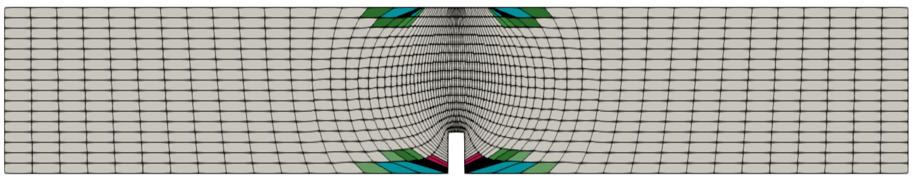
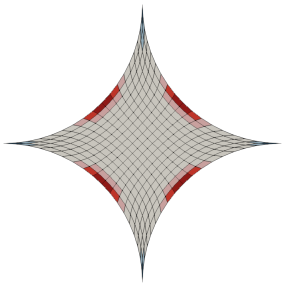
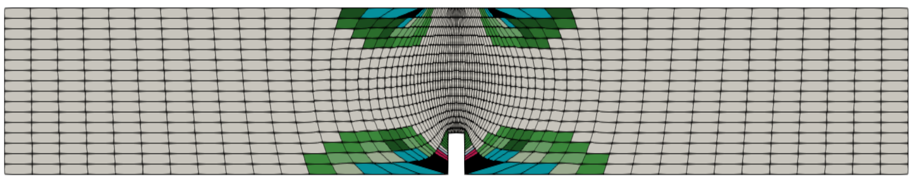
```
1  begin
2      Calculation of tolerances: Calculate the minimum tolerance $\varepsilon_{\min}$ and the
       maximum tolerance $\varepsilon_{\max}$;
3      for $\varepsilon_i \leftarrow \varepsilon_{\min}$ to $\varepsilon_{\max}$ do
4          Identification of low quality elements: Identify low quality elements in relation to the
           ratio between sides $E_r$, in relation to the internal angle $E_\alpha$ and to the
           compactness coefficient $E_c$, considering the tolerance $\varepsilon_i$ ;
5          Obtaining the set $E_{\cup_i}$: Obtain the set $E_{\cup_i}$, formed by the union of the sets
           $E_r$, $E_{alpha}$ e $E_c$, and its number of elements $|E_{\cup_i}|$;
6      end
7      Obtaining the exponential nonlinear regression function: Find the exponential nonlinear
       regression function that fits the tolerance graph $\varepsilon_i$ by the quantity of low
       quality elements $|E_{\cup_i}|$, applying the least squares method;
8      Calculation of tolerance: Calculate the desired tolerance $\overline{\varepsilon}$ from the
       exponential nonlinear regression function and the desired percentage $\overline{p}$;
9      Identification of low quality elements: Identify low quality elements in relation to the
       side-to-side ratio $\overline{E_r}$, between internal angles $\overline{E_\alpha}$ and coefficient
       of compactness $\overline{E_c}$, considering the desired tolerance $\overline{\varepsilon}$;
10     Getting other sets: Get the sets $\overline{E_\cup}$, $\overline{E_{r,\alpha}}$, $\overline{E_{r,c}}$,
       $\overline{E_{\alpha,c}}$, and $\overline{E_\cap}$, formed by operations involving sets
       $\overline{E_r}$, $\overline{E_\alpha}$, and $\overline{E_c}$;
11     Attribution of the value of $q$: Define a number $q$ for each category of elements,
       considering the sets obtained;
12     Results writing: Write the results;
13 end
```

One block meshes

Using geometries with cusp-shaped structures and with one obstacle, the meshes of one block are generated, as illustrated in the Table 1. The cusp-shaped geometry, the first column, consists of the inner region constructed by four circles, each circle tangent to two other circles.

The obstacle geometry, second column of Table 1, is rectangular. Both meshes were obtained by executing the Algorithm 1.

Table 1 – Identification of low quality elements, according to the metrics used, for geometries with a single block.

