# Quasi-isotropic initial triangulation of NURBS surfaces

**D. H. Adán**[1], **R. Cardoso**[2]

[1] *Daniel Herrero Adán (corresponding autor)*
*daniel2.herreroadan@live.uwe.ac.uk*
*Department of Engineering Design and Mathematics. University of the West of England Bristol.*
*Frenchay Campus, Coldharbour Lane, Bristol, BS16 1QY, United Kingdom*
*+44 (0) 7780020820*

[2] *Rui Cardoso*
*Rui.cardoso@brunel.ac.uk*
*Department of Mechanical and Aerospace Engineering. Brunel University London.*
*Kingston Lane, Uxbridge, Middlesex, UB8 3PH, United Kingdom*
*+44 (0) 1895268427*

## Abstract.

Isotropic triangulation of NURBS surfaces provides high quality triangular meshes, where all triangles are equilateral. This isotropy increases representation quality and analysis accuracy. We introduce a new algorithm to generate quasi-isotropic triangulation on NURBS surfaces at once, with no prior meshing. The procedure consists of one front made of vertexes that advances in a divergence manner avoiding front collision. Vertexes are calculated by intersecting arcs whose radius is estimated by trapezoidal rule integration of directional derivatives. The parameter space is discretized in partitions such that the error of trapezoidal rule is controlled efficiently. A new space, called pattern space, is used to infer the direction of the arcs' intersection. Derivatives, whose analytical computation is expensive, are estimated by NURBS surface fitting procedures, which increases the speed of the process. The resultant algorithm is robust and efficient. The mesh achieved possesses most of the triangles equilateral and with high uniformity of sizes. The performance is evaluated by measuring angles, vertex valences and size uniformity in different numerical examples.

**Keywords**. NURBS, isotropic triangulation, initial mesh, pattern space, outside limits vertexes.

| | | |
|---|---|---|
| Number of words: | Paper: | 7800 |
| | Appendixes: | 3400 |
| | Total: | 11200 |

## 1. INTRODUCTION

Non-uniform rational B-spline (NURBS) for curves and surfaces are ubiquitous in computer aided design (CAD) representation. In addition, NURBS became part of analysis due to the so-called Isogeometric Analysis (IGA) [1,2].

Surface representation in CAD environment is made of elements whose vertexes lie on the surface. This discretization into elements is called meshing or tessellation and represents an open problem still evolving [3]. Tessellation made of triangles, called triangulation, is widely used due to its facility of capturing any shape. Triangulation quality may be characterized by two parameters: angles of triangles corners and vertexes valences (number of triangles attached to each vertex), both measuring distortion of the triangles.

One triangulation is isotropic when it matches the two isotropy conditions: all its angles equal to 60 degrees and all valences are six. This situation only happens for a hypothetic triangulation with no boundaries, i.e. infinite mesh. We state that one bounded (non-infinite) triangulation is quasi-isotropic if only the triangles that are influenced by the contours do not match the isotropy conditions. The rest of the triangles, that are away from the boundaries form an isotropic mesh.

This work presents a new algorithm for computing quasi-isotropic triangulation on a given set of NURBS surfaces with no preliminary mesh. It provides high quality mesh regardless of the surface shape or parametrization.

### *1.1. Triangulation of parametric surfaces*

There are three types of triangulation techniques: direct, parametric and hybrid triangulations [4].

Direct approaches compute the vertexes of triangles on the surface physical space. The three main methods within this type are the Delaunay triangulation [5,6], the advancing front technique [7,8,9,6] and the octree division [10,11]. Collision of two different fronts is susceptible of appearing in advancing front methods, which generates conflicts for the computation of new triangle vertexes.

Parametric approaches compute the triangulation in the parametric domain [12,13,14]. These methods lack uniformity for the resultant triangles for the case when the parametrization is not uniform.

Hybrid approaches, which is a mix of the two previous methods, cover most of the publications within the last two decades. For example, in [15,16], surfaces are tessellated by primary coarse triangulation in the parameter space and then the quality is increased by the use of Delaunay methods. In [17], sequential triangulation was developed to reduce the memory usage of the CPU. An initial mesh was generated by Delaunay triangulation and then extra vertexes were added where curvature is more pronounced in the physical space. In [18] three different linear parametrization techniques for refining one initial triangulation were presented. The algorithm shown in [19] triangulated surfaces, minimizing the number of triangles and at the same time controlling the error from triangular discretization of the surface. The initial mesh assumed the edges were already discretized. The resulting tessellation has vertexes density which is a function of surface's curvature. In [20] an automatic triangulation was presented; it starts from a preliminary coarse triangulation that is refined and improved in two stages.

### *1.2. Isotropic meshes*

The ideal isotropic mesh is defined with vertex valences equal to six and all angles equal to 60 degrees. One mesh may be approached closer to an isotropic mesh by four local operations used iteratively: edge collapsing, edge splitting, edge flipping and vertex relocation.

Surazhsky et al. [21] developed an isotropic remeshing technique to be applied to an initial mesh in three stages: generation of vertexes, initial vertex partition and modification based on a density function to achieve isotropy. The error diffusion algorithm was used for initial geometry sampling and then that mesh was modified in order to approximate it closer to an isotropic arrangement [22]. Yang and Choi [23] introduced an efficient algorithm for the computation of restricted Voronoi diagrams (RVD) repeatedly so that it could come closer to isotropic triangulation. Isotropic meshes do not apply only to triangulation, but also to other type of meshes such as the ones with quads or hexahedral elements, see for example [24,25].

### *1.3. Proposed method*

In the mentioned triangulation techniques, a preliminary coarse mesh is first created and then modified to enhance the isotropy. Our method achieves quasi-isotropic mesh at once with no previous triangulation required. It estimates the physical coordinates of vertexes by using integration of paths in the parameter space. In the previous work of Tsai et

al. [26] a similar technique was used, but the difference with this current work is that the path can have any orientation rather than being restrained over to orthogonal parameter directions $\xi$ or $\eta$. The process consists of an advancing front method but, however, avoiding the colliding fronts since the front shape is always divergent in the physical space.

### *1.4. Article structure*

Section 2 introduces the theoretical background. Section 3 gives a general idea of the triangulation process and defines some concepts that are used in the rest of the work. Section 4 explains the discretization of NURBS curves, that will be used for construction of the surface edges. That procedure is the one-dimensional version of the surface triangulation. Triangulation of surfaces involves more steps than the discretization of curves and it is split into two main sections: section 5 explains the vertexes calculation and section 6 details the triangulation itself. Examples are provided in section 7 and, finally, section 8 presents conclusions and potential future work.

## 2. THEORETICAL FORMULATION

### *2.1. NURBS*

A NURBS entity (curve or surface) is defined in both the parameter and physical spaces. The number of dimensions for the parameter and physical spaces are $c$ and $d$ respectively, with $c < d$. For curves $c = 1$ and for surfaces $c = 2$. In this work we assume $d = 3$. **Figure 1** shows one NURBS surface example.

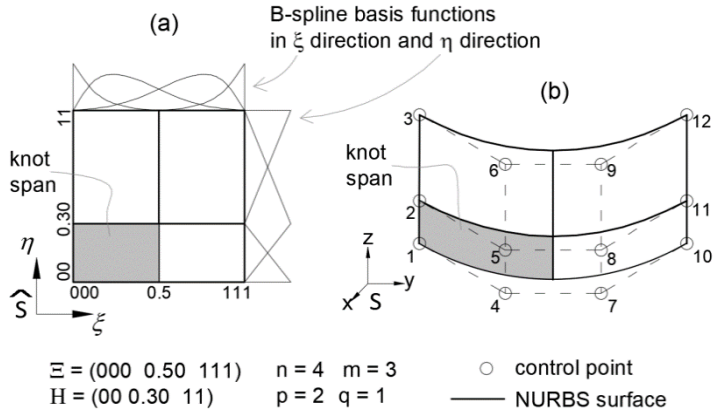


**Figure 1.** NURBS surface parameter (a) and physical (b) spaces.

NURBS entities are the evolution of Bézier entities [27,28] that are formed by linear combination of Bernštein polynomials [29](Bernšteın, 1912). NURBS entities have a set of control points whose coordinates in the physical space are defined by $\boldsymbol{P}$ and weights defined by $\boldsymbol{w}$. Each control point has attached one NURBS basis function. Parametrization is given by knot vectors, with one knot vector per each parameter direction. The knot vector is a sequence of numbers $\boldsymbol{\Xi} = \{\xi_1\ \xi_2\ \cdots\ \xi_a\ \cdots\ \xi_{n+p+1}\}$ with $\xi_i \leq \xi_{i+1}$. The components of $\boldsymbol{\Xi}$, called knots, are located in the parameter space. Stretches between knots are called knot spans. This work assumes knot values from 0 to 1 and open knot vectors, i.e. the first and last $p+1$ knots are repeated. The number of knots is equal to $p+n+1$, where $p$ is de degree of the NURBS functions and $n$ is the number of control points in the parameter direction. **Table 1** shows the nomenclature used for curves and surfaces.

**Table 1.** NURBS nomenclature.

| | **Curve** | **Surface** | |
|---|---|---|---|
| | | **Direction 1** | **Direction 2** |
| Parameter coordinates | $\xi$ | $\xi$ | $\eta$ |
| B-spline basis function | $N_i$ | $N_i$ | $M_j$ |
| NURBS basis function | $R_i$ | $R_{i,j}$ | |
| Number of control points | $n$ | $n$ | $m$ |
| Degree | $p$ | $p$ | $q$ |
| Knot vector | $\boldsymbol{\Xi}$ | $\boldsymbol{\Xi}$ | $H$ |
| Physical space | $\boldsymbol{C}$ | $\boldsymbol{S}$ | |
| Parameter space | $\widehat{\boldsymbol{C}}$ | $\widehat{\boldsymbol{S}}$ | |

A NURBS entity is generated by mapping $\mathbb{R}^c \rightarrow \mathbb{R}^d$ as detailed in equations (1) and (2) for curves and surfaces respectively, where $R$ are the NURBS basis functions.

$$\boldsymbol{C}(\xi) = \sum_{i=1}^{n} R_i^p(\xi)\boldsymbol{P}_i \tag{1}$$

$$\boldsymbol{S}(\xi,\eta) = \sum_{i=1}^{n}\sum_{j=1}^{m} R_{i,j}^{p,q}(\xi,\eta)\boldsymbol{P}_{i,j} \tag{2}$$

The NURBS basis functions are calculated as in (3) and (4).

$$R_i^p(\xi) = \frac{N_{i,p}(\xi)\, w_i}{\sum_{\hat{\imath}=1}^{n} N_{\hat{\imath},p}(\xi)\, w_{\hat{\imath}}} \tag{3}$$

$$R_{i,j}^{p,q}(\xi,\eta) = \frac{N_{i,p}(\xi)\, M_{j,q}(\eta)\, w_{i,j}}{\sum_{\hat{\imath}=1}^{n}\sum_{\hat{\jmath}=1}^{m} N_{\hat{\imath},p}(\xi)\, M_{\hat{\jmath},q}(\eta)\; w_{\hat{\imath},\hat{\jmath}}} \tag{4}$$

B-spline functions $N$ can be calculated with the Cox-De Boor iterative equations (5) and (6) [30,31]. **Figure 1** (a) also includes the B-spline functions in both directions.

For zero degree ($p = 0$):

$$N_{i,0}(\xi) = \begin{cases} 1 \text{ if } \xi_i \leq \xi < \xi_{i+1} \\ 0 \quad \text{otherwise} \end{cases} \tag{5}$$

For degrees 1 and higher ($p > 0$):

$$N_{i,p} = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i}\, N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}}\, N_{i+1,p-1}(\xi) \tag{6}$$

### *2.2. Length of paths on NURBS*

The physical path's length for a NURBS curve between parameter coordinates $\xi^a$ and $\xi^b$, corresponding to physical coordinates $\boldsymbol{x}^a$ and $\boldsymbol{x}^b$, is given by the integral detailed in equation (7).

$$L_{ab} = \int_{\xi^a}^{\xi^b} \left\| \boldsymbol{C}_{,\xi} \right\| d\xi \tag{7}$$

Where $\left\| \boldsymbol{C}_{,\xi} \right\|$ is the norm of the curve derivative w.r.t. parameter coordinate $\xi$. See **Figure 2** for clarity.

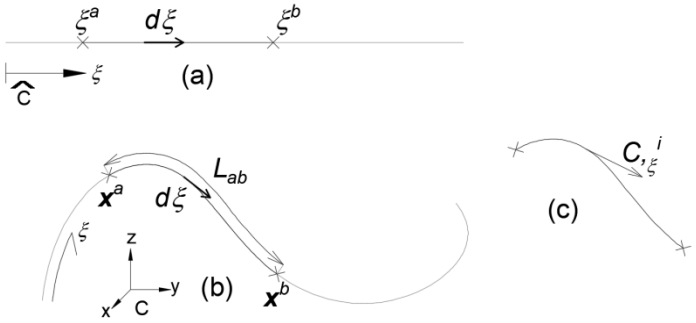


**Figure 2.** Curve path between $a$ and $b$ in parameter (a) and physical (b) spaces. Derivative at $i$th point (c).

The physical length of a path on a NURBS surface between parameter coordinates $\boldsymbol{\xi}^a$ to $\boldsymbol{\xi}^b$, that forms a $\theta$ angle w.r.t. the horizontal direction and

corresponds to physical coordinates $\boldsymbol{x}^a$ and $\boldsymbol{x}^b$, is given by the integral of the $\theta$-directional derivative norm along the path as follows:

$$L_{ab} = \int_{\xi^a}^{\xi^b} \| \boldsymbol{S}_{,\lambda} \| \, d\lambda \tag{8}$$

Where $d\lambda$ represents an infinitesimal increment in the parameter domain with orientation $\theta$, and $\boldsymbol{S}_{,\lambda}$ is the $\theta$ - directional derivative, i.e. $\boldsymbol{S}_{,\lambda} = (Sx_{,\lambda}, Sy_{,\lambda} Sz_{,\lambda})^T$ that is computed as per equation (9).

$$\boldsymbol{S}_{,\lambda} = \boldsymbol{S}_{,\xi} \cos\theta + \boldsymbol{S}_{,\eta} \sin\theta \tag{9}$$

The distance $L_{ab}$ lies onto the surface (physical space) but the shortest distance between $\boldsymbol{x}^a$ and $\boldsymbol{x}^b$ might be less (see **Figure 3** (b)).

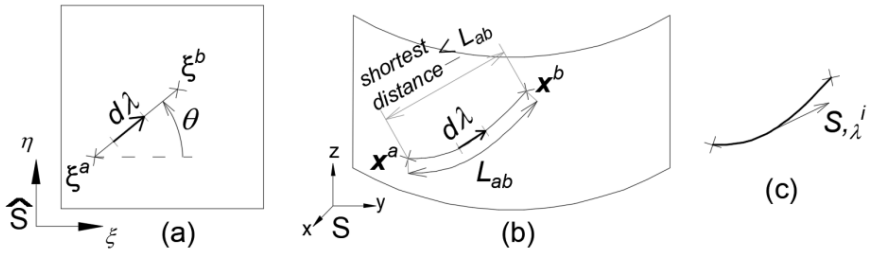


**Figure 3.** Surface path between $a$ and $b$ in parameter (a) and physical (b) spaces. Directional derivative at $i$th point (c).

We can generalize equations (7) and (8) by calling $\boldsymbol{H}$ to the NURBS entity and $\mu$ to the parameter ($\xi$, $\boldsymbol{\xi}$ or $\lambda$). Then these two expressions may be written as in equation (10), with $h = \|\boldsymbol{H}_{,\mu}\|$.

$$L_{ab} = \int_{\mu^a}^{\mu^b} h \, d\mu \tag{10}$$

### *2.3. Trapezoidal rule for path lengths*

The estimation of the path's length by the trapezoidal rule is expressed as:

$$L_{ab} \approx \frac{1}{2}(h^b + h^a) \, \Delta\mu \tag{11}$$

Under appropriate smoothness assumptions, there exists some point $\alpha$ in the integration interval such that the error is bounded, as expressed in equation (12) [32], where $\Delta\mu = (\mu^b - \mu^a)$. Since the location $\alpha$ is unknown, in this work we will evaluate the error at the initial and final locations for the following interval:

$$E \leq \frac{-1}{12} \left\| \boldsymbol{H}_{,\mu}^{\alpha} \right\|'' (\Delta\mu)^3 \tag{12}$$

The calculation of the derivatives $\left\| \boldsymbol{C}_{,\xi} \right\|''$ and $\left\| \boldsymbol{S}_{,\lambda} \right\|''$ is detailed in Appendix A. The error is expressed as percentage of the path's length:

$$Ep \leq 100 \frac{E}{L} \tag{13}$$

where $E$ is the absolute value of the error, computed as in equation (12) and $L$ is the estimated physical length of the path as in equation (11).

### 2.4. *Path Parameter Increment corresponding to a physical length (PPI)*

Let $L_{ab}$ be the length of a path that lies in the physical space of a NURBS entity whose end points are $a$ and $b$. Let $c$ be a third point along the path trajectory, either between the end points or beyond $b$ (see **Figure 4**). The Path Parameter Increment procedure (PPI) presented in this section finds the parameter coordinate $b$ ($\mu^b$), assuming that the physical coordinates of the three points and the parameter coordinates of $a$ and $c$ ($\mu^a$ and $\mu^c$) are known.

The trapezoidal rule between $a$ and $b$ is written as in equation (11) with $\Delta\mu$ and $h^b$ being unknowns in this case. To compute $\Delta\mu$ we use the third point $c$, whose derivative norm $h^c$ lies on the line $h$-$\Delta\mu$, as shown in **Figure 4** (c). Equation (14) is used, where it was assumed that $\Delta\mu = 0$ coincides with the $h^a$ location:

$$h(\Delta\mu) = \left(\frac{h^c - h^a}{\mu^c - \mu^a}\right) \Delta\mu + h^a \tag{14}$$

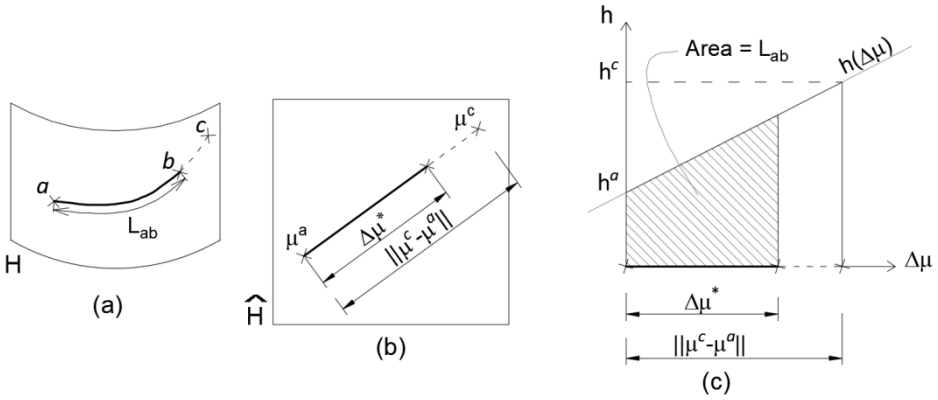


**Figure 4.** Path with points $a$, $b$ and $c$ in physical (a) and parameter (b) spaces. Line in the $h$-$\Delta\mu$ plane (c).

