Implementation of improvements to Isogeometric Analysis for trimmed-coupled solids directly extracted from Computer Aided Design environment

Daniel Herrero Adán

A thesis submitted in partial fulfilment of the requirements of the University of the West of England, Bristol for the degree of Doctor of Philosophy.

Faculty of Environment and Technology.

August 2021

DECLARATION

I declare that this thesis presented by me entitled 'Implementation of improvements to Isogeometric Analysis for trimmed-coupled solids directly extracted from Computer Aided Design environment' is my own investigation and has not previously been submitted for a degree or similar award at the University of the West of England or any other institution. To the best of my knowledge and belief, no material in this thesis has been previously published or written by another person, except where due reference is made.

Daniel Herrero Adán

Abstract

This research provides an algorithm to analyse structures composed of solids that are directly extracted from Computer Aided Design (CAD) environment. The analysis assumes linear elastic material and small displacements, being the result the displacements field of the structure under a set of forces and constraints. Since the equation to obtain such displacements is not analytically solvable, the solution is approximated by a linear combination of the so-called basis functions. The use of domains from CAD in analysis without additional meshing is the original purpose of the Isogeometric Analysis (IGA). The basis functions used by IGA are Non-uniform Rational B-splines (NURBS). Since any model developed in CAD is geometrically defined by NURBS, it already comes discretized into NURBS, and can be used directly by IGA to obtain the displacement field without additional meshing.

Therefore this work is based on IGA techniques. Solids are defined in CAD as a set of faces that wrap their volume. This arrangement is valid for representation, but not for analysis as the solid itself does not exist (only the hollow volume). Therefore the first challenge that appears in this work is the generation of the solid enclosed by those faces, or solid parametrization. In addition, the solids can be trimmed by surfaces to achieve the final shape of the domain. The existence of such trimming surfaces presents two challenging aspects. Firstly, the discretization with NURBS as they come from CAD is not valid anymore, being necessary to find an alternative discretization scheme. Secondly, the trimming surfaces that bound the discretization are unknown in the discretization space, being necessary to approximate them. The domain may be formed by multiple solids that are coupled in a weak manner using Lagrange multiplier approach on the coupling interfaces. Apart from tackling these problems, two novel techniques have been implemented in this algorithm: a new triangulation technique for surfaces to allow the representation of results on the faces of the solids, and a new approach to calculate coordinates in the solid parameter space (point projection) that brings more robustness to the computation.

The first part of the algorithm is devised to read the content of the CAD files and transform into suitable data for analysis. This transformation includes the solid parametrization. Then the trimming surfaces are approximated to allow the accurate definition of the limits of the discretization, which is carried out by tetrahedral mesh. The application of constraints and coupling with Lagrange multiplier approach follows the process to end up in the analysis results, which are the displacements of the domain. The stresses are also computed by a post-processor developed in this work. The performance of this algorithm is proved by a set of examples at the rear of the thesis.

Although the trimmed solids need discretization different from NURBS, which vanishes partially the IGA original purpose, this thesis stands as another step forward in the IGA development towards the CAD-analysis integration for solids.

	Number of words
Main body	38 360
Appendixes	22 070
Total	60 430

Acknowledgments

I would like to express my gratitude to the University of the West of England for its financial support over the first three years of this research. Thanks to my supervisory team, specifically to Dr Arnaud Marmier for his key advices at the key moments that deviated the path of my thesis for good, and to Dr Rui Cardoso who provided guidance in the theoretical background. Thanks also to the Graduate School of this university for their efficient support and advises (Marisa Downham, Claire McLaren, Samantha Watts among others), and to Trina Pitman, who has solved the problems to access to the campus derived from the COVID-19 restrictions.

The collaboration at CIMNE (International Centre for Numerical Methods in Engineering, Technical University of Cataluña) was crucial in the development of this research. Hence I must express my gratitude to Dr Riccardo Rossi, researcher at CIMNE and to Tobias Teschemacher, researcher at the Technical University of Munich and collaborator at CIMNE.

I must thank to my current company Hydrock, because they provided to me flexibility during the last two years to attend to the campus to finalize this research, and the opportunity to carry on working to be able to pay the bills. In particular, thanks to Antonio Juarez, Jake Beacham and Ryan Manders for their credit and confidence.

The colleagues (now friends) I found during this journey have been amazing. Thanks to Dr Barkha Javed, Charf Mahammedi and Dr Olukayode Olusola Awonuga for sharing the painful moments of this PhD process, improving the quality of the time spent at the campus and for having numerous funny moments.

During this time in Bristol I found amazing people which provided me support and even advice since some of them passed through this PhD experience, thank you all. In particular, thanks to Nikos Kapitsinis for revising chapter 1. I must express my gratitude to my friends from Madrid, which always demonstrated support and loyalty (David, Javi, Juan Andrés, Juan Pablo and Manolo).

Finally, and the most important, my gratitude to my family for supporting me whatever happens, and to my life partner, Fátima Palacios Fernández-Hijicos, for her love and for illuminating this path with her smile, enthusiasm, energy and support.

“En el arte de la marinería, más sabe el más simple marinero que el mayor letrado del mundo”

Los trabajos de Persiles y Sigismunda.

Miguel de Cervantes Saavedra, 1617.

A mis padres

CONTENTS

DETAILED INDEX

1. **Introduction**

1.1. Background and the research problem

This work lies within the Computational Mechanics field whose purpose is to simulate and predict the response of one domain under a set of constraints. In this work, the domain, called Ω, is one structure in static equilibrium composed of solids (and no other forms such as beams or shells). The solid is bounded by surfaces called Γ. The domain is subjected to constraints which are prescribed displacements and forces. The former are also called essential or Dirichlet boundary conditions and are applied on a portion of the boundary called Γ_D. The forces may be applied within the domain (body forces) or may be tractions on the surface, also known as natural or Neumann boundary conditions. The tractions are applied on surfaces Γ_N.

The response is the displacement field within the domain ($\boldsymbol{u}$). One example is illustrated in **Fig. 1.1**, where one domain subjected to boundary conditions and body forces is illustrated at the left-hand side. The response, depicted at the right-hand side, is the deformed configuration after the points of the domain have suffered displacements. Displacement at a and b locations are also shown, being decomposed into its orthogonal components for location a.

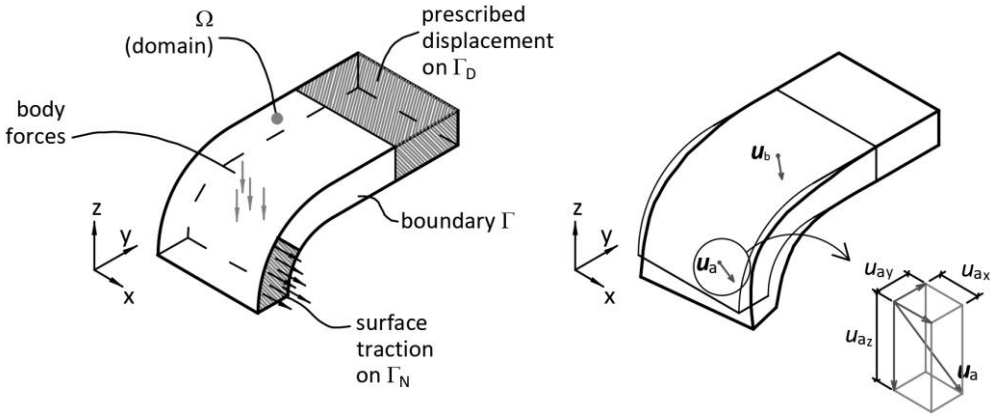


Fig. 1.1 Solid subjected to constraints (left) and its response (right).

The displacement field $\boldsymbol{u}$ is the solution of a partial differential equation (PDE) that describes de equilibrium of the domain. That PDE is, in general, not analytically solvable for complex domains and needs numerical approximation.

The stress field is also of interest and it is obtained from the derivatives of the displacement field. The material is assumed linear elastic and the displacements sufficiently small such that the geometrical non-linearities can be ignored. The small displacements allow taking as reference for

the integration of the governing equations either the un-deformed or the deformed configuration.

1.1.1. Design process and Isogeometric approach

In practice engineering, the design of a structure or mechanical part involves two stages as shown in **Fig. 1.2**. The first stage, **CAD modelling**, targets the definition of the geometry to comply with the functional requirements such us size or shape. In the second stage, **analysis**, the displacements and stresses under certain constraints are computed to check if the geometry and material are acceptable. The modelling stage is done by computer-aided design (CAD) tools (e.g. AutoCAD®, SolidWorks®, etc.) and the analysis requires solving the equilibrium PDE using numerical methods to approximate the solution.

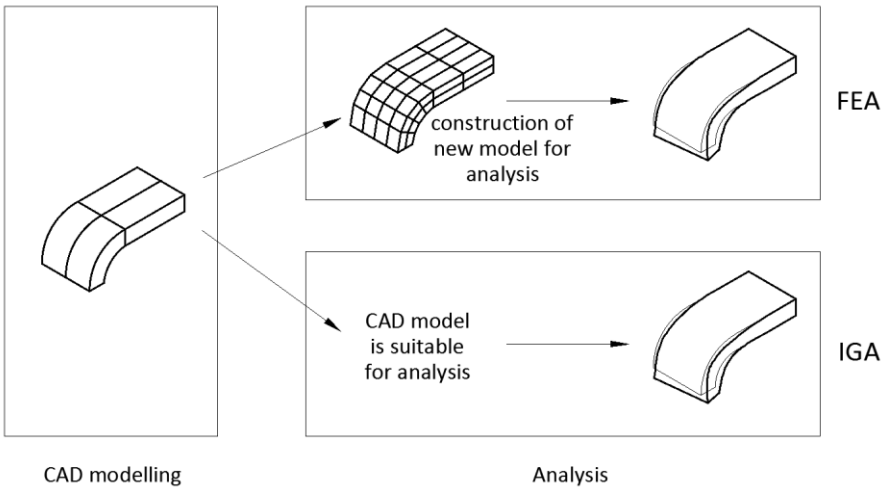


Fig. 1.2 Scheme of design process with two steps (CAD modelling and analysis). FEA and IGA are compared.

The most popular numerical method to approximate PDE is the finite element analysis (FEA), developed within the last 70 years (Clough, 2001). In FEA, the domain is divided into portions called *elements*, where the displacement field is calculated at certain locations of the elements called *nodes*. Then, the displacement field can be approximated within each element by interpolation between nodes using traditionally polynomials. This process of division into elements is called spatial discretization.

FEA has been exploited in many applications for structures, fluids, heat conduction or electromagnetism with excellent performance (Hughes, 2000; Zienkiewicz and Taylor, 2000). However, FEA has an important drawback: the CAD geometry developed in the modelling stage is not directly usable by FEA (Hughes, Cottrell and Bazilevs, 2005). By contrast, the suitable domain for FEA needs to be constructed again with elements defined by polynomials. In other words, the model needs to be done twice: one for modelling and another for analysis (**Fig. 1.2** top-right).

Isogeometric analysis (IGA), that was introduced in 2005 (Hughes, Cottrell and Bazilevs, 2005), was devised to use the CAD model directly for the analysis. IGA avoids constructing the domain twice since the model from CAD itself is suitable for analysis. This suitability occurs because IGA uses NURBS[1] functions instead traditional polynomials to approximate the displacement field being those NURBS the same functions used in CAD to define the model. Therefore, once the model is finished in CAD, it is ready for IGA with no additional treatment (**Fig. 1.2** bottom). In IGA there is also a spatial discretization, as in FEA, with the displacements computed at the so-called control points.

1.1.2. NURBS entities

Geometries in CAD are defined by two types of entities: curves and surfaces, that are visible for the CAD user in the so-called physical space. Let us call both of them NURBS entities. These entities are parametric, i.e. each position of the entity is mapped to the physical space (coordinates x, y, z) from its parameter space (coordinate ξ for curves and ξ, η for surfaces). **Fig. 1.3** shows one NURBS surface with its parameter and physical spaces. NURBS entities are discretized into *elements* that come from the parametrization: one element corresponds to one non-void knot span (Hughes, Cottrell and Bazilevs, 2005). One *element* is highlighted with grey colour in **Fig. 1.3**.