Cusp-shaped structures	**Rectangular plate with one obstacle**
$\overline{p} = 5\%$ of low quality elements	
	
$\overline{p} = 10\%$ of low quality elements	
	
$\overline{p} = 20\%$ of low quality elements	
	

Source: The authors.

Table 2 – Quantities of low quality elements, according to the metrics used, for geometries with a single block.

$\overline{p}$	$\lvert U' - \overline{E_{\cup}} \rvert$	$\lvert \overline{E_r} \rvert$	$\lvert \overline{E_\alpha} \rvert$	$\lvert \overline{E_c} \rvert$	$\lvert \overline{E_{r,\alpha}} \rvert$	$\lvert \overline{E_{r,c}} \rvert$	$\lvert \overline{E_{\alpha,c}} \rvert$	$\lvert U - U' \rvert$	$\lvert \overline{E_{\cap}} \rvert$
Cusp-shaped structures - 324 elements									
5%	302	17	0	5	0	0	0	0	0
10%	302	17	5	5	0	0	5	0	0
20%	274	25	25	5	0	0	5	0	0
Rectangular plate with one obstacle - 1280 elements									
5%	1262	9	11	17	4	9	10	0	4
10%	1241	11	35	19	7	11	15	0	7
20%	1093	19	181	43	15	19	37	0	15

Source: The authors.

When applying the Algorithm 3 on the cusp-shaped structures and rectangular plate with one obstacle, information about the quality of the mesh elements are obtained, where elements of low quality are highlighted and identified in Table 1, according to the color legend shown in Figure 6.

As expected, the low quality elements in the cusp-shaped mesh are located in the vicinity of the four vertices of the computational mesh, Table 1. Of the 324 elements of the mesh, 22 were of low quality. According to the metrics evaluated, for $\overline{p} = 5\%$, 17 low quality elements refer to the ratio between sides and 5 refer to the compactness coefficient. For $\overline{p} = 10$ and 20 %, low quality elements in relation to the internal angle were also identified, these results can be seen in the Table 2.

Similarly, for the 1280 elements of the rectangular plate with one obstacle, the location of low quality elements are showed in the Table 1. Details on the quantity of low quality elements, in relation to the metrics evaluated, they are presented in the Table 2.

It is observed that for $\overline{p} = 20\%$, the metric referring to the internal angle showed 181 elements of lower quality (in the order of 15% of the total elements of the mesh), located inferiorly in the vicinity of the obstacle and in the upper part of the mesh.

In this context, it should be noted that Figure 4 presents a comparison between the meshes of the rectangular plate with one obstacle generated by one block (Algorithm 1) and multi-blocks (Algorithm 2). It appears that the multi-block mesh has square elements, maintaining the quality of the elements.

Multi-block meshes

Using the geometries involving the structures NACA 64A 010 (HSU; JAMESON, 2002; NAOZUKA, 2018), turtle, plane, profile of a face, dog and frog, multi-block meshes are generated (NAOZUKA, 2018). The number of blocks used to generate each of the meshes is shown in the first row of Tables 3 and 4, varying between 2 and 25 blocks. Note that the meshes are ordered according to the number of blocks and the complexity of the geometry.

In the figures in Tables 3 and 4, the low quality elements are highlighted and identified according to the color legend of Figure 6. In general, it can be seen that low quality elements were identified in all the geometries presented, from the quality metrics defined in the work. Particularly, in the NACA 64A 010 experiment, low quality elements were identified at the ends of the airfoil.

Due to the complexity of the geometries, it was not possible to define a pattern in the location of the lower quality elements detected in the multi-block computational meshes. However, there is an evident concentration of elements in block 1 for the turtle, in block 2 for the plane, in block 7 for the face profile, in blocks 3 and 7 for the dog and in blocks 2, 3 and 10 for the frog.

Conclusion

This work presented the development of a two-dimensional mesh generator in generalized coordinates. As an original contribution, in relation to the methodology of the computational mesh generation process, we highlight the application of the parameterized linear spline method for the interpolation of the borders and the calculation of the weighted average of the borders to obtain the internal points of the mesh, which allowed faster convergence of the numerical resolution of the governing equation.