Substituting $h^b$ in equation (11) with the right-hand side of equation (14), we arrive to the quadratic equation for $\Delta\mu$, equation (15), where $m = \left(\frac{h^c - h^a}{\mu^c - \mu^a}\right)$. Among the two possible roots, the non-negative and within, or closest to, the interval $(\mu^a, \mu^c)$ corresponds to the searched increment $\Delta\mu^*$.

$$\Delta\mu^2\, m + \Delta\mu\, 2h^a - 2L_{ab} = 0 \tag{15}$$

Then, the coordinate $\mu^b$ is given by:

$$\mu^b = \mu^a + \Delta\mu^* \tag{16}$$

### *2.5. Orientation of a surface tangent vector in the parameter space*

Let $\boldsymbol{v}^k$ be a tangent vector to a surface at point $k$ in the physical space, the calculation of its orientation in the surface parameter space ($\theta$) will be presented in this section. $\boldsymbol{v}^k$ is a linear combination of *main derivatives* as shown in equation (17) (see **Figure 5**). Coefficients $c$ and $s$ are shortcuts to "$\mathcal{C}cos\theta$" and "$\mathcal{C}sin\theta$", respectively, with $\mathcal{C}$ being an unknown constant.

$$c\, \boldsymbol{S}^k_{,\xi} + s\, \boldsymbol{S}^k_{,\eta} = \boldsymbol{v}^k \tag{17}$$

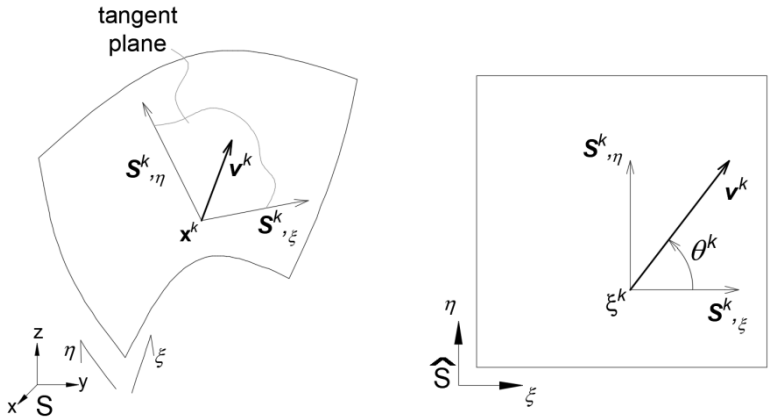


**Figure 5.** Vector $\boldsymbol{v}^k$ in in physical space and its orientation in the parameter space.

To compute the orientation $\theta$, the derivatives $\boldsymbol{S}^k_{,\xi}$ and $\boldsymbol{S}^k_{,\eta}$ are firstly calculated, then $c$ and $s$ are obtained from the system of equations (17). Finally, $\theta$ is calculated as in the following equation:

$$\theta = atan\left(\frac{S}{C}\right) \tag{18}$$

## 3. PRELIMINARIES

### *3.1. Pattern space*

In this work we introduce a new 2D space, named pattern space (S′), as a set of vertexes lying on a number of concentric regular hexagons separated by the distance $R_o = R\ sin60$. Vertexes are equally spaced at $R$. Due to the regular hexagonal arrangement, these vertexes form an isotropic triangulation in this space, see **Figure 6**.

The centre of the hexagons is located at origin (0,0). Hexagon one is the smallest with six vertexes and the rest of the hexagons grow concentrically with 12, 18, etc vertexes. Vertexes are numbered: the central is the first and the numbering increases for each hexagon that is generated. Inside one hexagon, numbers start at the right-hand side corner and move counter-clockwise (**Figure 6** shows some vertex numbers). Considering this additional new space for the hexagons (pattern space), three spaces are now involved for each surface: pattern, parameter and physical spaces.

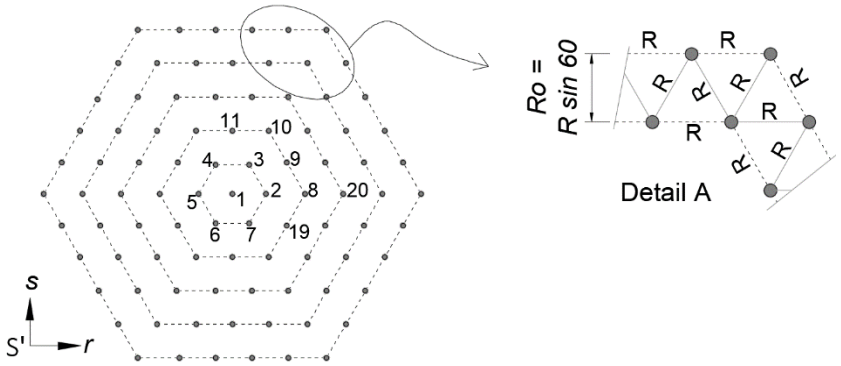


**Figure 6.** Pattern space with five contours (dashed lines). Dots represent vertexes.

### *3.2. A whole view: the QIT algorithm*

The Quasi-Isotropic initial Triangulation (*QIT*) algorithm purpose is to mesh a set of contiguous NURBS surfaces, each of them bounded by four edge curves, with conformal triangulations between contiguous surfaces and with a high degree of isotropy.

The strategy is to obtain the image of the pattern space vertexes onto the surface physical space, which leads naturally to a quasi-isotropic mesh given the pattern space arrangement indicated in section 3.1. The target distance between vertexes is called $R$ and is introduced by the user.

**Figure 7** details the flowchart for the process that is briefed in this section. The algorithm provides the vertexes coordinates and their relationship in the triangulation (the *connectivity matrix*). The inputs required for the *QIT* algorithm are:

- Target triangles edge distance ($R$): this is the distance that ideally all the triangles edges should have in the physical surface.
- Threshold distance in the physical space from the surface edges to discriminate *surface vertexes*.
- Tolerance for the error, in percentage, when computing path lengths (recall section 2.3).
- NURBS original data of the surfaces.

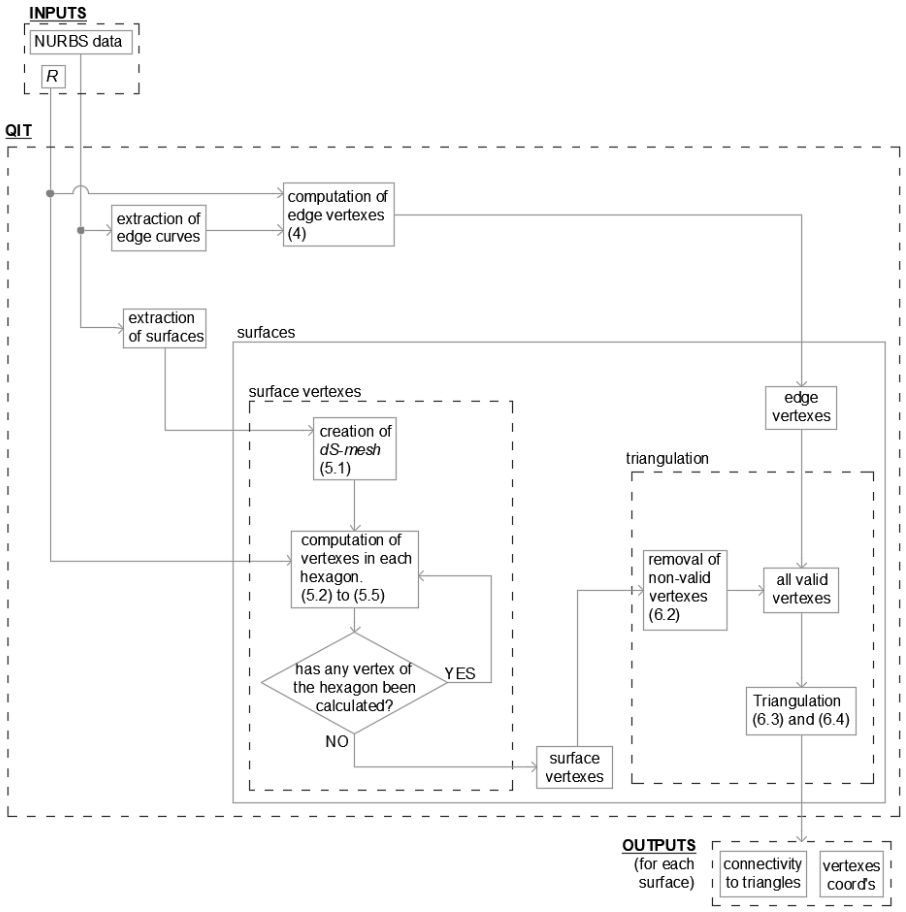


**Figure 7.** *QIT* algorithm flowchart. Related sections of the paper are in curved brackets.

Edges between two adjacent surfaces produce duplicated curves, one per surface. In order to compute the vertexes in both curves with the same

coordinates and achieve conformity between them, they must be considered as a single curve instead (see **Figure 8**). All curves are extracted and those duplicated are merged into one. Their vertexes are obtained according to the *R* distance, these are called *edge vertexes*. Section 4 delivers more details of this process.

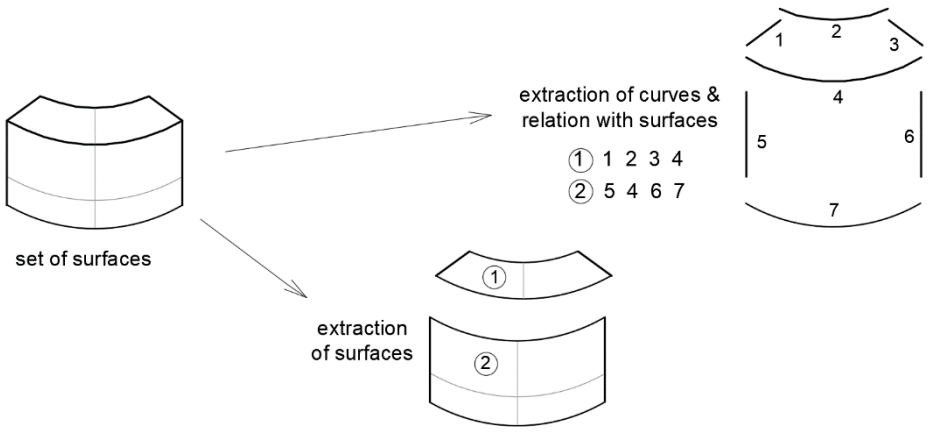


**Figure 8.** Extraction of surfaces and curves and their relationship.

Each surface is triangulated separately: the *surface vertexes* are calculated, the corresponding *edge vertexes* are added and all vertexes are triangulated. Surface vertexes calculation is outlined in **Figure 9**. The surface parameter space is discretized in a mesh called *dS-mesh* (section 5.1). The first vertex is set at mid location and the rest of the vertexes are calculated in a 2D hexagonal wave propagation manner. Propagation stops at the hexagon with no vertexes computed (see sections 5.4 and 5.5). This procedure links vertexes in pattern space with their image in the physical space using the parameter space in between (see sections 5.2 and 5.3).

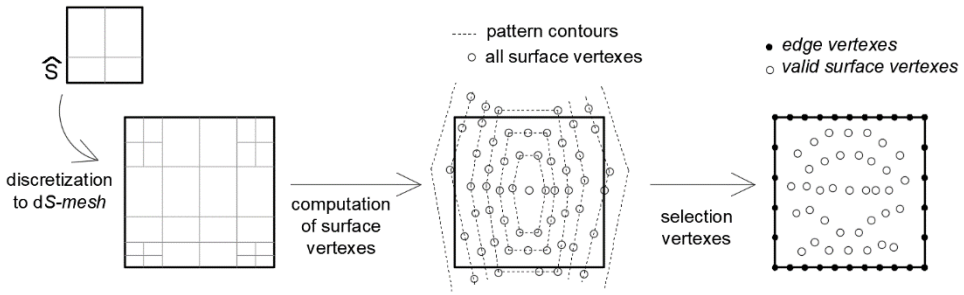


**Figure 9.** Computation of *surface vertexes*.

The advancing front algorithm is divergent in the physical space, therefore it is also in the parameter space (we assume the Jacobian of the NURBS

mapping to be strictly positive). That divergence is necessary because it avoids front collisions.

Previous to triangulation, *surface vertexes* outside the parameter limits are removed as well as those that are too close to the limits, since they would generate highly distorted triangles (see section 6.2). The remaining vertexes are called *valid*. Delaunay triangulation is carried out in the pattern space considering both, *edge* and valid *surface vertexes* (see section 6.3). Finally, the triangles at the edges of the surface might be improved by edge flipping, as shown in section 6.4. The triangulation process is illustrated in **Figure 10**.

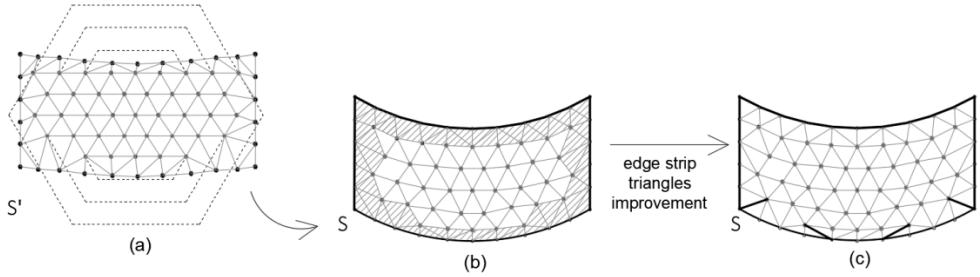


**Figure 10** Triangulation in the pattern space (a), image in the physical space (b) and improvement of edges (c).

### *3.3. Conventions and definitions*

Coordinates in the pattern, parameter and physical spaces are expressed as $\boldsymbol{r} = (r, s)$, $\boldsymbol{\xi} = (\xi, \eta)$ and $\boldsymbol{x} = (x, y, z)$, respectively. Coordinates at a specific point $a$ are written with $a$ as superscript, e.g. $\boldsymbol{r}^a$. The expression *vertex calculation* refers to the calculation of the vertex coordinates.

The definitions listed below are used within the next sections. **Figure 11** provides some examples of them for clarity.

- Path: straight line between two points in the parameter space, that has an image in the surface space, which is not straight in general.
- $\theta$ orientation: angle between a path and the horizontal axis in the parameter space.
- $\lambda$: surface parameter coordinate with orientation $\theta$.
- Path length: length of a path in the physical space (in general it is not the shortest).
- *Edge vertex*: vertex computed on edge curves.
- *Surface vertex*: vertex computed on the surface.
- *Main derivatives*: surface derivatives w.r.t. parameter directions $\xi$ and $\eta$.

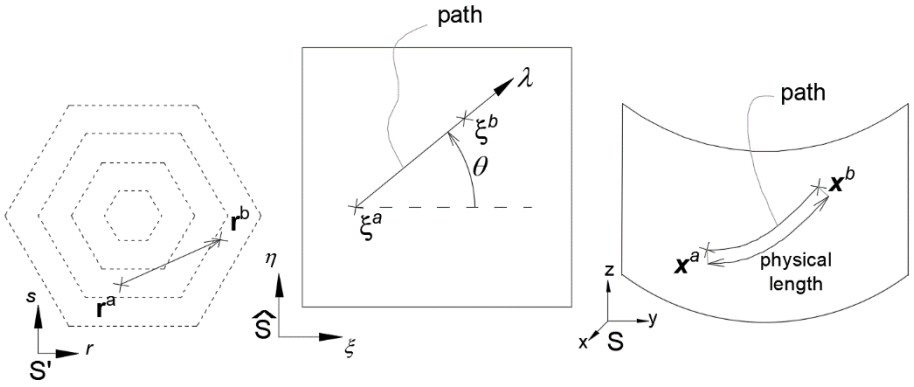


**Figure 11** Basic definitions used in the algorithm in pattern, parameter and physical spaces.

## 4. VERTEXES OF EDGE CURVES

The computation of vertexes for edge curves is explained in this section. The first step is a parameter space discretization into a *dC-mesh* in order to control the error of the estimated path lengths.

### *4.1. Discretization of the parameter space. The dC-mesh*

The *dC-mesh* is obtained by iterative division of the parameter space so that the error $E_p$ from equation (13) can be reduced for a path length estimation below a prescribed tolerance. We refer to the norm of the curve derivative $\|\boldsymbol{C}_{,\xi}\|$ by $h$ and its second derivative $\|\boldsymbol{C}_{,\xi}\|''$ by $h''$ (see Appendix A for the calculation of $h''$).

Initially, *dC-mesh* partitions coincide with non-void knot vector spans. Then, within each partition, the $E_p$ is computed and, in case it is greater than a prescribed tolerance, the element is halved. This iterative process ends whenever there is no partition with an error greater than the prescribed tolerance. The extremities of the partitions are called nodes. **Figure 12** provides one example. To compute the partitions length and error, $h$ and $h''$ are calculated for each node (recall equations (11) and (12)).

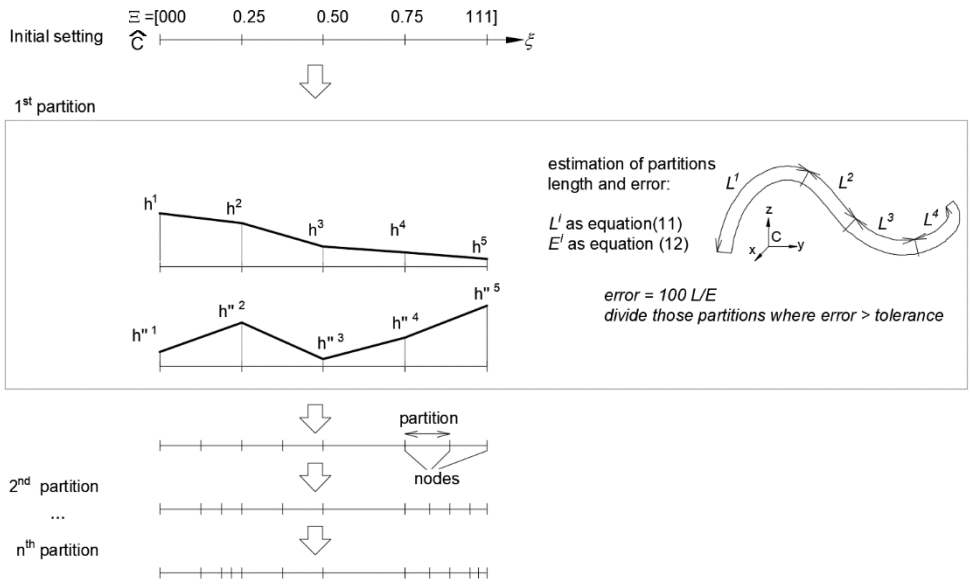


**Figure 12.** *dC-mesh* division process with first step detailed.

The error at some locations might be greater than the tolerance since the location $\alpha$, introduced in (12), is the initial or final node of the partition, whichever maximizes the error, but there might exist an intermediate value that leads to a higher error. In spite of this risk, results are satisfactory (refer to section 7).

### *4.2. Edge vertexes calculation*

To calculate the edge vertexes, the accumulated physical length up to each *dC-mesh* node, called $Lacc$, is estimated by using the trapezoidal rule from equation (11) applied to each partition, see **Figure 13**. The total estimated length of the curve is $L_{acc}^{end}$ and the accumulated length up to the previous node is $Lacc^{prev}$.

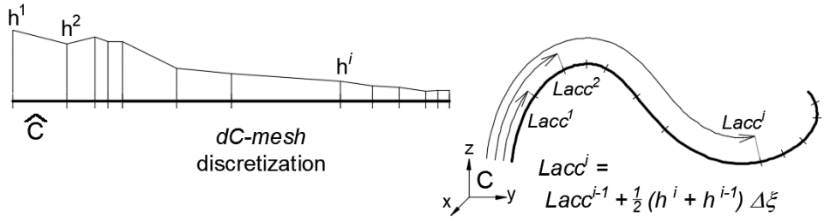


**Figure 13.** Computation of accumulated length to each dC-mesh node.

The target physical spacing between vertexes is not exactly $R$ but it is re-calculated to ensure that the resultant vertexes are equally spaced. The

updated spacing is called $Rc$ and is obtained as $Rc = L_{acc}^{end}/NS$, where $NS = round(L_{acc}^{end}/R)$ is the number of required segments between vertexes.