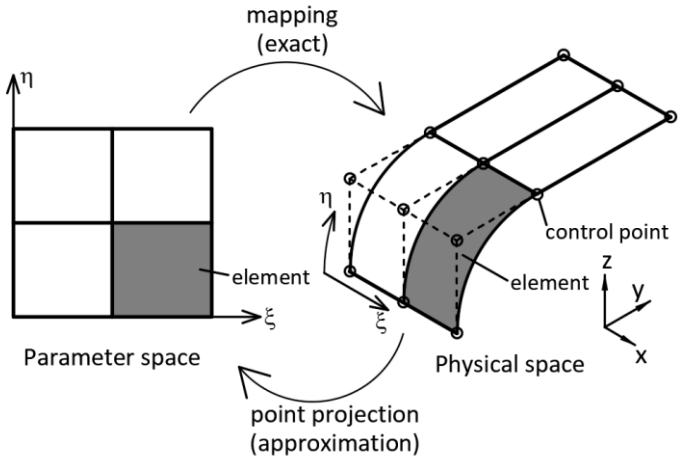


Fig. 1.3 NURBS surface.

The mapping from the parameter to the physical space is a linear combination of NURBS functions, whose coefficients are the coordinates of certain points called *control points*. The shape in the physical space is controlled by the control points positions. The mapping is represented in **Fig. 1.3** by the top arrow. This mapping is the same used by IGA to approximate the displacement field, and hence it allows using NURBS directly in analysis.

[1] NURBS is the acronym of non-uniform rational B-splines.

Mapping is one-directional from the parameter to the physical space, i.e. the inverse mapping has no definition. Therefore, it is not possible to find the exact parameter coordinates for a given physical coordinates. The unique possibility is to calculate an approximate parameter position by iterative techniques known as point projection (Piegl and Tiller, 1996). This approximation process is represented by the bottom arrow in **Fig. 1.3**.

In IGA, the elements arrangement of one NURBS entity is the same as the parameter space partition. This partition is given by the NURBS parametrization as detailed in Chapter 2.

Control points are linked to the parameter space arrangement. For surfaces they are generated as tensor product of control points in both parameter directions. The tensor product impedes the insertion of one single control point, i.e. it forces to insert a whole row of control points. In addition, this row must run parallel to one of the directions in the parameter space. **Fig. 1.4** gives four examples of control points insertions in a NURBS surface. Insertions (a) and (b) are permitted, since the row is parallel to ξ and η directions respectively. Insertion of isolated control points (case *c*) or rows not parallel to one parameter direction (case *d*) are not possible.

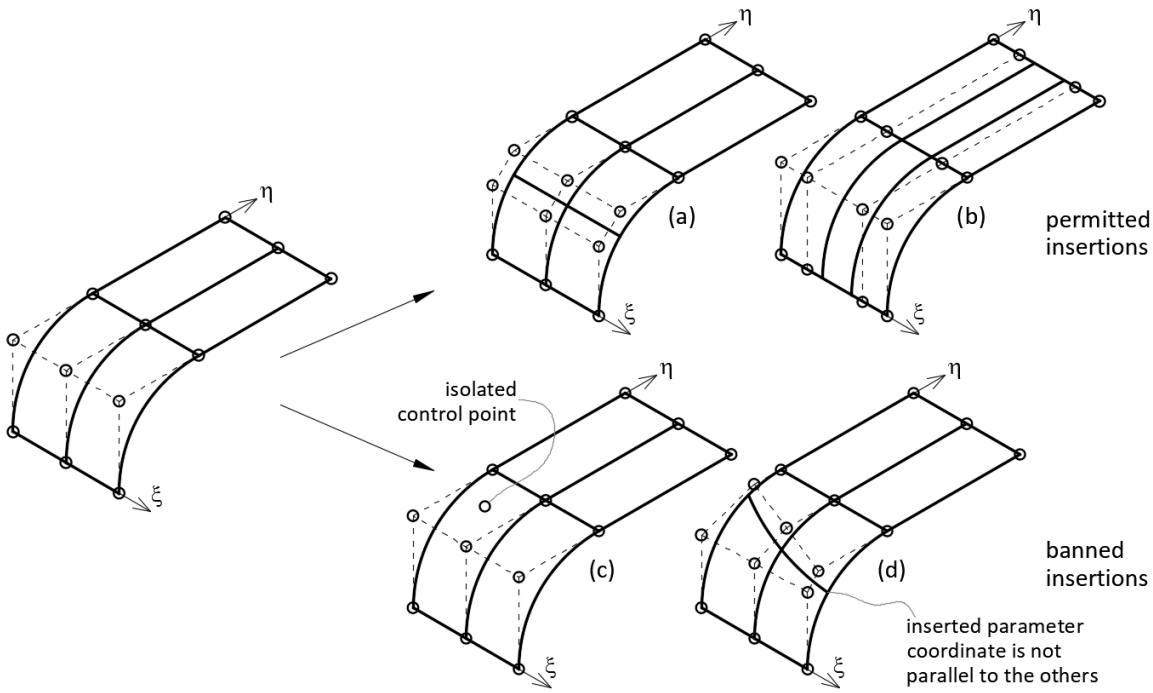


Fig. 1.4 Control points insertions in NURBS surface.

1.1.3. CAD information exchange

NURBS features, including the position of the control points, can be extracted as specific files called Initial Graphics Exchange Specification (IGES). These IGES files are devised to interchange information between different CAD packages.

1.1.4. Challenges in isogeometric analysis for solids

The parameter space of one NURBS surface, and therefore its *elements*, is contained in the IGES files. Then, IGA is applicable directly to NURBS surfaces, since the domain discretization (elements) was already defined during the CAD modelling stage. The only step that needs to be made is to transfer IGES files information from CAD packages to the analysis code[2].

For CAD volumes the case is different. The volume geometry is represented by a collection of surfaces forming a *hollow carcass* that wraps the solid. Therefore, the information stored in the IGES files is not the solid itself but its enclosing surfaces, as shown in **Fig. 1.5**. This geometric representation, called B-rep modelling, is well settled in CAD environment. B-rep description can be found in CAD reference books (Hoschek, Lasser and Schumaker, 1993; Stroud, 2006).

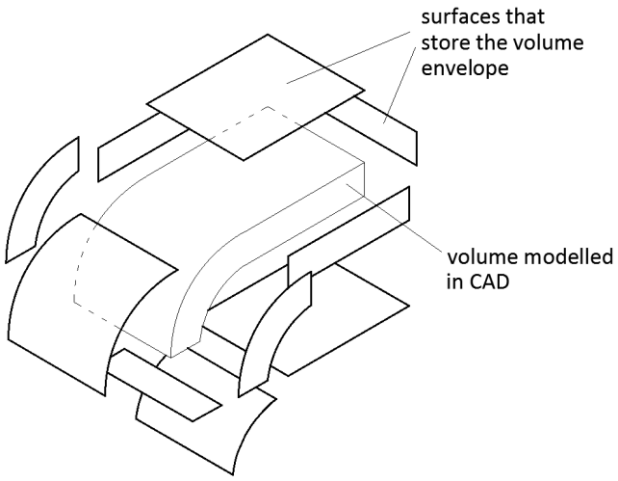


Fig. 1.5 The volume drawn in CAD is in reality a set of surfaces with no solid inside.

As a consequence of this volume representation, IGA is not applicable directly to solids because there is not a solid domain (Ω) but only the boundary surfaces (Γ). Subsequently the solid parameter space, its *elements*, and the domain itself do not exist (recall elements in IGA are generated from the parameter space, section 1.1.2).

Therefore, the first step to apply IGA to a volume modelled in CAD is to create that solid or *fill with domain* the volume enclosed by the envelope surfaces. This process is called **solid parametrization**, resulting in a NURBS solid, with its parameter space, physical space and control points as illustrated in **Fig. 1.6**. The non-existence of NURBS solids in CAD models constitutes one of the main challenges of this thesis.

[2] Rigorously speaking, NURBS surfaces may need some processing previous to the analysis if they are trimmed by curves. However, that process is simpler than volumes as will be shown in this section.

Two more pitfalls appear in IGA for solids that are outlined in this section. Let us assume that the volume is already parametrized. Then the solid elements and control points are given by the parametrization (see **Fig. 1.6**).

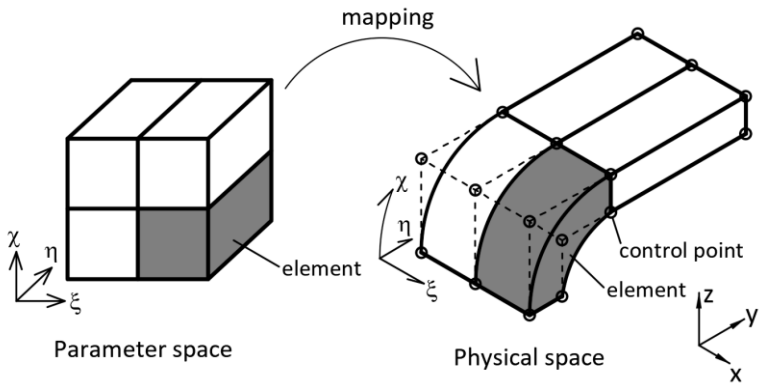


Fig. 1.6 Example of NURBS solid parameter and physical spaces. One element is highlighted in both spaces.

In the majority of the cases, solids in CAD need to be trimmed to achieve the final shape. The trimming brings another issue: the solid elements that come from parametrization do not match the trimmed version. One example is shown in **Fig. 1.7**. At the left-hand side the solid with spatial discretization as the parameter space is depicted, while at the right-hand side the trimmed solid to achieve final shape is shown. The discretization from the left configuration is not valid, observe for example the highlighted element that sticks out of the domain in the trimmed case. As a result of this situation, a **discretization** scheme different from the parametrization is required, with this issue being another challenge in IGA for solids.

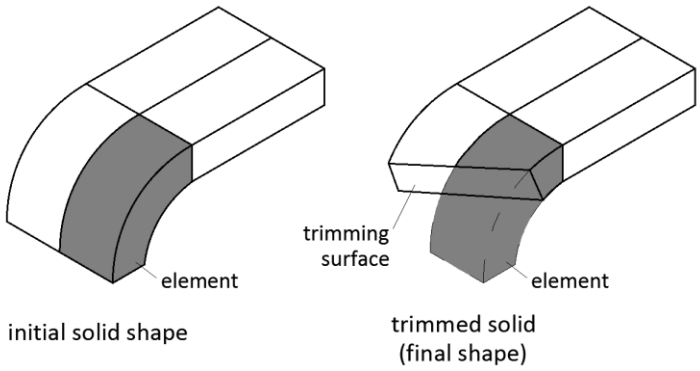


Fig. 1.7 Trimmed solid does not fit elements arrangement from the parametrization.

Further to trimming, coupling solids is often required to achieve the final shape. Two patches are conformal if their parametrization matches at the coupling interface, i.e. control points coincide, as shown in **Fig. 1.8** (left). Otherwise, if they have different parametrization at the interface, they are not conformal as shown in **Fig. 1.8** (right).

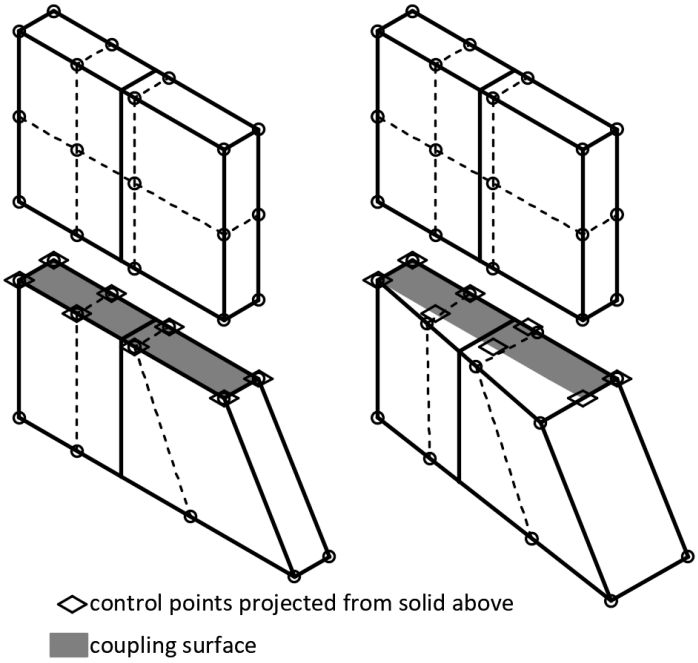


Fig. 1.8 Two conformal (left) and two not conformal (right) solids at the coupling surface

Coupling of two conformal solids is straightforward by linking coincident control points from both domains. It might be the case that previous insertion of control points is required to achieve conformity at the coupling interface. Control points insertion in NURBS domains is a well-settled technique (Piegl and Tiller, 1996). **Fig. 1.9** provides one example for the coupling of two conformal solids. At left, the interface to couple is shown in both solids. At the centre, control points in solid 2 are inserted to match arrangement of solid 1 at the coupling surface. At the right-hand side of the figure both solids are coupled with linked control points, represented with filled dots.