Note that the chosen programming language, and the libraries used, made possible, in addition to the optimization of the code, precision of the results and practicality in the visualization of the plots.

Knowing that the solution of the governing equations for the generation of two-dimensional meshes in generalized coordinates can generate elements of low quality, depending on the complexity of the geometry and the refinement employed, it is essential to analyze the quality of the meshes produced. In this context, the work presented a process of quality analysis of computational meshes in generalized coordinates, constituting another important contribution.

To carry out the quality analysis of the elements of the computational mesh, three evaluation criteria were applied: the ratios between the size of each side with the others, the internal angles and the compactness coefficient of the element. As the elements of the generated meshes are quadrilateral, then it was verified the similarity of the elements to a square, considered as an ideal element. Thus, using the programming languages Python and R, algorithms were developed to classify the quality of the elements.

According to the experiments carried out, the results were generally satisfactory, since the implemented algorithms accurately captured the low quality elements.

Table 3 – Identification of low quality elements, according to the metrics used, for geometries with multi-blocks - Part 1.

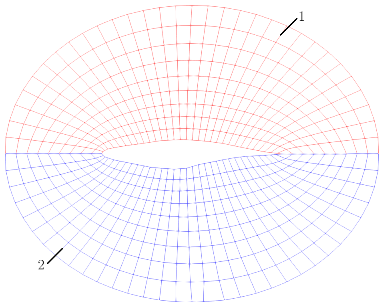
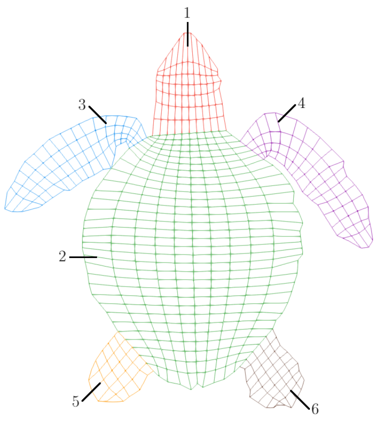
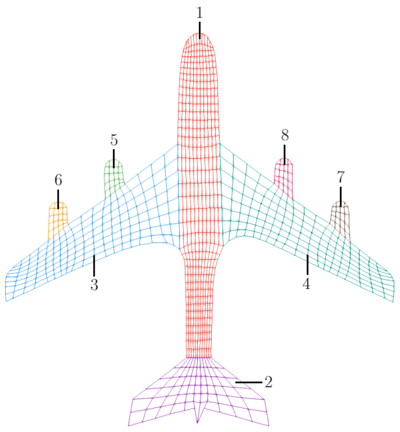
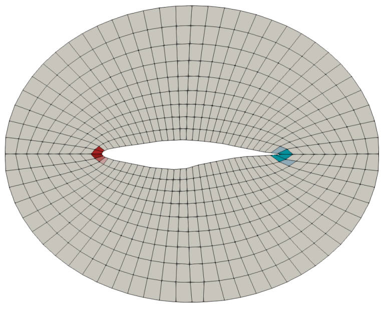
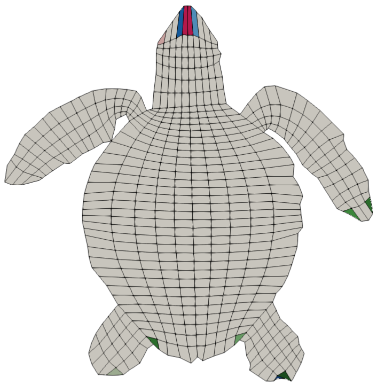
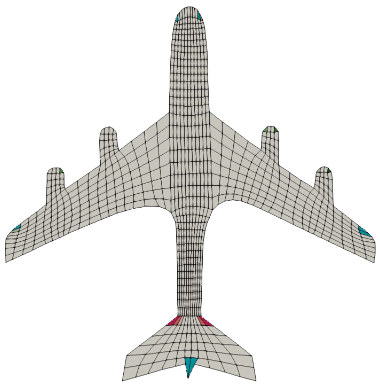
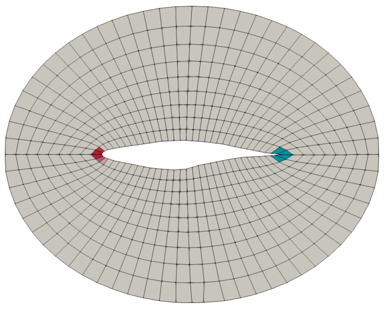
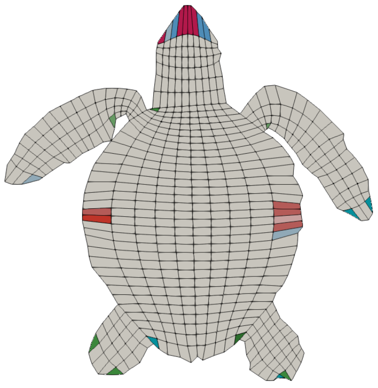
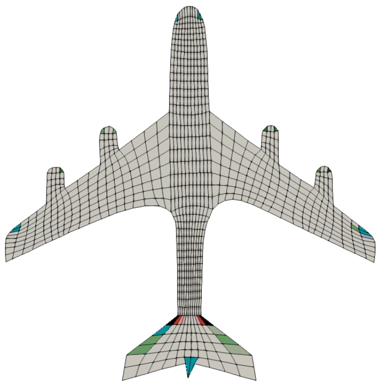
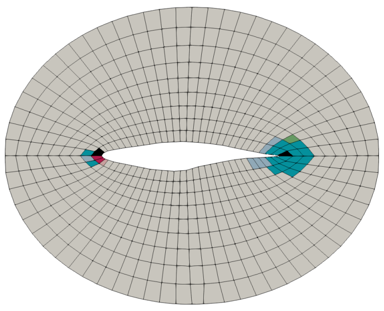
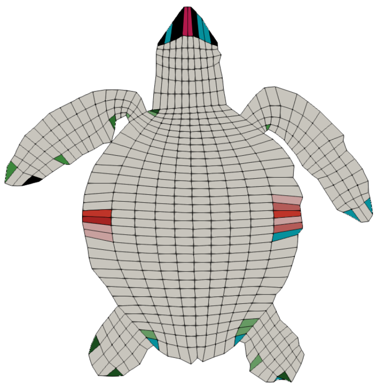
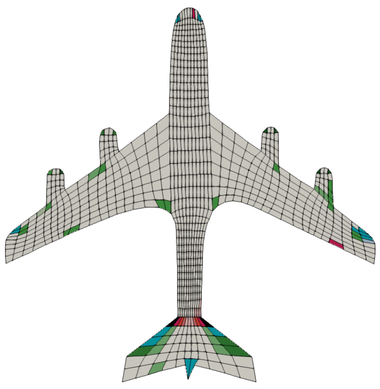
NACA 64A 010	**Turtle**	**Plane**
		
$\overline{p} = 5\%$ of low quality elements		
		
$\overline{p} = 10\%$ of low quality elements		
		
$\overline{p} = 20\%$ of low quality elements		
		

Table 4 – Identification of low quality elements, according to the metrics used, for geometries with multi-blocks - Part 2.