Accumulated target distances ($Ra$) are then sequentially searched. $Ra$ initially is set equal to $Rc$ and increases by $Rc$ in each step. The search first finds which partition of the *dC-mesh* contains $Ra$, using the accumulated physical lengths $Lacc$. Then, it estimates the parameter coordinate increment $\Delta\xi$ within that partition in order to achieve the distance $L = R_a - Lacc^{prev}$ by using the PPI algorithm from section 2.5. See one example in **Figure 14**.

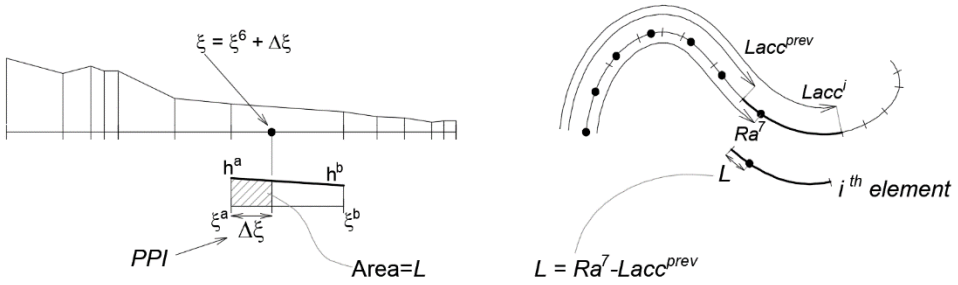


**Figure 14.** Example for the calculation of the seventh vertex.

The edge vertexes in the surface parameter space, required for triangulation (see section 6), are computed by using point projection techniques [34].

## 5. SURFACE VERTEXES

For the calculation of *surface vertexes* the surface must first be discretized into a *dS-mesh* (section 5.1). Sections 5.2 and 5.3 explain the two main algorithms used repeatedly in sections 5.4 and 5.5 for the computation of the surface vertexes.

### *5.1. Discretization of the parameter space. The dS-mesh*

The *dS-mesh* is obtained by iterative partition of the parameter space. Resultant rectangular partitions must be small enough such that the error in equations (12) and (13) for any patch length remains below a prescribed tolerance.

Six representative paths were selected within each partition of the *dS-mesh*, they are the four edges and its two diagonals. The partition error is the maximum error amongst these six paths. The second derivative $\|\boldsymbol{S}_{,\lambda}^{\alpha}\|''$, selected for error calculation (12), must be the maximum amongst the two

end points of the corresponding path. The calculation of these derivatives is detailed in Appendix A. **Figure 15** shows one example, with the fifth path detailed and where $\|\boldsymbol{S}_{,\lambda}\|''$ is called $h''$ for simplicity.

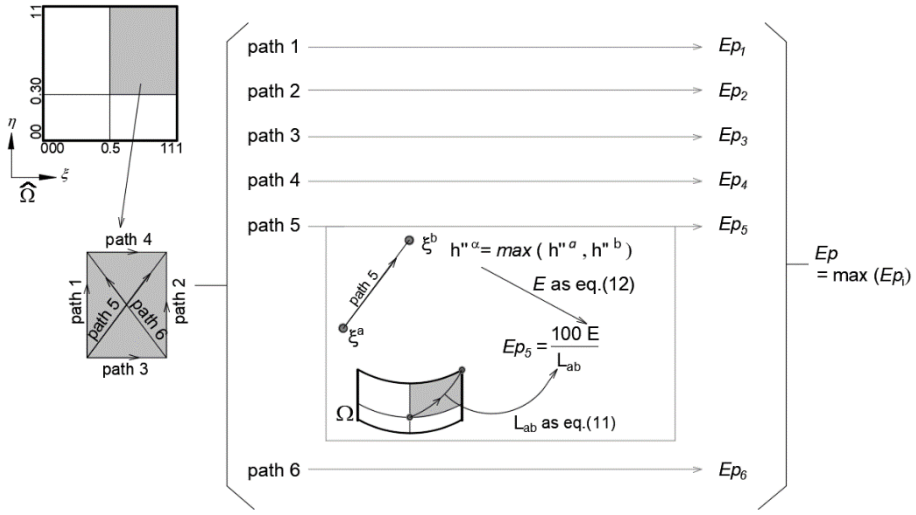


**Figure 15.** Error measurement in one partition of the *dS-mesh*.

Initially, partitions are the non-void knot spans. Error ($E_p$) is evaluated for each partition, which is divided into four rectangles if such error is greater than the tolerance. The division process ends whenever the error is smaller than the prescribed tolerance in all partitions.

Once the partitions are generated, the *dS-mesh* is extended beyond the surface parameter limits with so-called *perimeter partitions* and *corner partitions*. The *main derivatives* on these partitions are merely an extension of the derivatives at the edge limits. These partitions are semi-infinite, i.e. one end coincides with the surface parameter limit and the opposite goes to the infinite. This extension will be relevant in section 5.4. The whole process is depicted in **Figure 16**.

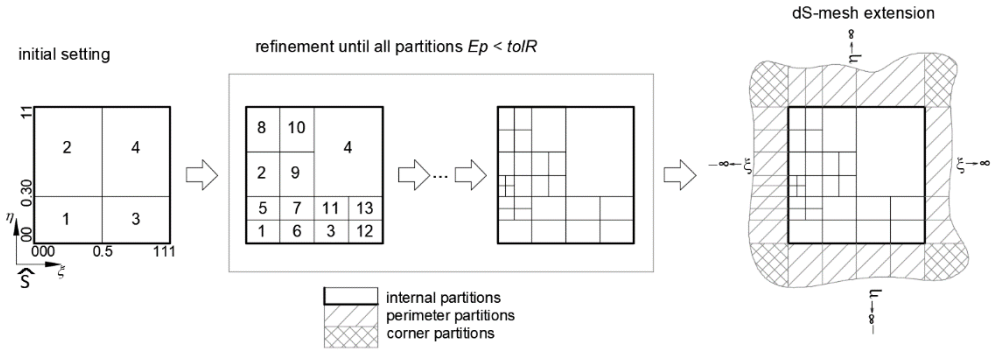


**Figure 16.** *dS-mesh* generation: initial setting, partition and extension with perimeter and corner partitions.

It is possible to find locations where the error is greater than the tolerance because the location $\alpha$ introduced in equation (12) lies at the start or at the end of the path (recall **Figure 15**) but there might exist an intermediate value that leads to a higher error. In addition, only six orientations for the paths are analysed event though there are infinite possibilities. In spite of this risk, results are satisfactory (refer to section 7).

### *5.2. End Parameter Position of a path given its physical length (EPP)*

This section explains the End Parameter Position procedure (EPP) that estimates the end location $\boldsymbol{\xi}^b$ of a path whose initial point coordinates $\boldsymbol{\xi}^a$, orientation $\theta$ and physical length are known a priori. The physical length is called target length and it is denoted by $R$, as shown in **Figure 17**.

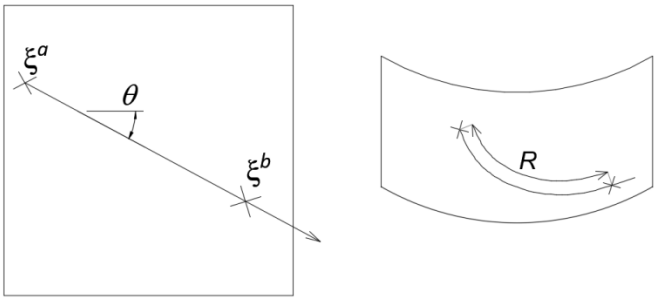


**Figure 17.** Path in parameter (left) and physical (right) spaces. The position $\boldsymbol{\xi}^b$ is the output of the EPP algorithm.

A semi-infinite line, starting at $\boldsymbol{\xi}^a$ and with orientation $\theta$ is defined (see **Figure 18**). The procedure is to move along this line computing at each time its intersection with *dS-mesh* edges and calculating the segment physical length by using the trapezoidal rule of equation (11). The $\theta$-directional derivatives required for equation (11) can be estimated as shown

in Appendix B. When the accumulated length of the segments goes beyond the target $R$, the parameter coordinate $\boldsymbol{\xi}^b$ is searched by PPI within the current segment (see section 2.5). Some examples are illustrated in **Figure 19**.

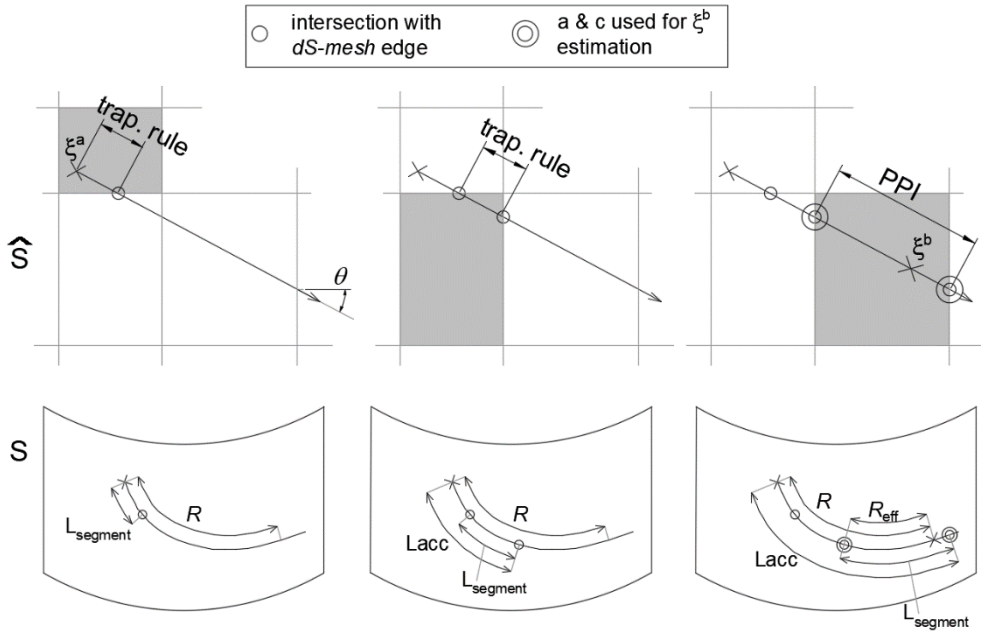


**Figure 18.** Estimation of different path increments to achieve the physical target distance $R$. Above represents the parameter space, with grey hatching the partition of *dS-mesh* involved. Below it is detailed the path in the physical space.

One special case happens when the ray passes the surface parameter limits and does not intersect any more partition edges. For this situation, the second trial point $c$ for PPI cannot be computed from the intersection with the *dS-mesh*. Instead, it is obtained by adding a certain distance along the $\theta$ direction. In this work, the diagonal length of the surface parameter space ($u_{diag}$) is used for that effect. **Figure 19** shows two examples representing this particular case.

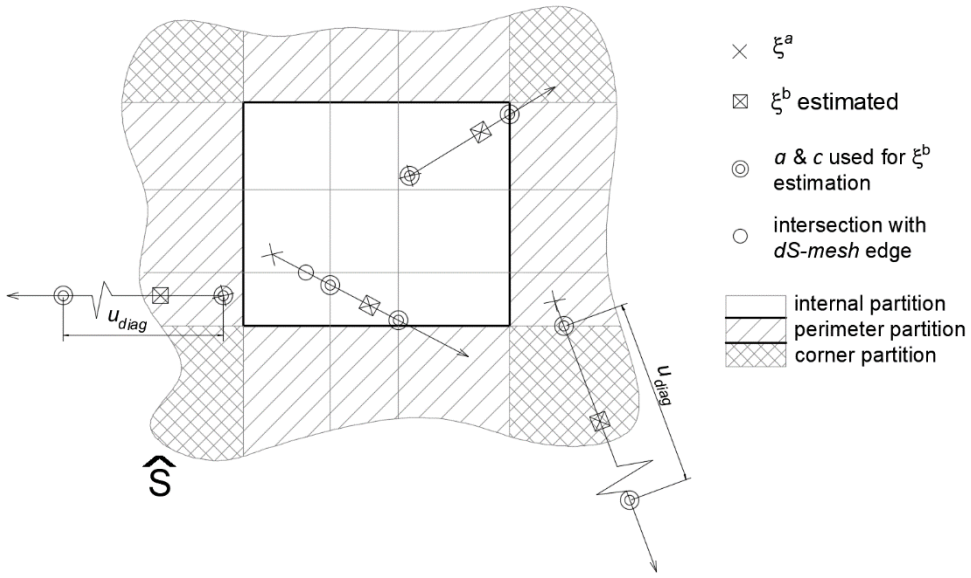


**Figure 19.** Examples of application of the EPP for some particular cases.

### *5.3. Intersection of two Arcs in Physical space (AIP)*

In this section a procedure called Arcs Intersection in Physical space (AIP) will compute the intersection (point $c$) of two arcs, $a$ and $b$, that lie on the physical space of the surface. Let us define one arc, onto the surface physical space centred at $\boldsymbol{x}^a$, by its radius ($R$), trial angle ($\beta$) and amplitude ($\beta^{amp}$), in this work $\beta^{amp} = 15$ degrees. The trial angle is the orientation of the arc bisector and the total arc angle is twice the amplitude (see **Figure 20**).

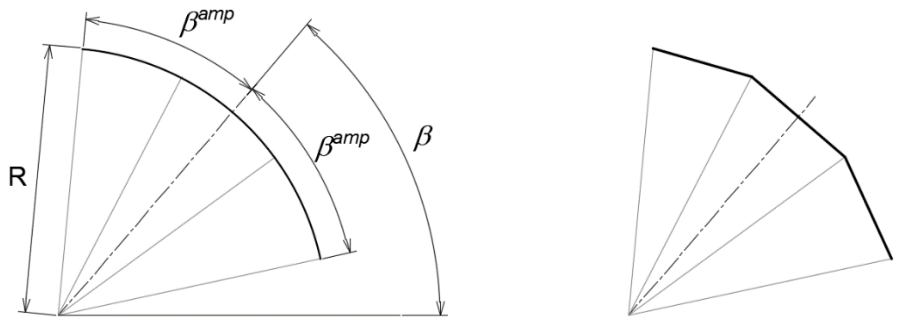


**Figure 20.** Definition of arc and discretization into three lines.

To find the intersection, arcs are first discretized in a number of lines ($NL$), as shown in **Figure 20** at the right. Hence, the number of points to compute per arc is $NL + 1$ (in this work $NL = 3$). Due to this discretization

procedure, a number of iterations is required to find the intersection point *c*. The iterative process ends when the difference between two consecutive intersections is less than a pre-established tolerance (in this work it is 1.0 %). After each iteration, the trial angles are re-oriented to the updated intersection and the amplitudes are also adjusted accordingly. The rest of this sub-section has two parts, one to explain the calculation of the end-points of the arc lines and another to describe the iterative process.

*Discretization of arcs:*

Let us define $\pi^a$ as the tangent plane to the surface at location $\boldsymbol{x}^a$, and $\boldsymbol{px}^a$ as the vector projected from vector $\boldsymbol{x}^b - \boldsymbol{x}^a$ onto $\pi^a$, with all of these vectors represented in the physical space (**Figure 21**).

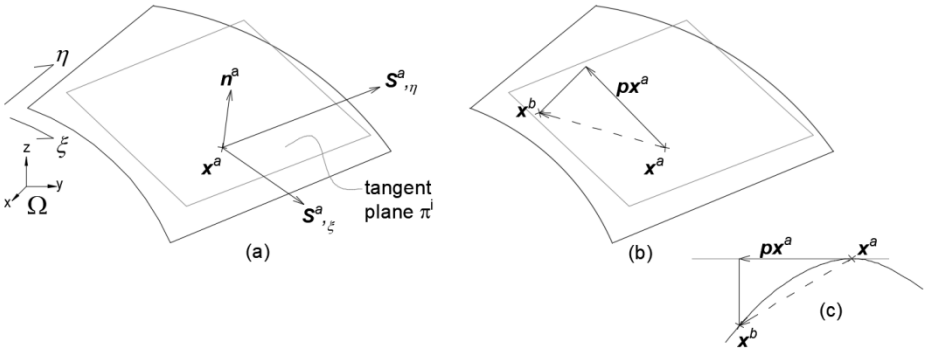


**Figure 21.** Tangent plane obtained by cross product of the derivatives to the surface (a). Projected vector $\boldsymbol{px}^a$ onto the $\pi^a$ plane (b). Front view of the projection's procedure at the right-bottom (c).

The trial angle $\beta^a$ is measured from vector $\boldsymbol{px}^a$ as shown in **Figure 22**. The value of $\beta^a$ for the first iteration is selected in the pattern space by using relative positions between pattern coordinates of points $\boldsymbol{r}^a$, $\boldsymbol{r}^b$ and $\boldsymbol{r}^c$. Their values are typically around -60 and +60 degrees for $\beta^a$ and $\beta^b$, respectively, except for the first hexagon (see section 5.4). The angles for the arc points vary from ${\beta^a}_1 = \beta^a - \beta^{amp}$ to ${\beta^a}_4 = \beta^a + \beta^{amp}$, with steps $\Delta\beta = 2\,\beta^{amp}/NL$, all within the $\pi^a$ plane.

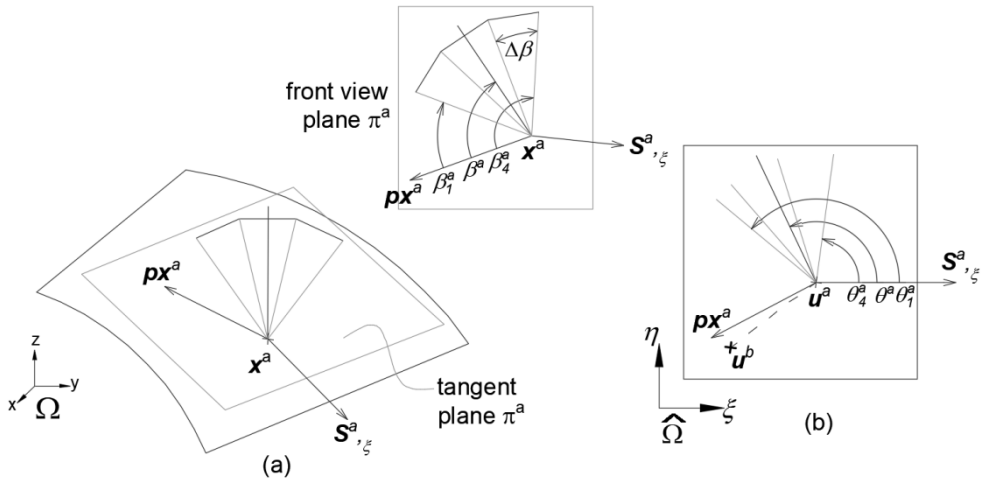


**Figure 22**. Arc angles in the $\pi^a$ plane (a) and their counterparts in the parameter space (b). Trial angle and angles for points 1 and 4 are also indicated.

Points for the discretized arcs are computed by the EPP algorithm from section 5.2, whose inputs needed are the location $\boldsymbol{\xi}^a$, the target distance $R^a$ and the orientation $\theta$ in parameter space. This angle corresponds to $\beta$, but defined w.r.t. the horizontal axis in the parameter space. The procedure to find $\theta$ from $\beta$ is depicted in **Figure 23** and its steps are explained below:

- Compute the tangent plane $\pi^a$ in the physical space. The normal vector to the plane is given by $\boldsymbol{n^a} = \boldsymbol{S}^a_{,\xi} \times \boldsymbol{S}^a_{,\eta}$ (for the computation of the derivatives see Appendix B).
- Find $\boldsymbol{px}^a$: the projection of $\boldsymbol{x}^a_b$ onto $\pi^a$ (for arc $b$ use $\boldsymbol{x}^b_a$).
- Form local base $\boldsymbol{B}^a$ with vectors $\boldsymbol{n^a}$, $\boldsymbol{px}^a$ and $\boldsymbol{w}$. Note that $\boldsymbol{w} = \boldsymbol{n^a} \times \boldsymbol{px}^a$.
- Compute vector $\boldsymbol{v}^a$ contained in plane $\pi^a$ that forms $\beta$ degrees with $\boldsymbol{px}^a$. This step involves computing vector $\boldsymbol{v}^{a\prime}$ in the local base $\boldsymbol{B}^a$ (at $\beta$ degrees from $\boldsymbol{px}^a$) and transforming to the global coordinate system to obtain $\boldsymbol{v}^a$.
- Obtain $\theta$, which is the orientation of $\boldsymbol{v}^a$ referred to the horizontal axis in the parameter space, as described in section 2.5.