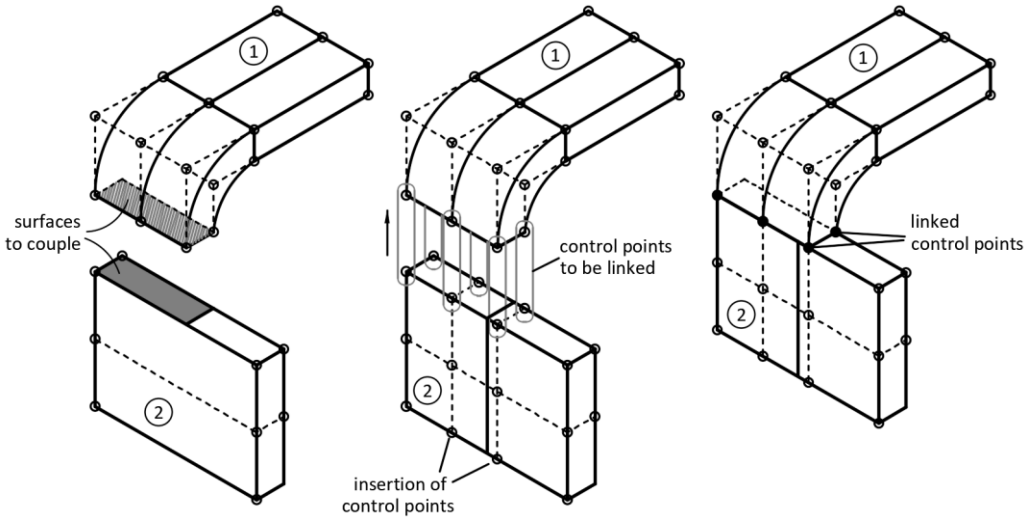


Fig. 1.9 Coupling of conformal solids.

Coupling non-conformal solids does not allow the mentioned control points linking approach. Control points conformity cannot be achieved by control points insertion since the parametrization is different in both solids at the coupling surface. This impossibility stems from

the fact that control points insertion must follow one of the parameter directions (recall **Fig. 1.4**). **Fig. 1.10** shows one example where the coupling surface is trimming one of the patches. At the left-hand side the coupling surface is identified. At the right-hand side both solids are coupled but the linking of the control points is not possible because there is no permitted insertion to reproduce the coupling surface. In general, when one solid is trimmed it loses any potential conformity with other solids.

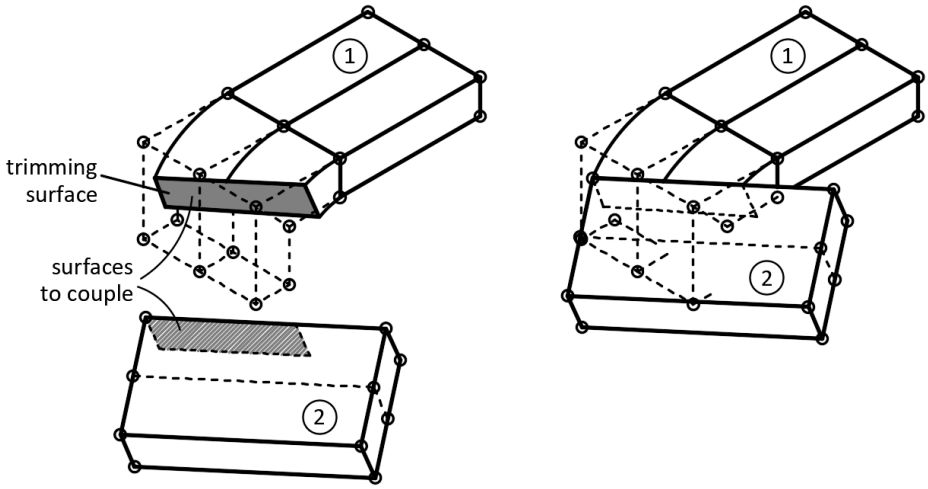


Fig. 1.10 Coupling of non-conformal solids.

The imposition of boundary conditions on a surface of the solid domain presents similar problems to coupling if such surface (the interface) is non-conformal with the solid. Therefore, alternative approach to **impose constraints to interfaces non-conformal with the solids** is required, being another major challenge tackled by this thesis.

To sum up, the three major challenges in IGA for solids are: **solid parametrization**, **discretization of trimmed solids** and **constraints to non-conformal interfaces**.

1.1.5. Why trimmed-coupled domains?

Creating NURBS solids suitable for IGA is challenging, especially for trimmed non-conformal solids. Then, why tackle this problem?

The answer lies in the CAD practise. When defining geometries in CAD, trimming and coupling appear as the most efficient strategies. For certain cases, the former strategy yields simpler domains, while for other cases, the second route is the easiest. Indeed, combining both techniques leads to a powerful tool to obtain any shape in a relatively simple manner, being the most common way to proceed in CAD. As a consequence, these trimmed-coupled domains should be suitable for IGA.

Two examples are presented in order to illustrate the power of trimmed-coupled techniques. Both are bi-dimensional for the sake of clarity, but the explained concepts can be extended to three-dimensional domains. Let us indicate that un-trimmed NURBS surfaces need to have typically four edges to keep their simplicity, it might be five in some circumstances. For un-trimmed NURBS solids six faces are required.

Fig. 1.11 illustrates the efficiency of trimming. A plate with a hole and notch is generated by five non-trimmed patches (left) and by a single trimmed patch. In the trimmed option the number of control points is lower, which involves low computational cost. Observe that the trimming length is relatively small. Trimming involves iterative processes that penalize the computational cost, therefore abusing of trimming is not recommended.

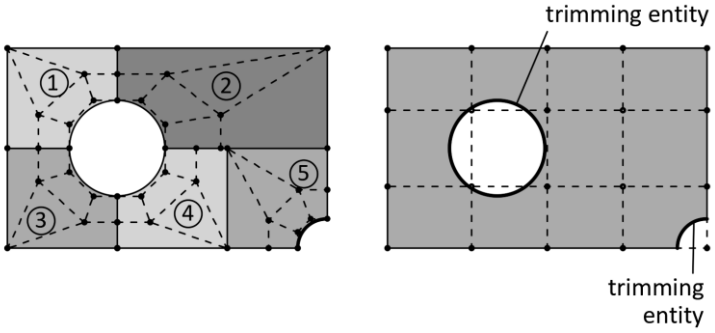


Fig. 1.11 Plate with hole and notch. Left: multi-patching. Right: single patch and trimming entities.

Fig. 1.12 illustrates the efficiency of coupling. The domain may be defined in two manners: coupling two un-trimmed patches (left) or trimming the whole contour of a single patch. In the second case the trimming contour would be too large which increases the computational cost. Hence the first option is more efficient.

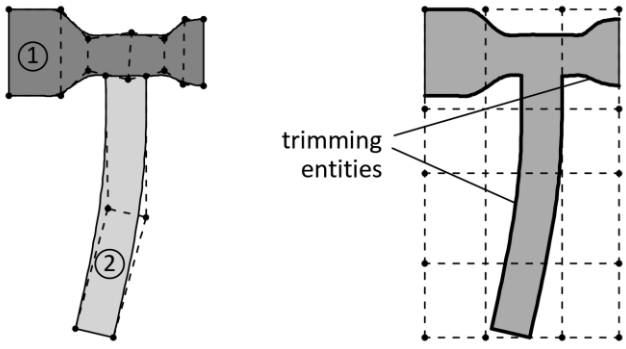


Fig. 1.12 Hammer. Left: coupling of two domains. Right: single trimmed domain.

1.2. Research aim and objectives

Given the difficulties on the IGA applied to solids exposed in the previous sections, next question arises: is it possible to apply IGA on trimmed-coupled solids generated in CAD?

The aim of this thesis is *to implement a procedure to read IGES files from trimmed-coupled volumes in CAD and generate from them a solid domain for elastic-linear IGA, without user intervention in the transformation from IGES files to analysis suitable data*.

The implementation of this procedure involves seven objectives:

a) Codification of an algorithm that encompasses all the processes: from IGES files interpretation to results visualization.

b) Creation of a *translator* from IGES files to numerical data readable by the algorithm.

c) Parametrization of the volumes that are enclosed by the surfaces given by the IGES files. The resultant parametrized volumes are analysis suitable solids.

d) Discretization of the solids with adaptability to capture any trimmed shape.

e) Application of constraints to surfaces non-conformal with the solid domain.

f) Improvement of point projection technique to increase its robustness.

g) Representation of the surfaces efficiently and independently of their parametrization.

1.3. Contribution of this thesis

We create an IGES translator, which reads the IGES files, transfer the information to numerical arrays readable by the code and generate a solid from the enveloping surfaces that wrap the volume. This process is presented in Chapter 4.

The solid parametrization is done by treating the NURBS control net as a 3D lattice. This lattice is deformed until it achieves the target shape. The quality of the parametrization is measured by Jacobian of the lattice cells instead the solid itself, which reduces the computational cost. The method is constrained to a solid with six faces volume and genus-zero topology. Wang and Qian (2014) used similar approach but their structure is not a lattice of control points but a solid composed of hexahedrons which is computationally more expensive. The solid parametrization is explained in section 4.2.

To carry out the discretization of the solids, we discretize the whole domain into linear and mixed-degree tetrahedrons, with the purpose of gaining accuracy at the trimmed surfaces and do not add additional nodes or control points that would raise the computational cost.

Mixed-degree tetrahedrons have some edges quadratic and other linear. Mixed-degree is used in those tetrahedrons that lie at the trimming surfaces. The approximation accuracy is based on Taylor´s series. The solid is not reparametrized as per works from Xia and Qian (2017). Instead the parametrization remains as it comes from CAD.

Discretization of trimmed solids into tetrahedrons involves constrained tetrahedralization (Lee and Lin, 1986). The imposition of facets to the tetrahedral mesh is no trivial and involves new nodes insertion (Shewchuk, 2009). Tetrahedralization of trimmed solids is explained in chapters 5 and 6.

The coupling of trimmed solids, as well as the Dirichlet boundary conditions, is applied by the Lagrange multiplier approach proposed by Apostolatos *et al.* (2014) but extended to solid domains. Constraints to trimmed solids are presented in Chapter 7.

The work by Zuo *et al.* (2015) is very similar to this thesis, but there are some differences that we remark in **Table 1.1**.

Table 1.1 Differences of this work with respect to Zuo *et al.* (2015)

Question	Zuo *et al.* (2015)	**This thesis**
Does the domain admit any shape?	No, volumes are created by extrusion, revolving or sweeping. User post-edition of surfaces is not permitted.	Yes, including customized surfaces
How is the solid parametrized?	By sweeping of one surface.	By lattice deformation or by linear sweeping (*sandwich* algorithm)
How is the solid integrated?	By octree decomposition	By mixed-degree tetrahedrons.

The point projection technique is used repeatedly in this thesis. If traditional methods (based on Newton-Raphson interations) are used in point projection, non-convergence might appear in many cases. To gain robustness we propose a version of the marching method (Hoschek *et al.*, 1993), which is extended to solids. This approach improves the robustness compared to the Newton-Raphson iterations. The method is presented in Appendix 10A.