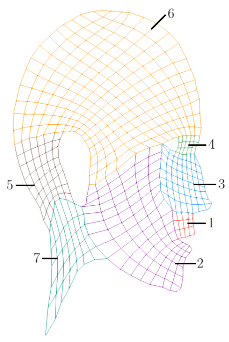
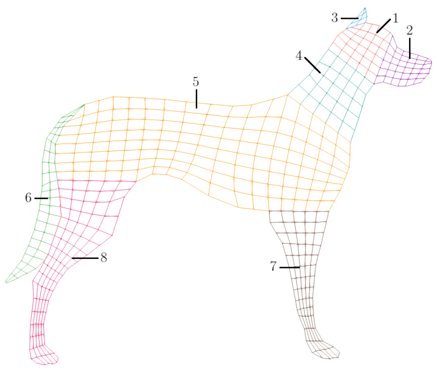
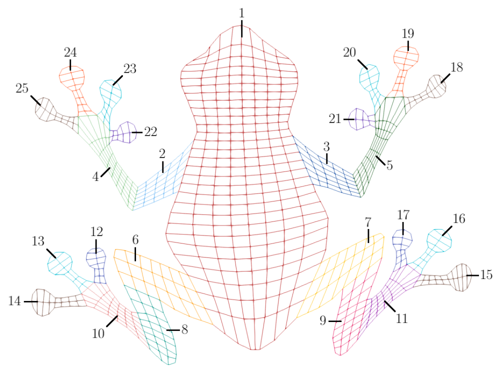
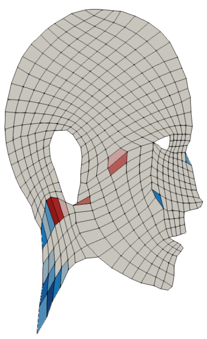
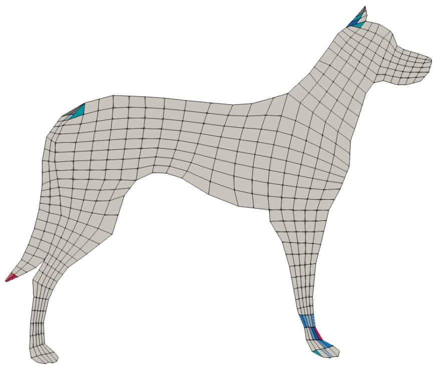
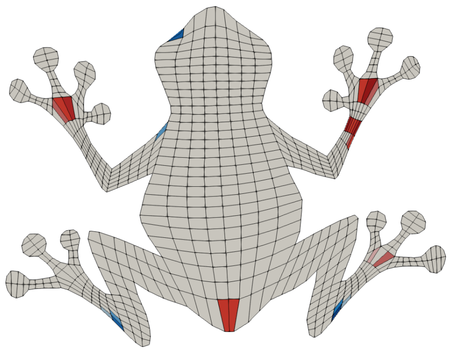
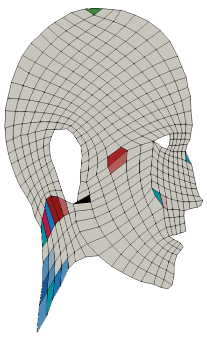
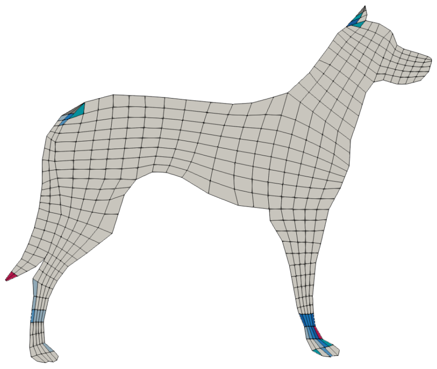
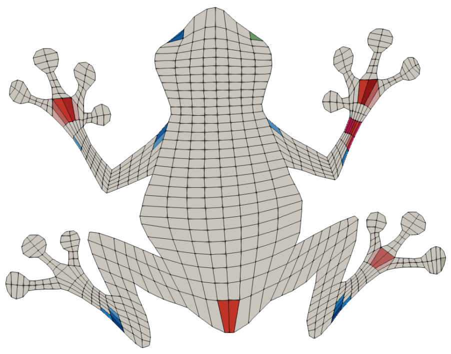
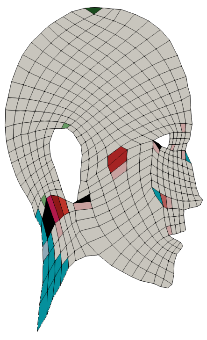
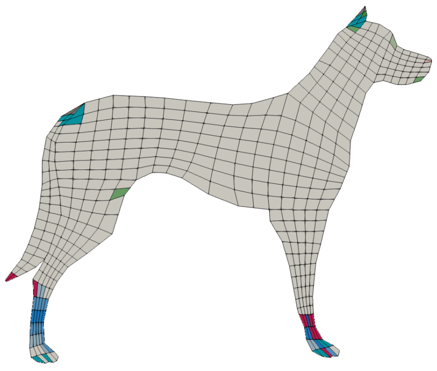
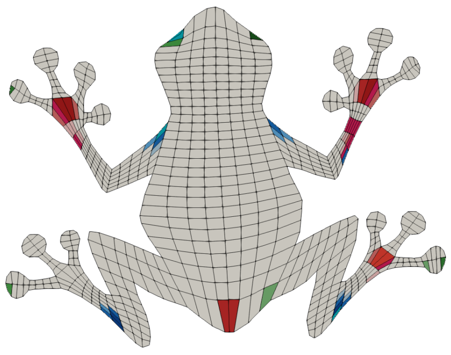
Face profile	Dog	Frog
		 1 19 24 23 20 18 25 21 22 2 3 4 5 17 16 13 12 6 7 15 14 10 8 9 11
$\overline{p} = 5\%$ of low quality elements		
		