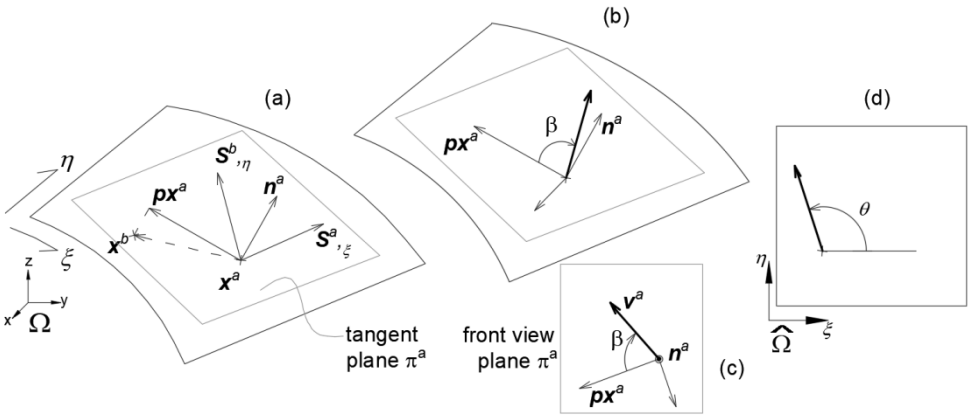


**Figure 23.** Calculation of angle $\theta$ correspondent to the $\beta$ angle. Computation of the $\pi^a$ plane and projected vector $\boldsymbol{px}^a$ (a); local base $\boldsymbol{B}^a$ (b); vector with $\beta$ angle $\boldsymbol{v}^a$ (c); corresponding angle $\theta$ in the parameter space (d).

The above-mentioned process is applied to the points of the arc for each iteration. The arc points obtained are equally spaced in the physical space but in the parameter space they can be distorted depending on the parametrization procedure used.

*Iterative process:*

Once the *a* and *b* arcs are discretized, the intersection of their lines can be calculated. The result is then compared against the previous intersection and, if it is greater than a threshold (1% in this work), arcs are re-defined and discretized again, and the intersection is re-calculated. If the difference is less than tolerance, then the process ends and the latest intersection is the one assumed valid.

After each iteration the trial angles are re-oriented to the latest computed intersection. The closer the initial trial angles ($\beta^a$ and $\beta^b$) are to the final answer the smaller number of iterations are required. Since their initial values are taken from the pattern space, they are very close to the final answer and the number of iterations are typically equal or less than three. In addition, the pattern space is used to know the side that $\boldsymbol{\xi}^c$ must hold w.r.t. the vector from $\boldsymbol{\xi}^a$ to $\boldsymbol{\xi}^b$ ($\boldsymbol{\xi}^a_b$) via the cross product $\boldsymbol{r}^a_b \times \boldsymbol{r}^a_c$. If the third component of this cross product is positive, then $\boldsymbol{\xi}^c$ must lie at the left-hand side of $\boldsymbol{\xi}^a_b$, otherwise it must lie at the right-hand side.

Four cases are possible to happen during the iterative process:

- *Case A*: intersection is found and it is in the correct side. If the error is greater than the tolerance then the next iteration is prepared: trial

angles are reoriented to the updated intersection and arc amplitudes are reduced after dividing by $NL$. See **Figure 24**.

- *Case B*: intersection is found but it is located on the wrong side. The angle amplitude is then doubled, see **Figure 25**.

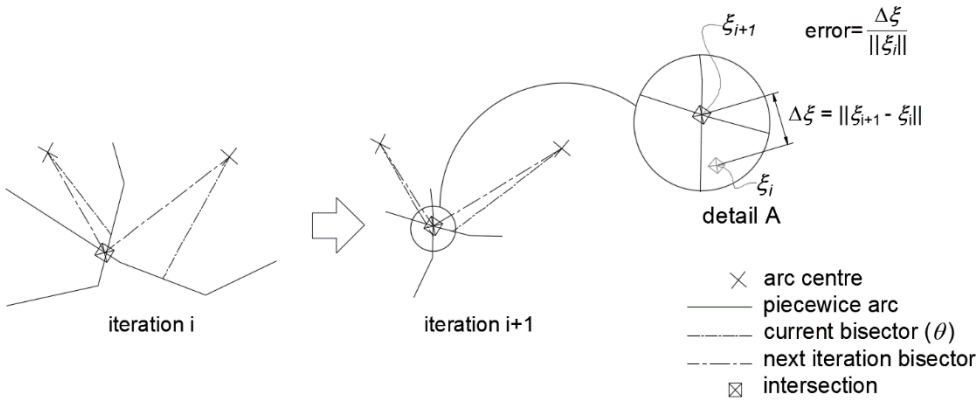


**Figure 24.** AIP case A in the parameter space.

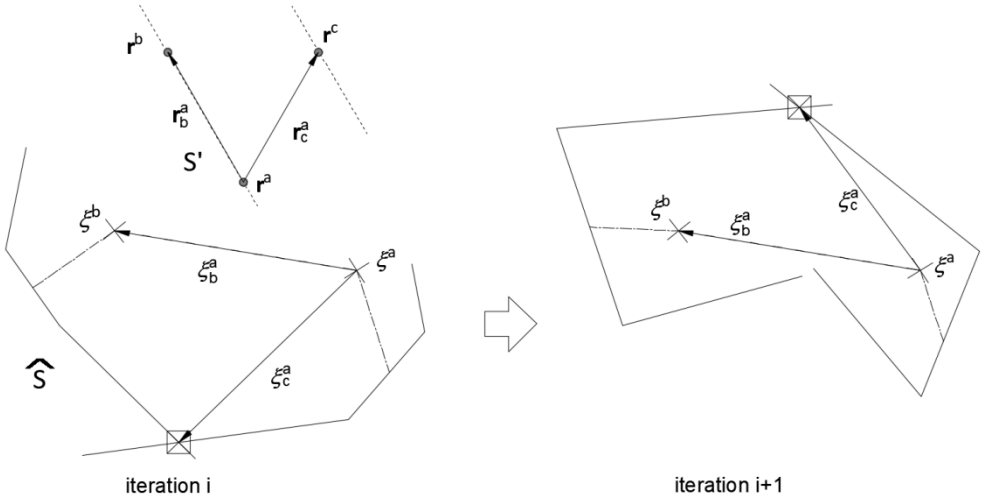


**Figure 25.** AIP case B in the parameter space. The pattern space is also shown for the *i*th iteration.

- *Case C*: no intersection is found. Both the angle amplitude and the number of segments per arc ($NL$) are doubled up for the next iteration, see **Figure 26**.

- *Case D*: the intersection was found previously but it was lost in the current iteration. The angle amplitude is then doubled for the next iteration.

For cases *B* to *D,* the arc amplitude might need to be increased to raise the possibilities of finding the intersection point.

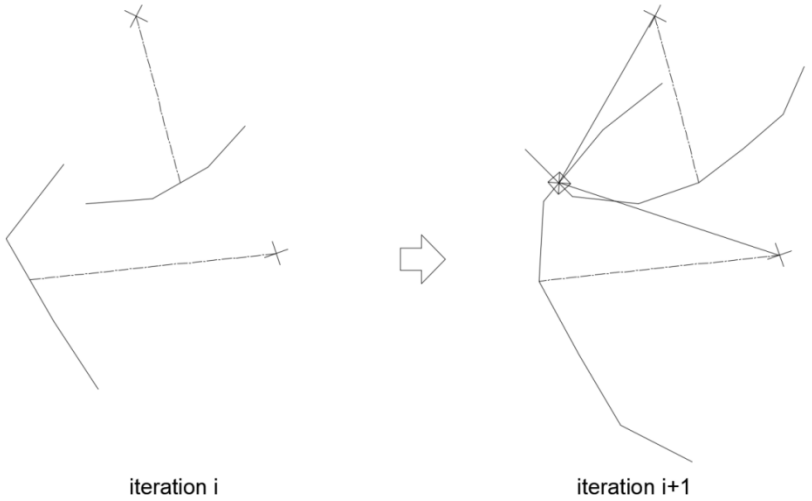


**Figure 26.** AIP case C in the parameter space.

### *5.4. Computation of vertexes*

This section explains how to compute the vertexes of the surfaces based on the EPP and AIP algorithms. First, the vertex is arbitrarily placed at the centre of the parameter and pattern spaces, i.e. $\boldsymbol{\xi}^1 = (0.5,0.5)$ and $\boldsymbol{r}^1 = (0,0)$. Second, the vertex is computed by EPP with target distance $R$, orientation $\theta = 0$ and initial location $\boldsymbol{\xi}^1$. The rest of the first hexagon vertexes ($\boldsymbol{\xi}^3$ to $\boldsymbol{\xi}^7$) are computed by the algorithm AIP from the previous vertex and $\boldsymbol{\xi}^1$, both with radius equal to $R$. Initial trial angles for AIP are $\beta^a = -60$ and $\beta^b = +60$. **Figure 27** illustrates the fourth vertex calculation in the pattern space, showing only the final iteration for clarity.

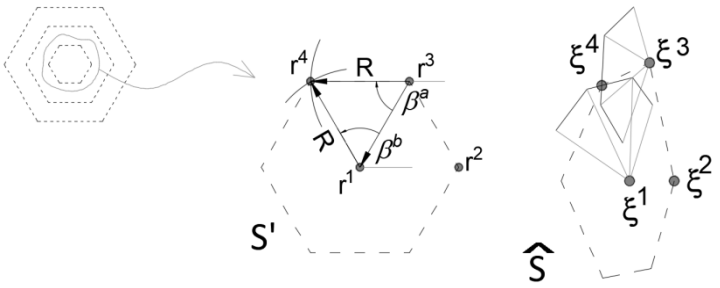

**Figure 27.** Fourth vertex computation.

The remaining vertexes are computed alongside the creation of the hexagons during their propagating motion in the form of a 2D hexagonal wave. Within one hexagon, each vertex is computed using two vertexes from previous hexagons as centre points for the intersection of the arcs (AIP). This pair, called *base vertexes*, is selected according to the current vertex position: side or corner (see **Figure 28**). The base vertexes for the former are the closest of the previous hexagon's side. The base vertexes for the latter are at both sides of the previous corner. If we call *a* and *b* to be the base vertexes and *c* to be the current vertex, the inputs required for AIP are described in **Table 2** for each type of vertex.

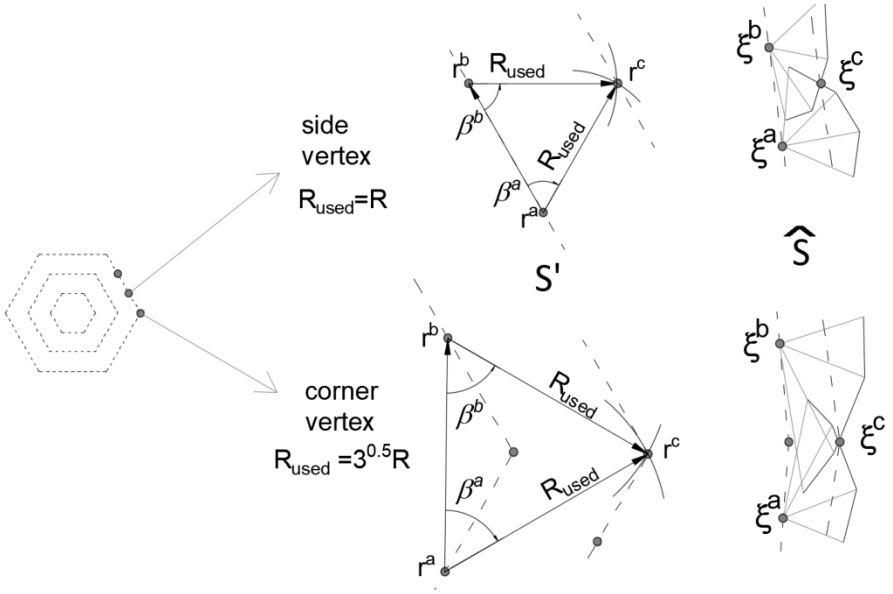


**Figure 28.** Vertex computation for some contours in the pattern and parameter spaces for side and corner vertexes. Only the final iteration arcs are depicted here for clarity.

**Table 2**. Inputs used for the intersection of arcs (AIP) to find each type of vertex.

| Current vertex location | Initial trial angles $\beta_a$ and $\beta_b$ | radius |
|---|---|---|
| Side | -60 and 60 | $R$ |
| corner | -60 and 60 | $\sqrt{3}\,R$ |

Not all the vertexes are computed: one vertex is computed if and only if at least one of its *base vertexes* lies inside the surface parameter limits. This rule avoids the computation of most of the vertexes that do not lie in the surface, reducing the computational cost considerably. Therefore, the unique hexagonal front that propagates from the central point will be cut and divided into two or more fronts by the surface boundaries during the

propagation process, but its divergence is still a possibility. The propagation of contours ends at the hexagon that has all its vertexes non-computed, *i.e.* all the *base vertexes*, from previous hexagons, lie outside the surface parameter limits.

One of the *base vertexes* might be outside the limits in the parameter domain. That vertex involves computations beyond the limits of the surface and this is why the *perimeter* and *corner elements* of a *dS-mesh* are necessary (recall section 5.1).

### *5.5. Recovering of non-computed base vertexes*

The procedure described in section 5.4 might lead to a vertex *c* with one *base vertex* non-computed. Let us call *a* and *b* to the computed and non-computed base vertexes respectively. If vertex *c* is to be calculated (we assume *a* inside surface limits) the base vertex *b* needs to be estimated. Two carry out this 'rescue' of vertex *b* we need first to identify its neighbours.

The vertexes that surround the *b* vertex in its first and second perimeters are localized using their relationship in the pattern space (see **Figure 29**). The relevant information required from these neighbour vertexes are their references, pattern distances and angles measured from vertex *b*. Two of them are then selected, giving priority to the first perimeter and to the pair that form 60 or 120 degrees with each other. The calculation of the *b* vertex is done by using AIP and the selected neighbour vertexes.

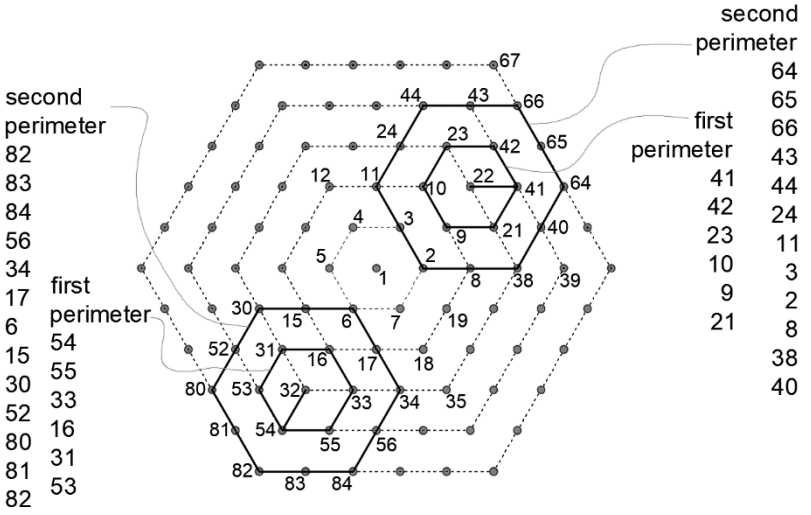


**Figure 29.** Pattern space representation of neighbour vertexes of 22 (side vertex) and 32 (corner vertex). First and second perimeters are indicated in solid line.

## 6. TRIANGULATION

This section details the surface triangulation. Section 6.1 explains the calculation of *edge vertexes* in the pattern space. Section 6.2 gives the criteria to detect non-valid surface vertexes for the triangulation. Section 6.3 shows the triangulation itself and section 6.4 details the improvement achieved at the edges.

The surface vertexes computed so far may be classified as follows. Let us define $\Delta x_{th}$ as a pre-established threshold distance in the physical space, which is measured from the surface edges (in this work $\Delta x_{th} = R/3$), then:

- *SI-vertex*: is a surface vertex which lies inside the parameter limits of the surface and is located further away of more than $\Delta x_{th}$ from the edges of the surface.
- *SE-vertex*: is a surface vertex inside the parameter limits of the surface and lies within a distance lower than $\Delta x_{th}$ when measured from the edges of the surface.
- *SO-vertex*: is a surface vertex which lies outside of the surface parameter limits.

### *6.1. Estimation of edge vertexes in the pattern space*

The *edge vertexes'* coordinates in the pattern space are part of the final triangulation procedure and, furthermore, they form the constraint that determines which triangles are inside or outside of the computable domain. Since these coordinates are unknown (*edge vertexes* were computed independently, see section 4) they need to be estimated.

We construct a triangulation with all surface vertexes (*SO*, *SE* and *SI-vertexes*) in the parameter space. This triangulation allows the mapping $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ from the surface parameter space to the pattern space. The parameter coordinates for the edge vertexes lie within this triangular net, therefore their pattern coordinates may be calculated by the following mapping:

$$\boldsymbol{r}^a = \sum_{i=1}^{3} N_i \boldsymbol{r}^i \tag{19}$$

Where shape functions $N_i$ are the area coordinates, as defined in equation (20), where $A_{total}$ is the area of the triangle and $A_i$ are the sub-areas attached to each node of the triangle. In **Figure 30**, one example for the computation of the pattern coordinates for vertex $a$ is illustrated.

$$N_i = \frac{A_i}{A_{total}} \tag{20}$$

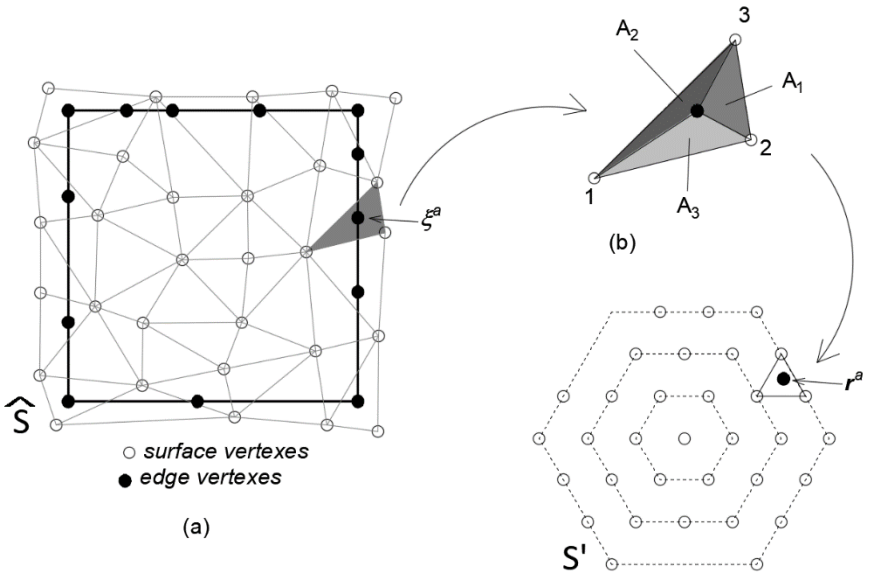


**Figure 30.** Triangulation of all vertexes in the parameter space (a) and calculation of the pattern coordinates of the edge vertex $a$ (b).