To represent the solids we use triangulation of their surfaces, then surface triangulation is used in this work. Most of the approaches of surface triangulation start from preliminary mesh that is refined to increase its isotropy. This thesis includes a new algorithm for computing quasi-isotropic triangulation[3] on a given set of NURBS surfaces at once, with no preliminary mesh. It provides high isotropy regardless of the surface shape or parametrization. There is one front that advances in a divergence manner such that front collisions do not happen. Quasi-isotropic triangulation algorithm is outlined in Appendix 10B, that is based on the work done by Adan and Cardoso (2020), which was developed during this thesis.

1.4. Thesis organization

This thesis is organized in nine chapters plus a number of appendixes.

Chapter 2 outlines the theoretical background, sets the state of the art and highlights the contributions of this thesis.

[3] In a quasi-isotropic triangulation only the triangles in the vicinity of the edges are not equilateral, the rest (inner triangles) are equilateral and with the same size.

Chapter 3 provides an overview of the algorithm and discloses information for the rest of the thesis facilitating the comprehension of the next chapters. This chapter is related to objective ***a*** since it briefs the whole code.

Chapter 4 describes the translation from IGES files to numerical data readable by the code and the volume parametrization, thus covering objectives ***b*** and ***c***.

Chapters 5 and 6 provide details of how the trimmed solids are discretized. In Chapter 5 the treatment to trimming surfaces is explained. The approximation to those surfaces is critical for achieving accurate discretization of the solid. In Chapter 6, the solid discretization itself is detailed. Chapters 5 and 6 cover the objective ***d***.

The application of constraints and coupling to surfaces of solids (objective ***e***) is explained in Chapter 7.

To validate the procedure a set of examples is exposed in Chapter 8. These examples are compared to an alternative model solved with FEA in order to check the validity of the results.

Finally, Chapter 9 highlights the main conclusions and future work.

The appendixes contain further details that facilitate the comprehension of this thesis. It is relevant to highlight Appendixes 10A and 10B.

Appendix 10A contains the improved point projection technique used in this research, which is related to the objective ***f***. In Appendix 10B the surfaces representation technique independent of the parametrization, which is related to the objective ***g***, is briefed. The algorithm developed in Appendix 10B was presented in a journal paper that has been published and referenced in that appendix. These two objectives are treated in appendixes, and not in the thesis' main body, because they are *transversal*, i.e. they support the algorithms provided in the other chapters.

The code is fully implemented in Matlab® from scratch, from the routines that read the IGES files to the representation of the results. The used CAD package is AutoCAD®.

All the routines, except for those ones embedded in Matlab®, are developed by the author and are attached to the thesis submission on line. The coding approach is object-oriented programming (OOP), which eases the variables and code organization. For more details on OOP refer to specialized bibliography (Budd, 2002). Schemes of the code, its functions and variables are shown in Appendix 3A. The code is in a *work in progress* state, hence some functions are not used in this thesis since they are devised for future applications. In addition, some routines may appear incomplete because they are prepared for future extension.

The purpose of this thesis is to present the fundamentals of the algorithms, but not the code itself. However, in chapters 3 to 7, there is a section that links, briefly, the formulation developed in the chapter to the main functions of the code.

1.5. Nomenclature assumptions

Formulation is given where possible in matrix notation. At some points tensor or indicial notation may be used. When representing matrices, components equal to zero might be omitted for the sake of clarity. Most common symbols used in the text are listed in **Table 1.2**, at the rear of this chapter.

The abbreviation of '*with respect to'* is indicated as w.r.t.

Some routines and variables of the code are mentioned in this thesis. To clearly distinguish them, are written in `courier type`.

Table 1.2 List of symbols

$\boldsymbol{b}$	Body forces vector
c	Number of parameter space dimensions
d	Number of physical space dimensions
$\hat{\boldsymbol{f}}$	Control points forces vector
$\boldsymbol{n}$	Surface normal versor
n, m, l	Number of control points in each parameter direction
p, q, r	Degree of basis functions
$\boldsymbol{q}$	Traction vector
$\bar{\boldsymbol{t}}$	Prescribed traction
$\boldsymbol{u}$	Displacement field
$\boldsymbol{u}^h$	Displacement field approximation
$\hat{\boldsymbol{u}}$	Control points displacement vector
$\overline{\boldsymbol{u}}$	Prescribed displacement
w	Control point weight, Gauss point weight
$\boldsymbol{w}$	Weighting function
$\boldsymbol{w}^h$	Weighting function approximation
$\boldsymbol{x}$	Spatial coordinates
$\boldsymbol{B}$	Strain-displacement matrix
$\boldsymbol{D}$	Elastic constitutive tensor
E	Young modulus
$\mathcal{J}$	Jacobian
$\boldsymbol{K}$	Stiffness matrix
$\aleph$	Aggregated number of control points
N, M, L	B-spline basis function
$\boldsymbol{P}$	Control point coordinates
R	NURBS basis function

Table 1.2 (cont.)

$\boldsymbol{\sigma}$	Stress tensor
∂	Strain operator
$\boldsymbol{\varepsilon}$	Strain tensor
λ	First parameter of Lamé
$\boldsymbol{\lambda}$	Lagrange multiplier field
$\hat{\boldsymbol{\lambda}}$	Control points Lagrange multiplier vector
μ	Shear modulus
ν	Coefficient of Poisson
ξ, η, χ	Parameter space coordinates
Γ	Boundary
Ω	Physical space
$\hat{\Omega}$	Parameter space
$\tilde{\Omega}$	Parent space
Ω'	Index space
Ξ, H, X	Knot vectors

2. **Theoretical background and literature review**

This chapter reviews the fundamentals behind the algorithms developed in this thesis and the related works. The problem to be solved by the algorithm is the equilibrium equation in Continuum Mechanics, which constitutes the Initial Boundary Value Problem (IBVP). The equilibrium equation is a Partial Differential Equation (PDE). First attempts to solve analytically the equilibrium equation appeared in the XVII century (Timoshenko, 1983) when they were applied to a variety of structural types (trusses, beams, plates etc.) until the XX century. These analytical methods provide the exact solution to the problem, but they are restrained to relatively simple shapes. For complex domains the analytical solution is, in general, not available. The Finite Element Analysis (FEA) appeared in the 50's of past century to tackle this problem (Clough, 2001).

The FEA is a Galerkin´s method, as well as the Isogeometric Analysis (IGA). The Galerkin's method discretizes the domain into portions or elements (spatial discretization) and approximates the solution at certain points called control points[4]. The domain is subjected to boundary conditions and body forces. The boundary conditions can be imposed displacements (Dirichlet boundary conditions) or imposed tractions (Neumann boundary conditions).

The spatial discretization allows the boundary conditions to be applied to the control points and the IBVP is transformed into a linear system of equations as follows:

$$\boldsymbol{K}\,\hat{\boldsymbol{u}} = \hat{\boldsymbol{f}} \tag{2.1}$$

where:

$\boldsymbol{K}$ represents the *resistance* of the control points to the displacement (stiffness);

$\hat{\boldsymbol{u}}$ are the displacements of the control points due the forces $\hat{\boldsymbol{f}}$ applied on the control points.

The Galerkin´s method is well settled in the field (Hughes, 2000; Zienkiewicz and Taylor, 2000; Felippa, 2004; Bathe, 2006). The Appendix 2B explains in detail how to obtain the linear system of equations (2.1) from the PDE of equilibrium. In this thesis, the approach to impose the boundary conditions and couplings follows the work from Apostolatos *et al.* (2014).

Fig. 2.1 shows one domain under boundary conditions (a) that is discretized in two portions with the boundary conditions applied on the control points (b). The results are the displacement of the control points (c).

[4] In FEA they are called nodes, but we will follow the nomenclature for IGA.

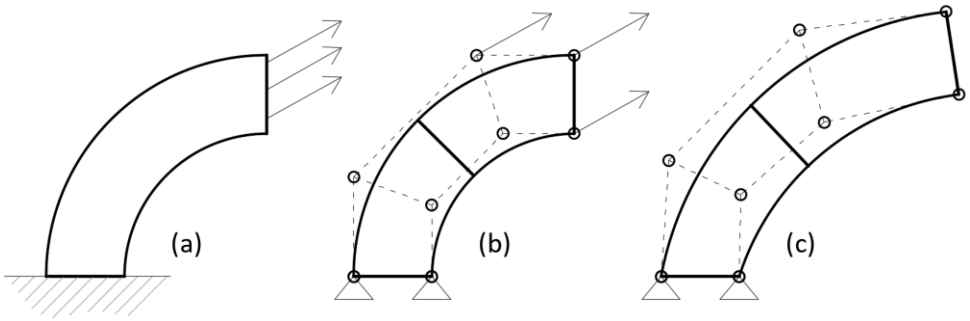


Fig. 2.1 Discretization of the domain and results of the analysis.

Once the displacements of the control points are calculated, the displacement field within each element ($\boldsymbol{u}^{he}$) is estimated by linear combination of the displacements of the involved control points as follows:

$$\boldsymbol{u}^{he} = \boldsymbol{R}^{e}\, \widehat{\boldsymbol{u}}^{e} \tag{2.2}$$

where:

$\boldsymbol{R}^{e}$: is the matrix of the so-called basis functions. Each control point has associated one basis function;

$\widehat{\boldsymbol{u}}^{e}$: is the vector of the displacements at the control points that influence on the element.

The Computational Mechanics deals with the development and implementation of these approximation methods. Then, this work lies in the Computational Mechanics field.

The rest of this chapter covers the next contents. In section 2.1 the Isogeometric Analysis and its application for elasticity are introduced. Sections 2.2 to 2.7 revise the topics directly related to the chapters and annexes of this thesis as listed below:

Section 2.2: Basics of CAD information and its exchange formats.

Section 2.3: Parametrization of solid NURBS.

Section 2.4: Integration of trimmed solids NURBS.

Section 2.5: Constraints to non-conformal domains.

Section 2.6: Point projection techniques.

Section 2.7: Triangulation of NURBS surfaces.

2.1. Isogeometric analysis

Isogeometric analysis (IGA) was introduced by Hughes, Cottrell and Bazilevs (2005) to use CAD domains directly for analysis. IGA has rapidly extended to areas such as structural analysis, vibration, fluid dynamics or fluid structure interaction (see for example Dede, Jaggli and Quarteroni, 2015; Weeger, Wever and Simeon, 2013; Bazilevs, Hsu and Scott, 2012; Kiendl *et al.*, 2010). IGA has been also applied for fracture mechanics, in particular using extended isogeometric analysis (XIGA), as shown by Ghorashi *et al.* (2015); Ghorashi, Valizadeh and Mohammadi (2012) and Singh *et al.* (2018).

Knot insertion (*h*-refinement) and degree elevation (*p*-refinement) are suitable for NURBS (Piegl and Tiller, 1996) in a similar manner as FEA. In addition, NURBS possesses the *k*-refinement technique (Cottrell, Hughes and Bazilevs, 2009), which is a sequence of *p*-refinement and *h*-refinement. This combined refinement raises the degree of NURBS and the number of knots keeping the inter-elemental continuity equal to *p-1* (*p* is the NURBS degree).

Variations and improvements for IGA have appeared in the last decade. This section provides the most relevant publications for IGA, but previously briefs the fundamentals of IGA itself. Since IGA uses Non-uniform rational B-splines, they are firstly introduced.

2.1.1. Non-uniform rational B-splines (NURBS)

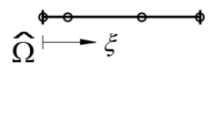
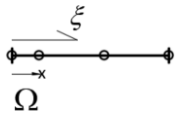
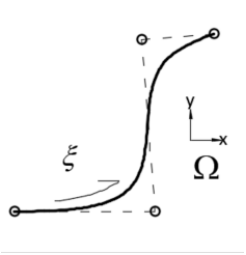
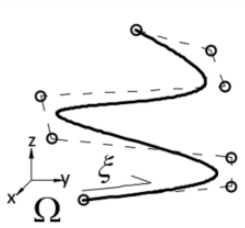
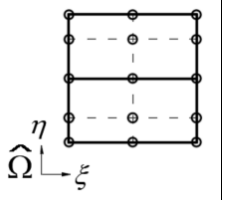
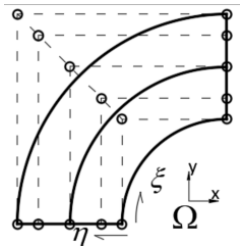
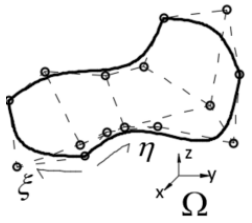
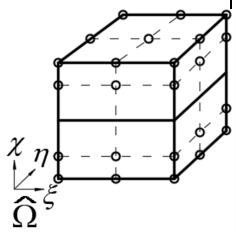
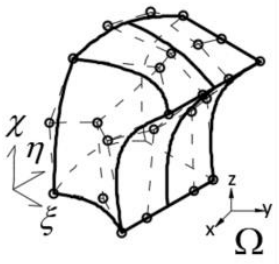
NURBS entities[5] are defined in the parameter ($\hat{\Omega}$) and the physical (Ω) spaces. The parameter space lies in $\mathbb{R}^c$, with c equal to one, two and three for curves, surfaces and solids respectively. The physical space lies in $\mathbb{R}^d$ such that $d \geq c$. **Table 2.1** shows one example for each permitted combination of c and d.