$\overline{p} = 10\%$ of low quality elements		
		
$\overline{p} = 20\%$ of low quality elements		
		

Table 5 – Quantities of low quality elements, according to the metrics used, for geometries with multi-blocks.

$\overline{p}$	$\lvert U'-\overline{E_\cup}\rvert$	$\lvert\overline{E_r}\rvert$	$\lvert\overline{E_\alpha}\rvert$	$\lvert\overline{E_c}\rvert$	$\lvert\overline{E_{r,\alpha}}\rvert$	$\lvert\overline{E_{r,c}}\rvert$	$\lvert\overline{E_{\alpha,c}}\rvert$	$\lvert U-U'\rvert$	$\lvert\overline{E_\cap}\rvert$
NACA - 2 blocks and 576 elements									
5%	567	5	3	4	0	0	3	0	0
10%	567	5	4	5	0	1	4	0	0
20%	545	7	27	23	4	5	21	0	4
Turtle - 6 blocks and 581 elements									
5%	570	3	0	10	0	2	0	0	0
10%	553	11	11	15	0	5	4	0	0
20%	543	15	28	26	5	13	18	0	5
Plane - 8 blocks and 868 elements									
5%	850	5	13	9	0	2	7	0	0
10%	841	7	19	10	0	2	7	0	0
20%	798	15	56	20	2	6	15	0	2
Face profile - 7 blocks and 474 elements									
5%	457	6	0	12	0	1	0	0	0
10%	452	10	6	14	1	3	5	0	1
20%	440	16	21	21	4	5	19	0	4
Dog - 8 blocks and 258 elements									
5%	414	3	4	18	0	2	4	4	0
10%	401	3	5	32	0	3	5	4	0
20%	370	12	20	57	0	11	15	4	0
Frog - 25 blocks and 566 elements									
5%	727	20	0	8	0	0	0	0	0
10%	712	28	2	18	0	5	0	0	0
20%	685	38	11	42	0	18	3	0	0

Source: The authors.

Acknowledgments

The first author would like to thank CAPES (N° 1602873/2016-4) for financial support. The authors wish to thank the Laboratory Simulation and Numerical Analysis (LabSAN) of Londrina State University.

References

ALMEIDA, J.; LOBAO, D.; STAMPA, C.; ALVAREZ, G. Multi-block technique applied to Navier-Stokes equations in two dimensions, *Semina*: Ciências Exatas e Tecnológicas, Londrina, v. 39, n. 2, p. 115-124, 2018. DOI: http://dx.doi.org/10.5433/1679-0375.2018v39n2p115.