The triangulation explained in this section is not the final aim chased by the *QIT* procedure but it is a temporary triangulation that permits the estimation of the pattern coordinates of edge vertexes with some degree of accuracy.

### *6.2. Removal of non-valid surface vertexes*

Only *SI-vertexes* are considered in the triangulation (valid vertexes). Meanwhile *SO* and *SE-vertexes* are non-valid. *SO-vertexes* are detected because they are located on the outside of the surface limits. *SE-vertexes* are closer than $\Delta x_{th}$ to the edges of the surface. To measure the distance from one surface vertex to the edges, the closest pair of edge vertexes needs to be found. Then the distance from the vertex to the segment between both edge vertexes is computed in the physical space.

### *6.3. Delaunay triangulation in the pattern space*

Triangulation is done in the pattern space mainly for two reasons:

- It will be quasi-isotropic, given the vertexes arrangement of this space.

- It is a 2D plane space that facilitates the entire process.

The resultant triangulation in the physical space will inherit the same features of the triangulation on the pattern space since the location of its vertexes follows the same scheme.

*Valid surface* and *edge vertexes* obtained in previous steps are used in this section for the Delaunay triangulation. *Edge vertexes* impose constraints to the triangulation: they form the perimeter of the domain (see **Figure 31**).

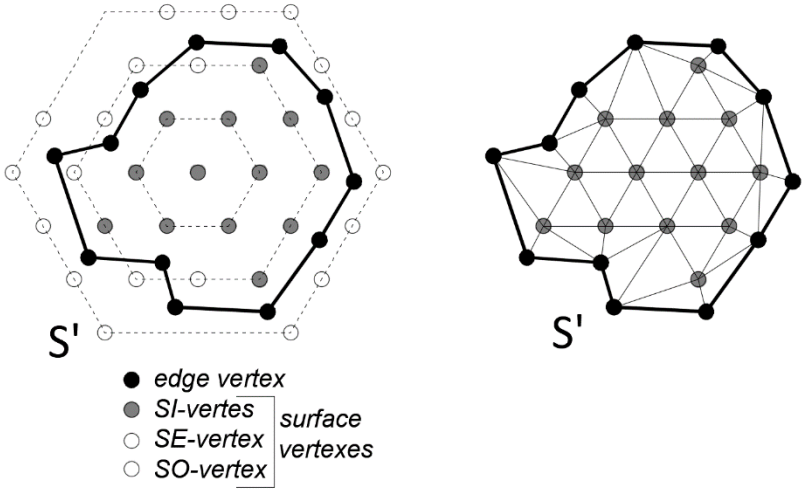


**Figure 31.** All vertexes and resultant triangulation with valid vertexes in the pattern space.

### *6.4. Edge strip triangles amendment*

Let us call *edge triangles strip* to triangles that have at least one *edge vertex*. As the pattern coordinates of *edge vertexes* were already estimated (section 6.2) they might not yield the highest quality triangulation in the physical space along this strip.

A localized improvement through the *edge triangles strip* is needed to reduce their distortion. Since not all the triangles are to check but only the edge strip ones, the process is computationally cheap. Triangles are selected in pairs, forming one quadrilateral, in advancing sequence along the four different edges separately. In each quadrilateral, both diagonals are measured in the physical space and the shortest diagonal is selected, which might coincide with the original or might not (the diagonal is then flipped). Quadrilaterals with at least one angle greater than 180º are not checked. **Figure 32** illustrates one example for demonstration purposes, where only one edge strip is detailed for clarity.

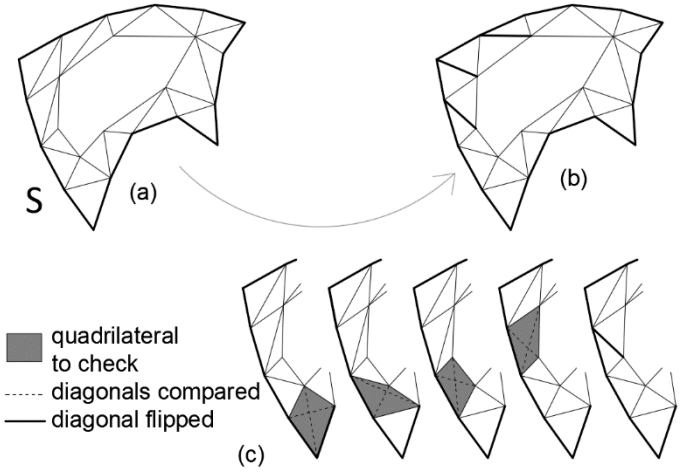


**Figure 32.** Edge triangles strip before (a) and after (b) improvement. Quadrilaterals advancing at left edge (c).

## 7. NUMERICAL EXAMPLES

The aim of the two examples presented in this section is to demonstrate the performance of *QIT*. Geometry and algorithm input details are listed in Appendix D. For both examples, the resultant mesh from the proposed *QIT* algorithm is compared with the equivalent highest quality triangulation ideally achievable, that we call *BIT* (acronym for 'Bounded Isotropic Triangulation'). Details of such triangulation are provided in Appendix C, but here we list the most relevant features:

- All angles are sixty degrees.

- All triangles have the same area: $A_{BIT} = \frac{R^2}{4} tan60$.

- Vertexes valences frequency is the closest possible to the ideal case: two vertexes of valence 2, two with valence 4, a few with valence 3 and the rest with valence 6.

To characterize the triangulation performance, we set intervals for angles, triangle sizes and valences, and count the number of instances in each interval to obtain the frequency, expressed in percentage. The frequencies are plotted and compared against the *BIT* reference solutions. In addition, the so-called quality index $Q$ described in equation (21), is computed. This is a numerical indicator in percentage of how close the triangulation is to the *BIT* reference solution. The ideal value is 100 %.

$$Q = 25\frac{f_s^a}{100} + 25\frac{f_n^a}{100} + 25\frac{f_{v3}^a}{f_{v3}^b} + 25\frac{f_{v6}^a}{f_{v6}^b} \quad (21)$$

The inputs for equation (21) are frequencies, in percentage, for:

$f_s^a$: triangles of *QIT,* with sizes in the same interval as *BIT* size ($A_{BIT}$);

$f_n^a$: angles of *QIT* in the same interval of 60 degrees;

$f_{v3}^a$ and $f_{v3}^b$: vertexes with the valence number equal to 3 for both *QIT* and *BIT*;

$f_{v6}^a$ and $f_{v6}^b$: vertexes with the valence number equal to 6 for both *QIT* and *BIT*.

Note that the frequencies for the size and angles for the *BIT* are 100 %.

### *7.1. Single surface with severe distortion in the parameter space*

A single surface with abrupt increments in parameter space is meshed with our *QIT* algorithm. The algorithm was performed for two different sizes: $R = 12$ and $R = 5$. Derivatives were estimated by spline surface fitting, as described in Appendix B. In spite of the distortion in the parameter space, the resultant triangulations remain mostly isotropic, only some few triangles appear to be distorted due to the presence of edges. The quality factor $Q$ is greater when $R = 5$ because the number of triangles affected by the edges is less than in the other case when $R$ = 12. This indicates that if the edges have a small influence on the overall surface's domain then the closer to the *BIT* reference solution is the *QIT* triangulation and, therefore, it proves that the *QIT* method brings onto the surface physical space an accurate 'image' of the pattern space.

**Figure 33** shows the surface in the physical space with the knot spans depicted and one triangulation with vertexes equally spaced in the parameter space to highlight the distortion in the parametrization. **Figure 34** illustrates the *QIT* method with both $R = 12$ and $R = 5$. **Figure 35** shows the propagation of contours in the parameter space, where the divergent nature of the front can be clearly seen, note also some of the contours go further off the surface limits. **Figures 36** to **38** show the frequency plots.

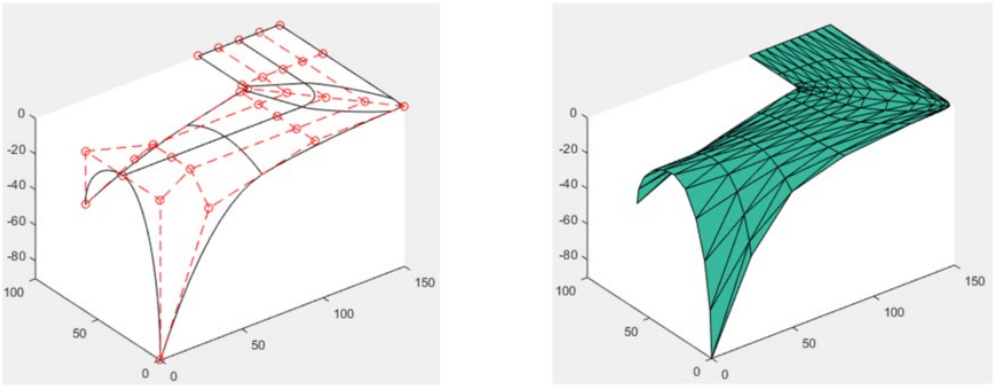


**Figure 33.** Left: surface for triangulation. Right: triangulation with nodes equally spaced in the parameter space.

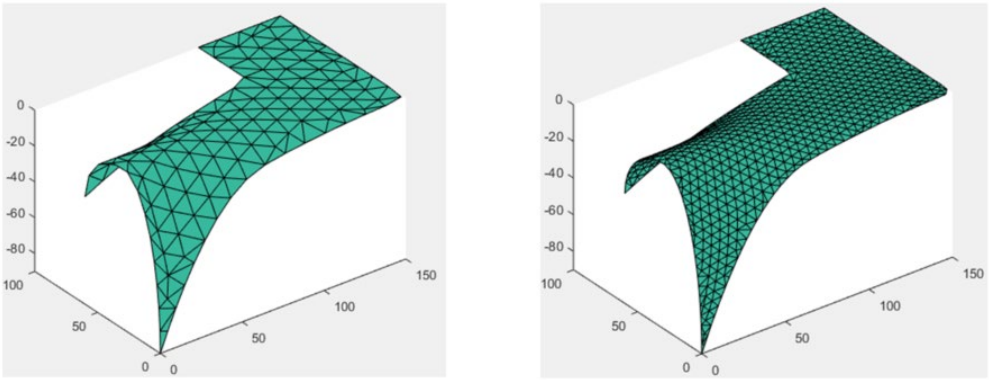


**Figure 34.** Resultant *QIT* triangulation. Left: $R = 12$. Right: $R = 5$.

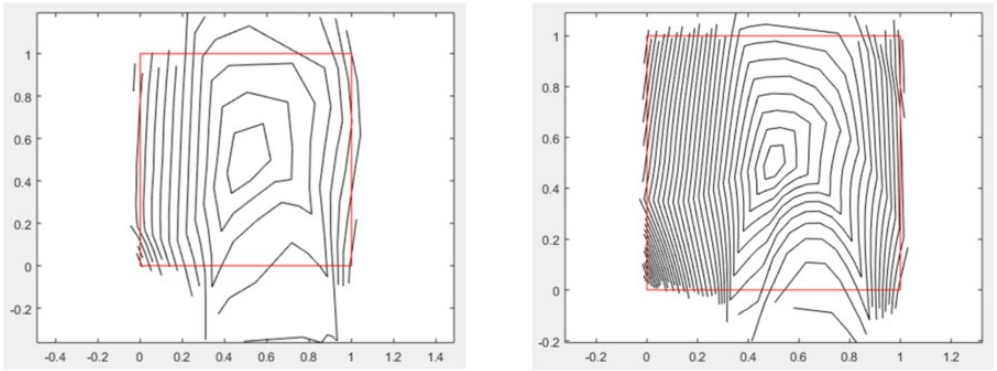


**Figure 35.** Contours propagation in the parameter space, red lines are the surface limits. Left: $R = 12$. Right: $R = 5$.

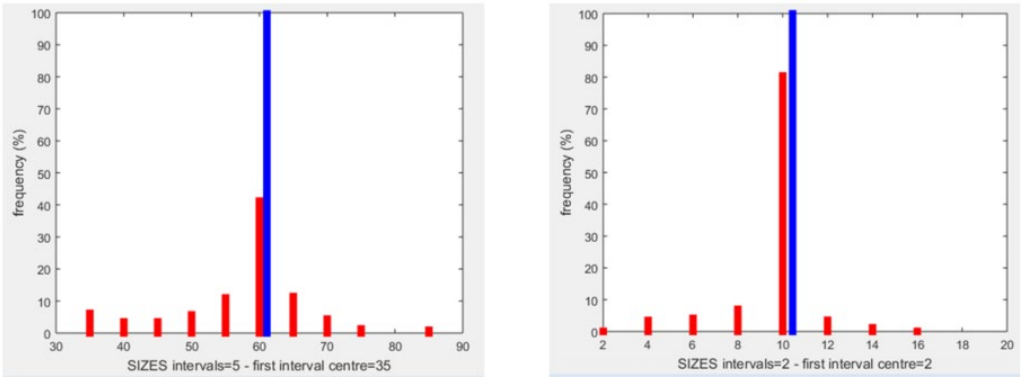


**Figure 36.** Frequency plots for triangle sizes. Red represents the *QIT* while blue represents the *BIT*. Left: $R = 12$. Right: $R = 5$.

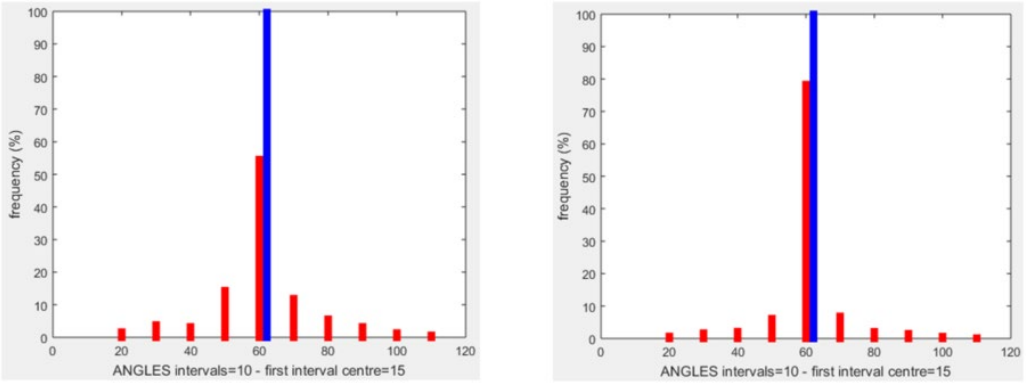


**Figure 37.** Frequency plots for angles. Red represents the *QIT* while blue represents the *BIT*. Left: $R = 12$. Right: $R = 5$.

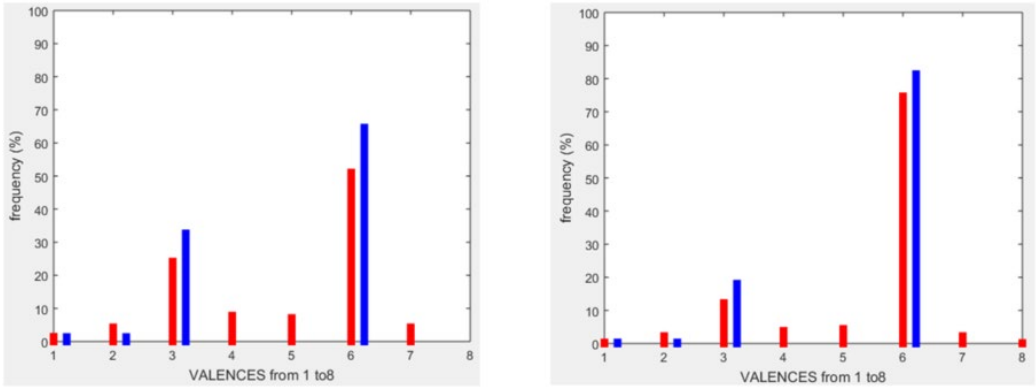


**Figure 38.** Frequency plots for valences. Red represents the *QIT* while blue represents the *BIT*. Left: $R = 12$. Right: $R = 5$.

The quality index (Q) is computed below for both cases. Note how the quality raises from 62 to 80 % when the target distance decreases from 12 to 5, i.e. if a finer mesh is used then it gets closer to the ideal *BIT*.

$$Q = 25\frac{41}{100} + 25\frac{55}{100} + 25\frac{24}{33} + 25\frac{51}{65} = 62\ \%$$

$$Q = 25\frac{78}{100} + 25\frac{80}{100} + 25\frac{12}{18} + 25\frac{75}{81} = 80\ \%$$

To illustrate the influence of the tolerance in the computational cost, **Figure 39** is used. It includes the plot for the relative computational time (tr) for edge vertex tolerances of 0.25, 0.50, 1.0, 2.0 and 4.0 % (vertical axis on the right-hand side of the plot). The surface vertex tolerances are 4 times larger, e.g. 2.0 % for the tolerance for edge vertexes, the tolerance for surface vertexes is 8.0 %. The triangle size used was $R = 12$. That relative computational time is referred to the tolerance of 1.0 % for edges. It also includes the quality index (Q), which is plotted in the vertical axis on the left-hand side of the plot. It can be clearly seen that it decreases as the tolerance becomes larger, as expected.

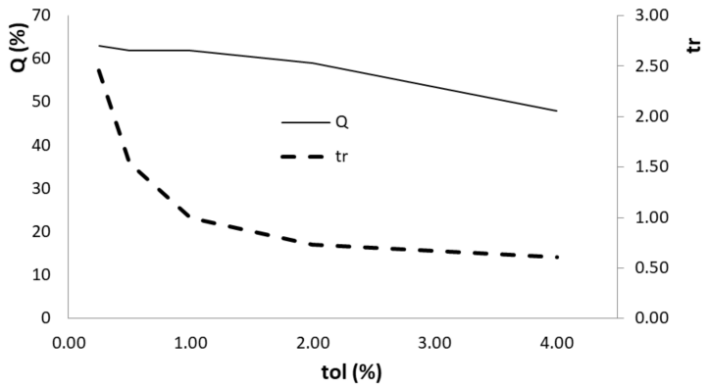


**Figure 39.** Relative computational time (tr) and quality index (Q) versus tolerances.

The quality improvement with the tolerance restriction can also be seen in **Figure 40**, where the resultant meshes are depicted for tolerances of 4.0 and 0.25 %.