The parameter and physical coordinates are referred as $\boldsymbol{\xi}$ and $\boldsymbol{x}$ respectively. In this thesis, the physical space is three-dimensional ($d = 3$) unless noted otherwise.

The parametrization of one NURBS entity is given by the knot vector in each parameter direction. One knot vector is a set of non-decreasing numbers (knots), i.e. $\boldsymbol{\Xi} = \{\xi_1 \ \xi_2 \ \cdots \ \xi_a \ \cdots \ \xi_{n+p+1}\}$ with $\xi_i \leq \xi_{i+1}$. If the first and last $p+1$ knots are repeated, it is called open knot vector, which is the case of this thesis. In this work the knots vary from 0 to 1. The number of knots in each parameter direction is $n+p+1$, being n the number of control points and p the degree of the basis functions in the corresponding parameter direction.

[5] *Entity* refers to curve, surface or solid.

Table 2.1 Physical spaces representable for each parameter space.

c	parameter representation	Number of physical dimensions and examples		
		$d = 1$	$d = 2$	$d = 3$
1				
2		-		
3		-	-	

One NURBS entity has a set of control points whose coordinates define its shape in the physical space. The lines jointing these points form the control net. Each control point has attached one NURBS basis function whose degree is denoted by p. NURBS basis functions are made of B-spline functions (Piegl and Tiller, 1996).

B-spline functions are computed with the parametric equations (2.3) and (2.4) also known as Cox-De Boor equations (Cox, 1972; De Boor, 1972). **Fig. 2.2** illustrates a set of these functions.

For zero degree (p = 0):

$$ {}^{0}N_i(\xi) = \begin{cases} 1 \text{ if } \xi_i \leq \xi < \xi_{i+1} \\ 0 \quad \text{otherwise} \end{cases} \tag{2.3} $$

For degrees 1 and higher (p > 0) these functions are obtained recursively:

$$ {}^{p}N_i = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} \left({}^{p-1}N_i\right)(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} \left({}^{p-1}N_{i+1}\right)(\xi) \tag{2.4} $$

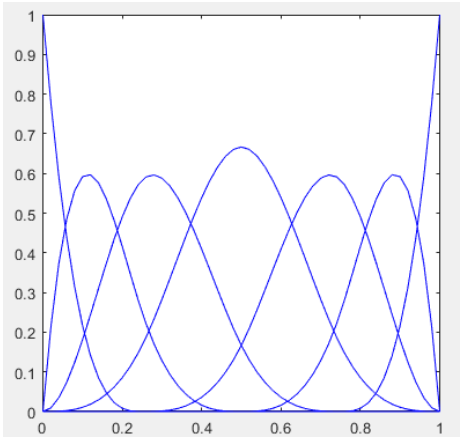


Fig. 2.2 Seven cubic basis functions.

NURBS functions (R) are obtained from B-splines functions by considering an extra dimension for the control points called weight (w). The B-splines functions are weighted obtaining rational B-splines as per equations (2.5), (2.6) and (2.7), that are for curves, surfaces and volumes respectively. **Table 2.2** provides the nomenclature followed in this work for NURBS entities.

$$ {}^{p}R_i(\xi) = \frac{{}^{p}N_i\,(\xi)\; w_i}{\sum_{\hat{\imath}=1}^{n} {}^{p}N_{\hat{\imath}}(\xi)\; w_{\hat{\imath}}} \tag{2.5} $$

$$ {}^{p,q}R_{i,j}(\xi,\eta) = \frac{{}^{p}N_i(\xi)\quad {}^{q}M_j(\eta)\;\; w_{i,j}}{\sum_{\hat{\imath}=1}^{n}\sum_{\hat{\jmath}=1}^{m} {}^{p}N_{\hat{\imath}}(\xi)\quad {}^{q}M_{\hat{\jmath}}(\eta)\;\; w_{\hat{\imath},\hat{\jmath}}} \tag{2.6} $$

$$ {}^{p,q,r}R_{i,j,k}(\xi,\eta,\chi) = \frac{{}^{p}N_i(\xi)\quad {}^{q}M_j(\eta)\quad {}^{r}L_k(\chi)\;\; w_{i,j,k}}{\sum_{\hat{\imath}=1}^{n}\sum_{\hat{\jmath}=1}^{m}\sum_{\hat{k}=1}^{l} {}^{p}N_{\hat{\imath}}(\xi)\quad {}^{q}M_{\hat{\jmath}}(\eta)\quad {}^{r}L_{\hat{k}}(\chi)\;\; w_{\hat{\imath},\hat{\jmath},\hat{k}}} \tag{2.7} $$

Table 2.2 Nomenclature for NURBS entities

Entity	**volume**		
	surface		-
	curve	-	-
Parameter direction	**1**	**2**	**3**
Coordinate	ξ	η	χ
B-spline basis function	N	M	L
NURBS basis function	R		
Number of control points	n	m	l
Degree	p	q	r
Knot vector	$\boldsymbol{\Xi}$	H	X

NURBS entities are obtained by linear combination of NURBS functions, using the coordinates of control points as coefficients. Equations (2.8) to (2.10) provide the expressions for NURBS curves, surfaces and volumes.

$$\boldsymbol{C}(\xi) = \sum_{i=1}^{n} R_i(\xi)\boldsymbol{P}_i \tag{2.8}$$

$$\boldsymbol{S}(\xi,\eta) = \sum_{i=1}^{n}\sum_{j=1}^{m} R_{i,j}(\xi,\eta)\boldsymbol{P}_{i,j} \tag{2.9}$$

$$\boldsymbol{V}(\xi,\eta,\chi) = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} R_{i,j,k}(\xi,\eta,\chi)\boldsymbol{P}_{i,j,k} \tag{2.10}$$

where:

$\boldsymbol{P}_i$ are the ith control point coordinates for curves;

$\boldsymbol{P}_{i,j}$ are the i-jth control point coordinates for surfaces;

$\boldsymbol{P}_{i,j,k}$ are the i-j-kth control point coordinates for volumes.

These three equations might be expressed for any general entity $\boldsymbol{\Omega}$ as (2.11)[6], where the aggregated index I may refer to (i), (i,j) or (i,j,k) for curves, surfaces and volumes respectively, and $\aleph$ is the total number of control points. Equation (2.11) represents a mapping from the parameter to the physical space.

$$\boldsymbol{\Omega} = \sum_{I=1}^{\aleph} R_I \boldsymbol{P}_I \tag{2.11}$$

NURBS surfaces and solids are formed by the tensor product of two and three NURBS curves respectively. Therefore, their parameter domains are calculated as per equations (2.12) and (2.13). **Fig. 2.3** illustrates one NURBS surface which spans the parameter space $\{0\ 1\} \otimes \{0\ 1\}$. Each non-void knot span can be seen as one element (in **Fig. 2.3** one element is grey hatched).

$$\boldsymbol{S}_{\widehat{\boldsymbol{\Omega}}} = \boldsymbol{\Xi} \otimes \boldsymbol{H} \tag{2.12}$$

$$\boldsymbol{V}_{\widehat{\boldsymbol{\Omega}}} = \boldsymbol{\Xi} \otimes \boldsymbol{H} \otimes \boldsymbol{X} \tag{2.13}$$

[6] The dependency on $\boldsymbol{\xi}$ is removed for clarity.

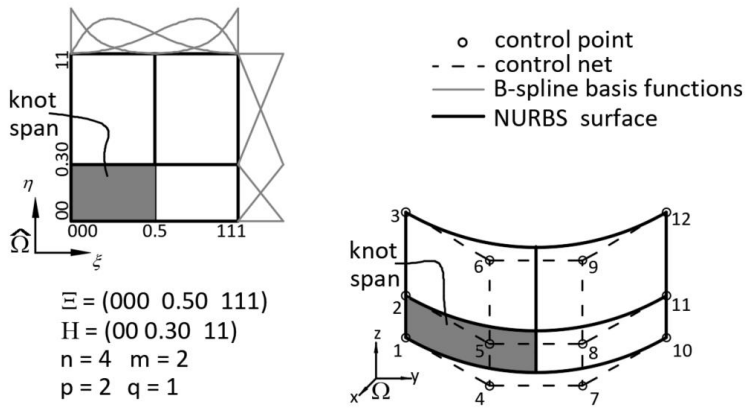


Fig. 2.3 NURBS surface in parameter and physical spaces.

NURBS entities are the evolution of Bézier entities. Bézier curves (Bézier, 1970; Forrest, 1972) were used by mid-20th century for engineering representation. Bézier curves are parametric curves formed by linear combination of basis functions that are Bernstein polynomials (Bernšteın, 1912). **Fig. 2.4** shows one example of bi-dimensional curve.

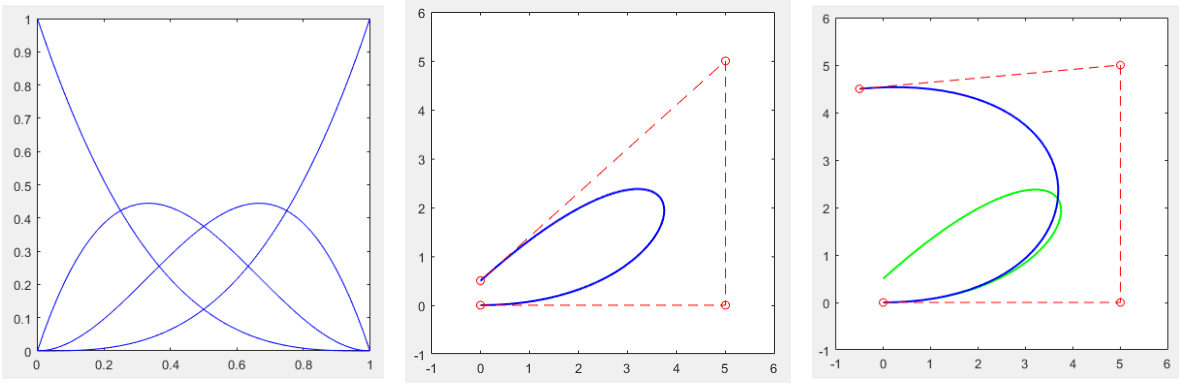


Fig. 2.4 Bi-dimensional Bézier curve of cubic degree. Left: Bernstein polynomials. Centre: curve (blue thick) and control points (red circles). Right: modification of curve (green is the previous curve).

Bézier entities lack localized shape control: a single control point modification varies the whole entity, i.e. it is not possible to carry out a localized shape variation (**Fig. 2.4** right). B-splines overcome this pitfall since they are a sequence of Bernstein polynomials with *small support*, i.e. each basis function does not span along the whole domain but onto a localized portion. Hence, local shape modification is allowed (see **Fig. 2.5**).