BELINELLI, E. O.; NATTI, P. L.; ROMEIRO, N. M. L.; CIRILO, E. R.; FANTIN, L. H.; OLIVEIRA, K. B.; CANTERI, M. G.; NATTI, E. R. T. Geração de malha para descrever a dispersão da ferrugem da soja no Paraná. *In*: RIBEIRO, Júlio Cesar (org.). *Ciências exatas e da terra*: conhecimentos estratégicos para o desenvolvimento do país. Ponta Grossa: Atena Editora, 2020. p. 225-239.

BOROUCHAKI, H.; FREY, P. J. Adaptive triangular–quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering*, Chichester, v. 45, n. 5, p. 915-934, 1998.

BURDEN, R. L.; FAIRES, J. D.; BURDEN, A. M. *Numerical analysis*. Boston: Cengage Learning, 2015.

CAI, X.; LANGTANGEN, H. P.; MOE, H. On the performance of the Python programming language for serial and parallel scientific computations. *Scientific Programming*, New York, v. 13, n. 1, p. 31-56, 2005.

CIRILO, E. R.; BORTOLI, A. L. Cubic splines for trachea and bronchial tubes grid generation. *Semina*: Ciências Exatas e Tecnológicas, Londrina, v. 27, n. 2, p. 147-155, 2006. DOI: http://dx.doi.org/10.5433/1679-0375.2006v27n2p147.

CIRILO, E. R.; BARBA, A. N. D.; NATTI, P. L.; ROMEIRO, N. M. L. A numerical model based on the curvilinear coordinate system for the MAC method simplified. *Semina*: Ciências Exatas e Tecnológicas, Londrina, v. 39, n. 2 p. 87, 2018.

COELHO, F. U.; LOURENCO, M. L. *Curso de álgebra linear*. São Paulo: Edusp, 2001.

DE BORTOLI, A. L. *Introdução à dinâmica de fluidos computacional*l. Porto Alegre: Editora da UFRG, 2000.

FORTUNA, A. O. *Técnicas computacionais para dinâmica de fluidos*: Conceitos básicos e aplicações. São Paulo: Edusp, 2012.

GONZALEZ, R. C.; WOODS, R. E. *Processamento digital de imagens*. São Paulo: Addison Wesley Pearson Brasil, 2011.

GOSE, E.; JOHNSONBAUGH, R.; JOST, S. *Pattern recognition and image analysis*. Saddle River: Prentice-Hall, 1996.

HSU, J.; JAMESON, A. An implicit-explicit hybrid scheme for calculating complex unsteady flows. *American Institute of Aeronautics and Astronautics*, Reston, p. 1-10, 2002. DOI: https://doi.org/10.2514/6.2002-714.

KOOMULLIL, R.; SONI B.; SINGH R. A comprehensive generalized mesh system for CFD applications. *Mathematics and Computers in Simulation*, Amsterdam, v. 78, p. 605-617, 2008.

JOHNEN, A.; ERNST, D.; GEUZAINE, C. Sequential decision-making approach for quadrangular mesh generation, *Engineering with Computers*, New York , v. 31, n. 4, p. 729-735, 2015. DOI: http://dx.doi.org/10.1007/s00366-014-0383-9.

LAIPING, Z.; ZHONG, Z.; XINGHUA, C., XITHAMES, H. A 3D hybrid grid generation technique and a multigrid/parallel algorithm based on anisotropic agglomeration approach. *Chinese Journal of Aeronautics*, [*s. l.*], v. 26, n. 1, p. 47-62, 2013.

MAGANIN, J.; ROMEIRO, N. M. L.; CIRILO, E. R.; NATTI, P. L. Simulation of a mathematical model of tumoral growth using finite differences. *Brazilian Journal of Development*, Curitiba, v. 6, p. 87696-87709, 2020. DOI: https://doi.org/10.34117/bjdv6n11-261

MALISKA, C. R. *Transferência de calor e mecânica dos fluidos computacional*. São Paulo: LTC, 2004.

NAOZUKA, G. T. *Geração e análise de qualidade de malhas computacionais em coordenadas curvilíneas*. 2018. Master's (Dissertation in Computer Science) - Londrina State University, Londrina, 2018.