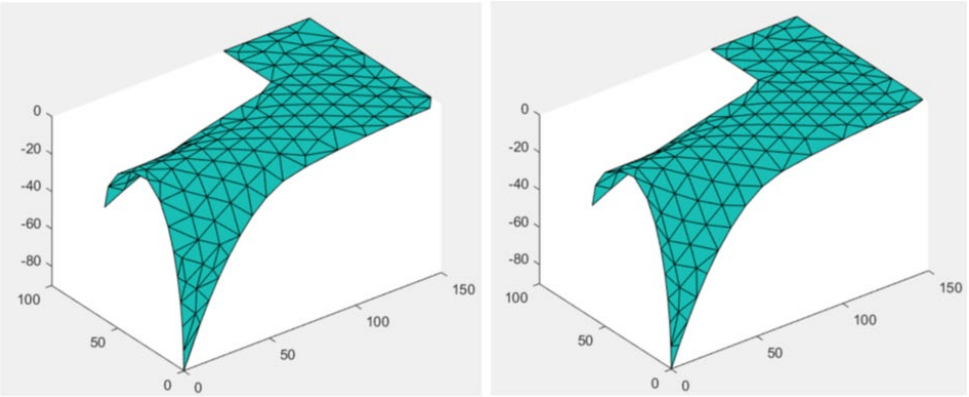


**Figure 40.** Triangulation for tolerances of 4.0 % (left) and 0.25 % (right).

### *7.2. Three contiguous surfaces*

This example shows how three contiguous surfaces are conformal triangulated using the *QIT* algorithm, i.e. their shared edges have the same curve discretization. The target distance used was $R = 5$. **Figure 41** gives the surfaces in the physical space with knot spans and control points (left) and the computed *edge vertexes* (right). **Figure 42** shows the final result after triangulation (left), where the general isotropy and uniformity of the triangulation can be clearly observed. On the right side of that figure, the edge shared by contiguous surfaces is detailed, where conformal meshes can be observed. Finally, **Figure 43** provides the frequency plots showing again the tendency of the *QIT* algorithm to achieve mesh isotropy close to the perfect solution delivered by *BIT*. The quality index for this case is 73 %, as calculated below.

$$Q = 25\frac{54}{100} + 25\frac{77}{100} + 25\frac{9}{13} + 25\frac{80}{86} = 73\ \%$$

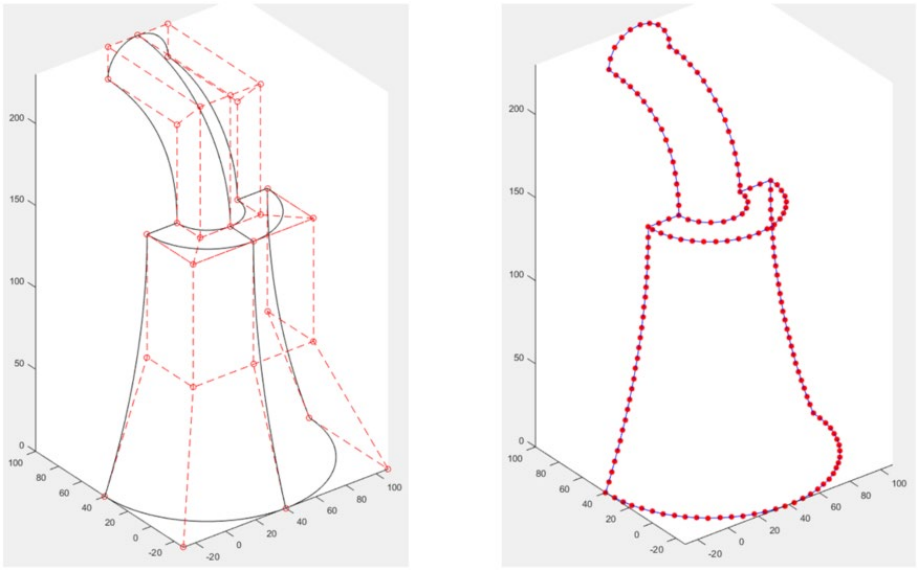

**Figure 41.** Left: surfaces for triangulation. Right: *edge vertexes* resultant from the *QIT* algorithm.

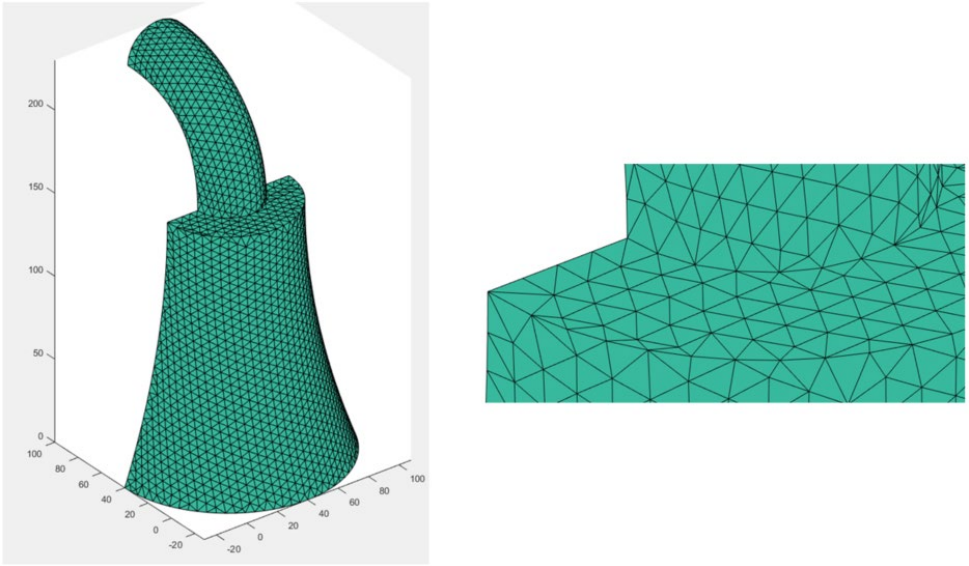

**Figure 42.** Left: resultant mesh from the *QIT* algorithm. Right: detail for the merging of the mesh for different surfaces.

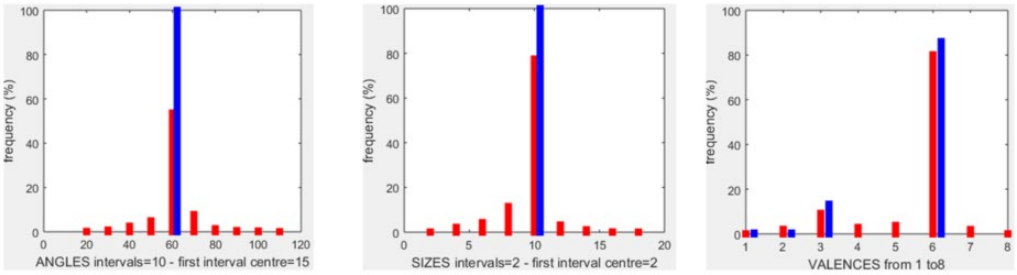


**Figure 43.** Frequencies of sizes, angles and valences. Red represents our *QIT* and blue *BIT*

## 8. CONCLUSIONS AND FUTURE WORK

A new procedure for triangulating NURBS surfaces is presented in this work. It provides a quasi-isotropic triangular mesh at once, with no preliminary tessellation, based on a divergent advancing front technique that avoids front collisions. Each new vertex position is calculated using trapezoidal numerical integration, which provides simplicity and therefore efficiency. The error committed in this approximation is controlled by previous discretization of the parameter space. When there is more than one surface involved, their meshes are conformal at the shared curve because vertexes of such curve are computed once and applied for both surfaces. Derivatives are required repeatedly for this algorithm. In order to improve the efficiency, alternatives to the analytical calculation of these derivatives are proposed in Appendix B.

The examples proposed demonstrated that the method delivers high quality triangulations that tend to be isotropic, regardless of the shape or parametrization used. Potential extensions or improvements of the method are listed below:

- This procedure applies to non-trimmed surfaces. Application to trimmed surfaces is still pending.
- Triangulations obtained by the algorithm presented here might be the initial stage for further refinements at certain zones such as high curvature areas or where analysis results (e.g. strains) are expected to present sudden variations.

## 9. ACKNOWLEDGMENTS

The authors gratefully acknowledge the Department of Engineering Design and Mathematics of the University of the West of England that partially founded this research. They also acknowledge to Dr. Arnaud Marmier, senior lecturer at the same university, for his comments on this work.

## 10. APPENDIXES

### *Appendix A: Derivatives of a function that is as norm of first derivative of another function*

Let $f(u)$ be a function defined as the norm of first derivative of another function $\boldsymbol{g}(u)\colon \mathbb{R}^1 \to \mathbb{R}^d$. For the sake of clarity we remove the free variable from the notation, then $f(u)$ is expressed as $f$, $\boldsymbol{g}(u)$ as $\boldsymbol{g}$ and so on.

$$\boldsymbol{g} = g_i \quad \forall\, i = 1\ to\ d \tag{A.1}$$

$$f = \|\boldsymbol{g}'\| = \left(\sum_{i=1}^{d} {g_i}'^2\right)^{1/2} \tag{A.2}$$

First derivative of $f$ is computed by simple differentiation of (A.2), that yields (A.3).

$$f' = \frac{\sum_{i=1}^{d}({g_i}'\, {g_i}'')}{f} \tag{A.3}$$

Applying differentiation again to (A.3) we obtain the second derivative of $f$, written in (A.4).

$$f'' = \frac{\sum_{i=1}^{d}(g_i'' g_i'' + {g_i}'\, {g_i}''')\ f - \sum_{i=1}^{d}({g_i}'\, {g_i}'')\ f'}{f^2} \tag{A.4}$$

For NURBS surfaces $\boldsymbol{S}(\xi,\eta)$ the directional derivatives are functions of *main derivatives* ($\boldsymbol{S}_{,\xi}$ and $\boldsymbol{S}_{,\lambda}$) and are not trivial. First directional derivative $g_i' = \boldsymbol{S}_{,\lambda}$ is (A.5).

$$\boldsymbol{S}_{,\lambda} = \boldsymbol{S}_{,\xi}\, cos\,\theta + \boldsymbol{S}_{,\eta}\, sin\,\theta \tag{A.5}$$

Second $g_i'' = \boldsymbol{S}_{,\lambda\lambda}$ and third $g_i''' = \boldsymbol{S}_{,\lambda\lambda\lambda}$ directional derivatives are explained here. Let $\boldsymbol{v}$ be a vector with orientation $\theta$ and $\|\boldsymbol{v}\| = 1$, i.e. $v_1 = \cos\theta$ and $v_2 = \sin\theta$. Let $S(\boldsymbol{\xi})$ be a function such that $\boldsymbol{g}(u)\colon \mathbb{R}^2 \to \mathbb{R}^1$ with $\boldsymbol{\xi} = (\xi,\eta)^T$. Derivatives w.r.t. $\xi$ and $\eta$ at location $\boldsymbol{\xi}_0$ may be calculated as expressions (A.6) and (A.7).

$$S_{,\xi} = \lim_{h\to 0}\frac{S(\boldsymbol{\xi}_0 + (h,0)) - S(\boldsymbol{\xi}_0)}{h} = \lim_{h\to 0}\frac{S(\xi_0 + h, \eta_0) - S(\xi_0,\eta_0)}{h} \tag{A.6}$$

$$S_{,\eta} = \lim_{h\to 0} \frac{S(\boldsymbol{\xi}_0 + (0,h)) - S(\boldsymbol{\xi}_0)}{h} = \lim_{h\to 0} \frac{S(\xi_0, \eta_0 + h) - S(\xi_0, \eta_0)}{h} \tag{A.7}$$

Directional first derivative of $S$ with $\theta$ orientation at location $\boldsymbol{\xi}_0$ is given by equation (A.8), which is equivalent to (A.9).

$$S_{,v} = \lim_{h\to 0} \frac{S(\boldsymbol{\xi}_0 + h\boldsymbol{v}) - S(\boldsymbol{\xi}_0)}{h} = \lim_{h\to 0} \frac{S(\xi_0 + hv_1, \eta_0 + hv_2) - S(\xi_0, \eta_0)}{h} \tag{A.8}$$

$$S_{,v} = \boldsymbol{\nabla S} \cdot \boldsymbol{v} = \begin{Bmatrix} S_{,\xi} \\ S_{,\eta} \end{Bmatrix}^T \begin{Bmatrix} v_1 \\ v_2 \end{Bmatrix} \tag{A.9}$$

Directional second derivative is obtained as follows:

$$S_{,vv} = \lim_{h\to 0} \frac{\left(\lim_{h\to 0} \frac{S(\boldsymbol{\xi}_0 + h\boldsymbol{v} + h\boldsymbol{v}) - S(\boldsymbol{\xi}_0 + h\boldsymbol{v})}{h}\right) - \left(\lim_{h\to 0} \frac{S(\boldsymbol{\xi}_0 + h\boldsymbol{v}) - S(\boldsymbol{\xi}_0)}{h}\right)}{h} \tag{A.10}$$

$$S_{,vv} = \lim_{h\to 0} \frac{S(\boldsymbol{\xi}_0 + 2h\boldsymbol{v}) - 2S(\boldsymbol{\xi}_0 + h\boldsymbol{v}) + S(\boldsymbol{\xi}_0)}{h^2} \tag{A.11}$$

Developing equation (A.11) and grouping terms we arrive to the bilinear form (A.12), which is equivalent to (A.13).

$$S_{,vv} = \boldsymbol{v}^T \boldsymbol{H}\, \boldsymbol{v} = \{v_1 \quad v_2\} \begin{bmatrix} S_{,\xi\xi} & S_{,\xi\eta} \\ S_{,\xi\eta} & S_{,\eta\eta} \end{bmatrix} \begin{Bmatrix} v_1 \\ v_2 \end{Bmatrix} \tag{A.12}$$

$$S_{,vv} = v_i v_j \; S_{,ij} \quad \forall\, i,j = 1,2 \tag{A.13}$$

In (A.13), sub-index of $S_{,ij}$ indicates derivatives w.r.t. $\xi$ (sub-index =1) or $\eta$ (sub-index=2). Same procedure for third directional derivative yields equation (A.14):

$$S_{,vvv} = \lim_{h\to 0} \frac{S(\boldsymbol{\xi}_0 + 3h\boldsymbol{v}) - 3S(\boldsymbol{\xi}_0 + 2h\boldsymbol{v}) + 3S(\boldsymbol{\xi}_0 + h\boldsymbol{v}) + S(\boldsymbol{\xi}_0)}{h^3} \tag{A.14}$$

Developing (A.14) and grouping terms, the third directional derivative can be expressed as (B.15).

$$S_{,vvv} = v_i v_j v_k \; S_{,ijk} \quad \forall\, i,j = 1,2 \tag{A.15}$$

Where sub-index of $S_{,ijk}$ indicates derivatives w.r.t. $\xi$ (sub-index =1) or $\eta$ (sub-index=2).

So far, third directional derivatives expression (A.9), (A.13) and (A.15) are deducted for functions $S: \mathbb{R}^2 \rightarrow \mathbb{R}^1$. Application for function $\boldsymbol{S}: \mathbb{R}^2 \rightarrow \mathbb{R}^d$ is direct. Each of the $d$ components of the directional derivative can be calculated separately by equations (A.9), (A.13) and (A.15). For example for $d = 3$ ($\boldsymbol{x} = x, y, z$) third directional derivative has three components as per equation (A.16).

$$\begin{aligned} S^x{}_{,vvv} &= v_i v_j v_k \; S^x{}_{,ijk} \quad \forall\, i,j = 1,2 \\ S^y{}_{,vvv} &= v_i v_j v_k \; S^y{}_{,ijk} \quad \forall\, i,j = 1,2 \\ S^z{}_{,vvv} &= v_i v_j v_k \; S^z{}_{,ijk} \quad \forall\, i,j = 1,2 \end{aligned} \tag{A.16}$$

## *Appendix B: Surface derivatives estimation*

Analytical calculation of NURBS derivatives is computationally expensive. To increase the algorithm speed we propose two alternatives. In both cases analytical derivatives are calculated previously at certain locations (sample points) and then a surface is fitted to them. The first presented method fits spline surfaces to those sample points. The second method uses the *dS-mesh* nodes as sample points to linearly interpolate between them. We recall that $\theta$-directional derivative is computed as per equation (B.1).

$$\boldsymbol{S}_{,\lambda} = \boldsymbol{S}_{,\xi} \cos\theta + \boldsymbol{S}_{,\eta} \sin\theta \tag{B.1}$$

### *B.1 Derivatives computed from fitted B-spline surfaces*

#### *B.1.1 Basic definitions*

Let $\boldsymbol{F}(\xi, \eta): \mathbb{R}^2 \rightarrow \mathbb{R}^6$ be a function that store the NURBS surface $\boldsymbol{S}(\xi, \eta)$ derivatives fields, i.e. $S_{j,\beta}$ with $j = 1,2,3$ that corresponds to $x, y, z$ components; and $\beta = 1,2$ for derivatives w.r.t. $\xi$ and $\eta$. The domain of $\boldsymbol{F}$ is the parameter space of $\boldsymbol{S}$. The $i$th component of $\boldsymbol{F}$ corresponds to $jx\beta$. **Figure B.1** shows two examples.

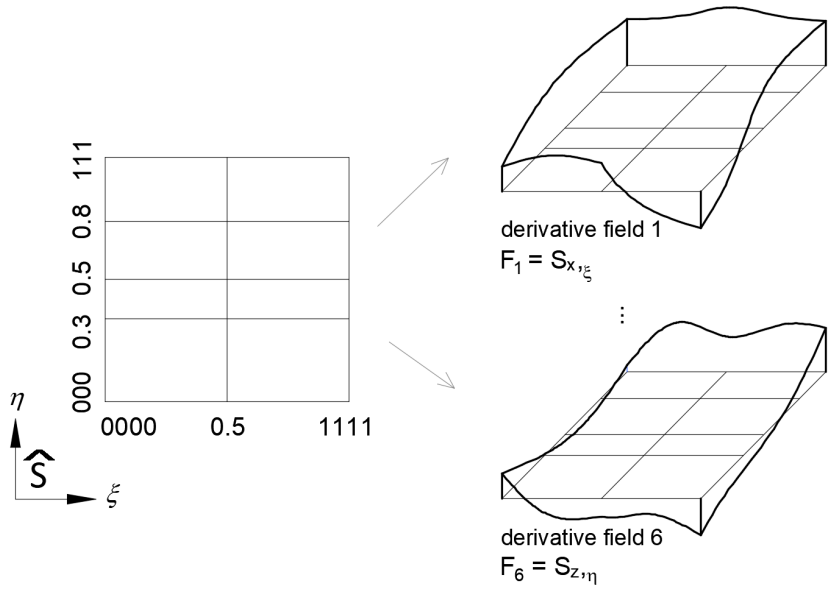


**Figure B.1.** Derivatives fields of NURBS surface showing the first and last derivatives components.

We define in each knot span of $\boldsymbol{S}$ a set of six spline surfaces to approximate the six components of $\boldsymbol{F}$. Spline[1] surface ${\boldsymbol{T}^i}_k: \mathbb{R}^2 \to \mathbb{R}^3$ is to be fitted to the $(\beta xj)$th derivative component within the $k$th knot span of $\boldsymbol{S}$. We will refer to each of those sets of six spline surfaces as $k$-set.

${\boldsymbol{T}^i}_k$ has the parameter space $\hat{\mathrm{S}}_k$ with components $(u, v)$ and maps onto $\mathbb{R}^3$, with two first components, called *plan coordinates*, equal to $(u, v)$ and the third component, called *height* ($\zeta$), with the $F_i$ derivative estimation, (see **Figure B.2**).

[1] We refer to *B-spline* as *spline* for brevity.

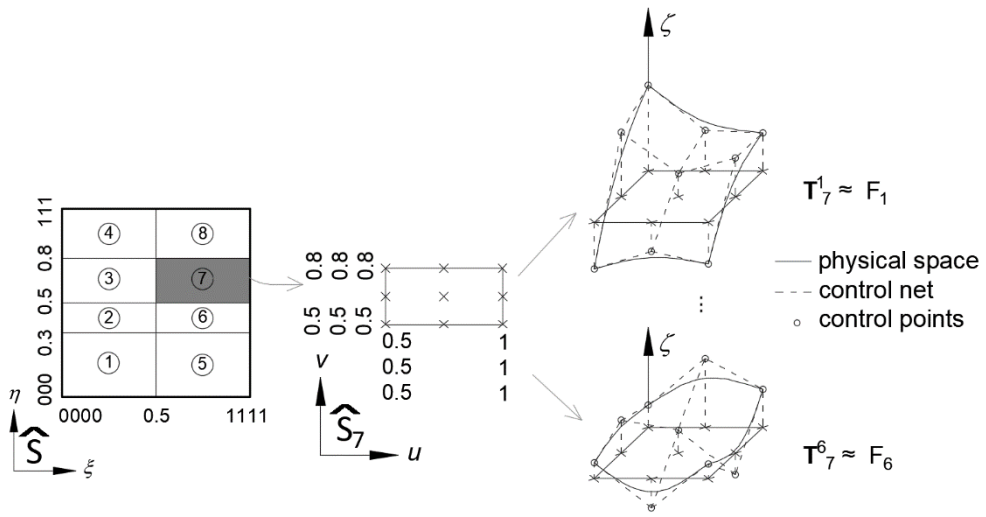


**Figure B.2.** *k*-set for the seventh span of the NURBS surface (first and last components are shown).

Features of each *k*-set are listed below:

- Parameter space domain coincides with the correspondent $\boldsymbol{S}$ knot span domain: $\hat{\mathrm{S}}_k = (\xi_{k1}, \xi_{k2}) \otimes (\eta_{k1}, \eta_{k2})$, where $\xi_{k1}, \xi_{k2}, \eta_{k1}$ and $\eta_{k2}$ are the *k*th knot span limits.
- Control points *plan coordinates* coincide with their parameter coordinates $(u, v)$, therefore one parameter location for $\boldsymbol{T}^i{}_k$ coincides with its physical *plan coordinates* and with the parameter coordinates of $\boldsymbol{S}$.
- Control points are equally spaced on plan in each direction, i.e. *plan coordinates* form a regular net on $\hat{\mathrm{S}}_k$.
- The six splines of the *k*-set share the same *plan coordinates*, hence they share parametrization.
- The six splines of the *k*-set share basis functions, i.e. they use the same knot spans, degrees and number of control points.
- Control points *heights* are to be fitted to the correspondent derivatives field, e.g. $\boldsymbol{T}^3{}_k$ fits to $F_3 = S_{1,3} = S_{z,\xi}$.