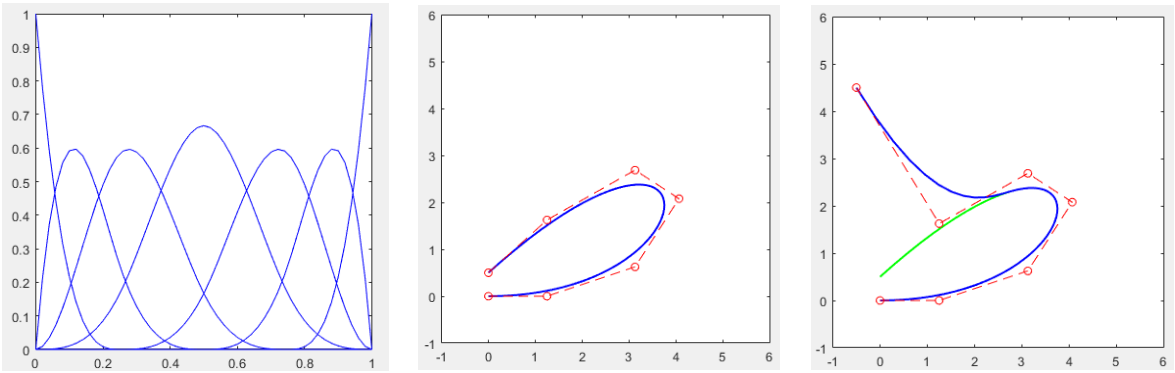


Fig. 2.5 Bi-dimensional B-spline curve of cubic degree and seven control points. Left: basis functions, whose support, in general, is not the whole span. Centre: curve (blue thick) and control points (red circles). Right: local modification of curve (green is the original curve).

2.1.2. Aspects of implementation of IGA

Spaces in IGA

Four spaces are defined in IGA: physical (Ω), parameter ($\widehat{\Omega}$), index (Ω^{I}) and parent space ($\widetilde{\Omega}$). The physical space contains the actual geometry mapped from the parameter space. The parameter space contains the arrangement of the knot vectors where only non-zero knot spans are depicted. In the index space all the knot spans are depicted (including zero knot spans). The spacing between knots in the parameter space is proportional to its span length, but in the index space all the knots are equally spaced. Each knot span has its own parent space which will be used for integration by Gauss rule. **Fig. 2.6** illustrates these four spaces for a typical NURBS surface. The fourth element (knot span) is hatched in grey colour.

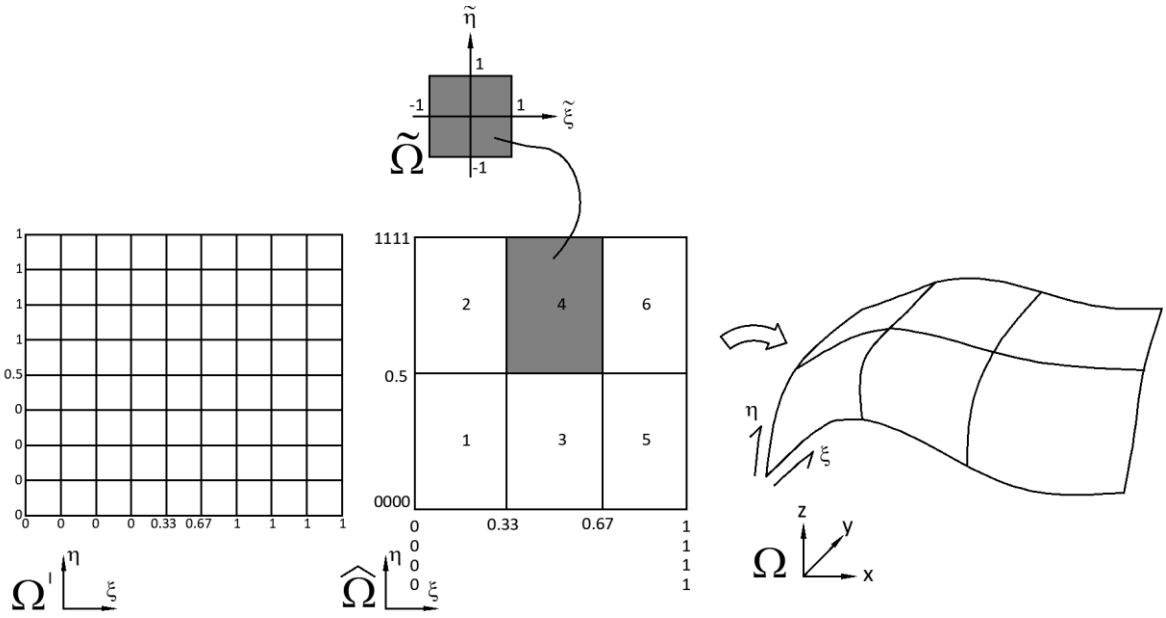


Fig. 2.6 Spaces in IGA.

The index space is relevant for local insertion of control points in the so called T-splines (Bazilevs *et al.*, 2010). Since this thesis works with B-splines, and not T-splines, the index space is not used.

Spatial discretization using NURBS

IGA is a Galerkin's method that uses NURBS as basis functions. The PDE solution is approximated by spatial discretization of the domain with the displacement field estimated with the equation (2.2).

IGA is isoparametric: same basis functions are used for both, geometry definition and approximation to the solution of the PDE. That is patent by comparing equation (2.2) against equation (2.11), where both the domain Ω and the solution approximation $\boldsymbol{u}^{he}$ are defined as linear combination of the basis functions R_i. The key of IGA is that NURBS are used in CAD for defining the geometry. Therefore, IGA potentially allows to use domains drawn in CAD directly for analysis without meshing, since the division into elements (mesh), is already generated when the CAD model is drawn.

Originally in IGA, the parent spaces were square and hexahedrons[7], which follows the arrangement of knot spans in NURBS surfaces and solids (Hughes, Cottrell and Bazilevs, 2005). However, for trimmed domains those parent spaces might not be acceptable as shown in section 2.4.

Numerical integration in IGA

The components of the system of equations (2.1) are obtained by integration of the domain and the boundary entities[8]. These integrals are approximated by the Gauss-Legendre quadrature[9]. Let us call φ to the integrand, then the integral $\mathbb{I}_\varphi$ is approximated as:

$$\mathbb{I}_\varphi \approx \sum_{g=1}^{N_g} \varphi_g \mathcal{J}_g w_g \tag{2.14}$$

where:

sub-index g indicates that the value is computed at the Gauss point location;

N_g is the number of Gauss points involved;

φ_g is the function evaluated at the gth Gauss point;

w_g is the Gauss point weight.

[7] Appendix 2E reviews the most common parent elements.
[8] See Appendix 2B.
[9] *Gauss-Legendre quadrature* will be referred as *Gauss rule* for brevity.

The Jacobian ($\mathcal{J}_g$) is required because the domain is mapped onto the parent domain to set the same integration limits for all the integrals to facilitate the implementation. The locations of the Gauss points and its weights depend on the type of parent element and on the number of Gauss points per element[10].

In IGA there are two mappings involved: from the parent to the parameter space, and from the parameter to the physical space, which introduce the Jacobians $\boldsymbol{\mathcal{J}}_2$ and $\boldsymbol{\mathcal{J}}_1$ respectively (see **Fig. 2.7**). The form of both Jacobians depends on the parent element and on the type of NURBS entities.

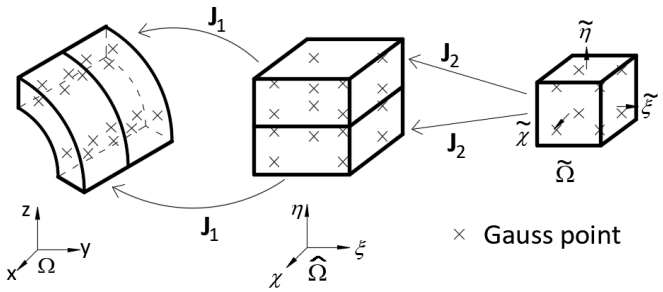


Fig. 2.7 Jacobians involved in IGA.

With these two Jacobians at hand, the Gauss quadrature in IGA takes the form:

$$\mathbb{I}_{\varphi} \approx \sum_{g=1}^{N_g} \varphi_g \, \mathcal{J}_{1g} \, \mathcal{J}_{2g} \, w_g \tag{2.15}$$

where:

$\mathcal{J}_{1g}\, \mathcal{J}_{2g}$ are the Jacobians evaluated at the Gauss point location.

Strain-displacement matrix

The components of the strain-displacement matrix ($\boldsymbol{B}$) are the derivatives of basis function w.r.t. the physical space which are calculated as follows:

$$\begin{Bmatrix} R_{i,x} \\ R_{i,y} \\ R_{i,z} \end{Bmatrix} = \boldsymbol{\mathcal{J}}_1^{-1} \begin{Bmatrix} R_{i,\xi} \\ R_{i,\eta} \\ R_{i,\chi} \end{Bmatrix} \tag{2.16}$$

The Jacobian $\boldsymbol{\mathcal{J}}_1$ is required because basis functions (R_i) depend on the parameter coordinates[11].

[10] Jacobian calculation is given in Appendix 2F and Gauss points arrangements in Appendix 6A.
[11] See Appendix 2B for details of the strain-displacement matrix.

2.1.3. State of the art of IGA

IGA for different types of domains

For structures of beam type, IGA is particularly suitable for curve beams because it captures the exact geometry. See for example works done by Nagy, Abdalla and Gürdal (2010); Luu, Kim and Lee (2015); and Cazzani, Malagù and Turco (2016).

Shell structures have been extremely developed on IGA since NURBS represent exact geometry and the parametrization itself provides curvilinear coordinates. However, special care must be considered since the degrees of freedom (control points) do not lie on the physical shell. The single patch NURBS domains fit the Kirchhoff-Love C^1 continuity requirement between knots spans, as the basis functions are C^{p-m} continuous across those boundaries. IGA applied to Kirchhoff-Love shells was introduced by Kiendl *et al.* (2009), and extended to multi-patching with C^1 continuity by the strip method by Kiendl *et al.* (2010). Reissner-Mindlin shells have been also developed in IGA by Uhm and Youn (2009); and Benson *et al.* (2010), and both types mixed by Benson *et al.* (2013). Solid shells with IGA were presented by Bouclier, Elguedj and Combescure (2013); Hosseini *et al.* (2013); and Bouclier, Elguedj and Combescure (2015). Laminated materials are suitable for IGA as shown in Thai *et al.* (2012); Yin *et al.* (2015); and Leonetti *et al.* (2018)

Literature for IGA applied to solid type structures is scarcer. Yusuf *et al.* (2015); and Lai *et al.* (2017) applied IGA to solids achieving satisfactory results, but still some related processes such as parametrization or integration of trimmed solids are not fully addressed. Further exploration of these items is provided in sections 2.3 and 2.4 of this thesis.

In the last 8-10 years two techniques has been developed to improve some of the IGA limitations: Bézier extraction and local refinement. These two techniques are not used in this thesis. However, most relevant publications on them are briefed because they are two major aspects in the current IGA development.

Bézier extraction

Since FEA has been implemented in the industry for a few decades, it would be interesting to insert IGA within FEA routines, then all the infrastructure already existing can be reused for IGA. Bézier extraction operator maps a piece-wise base made of Bernstein polynomials into B-spline basis functions. Borden *et al.* (2011) proposed Bezier extraction to NURBS. Scott *et al.* (2011) extended the technique to T-splines. Bernstein polynomials can be treated exactly the same as conventional polynomials, and therefore they can be implemented in FEA routines. Bézier extraction has been applied also for local refinement procedures (de Borst and Chen, 2018).

Local refinement

Knot insertion in NURBS suffers an important drawback for 2D and 3D domains: any control point insertion cannot be carried out solely, but a full row of control points needs to be inserted due to the B-splines tensor product nature. Some of the new inserted control points may result superfluous raising the computational cost without adding more accuracy. To circumvent this limitation variations of B-splines have been introduced.

Sederberg *et al.* (2004) firstly introduced T-splines for representation purposes, afterwards Bazilevs *et al.* (2010) used them for analysis. Not any arrangement is valid in T-splines: the resulting blending functions need to form linearly independent basis, becoming *analysis suitable T-mesh* (ASTM) (X. Li *et al.*, 2012). Proposals to find ASTM in two and three-dimensional domains are given by Scott *et al.* (2012); and Morgenstern (2016) respectively. T-splines have been used to simulate damage mechanics (Verhoosel, Scott, Hughes *et al.*, 2011) and cohesive fracture problems (Verhoosel, Scott, Borst *et al.*, 2011). Bezier extraction has been used to obtain ASTM (May, Vignollet and De Borst, 2015).