PARDO, S. R.; NATTI, P. L.; ROMEIRO, N. M. L.; CIRILO, E. R. A transport modeling of the carbon-nitrogen cycle at Igapó I Lake-Londrina, Paraná State, Brazil. *Acta Scientiarum*: Technology, Maringá, v. 34, p. 217-226, 2012. DOI: 10.4025/actascitechnol.v34i2.11792

PARK, J.; SHONTZ, J. M. Two derivative-free optimization algorithms for mesh quality improvement. *Procedia Computer Science*, [Amsterdam], v. 1, n. 1, p. 387-396, 2010. DOI: https://doi.org/10.1016/j.procs.2010.04.042.

R CORE TEAM. *A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna: R Core Team, 2020. Available from: <https://www.R-project.org/>. Acess in: 20 aug. 2020

ROMEIRO, N. L. M.; CIRILO, E. R.; NATTI, P. L.; DOY OKAMOTO, L. M.; JULIANOTTI, T. Chimney height, a determining factor in the dispersion of pollutants and their concentration. *World Journal of Engineering and Technology*, [*s. l.*], v. 9, p. 173-193, 2021. DOI: 10.4236/wjet.2021.91013.

ROMEIRO, N. L. M.; CASTRO, R. G. S.; CIRILO, E. R.; NATTI, P. L. Local calibration of coliforms parameters of water quality problem at Igapó I Lake - Londrina, Paraná, Brazil. *Ecological Modelling*, Amsterdam, v. 222, p. 1888-1896, 2011. DOI: https://doi.org/10.1016/j.ecolmodel.2011.03.018.

ROMEIRO, N. L. M.; MANGILI, F. B.; COSTANZI, R. N.; CIRILO, E. R.; NATTI, P. L. Numerical simulation of BOD5 dynamics in Igapó I lake, Londrina, Paraná, Brazil: Experimental measurement and mathematical modeling. *Semina*: Ciências Exatas e Tecnológicas, Londrina, v. 38, p. 50-58, 2017. DOI: http://dx.doi.org/10.5433/1679-0375.2017v38n2p50.

RUSSUM, G. *Python reference manual, Centre for Mathematics and Computer Science*. Amsterdam: The Netherlands, 1995. Available from: <http://www.ncstrl.org:8900/ncstrl/servlet/search?formname=detail&id=oai%3Ancstrlh%3Aercim_cwi%3Aercim.cwi%2F%2FCS-R9525>. Access in: oct. 2018.

SAITA, T. M.; NATTI, P. L.; CIRILO, E. R.; ROMEIRO, N. L. M.; CANDEZANO, M. A. C.; ACUNA, R. A. B.; MORENO, L. C. G. Simulação numérica da dinâmica de coliformes fecais no lago Luruaco, Colômbia. *Trends in Applied and Computational Mathematics*, São Carlos, v. 18, p. 435- 447, 2017. DOI: https://doi.org/10.5540/tema.2017.018.03.0435.

SCIPY-NUMPY. Available from: <http://www.numpy.org/>. Access in: oct. 2020.

SCIPY-MATPLOTLIB. Available from: <http://matplotlib.org/>. Access in: oct. 2020.

THOMPSON, J. F.; THAMES, F. C.; MASTIN, C. W. TOMCAT - A Code for Numerical Generation of Boundary Fitted Curvilinear Coordinate Systems on Fields Containing Any Number of Arbitrary Two-Dimensional Bodies. *Journal of Computational Physics*, San Diego, v. 24, p. 274- 302, 1977. DOI: https://doi.org/10.1016/0021-9991(77)90038-9.

THOMPSON, J. F.; WARSI, Z. U. A.; MASTIN, C. W. *Numerical grid generation*: foundations and applications. New York: Elsevier Science Publishing, 1985.

THOMPSON, J. F.; SONI, B. K.; WEATHERILL, N. P. *Handbook of Grid Generation*. Florida: CRC Press, 1998.

Received: Oct. 17, 2020
Accepted: Apr. 30, 2021
Published: May 14, 2021