There is one *k*-set defined separately for each knot span of $\boldsymbol{S}$ in order to guarantee that those splines are fitted to a smooth field avoiding any potential $C^0$ transition between knot spans. The fitted splines in this work are quadratic. Previously to fit $\boldsymbol{T}^i{}_k$ splines to derivatives fields $S_{j,\beta}$, we

need to define the number of control points in each direction, which is driven by the error estimation as shown in section B.2.2.

*B.1.2 Number of control points*

Explanations in this section are given for one *k*-set and one derivative field $F_i$. Sub-index on $F$ is removed for clarity. The number of control points is driven by the estimation of error. Absolute error is given by equation (B.2), that is deducted in section B.2.5.

$$E \leq \left| \frac{1}{3!} \left( F^{\boldsymbol{\alpha}}{}_{,\xi\xi\xi}\ \Delta\xi_r{}^3 + 3F^{\boldsymbol{\alpha}}{}_{,\xi\xi\eta}\ \Delta\xi_r{}^2 \Delta\eta_r + 3F^{\boldsymbol{\alpha}}{}_{,\eta\eta\xi}\ \Delta\eta_r{}^2 \Delta\xi_r + F^{\boldsymbol{\alpha}}{}_{,\eta\eta\eta}\ \Delta\eta_r{}^3 \right) \right| \qquad \text{(B.2)}$$

Where $\Delta\xi_r$ and $\Delta\eta_r$ are the representative increments (see section B.2.5, equation B.28) and derivatives are at location $\boldsymbol{\alpha} = (\xi_\alpha, \eta_\alpha)$ that belongs to the knot span sub-domain $\hat{\mathrm{S}}_k$ and maximises the error. Relative error in percentage is obtained as equation (B.3), being $\bar{F}$ the root mean square over the whole knot span (B.4) that might be estimated by Gauss quadrature.

$$E_r = 100\ E/\bar{F} \qquad \text{(B.3)}$$

$$\bar{F} = \sqrt{\frac{1}{Area_{\hat{\mathrm{S}}_k}} \int_{\hat{\mathrm{S}}_k} F\ d\hat{\mathrm{S}}_k} \qquad \text{(B.4)}$$

$E_r$ in the derivative estimation is to be equal or less than the prescribed tolerance. This condition will determinate the number of control points for the *k*-set following next steps:

- Initial number of control points corresponds to $\boldsymbol{T}^i{}_k$ spline with a single knot span, since $\boldsymbol{T}^i{}_k$ is quadratic, initial number of control points is three in each direction. Therefore initial representative plan increments are $\Delta\xi^0{}_r = 0.72\ (\xi_{k2} - \xi_{k1})/2$ and $\Delta\eta^0{}_r = 0.72\ (\eta_{k2} - \eta_{k1})/2$, where $\xi_{k1}, \xi_{k2}, \eta_{k1}$ and $\eta_{k2}$ are the knot span limits.
- Third derivatives are needed, but $\boldsymbol{\alpha}$ location is unknown, then we calculate exact derivatives values $F_{,\xi\xi\xi}$, $F_{,\xi\xi\eta}$, $F_{,\eta\eta\xi}$ and $F_{,\eta\eta\eta}$ at locations of a net of $s \times s$ equally spaced (in this work $s = 3$).
- $E_r$ is computed with equations (B.2), (B.3) and (B.4) for each of these $s \times s$ points using the initial representative increments $\Delta\xi^0{}_r$ and $\Delta\eta^0{}_r$. We consider only the highest value among the $s \times s$ errors.

- The ratio $d = E_r\ /tolerance$ is calculated.
- To reduce our error by a $d$ factor, we can only reduce the representative increments as shown in equation (B.5).

$$E/d \leq \left|\frac{1}{3!}\left(F^{\alpha}{}_{,\xi\xi\xi}\ \Delta\xi_r{}^3/d + 3F^{\alpha}{}_{,\xi\xi\eta}\,\Delta\xi_r{}^2\Delta\eta_r/d + 3F^{\alpha}{}_{,\eta\eta\xi}\,\Delta\eta_r{}^2\Delta\xi_r/d + F^{\alpha}{}_{,\eta\eta\eta}\,\Delta\eta_r{}^3/d\right)\right| \tag{B.5}$$

- We use first and last summands of (B.5) to estimate updated increments in each direction to reduce the error below tolerance, as shown in equations (B.6) and (B.7).

$$\Delta\xi_r = \left(\frac{\Delta\xi_r^{0\,3}}{d}\right)^{1/3} = \frac{0.72\ (\xi_{k2} - \xi_{k1})/2}{d^{1/3}} \tag{B.6}$$

$$\Delta\eta_r = \left(\frac{\Delta\eta_r^{0\,3}}{d}\right)^{1/3} = \frac{0.72\ (\eta_{k2} - \eta_{k1})/2}{d^{1/3}} \tag{B.7}$$

- With these representative increments ($\Delta\xi_r$, $\Delta\eta_r$) the actual increments ($\Delta\xi$, $\Delta\eta$) are obtained dividing by 0.72 and then the number of control points in each direction is calculated as (B.8) and (B.9).

$$n = max\left[3, round\left(\frac{(\xi_{k2} - \xi_{k1})}{\Delta\xi_r/0.72}\right)\right] \tag{B.8}$$

$$m = max\left[3, round\left(\frac{(\eta_{k2} - \eta_{k1})}{\Delta\eta_r/0.72}\right)\right] \tag{B.9}$$

The number of control points $n$ and $m$ are shared by the six splines of the *k*-set.

*B.1.3 Surface fitting*

Once the number of control points is obtained all the splines features of the *k*-set are already defined with the exception of control points *heights* ($\zeta$). These coordinates are obtained by surface fitting techniques. Matrix **A** (B.10) is computed only once for the *k*-set, since basis functions are shared by the six splines. Computation of matrix **A** needs parameter coordinates of control points. As stated before, these parameter coordinates coincide with their *plan coordinates*: $u, v$.

$$\mathbf{A} = \begin{bmatrix} N_1(u_1)M_1(v_1) & \dots & N_1(u_1)M_m(v_m) \\ \vdots & & \vdots \\ N_n(u_n)M_1(v_1) & \dots & N_n(u_n)M_m(v_m) \end{bmatrix} \tag{B.10}$$

To compute *heights* of each of the six splines control points, we use equations (B.11) and (B.12), where $a$ may be substituted by $x$, $y$ or $z$, and the exact values at control points: ${F_{x,\xi}}^{11}, {F_{x,\xi}}^{12}, \dots, {F_{z,\eta}}^{nm}$, are needed. These exact values are computed analytically prior to this operation.

$$\begin{Bmatrix} {\zeta^1}_{11} \\ \vdots \\ {\zeta^1}_{nm} \end{Bmatrix} = \mathbf{A}^{-1} \begin{Bmatrix} {F_{a,\xi}}^{11} \\ \vdots \\ {F_{a,\xi}}^{nm} \end{Bmatrix} \tag{B.11}$$

$$\begin{Bmatrix} {\zeta^6}_{11} \\ \vdots \\ {\zeta^6}_{nm} \end{Bmatrix} = \boldsymbol{A}^{-1} \begin{Bmatrix} {F_{a,\eta}}^{11} \\ \vdots \\ {F_{a,\eta}}^{nm} \end{Bmatrix} \tag{B.12}$$

Once the *height* of control points are calculated, we achieve all the features of the six splines of ${\boldsymbol{T}^i}_k$ that approximates the components of $\boldsymbol{S}_{,\xi}$ and $\boldsymbol{S}_{,\eta}$ with error equal or less than the tolerance. In addition, the input parameter coordinates for $\boldsymbol{S}$ and for ${\boldsymbol{T}^i}_k$ are the same: $(\xi, \eta) = (u, v)$.

*B.1.4 Estimation of directional derivatives using fitted splines*

The norm of $\theta$-directional derivative can be estimated at location $\boldsymbol{\xi}^a$ using the fitted spline surfaces ${\boldsymbol{T}^i}_k$. Firstly the surface $\boldsymbol{S}$ knot span where $\boldsymbol{\xi}^a$ lies is identified in order to select the corresponding $k$-set. Then the six components of both derivatives are calculated entering in each spline surface with the same $\boldsymbol{\xi}^a$ coordinates. Note that basis functions are to be calculated only once, as the six splines share them. Estimation of derivatives vectors $\boldsymbol{S}_{,\xi}$ and $\boldsymbol{S}_{,\eta}$ are assembled and $\theta$-directional derivative is computed as equation (B.1).

*B.1.5 Error in approximation with spline surface*

This section demonstrates that the error when fitting a bi-quadratic spline surface to a function $F(\xi, \eta)$ within the rectangular domain $(\xi_1, \xi_2) \otimes (\eta_1, \eta_2)$ is calculated as expression (B.13).

$$E \leq \left| \frac{1}{3!} \left( {F^{\boldsymbol{a}}}_{,\xi\xi\xi} \, {\Delta\xi_r}^3 + 3{F^{\boldsymbol{a}}}_{,\xi\xi\eta} \, {\Delta\xi_r}^2 \Delta\eta_r + 3{F^{\boldsymbol{a}}}_{,\eta\eta\xi} \, {\Delta\eta_r}^2 \Delta\xi_r + {F^{\boldsymbol{a}}}_{,\eta\eta\eta} \, {\Delta\eta_r}^3 \right) \right| \tag{B.13}$$

Where $\boldsymbol{\alpha} = (\xi_\alpha, \eta_\alpha)$ is an unknown location in $(\xi_1, \xi_2) \otimes (\eta_1, \eta_2)$ whose derivatives $F^{\boldsymbol{\alpha}}{}_{,\xi\xi\xi}$, $F^{\boldsymbol{\alpha}}{}_{,\xi\xi\eta}$, $F^{\boldsymbol{\alpha}}{}_{,\eta\eta\xi}$ and $F^{\boldsymbol{\alpha}}{}_{,\eta\eta\eta}$ lead to the maximum error, and $\Delta\xi_r = 0.72\,\Delta\xi$ and $\Delta\eta_r = 0.72\,\Delta\eta$ are the representative increments, being $\Delta\xi$ and $\Delta\eta$ the increments in $\xi$ and $\eta$ directions between a regular spaced set of control points. $F$ values at parameter coordinates corresponding to control points must be analytically calculated.

We start with the error of a $p$-degree polynomial interpolation to a function $f(\xi)\colon \mathbb{R}^1 \to \mathbb{R}^1$ using a set of $p+1$ points. That interpolation can be expressed in Newton's polynomials form (B.14).

$$\begin{aligned} q(\xi) = {} & f(\xi_0) + f[\xi_1, \xi_0](\xi - \xi_0) + f[\xi_2, \xi_1, \xi_0](\xi - \xi_1)(\xi - \xi_0) + \cdots \\ & + f[\xi_p, \cdots, \xi_0](\xi - \xi_p) \cdots (\xi - \xi_0) \end{aligned} \tag{B.14}$$

Where $f$ is known at locations $\xi_0, \xi_1, \ldots, \xi_p$ and the finite difference are obtained as (B.15), being the first one (B.20).

$$f[\xi_p, \xi_{p-1}, \cdots, \xi_1, \xi_0] = \frac{f[\xi_p, \xi_{p-1}, \cdots, \xi_1] - f[\xi_{p-1}, \cdots, \xi_1, \xi_0]}{\xi_n - \xi_0} \tag{B.15}$$

$$f[\xi_1, \xi_0] = \frac{f(\xi_1) - f(\xi_0)}{\xi_1 - \xi_0} \tag{B.16}$$

Equation (B.14) has the same structure as Taylor's polynomial and the error committed in this interpolation has a similar expression to Taylor's error [33], which is expressed in its Lagrange form as (B.17).

$$E_p \leq \frac{1}{(p+1)!} f^{(p+1)}(\xi_\alpha) \prod_{i=0}^{p} (\xi - \xi_i) \tag{B.17}$$

Where $\xi_\alpha$ is an unknown location within $(\xi_0, \xi_p)$ whose $p+1$ derivative maximises the error, and $\xi$ is the location where we want to estimate $f$ value using the polynomial interpolation.

For the particular case of $p = 2$, the error is expressed as (B.18).

$$E_2 \leq \frac{1}{3!} f^{(3)}(\xi_\alpha)(\xi - \xi_0)(\xi - \xi_1)(\xi - \xi_2) \tag{B.18}$$

The error when using spline instead a polynomial has similar expression as demonstrated further on. Let $C(\xi)$ be a 2-degree spline defined within

certain knot span. Since degree is 2, the number of basis functions influential on that span are three, which we call $N_1(\xi)$, $N_2(\xi)$ and $N_3(\xi)$. Then any $C(\xi)$ value is calculated as (B.19), where $z_i$ are the control points coordinates.

$$C(\xi) = N_1(\xi)\, z_1 + N_2(\xi)\, z_2 + N_3(\xi)\, z_3 \tag{B.19}$$

Recall that each basis function is a 2-degree polynomial within the span [34], hence each basic function approximates to a function $f_i(\xi)$, i.e. $N_i(\xi) \approx f_i(\xi)$, so that their linear combination with control points coordinates $z_i$ result the function $f(\xi)$ as shown in equation (B.20).

$$f(\xi) = f_1(\xi)\, z_1 + f_2(\xi)\, z_2 + f_3(\xi)\, z_3 \tag{B.20}$$

Therefore the error committed within each basis function $N_i$ has the same expression than the Newton's polynomial approximation (B.21).

$$E_{2_i} \leq \frac{1}{3!} {f_i}^{\alpha}{}_{,\xi\xi\xi}\, (\xi - \xi_0)(\xi - \xi_1)(\xi - \xi_2) \tag{B.21}$$

Where ${f_i}^{\alpha}{}_{,\xi\xi\xi}$ indicates third derivative of $f_i$ at $\xi_\alpha$, being $\xi_\alpha$ an unknown location within $(\xi_0, \xi_2)$ whose ${f_i}^{\alpha}{}_{,\xi\xi\xi}$ maximises the error, $\xi$ is the location where we want to estimate $f$ value using the spline and $\xi_0, \xi_1$ and $\xi_2$ are the parameter coordinates of control points. The error when using $C$ is therefore the linear combination of the three errors as expressed in equation (B.22).

$$E_{2_C} \leq \frac{1}{3!} \left( {f_1}^{\alpha}{}_{,\xi\xi\xi}\ \ z_1 + \ {f_2}^{\alpha}{}_{,\xi\xi\xi}\ \ z_2 + {f_3}^{\alpha}{}_{,\xi\xi\xi}\ \ z_3 \right)\ (\xi - \xi_0)(\xi - \xi_1)(\xi - \xi_2) \tag{B.22}$$

In (B.26) the expression within left brackets is the third derivative of $f$, hence that equation can be expressed as (B.23).

$$E_{2_C} \leq \frac{1}{3!} f^{\alpha}{}_{,\xi\xi\xi}\, (\xi - \xi_0)(\xi - \xi_1)(\xi - \xi_2) \tag{B.23}$$

Note that (B.23) has the same structure as equation (B.18). To generalize (B.23) for any location within the knot span, and assuming control points with parameter coordinates equally spaced $\Delta\xi$, we locate a representative $\xi$ at mid point of one of the intervals and calculate the representative interval $\Delta\xi_r$ as equation (B.24). **Figure B.3** illustrates the location of this representative $\xi$ coordinate.

$$\Delta\xi_r = \left(\frac{1}{2}\Delta\xi\ \frac{1}{2}\Delta\xi\ \frac{3}{2}\Delta\xi\right)^{1/3} \approx 0.72\,\Delta\xi \tag{B.24}$$

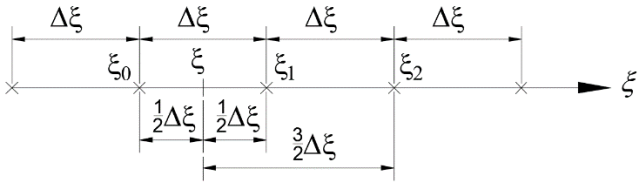


**Figure B.3** Location of representative coordinate $\xi$.

Using the representative increment $\Delta\xi_r$ equation (B.23) might be expressed as (B.25). Which establishes the estimated maximum error for any location for a 2-degree spline curve fitting to a set of equally spaced points.

$$E_{2_C} \leq \frac{1}{3!} f^{\alpha}{}_{,\xi\xi\xi}\,\Delta\xi_r{}^3 \tag{B.25}$$

Extension to spline surface $T(\xi,\eta)$ that approximates to a scalar function $F(\xi,\eta)$ involves chain rule for derivatives calculation. Here we show directly the error result for the sake of brevity. Equation (B.26) shows error for $p$-degree and equation (B.27) is particularized for 2-degree case with representative increments. The fitted surface $T(\xi,\eta)$ is to have a rectangular domain shared with $F(\xi,\eta)$ defined as $(\xi_1,\xi_2)\otimes(\eta_1,\eta_2)$.

$$E_{p_S} \leq \sum_{m=0}^{p+1} \binom{p}{m} \frac{\partial^{p+1}}{\partial\xi^m \partial\eta^{p+1-m}} F(\xi_\alpha,\eta_\alpha) \prod_{i=0}^{m}(\xi-\xi_i) \prod_{j=0}^{p+1-m}(\eta-\eta_j) \tag{B.26}$$

$$E_{2_S} \leq \left|\frac{1}{3!}\left(F^{\boldsymbol{\alpha}}{}_{,\xi\xi\xi}\ \Delta\xi_r{}^3 + 3F^{\boldsymbol{\alpha}}{}_{,\xi\xi\eta}\,\Delta\xi_r{}^2\Delta\eta_r + 3F^{\boldsymbol{\alpha}}{}_{,\eta\eta\xi}\,\Delta\eta_r{}^2\Delta\xi_r + F^{\boldsymbol{\alpha}}{}_{,\eta\eta\eta}\,\Delta\eta_r{}^3\right)\right| \tag{B.27}$$

In (B.27) $\Delta\xi_r$ and $\Delta\eta_r$ are the representative increments, calculated as (B.24). The derivatives of $F$ are at one location $\boldsymbol{\alpha} = (\xi_\alpha,\eta_\alpha)$ within the domain $(\xi_1,\xi_2)\otimes(\eta_1,\eta_2)$ such hat the computed error is maximum.