Other types of splines have been introduced to allow local refinement. Locally refined B-splines were introduced by Dokken (2010); and Johannessen, Kvamsdal and Dokken (2014), hierarchical T-splines by Evans *et al.* (2015) and hierarchical B-splines by Vuong *et al.* (2011).

2.2. CAD entities and the IGES files

The graphical information contained in one CAD drawing can be extracted in two types of files: IGES and STEP. The former, which is an acronym of Initial Graphic Exchange Specification, was born in 1979 from the necessity of common language between different CAD packages (Goldstein, Kemmerer and Parks, 1998). The first handbook was published in 1980 (Nagel *et al.*, 1980) and the last version by Kennicott (1995). IGES file stores the information in a text format that is detailed in Appendix 2G.

STEP files, that are more recent, overcome some pitfalls of the IGES such as ambiguity, loss of information during exchange, or potential incompatibility with earlier IGES version (Marussig and Hughes, 2018). Since most of the efforts have focused on STEP files (Pratt, Anderson and Ranger, 2005; Kim, Seo and Youn, 2009; Pratt and Kim, 2006; Skytt and Haenisch, 2013) we focus on IGES. One can read the content of a CAD drawing from IGES and use that information to construct an analysis suitable domain for IGA as demonstrated in this thesis.

This section outlines the IGES files content for **solids** and **bounded surfaces**. For further information refer to work from Kennicott (1995). Here *object* refers to any IGES graphical object,

e.g. vertexes, surfaces, curves. The term *NURBS features* encompasses the number of control points, degree, knot vectors and control points coordinates.

IGES information is stored in a hierarchical tree scheme, with *larger* objects containing the references for *smaller* ones. The rough idea of this arrangement, for solids, is that one solid is a set of shells that contain several faces, each face contains several curves and each curve contains two end-vertexes.

Curves are used to contour the surfaces leaving the visible (or computable) portion at the left-hand side. They might trim the surface or not. This type of representation is called B-rep. These surfaces have a NURBS surface in the background and one or more contours loops of curves to define the visible portion (see section 2.2.3). For further information on B-rep surfaces refer to the work by Stroud (2006).

For clear identification of the objects, they are written in ***bold-italic*** in this section. **Table 2.3** provides the name and type number of each object in the IGES system.

Table 2.3 Type number corresponding to IGES objects.

Type number in IGES system	**Graphic object**
186	***manifold solid B-rep***
514	***shell***
510	***face***
128	***rational B-spline surface*** (***surface***)
508	***loop***
504	***edges***
126	***curve***
502	***vertex***
143	***bounded surface***
141	***boundary***

2.2.1. Content of IGES files for solids

This section describes the arrangement of the objects that form one solid[12]. **Fig. 2.8** shows one example with two solids. **Fig. 2.9** depicts the corresponding tree scheme. This example will be used in the forthcoming explanation for clarity.

[12] As we will see, the solid itself does not exist in the IGES but only its *enveloping surfaces*.

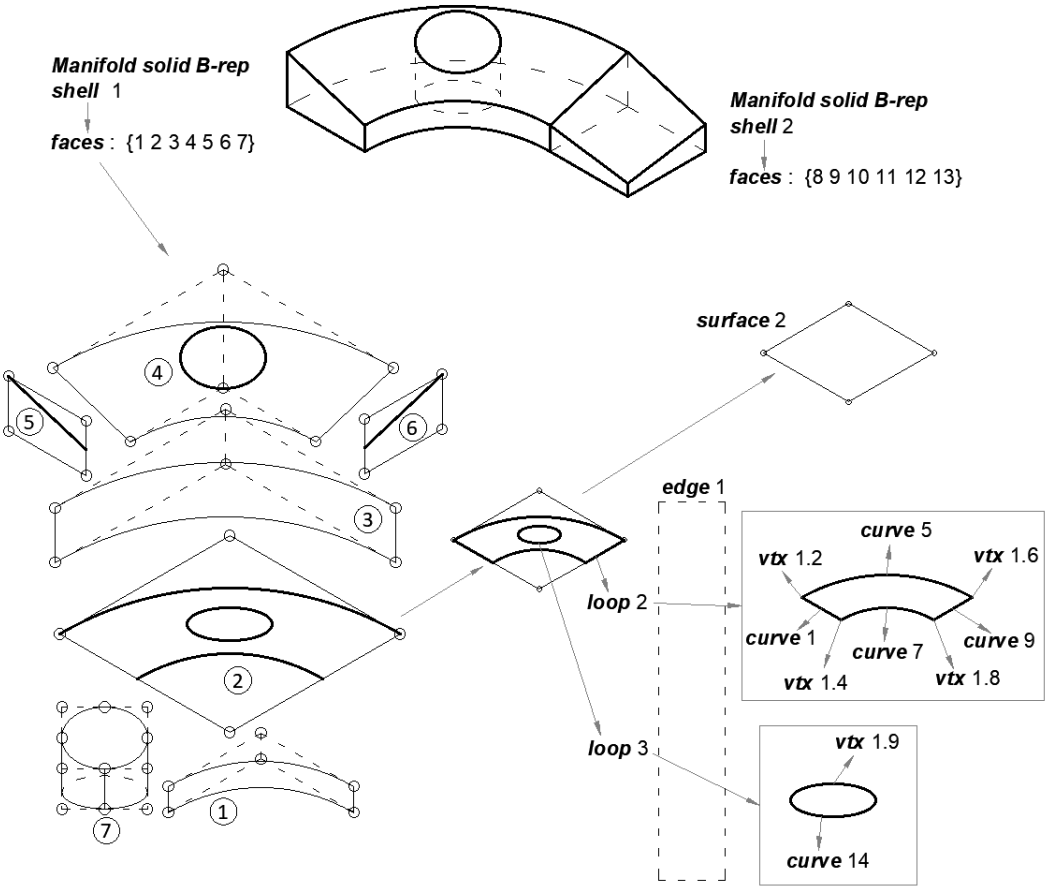


Fig. 2.8 CAD drawing composed of two solids.

Let us suppose a CAD drawing with two solids[13]. Each solid is stored in one ***Manifold solid B-rep***, called here *patch* for brevity. Each patch points to one ***shell*** and each shell points to a set of ***faces*** that form the boundary that *wraps* of the patch volume. **Fig. 2.8** shows two patches, with seven and six faces each. Only faces of patch 1 are shown separately.

Each ***face*** points to one single ***rational B-spline surface*** (or ***surface***) and to one or more ***loops***. See face 2 in **Fig. 2.8** which has attached surface 2 and loops 2 and 3. Each ***surface*** stores its NURBS features and a flag indicating if it is a closed or not. For example, in **Fig 2.8** face 7 is the only closed surface.

[13] We use two solids but the explanation can be extended to any number of them.

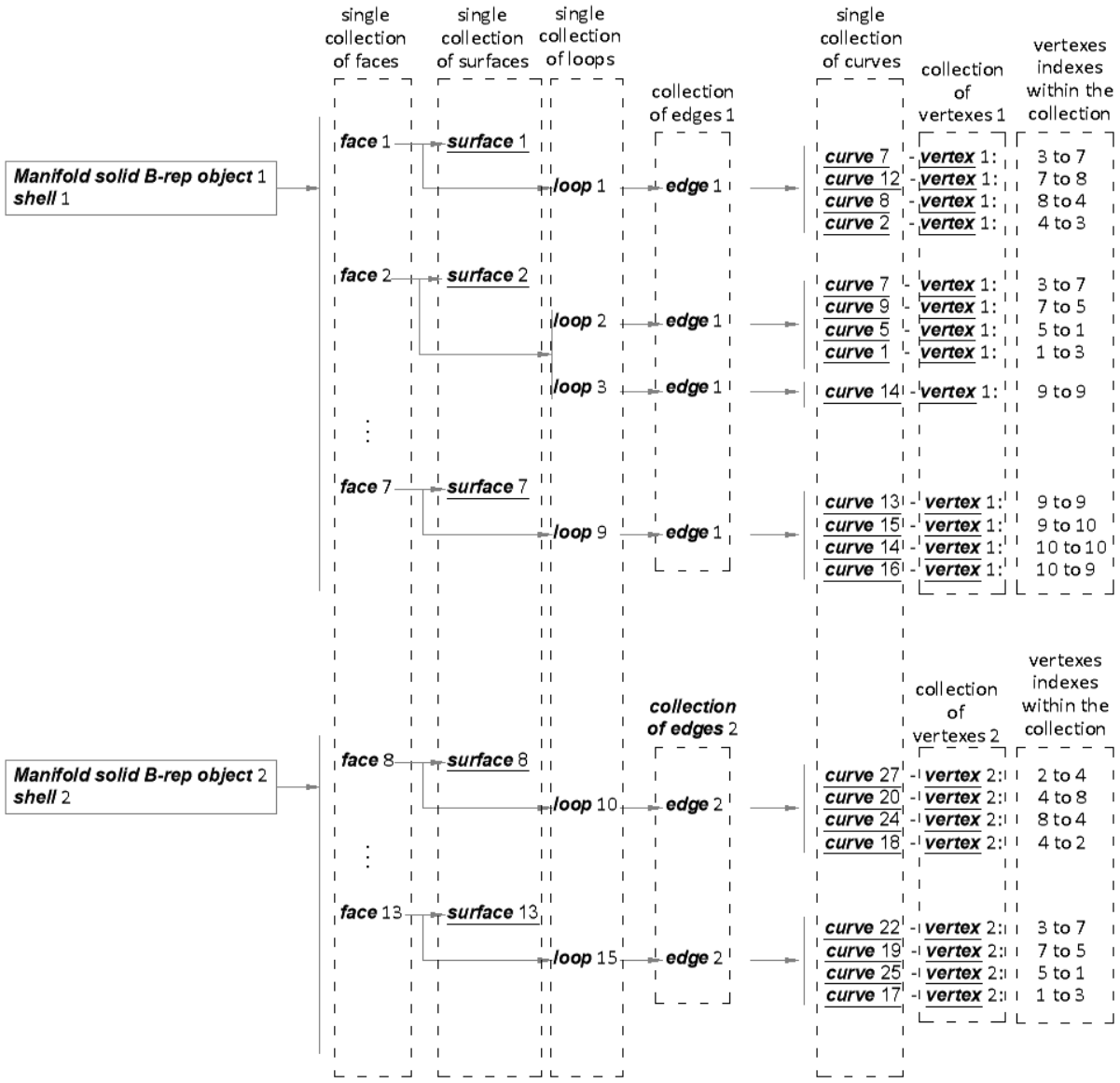


Fig. 2.9 Example of tree scheme. The objects underlined contain information used in this thesis.

One ***loop*** is a closed contour of ***curves*** that lie onto the ***surface***. The ***loop*** points to ***curves*** and ***vertexes*** through the ***edge*** objects. There is one ***edge*** per patch. Each ***edge*** contains the number of ***curves*** and points to them and to the start and terminating ***vertex*** of each curve.

Each ***curve*** hosts its NURBS features and a flag indicating if it is a closed or not. ***Vertexes*** are stored in different collections, one collection per patch, and they contain their coordinates. ***Vertexes*** are attached to the initial and final points of each ***curve***, by a reference that is contained in the ***edge***.

Fig. 2.8 shows loops 2 and 3, that are attached to face 2. Each of them points, via edge 1, to curves and vertexes forming a closed contour. Edge 1 also links each curve to its start and terminating vertexes, for example curve 5 has vertexes 6 and 2 allocated. Both vertexes belong to vertexes collection 1.

Fig. 2.9 illustrates the relationship between different objects in a tree-fashion. ***Faces***, ***surfaces***, ***loops*** and ***curves*** are stored each one in a unique group independently from the patches. They are linked to each other and to patches using pointers. By contrast ***edges*** and ***vertexes*** are grouped by patches: there is one ***edge*** group and one ***vertex*** group per patch. Within each group, each item (***edges*** or ***vertex***) is identified by one index.

The type of relations between these objects is listed below (see **Fig. 2.9** for clarity):

- 1 to 1 for ***manifold solid B-rep–shell*** and ***face–surface***.
- 1 to n for ***shell–faces***, where n is the number of faces.
- 1 to n for ***face–loops***, where n is the number of loops.
- 1 to n for ***loop–curves***, where n is the number of curves.
- 1 to 2 for ***curve–vertexes***.