In our case in particular, functions to approximate is $F_i = S_{j,\beta}$. Sub-indexes values are $j = 1,2,3$ for $x$, $y$ and $z$; and $\beta = 1,2$ for $\xi$ and $\eta$. Some examples of expressions for derivatives required in equation (B.27) are provided below:

$$F_{1,\xi\xi\xi} = S_{x,\xi\xi\xi\xi}$$

$$F_{1,\xi\xi\eta} = S_{x,\xi\xi\xi\eta}$$

$$F_{4,\eta\eta\xi} = S_{x,\eta\eta\eta\xi}$$

$$F_{6,\eta\eta\xi} = S_{z,\eta\eta\eta\xi}$$

*B.2 Directional derivatives computed from the dS-mesh*

The norm of $\theta$-directional derivative can be estimated at certain location $\boldsymbol{\xi}^a$ from the *dS-mesh* by linear interpolation by following next three steps. Firstly partition of *dS-mesh* where $\boldsymbol{\xi}^a$ lies is identified, secondly the *main derivatives* ($\boldsymbol{S}_{,\xi}$ and $\boldsymbol{S}_{,\eta}$) of that element nodes are extracted (they are analytically calculated previously) and the $\theta$-directional derivative is computed at each node as per equation (B.1). Third, the norm of $\theta$-directional derivative at $\boldsymbol{\xi}^a$ is estimated by bi-linear interpolation of norms from the element nodes, as equation (B.28).

$$\left\| \boldsymbol{S}_{,\lambda}^{\xi^a} \right\| \approx \sum_{k=1}^{4} N^k(\boldsymbol{\xi}^a) \left\| \boldsymbol{S}_{,\lambda}^{k} \right\| \tag{B.28}$$

Where $N^k(\boldsymbol{\xi}^a)$ and $\boldsymbol{S}_{,\lambda}^{k}$ are the value of the $k^{th}$ basis function at position $\boldsymbol{\xi}^a$ and the derivative value at $k^{th}$ node respectively.

The *dS-mesh* is non-conformal, therefore if the location $\boldsymbol{\xi}^a$ lies at the edge between two elements, only the smallest one, more accurate, is considered.

This procedure leads to a high speed directional derivative estimation. However recall that *dS-mesh* was refined to control error for physical length computation, equation (B.29), and not for derivatives itself. The error of the derivative linear interpolation is calculated as (B.30).

$$E \leq \frac{-1}{12} \left\| \boldsymbol{S}_{,\lambda}^{\alpha} \right\|'' \left\| \boldsymbol{\xi}^b - \boldsymbol{\xi}^a \right\|^3 \tag{B.29}$$

$$E_d \leq \frac{1}{2} \left\| \boldsymbol{S}_{,\lambda}^{\alpha} \right\|'' \left\| \boldsymbol{\xi}^b - \boldsymbol{\xi}^a \right\| \tag{B.30}$$

The relationship between both errors is:

$$\frac{E_d}{E} \leq \frac{3}{\left\| \boldsymbol{\xi}^b - \boldsymbol{\xi}^a \right\|^2} \tag{B.31}$$

It is clear that the estimated error for derivatives interpolation is greater than error for path length trapezoidal rule (we assume $\|\boldsymbol{\xi}^b - \boldsymbol{\xi}^a\| < 1$). Since *dS-mesh* is generated to control error (B.29) and not (B.30), the error or this estimation of derivatives is not fully controlled. Therefore this method, that is faster than the splines fitting (section B.1) can be used only if the accuracy is not critical in the triangulation process.

## *Appendix C: Bounded isotropic triangulation*

This appendix defines the bounded isotropic triangulation (*BIT*) and explains how to obtain a *BIT* and its characterization parameters (angles, triangles sizes and valences) corresponding to any triangulation (*AT)*. *BIT* is the ideal isotropic triangulation version of *AT* and therefore the highest quality triangulation ideally achievable.

### *C.1 Bounded isotropic triangulation (BIT)*

*BIT* is a portion of unbounded isotropic triangulation. The latter extends to infinite, i.e. presents no boundary edges, all angles are sixty degrees, all nodes valences are six and all triangles are the same size (**Figure C.1** (a)). The former is bounded by four edges forming a rhomboid (**Figure C.1** (b)), therefore not all valences are six, however sizes and angles are preserved as ideal isotropic. Parameters that characterize the *BIT* are the number of rows and columns, designated as $r_b$ and $c_b$. **Figure C.1** provides one example.

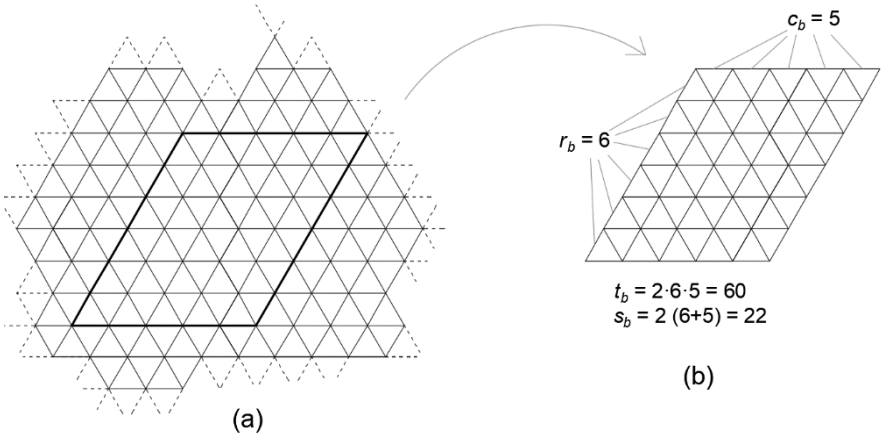


**Figure C.1.** Ideal isotropic triangulation (a) and extraction of 6 x 5 *BIT* (b).

The number of triangles ($t_b$) and number contour segments ($s_b$) of *BIT* are computed from $r_b$ and $c_b$ as shown in equations (C.1) and (C.2).

$$t_b = 2\, r_b\, c_b \tag{C.1}$$

$$s_b = 2\,(r_b + c_b) \tag{C.2}$$

As mention, *BIT* has all angles equal to sixty degrees and all triangles are the same size, but not all vertexes valences are six. Vertexes valences are calculated as per **Table C.1**.

**Table C.1** Frequency (number of instances) of vertexes valences.

| Valence | Frequency |
|---|---|
| 1 | 2 |
| 2 | 2 |
| 3 | $2((c_b - 1) + (r_b - 1))$ |
| 4 | 0 |
| 5 | 0 |
| 6 | $(c_b - 1) \times (r_b - 1)$ |

*C.2 Computation of BIT correspondent to any triangulation (AT)*

Given *AT* mesh we can find its correspondent *BIT* using equations (C.1) and (C.2). By manipulation of them we arrive to expressions (C.3).

$$\begin{aligned} r_b &= \frac{s_a}{4} + \sqrt{\frac{{s_a}^2}{16} - \frac{t_a}{2}} \\ s_b &= \frac{s_a}{4} - \sqrt{\frac{{s_a}^2}{16} - \frac{t_a}{2}} \end{aligned} \tag{C.3}$$

In equations (C.3) we input the *AT* number of triangles and edge segments ($t_a$ and $s_a$) and obtain its correspondent *BIT* number of rows and columns ($r_b$ and $c_b$) and afterwards the *BIT* vertexes valences frequency as per **Table C.1**. **Figure C.2** provides one example of the *BIT* associated to *AT*.

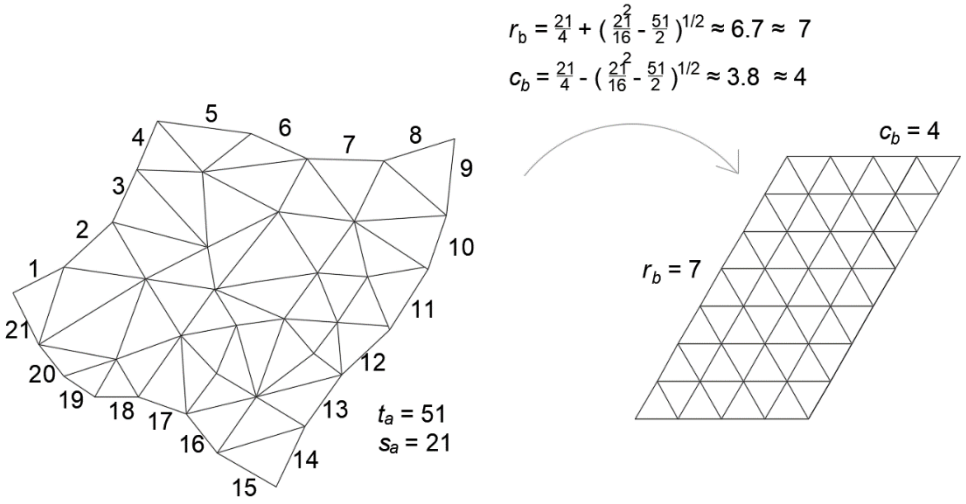


**Figure C.2.** Computation of *BIT* (right) for a given *AT* (left). Contour segments of *AT* are numbered.

*BIT* is the closest version of *AT* to an ideal isotropic triangulation, then the closest the parameters (angles, sizes and valences) of *AT* are to its *BIT* the higher quality presents the former.

## *Appendix D: Data for numerical examples*

### *D.1 Example 7.1: single surface*

**Table D.1.** *QIT* inputs.

| *R* | Threshold distance to edge | Tolerance for curves (%) | Tolerance for surfaces (%) | Tolerance for derivatives (%) |
|---|---|---|---|---|
| 12 and 6 | 4 and 2 | 1.0 | 4.0 | 15.0 |

**Table D.2.** Surface NURBS features.

| | |
|---|---|
| No. control points | $n = 6 \quad m = 5$ |
| Degrees | $p = 2 \quad q = 2$ |
| Knot vectors | $\Xi = \{000\ 0.25\ 0.50\ 0.75\ 111\}$<br>$H = \{000\ 0.50\ 0.50\ 111\}$ |

**Table D.3.** Control points coordinates and weights.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0, 0, -90, 1 | 0, 0, 0, 0.707 | 0, 30, 0, 1 | 0, 60, 0, 0.707 | 0, 60, -30, 1 |
| 2 | 30, 0, -15, 1 | 30, 15, 0, 1 | 30, 30, 0, 1 | 30, 45, 0, 1 | 30, 60, -15, 1 |
| 3 | 95, 0, 0, 1 | 95, 15, 0, 1 | 95, 30, 0, 1 | 95, 45, 0, 1 | 95, 60, 0, 1 |
| 4 | 150, 0, 0, 6 | 137.5, 15, 0, 1 | 125, 30, 0, 1 | 112.5, 45, 0, 1 | 100, 60, 0, 1 |
| 5 | 150, 65, 0, 1 | 137.5, 65, 0, 1 | 125, 65, 0, 1 | 112.5, 65, 0, 1 | 100, 65, 0, 1 |
| 6 | 150, 100, 0, 1 | 137.5, 100, 0, 1 | 125, 100, 0, 1 | 112.5, 100, 0, 1 | 100, 100, 0, 1 |

## *D.2 Example 7.2: three contiguous surfaces*

**Table D.4.** *QIT* inputs.

| $R$ | Threshold distance to edge | Tolerance for curves (%) | Tolerance for surfaces (%) | Tolerance for derivatives (%) |
|---|---|---|---|---|
| 5 | 1.67 | 1.0 | 4.0 | 15.0 |

- Bottom surface:

**Table D.5.** Surface NURBS features.

| | |
|---|---|
| No. control points | $n = 3 \quad m = 5$ |
| Degrees | $p = 2 \quad q = 2$ |
| Knot vectors | $\Xi = \{000 \ 111\}$<br>$H = \{000\ 0.50\ 0.50\ 111\}$ |

**Table D.6.** Control points coordinates and weights.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 108, 40, 0, 1 | 108, -28, 0, 0.707 | 40, -28, 0, 1 | -28,-28, 0, 0.707 | -28, 40, 0, 1 |
| 2 | 80, 40, 75, 1 | 80, 0, 75, 0.707 | 40, 0, 75, 1 | 0, 0, 75, 0.707 | 0, 40, 75, 1 |
| 3 | 80, 40, 150, 1 | 80, 0, 150, 0.707 | 40, 0, 150, 1 | 0, 0, 150, 0.707 | 0, 40, 150, 1 |

- Mid surface:

**Table D.7.** Surface NURBS features.

| | |
|---|---|
| No. control points | $n = 5 \quad m = 2$ |
| Degrees | $p = 2 \quad q = 1$ |
| Knot vectors | $\Xi = \{000\ 0.50\ 0.50\ 111\}$<br>$H = \{00 \ 11\}$ |

**Table D.8.** Control points coordinates and weights.

| | 1 | 2 |
|---|---|---|
| 1 | 0, 40, 150, 1 | 20, 40, 150, 1 |
| 2 | 0, 0, 150, 0.707 | 20, 20, 150, 0.707 |
| 3 | 40, 0, 150, 1 | 40, 20, 150, 1 |
| 4 | 80, 0, 150, 0.707 | 60, 20, 150, 0.707 |
| 5 | 80, 40, 150, 1 | 60, 40, 150, 1 |

- Top surface:

**Table D.9.** Surface NURBS features.

| | |
|---|---|
| No. control points | $n = 3 \quad m = 5$ |
| Degrees | $p = 2 \quad q = 2$ |
| Knot vectors | $\Xi = \{000 \ 111\}$<br>$H = \{000\ 0.50\ 0.50\ 111\}$ |

**Table D.10.** Control points coordinates and weights.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 60, 40, 150, 1 | 60, 20, 150, 0.707 | 40, 20, 150, 1 | 20, 20, 150, 0.707 | 20, 40, 150, 1 |
| 2 | 60, 40, 210, 0.707 | 60, 20, 230, 0.50 | 40, 20, 230, 0.707 | 20, 20, 230, 0.707 | 20, 40, 210, 1 |
| 3 | 60, 100, 210, 1 | 60, 100, 230, 0.707 | 40, 100, 230, 1 | 20, 100, 230, 0.707 | 20, 100, 210, 1 |

## 11. REFERENCES


[1] T.J.R. Hughes, J.A. Cottrell, Y. Bazilevs, 'Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement'. Computer Methods in Applied Mechanics and Engineering, **194**(39), pp. 4135-4195, 2005

[2] J.A. Cottrell, T.J.R. Hughes, Y. Bazilevs, 'Isogeometric analysis: toward integration of CAD and FEA', Oxford: Wiley, 2009.

[3] T.J. Baker, 'Mesh generation: Art or science?', Progress in Aerospace Sciences, **41**(1), pp. 29-63, 2005.

[4] K. Shimada, 'Current issues and trends in meshing and geometric processing for computational engineering analyses', Journal of Computing and Information Science in Engineering, **11**(2), pp. 021008, 2011.

[5] Q. Du, V. Faber, M. Gunzburger, 'Centroidal Voronoi tessellations: Applications and algorithms', SIAM Review, **41**(4), pp. 637-676, 1999.

[6] P.J. Frey, H. Borouchaki, P. George, '3D Delaunay mesh generation coupled with an advancing-front approach', Computer Methods in Applied Mechanics and Engineering, **157**(1-2), pp. 115-131, 1998.

[7] R. Löhner, P. Parikh, 'Generation of three-dimensional unstructured grids by the advancing-front method', International Journal for Numerical Methods in Fluids, **8**(10), pp. 1135-1149, 1988.

[8] K. Nakahashi, D. Sharov, 'Direct surface triangulation using the advancing front method', 12th Computational Fluid Dynamics Conference 1995, pp. 1686, 1995.

[9] J.R. Tristano, S.J. Owen, S.A. Canann, 'Advancing Front Surface Mesh Generation in Parametric Space Using a Riemannian Surface Definition', IMR 1998, pp. 429-445, 1998.

[10] M.A. Yerry, M.S. Shephard, 'Automatic three-dimensional mesh generation by the modified-octree technique', International Journal for Numerical Methods in Engineering, **20**(11), pp. 1965-1990, 1984.

[11] M.S. Shephard, M.K. Georges, 'Automatic three-dimensional mesh generation by the finite octree technique', International Journal for Numerical Methods in Engineering, **32**(4), pp. 709-749, 1991.

[12] X. Sheng, B.E. Hirsch, 'Triangulation of trimmed surfaces in parametric space', Computer-Aided Design, **24**(8), pp. 437-444, 1992.

[13] H. Borouchaki, P. Laug, P. George, 'Parametric surface meshing using a combined advancing-front generalized Delaunay approach', International Journal for Numerical Methods in Engineering, **49**(1-2), pp. 233-259, 2000.

[14] R.J. Cripps, S. Parwana, 'A robust efficient tracing scheme for triangulating trimmed parametric surfaces', Computer-Aided Design, **43**(1), pp. 12-20, 2011.

[15] E. Béchet, J. Cuilliere, F Trochu, 'Generation of a finite element MESH from stereolithography (STL) files', Computer-Aided Design, **34**(1), pp. 1-17, 2002.

[16] D. Wang, O. Hassan, K. Morgan, N. Weatherill, 'EQSM: An efficient high quality surface grid generation method based on remeshing', Computer Methods in Applied Mechanics and Engineering, **195**(41-43), pp. 5621-5633, 2006.

[17] S.W. Yang, Y. Choi, 'Triangulation of CAD data for visualization using a compact array-based triangle data structure', Computers & Graphics, **34**(4), pp. 424-429, 2010.

[18] E. Marchandise, J. Remacle, C. Geuzaine, 'Optimal parametrizations for surface remeshing', Engineering with Computers, **30**(3), pp. 383-402, 2014.

[19] R. Aubry, S. Dey, E.L. Mestreau, B.K. Karamete, D. Gayman, 'A robust conforming NURBS tessellation for industrial applications based on a mesh generation approach', Computer-Aided Design, **63**, pp. 26-38, 2015.

[20] J. Guo , F. Ding, X. Jia, D. Yan, 'Automatic and high-quality surface mesh generation for CAD models', Computer-Aided Design, **109**, pp. 49-59, 2019.

[21] V. Surazhsky, P. Alliez, C. Gotsman, 'Isotropic remeshing of surfaces: a local parameterization approach', 2003.

[22] P. Alliez, E.C. De Verdire, O. Devillers, M. Isenburg, 'Isotropic surface remeshing', 2003 Shape Modeling International. 2003, IEEE, pp. 49-5, 2003.

[23] S.W. Yang, Y. Choi, 'Triangulation of CAD data for visualization using a compact array-based triangle data structure', Computers & Graphics, **34**(4), pp. 424-429, 2010.

[24] Y. Li, W. Wang, R. Ling, C. Tu, 'Shape optimization of quad mesh elements', Computers & Graphics, **35**(3), pp. 444-451, 2011.

[25] M.S. Ebeida, A. Patney, J.D. Owens, E. Mestreau, 'Isotropic conforming refinement of quadrilateral and hexahedral meshes using two-refinement templates', International Journal for Numerical Methods in Engineering, **88**(10), pp. 974-985, 2011.

[26] M. Tsai, C. Cheng, M. Cheng, 'A real-time NURBS surface interpolator for precision three-axis CNC machining', International Journal of Machine Tools and Manufacture, **43**(12), pp. 1217-1227, 2003.

[27] P. Bézier, 'Numerical control: mathematics and applications', 1970.

[28] A.R. Forrest, 'Interactive interpolation and approximation by Bézier polynomials', The Computer Journal, **15**(1), pp. 71-79, 1972.

[29] S. Bernstein, 'Démonstration du théoreme de Weierstrass fondée sur le calcul des probabilities', Comm.Soc.Math.Kharkov, **13**, pp. 1-2, 1912.

[30] M.G. Cox, 'The numerical evaluation of B-splines', IMA Journal of Applied Mathematics, **10**(2), pp. 134-149, 1972.

[31] C. De Boor, 'On calculating with B-splines', Journal of Approximation theory, **6**(1), pp. 50-62, 1972.

[32] E. Isaacson, H.B. Keller, 'Analysis of numerical methods', John Wiley & Sons, 1966.

[33] S.C. Chapra, R.P. Canale, 'Numerical methods for engineers', Boston: McGraw-Hill Higher Education, 2010.

[34] L. Piegl, W. Tiller, 'The NURBS Book', 2 edn. Springer-Verlag, 1996.