The arrangement is such that one object cannot be pointed from two different objects (except vertexes as explained below). This arrangement leads to duplicated curves at intersection of surfaces, one curve per face but with opposite senses. In **Fig. 2.10** the intersection curve between faces 2 and 3 is not unique, instead curve 5 lies on face 2, and curve 21 on face 3, both with opposite senses. The ***vertexes*** may be shared by several ***curves***. In **Fig. 2.10** the sixth vertex of collection 1 is pointed by curves 5, 9, 21 and 22.

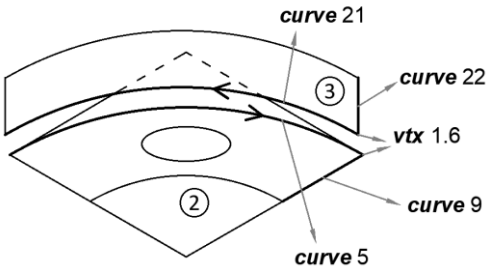


Fig. 2.10 Curves and vertexes.

The underlined items in **Fig. 2.9** contain the information used in this thesis, which is:

- NURBS features from ***rational B-splines surfaces***;
- NURBS features from ***curves***;
- physical coordinates of ***vertexes.***

2.2.2. Content of IGES files for bounded surfaces

This section describes how objects are related in one ***bounded surface***. These surfaces are not part of one solid, instead they are either drawn from scratch (using CAD available techniques as line extrusion or sweep) or extracted and separated from one existing solid ***face***.

Fig. 2.11 shows one example. One ***bounded surface*** points to a set of ***boundaries*** and one ***rational B-spline surface***. One ***boundary*** stores the number of ***curves*** and their references that

form the contour to define the visible surface. Each ***curve*** contains its NURBS features. One ***rational B-spline surface*** contains the surface NURBS features. This thesis uses the NURBS information from both of them: ***curves*** and ***rational B-spline surfaces***.

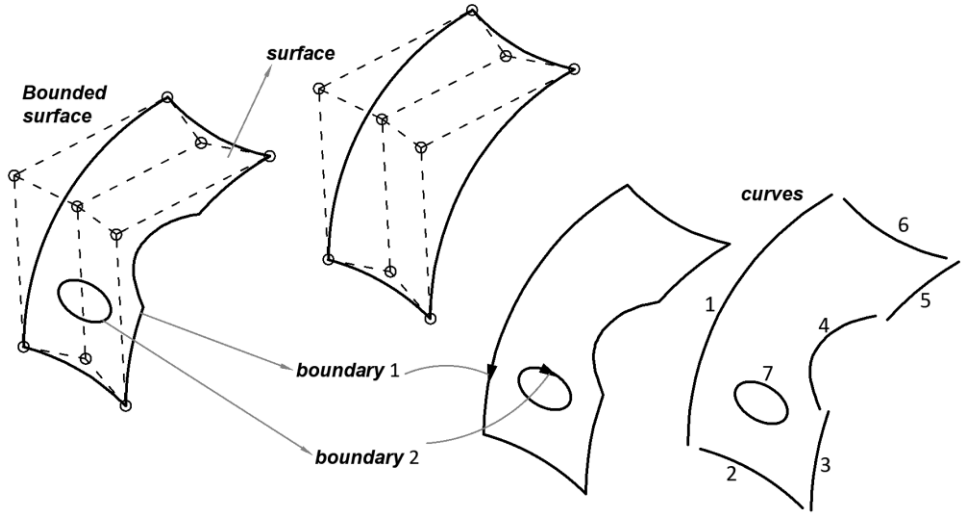


Fig. 2.11 Example bounded surface with two contour loops.

2.2.3. B-rep arrangement and trimmed surfaces in CAD

The *background surfaces* of a ***face*** and a ***bounded surface*** are contoured by a set of curves, called ***loops*** and ***boundaries*** respectively (see section 2.2.1 and 2.2.2). There may be one or several contours, being the first one always the outer contour.

The contours might trim the background surface. If it is not the case, then the contour is unique and coincident with the background surface edge limits, being an *edging* contour. If there is a single contour it might be edging or trimming (see **Fig. 2.12** *a* and *b*). If there are several contours, only the first listed on the IGES files can be *edging*, the posterior contours are *trimming* (see **Fig. 2.12** *c* and *d*). This arrangement of trimmed surfaces by contours is called B-rep.

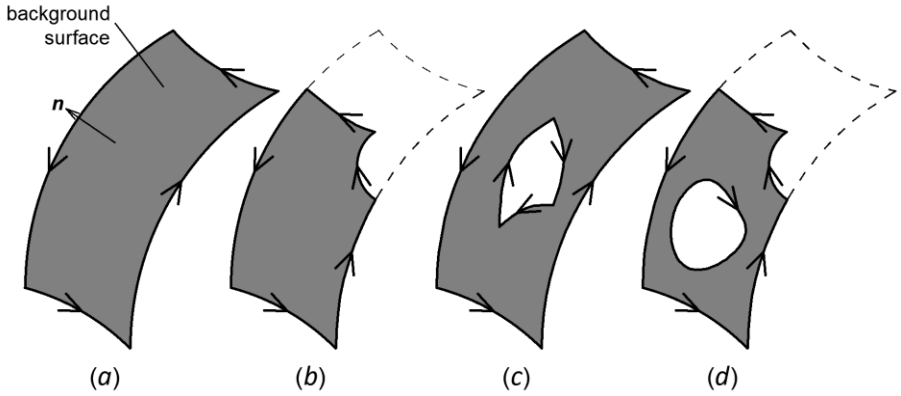
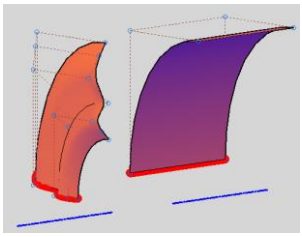
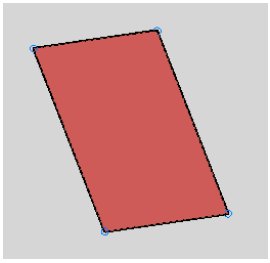
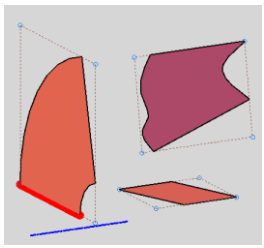
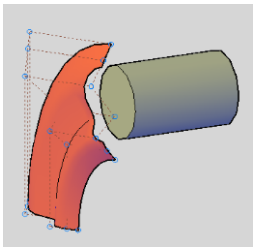


Fig. 2.12 Four cases of contours: (*a*) single edging contour, (*b*) single trimming contour, (*c*) two contours being the outer edging, (*d*) two contours being the outer trimming. Visible surface is hatched in grey.

Table 2.4 describes when one background surface is trimmed or not depending on its construction. The criterion shown applies to ***bounded surfaces*** and to ***faces*** of solids. **Fig. 2.13**, illustrates one example of a solid with four ***faces*** extracted, which are trimmed or not according to **Table 2.4**.

Table 2.4 Cases of trimmed and non-trimmed surfaces.

Surface	Construction	Examples*
Non-trimmed	Non-planar surface generated by extrusion, revolution or sweep of a line (these are typical tools of CAD packages)	
	Planar surface rectangular	
Trimmed	Planar surface not rectangular	
	Any surface intersected and trimmed by other object	

* Thick blue line is the revolving axis and thick red the generatrix. Background control net is represented.

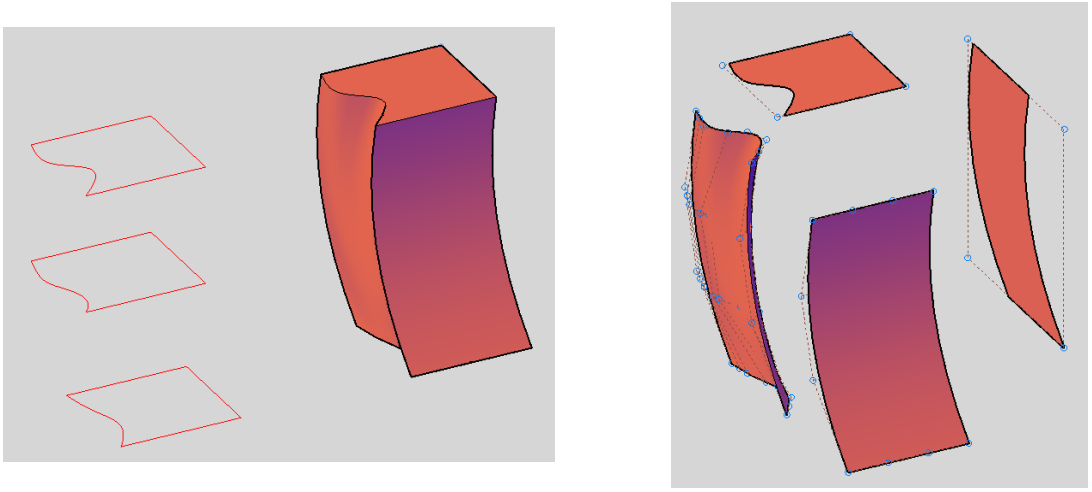


Fig. 2.13 Left: generation of solid by loft of three contours. Right: view of four extracted faces with their control net.

2.3. Solid parametrization

Achieving analysis-suitable solids for IGA is challenging and is still under development. Volumes drawn in CAD are represented by a collection of boundary surfaces instead a tri-variate parametrized solid, therefore one needs to generate that parametrization suitable for IGA. One parametrization is valid when the mapping is not folding, i.e. there is not self-intersection (injective parametrization). That involves the determinant of the Jacobian to have the same sign at any location of the domain or, in other words, the Jacobian must not vanish. Many efforts have been done in the last decade on this topic.

Cohen *et al.* (2010) provided guidelines for the CAD modeller to achieve domains, either surface or solid, that enhance the quality of the posterior parametrization for IGA. In particular for solids, they proposed the generation of them by sweep to obtain a *cylinder-like* objects.

Some of the existing works use as input for the parametrization a set of triangulated boundary surfaces that encloses the volume to parametrize, as shown by Martin, Cohen and Kirby (2009); and Escobar *et al.* (2011). This approach can use T-splines to refine locally the parametrized solid as shown by Zhang, Wang and Hughes (2012); and Zhang, Wang and Hughes (2013).

Another approach is to start with a geometry encompassing the solid to parametrize and reduce gradually its volume to fit the solid. Chan, Anitescu and Rabczuk (2017) used this approach with the advantage of PHT-splines for local refinement. Chen *et al.* (2019) applied polycubes to this approach.

Xu, Mourrain, Duvigneau and Galligo (2013a) parametrized solids assuming six NURBS surfaces enclosing the domain. The solution of the problem is known in advance and the control points

positions are based on minimizing the difference between the IGA solution and the exact solution. The method is an iterative process that stops when the error reaches a value lower than a tolerance. Xu, Mourrain, Duvigneau and Galligo (2013b) extended to problems with unknown solution. In both cases they used the coons method (Farin and Hansford, 1999).

Wang and Qian (2014) proposed an optimization of the Jacobian based on gradient-based optimization approach. The technique consists of making positive the Jacobians of all Bézier elements of the patch. The initial control points configuration is improved so that they reduce the number of iterations. They used two different methods for initial control points location: coons patch interpolation (Provatidis, 2005; Shih *et al.*, 2005) and deformation method. In the deformation method one initial enveloping cuboid is deformed until achieves the final shape, using FEA for solving the deformation.

In this work we introduce the *lattice algorithm* to create the control net of the solid, as explained in section 4.2.

2.4. Integration of trimmed domains

NURBS untrimmed domains might be discretized into portions that coincide with the non-void knot spans (Cottrell *et al.*, 2009), being the Gauss points location derived from such arrangement. Hughes, Reali and Sangalli (2010); and Auricchio *et al.* (2012) proposed the *half point rule* that reduces the number of required Gauss points. However the reduction of computational cost is not so effective since the location of Gauss points according to this rule requires preliminary calculation.

Integration of trimmed domains cannot follow the knot span scheme neither the *half point rule* because they would include non-computable portions of the domain in the integration. Different solutions have been proposed within the last decade.

The first attempts were devised for trimmed surfaces. Kim *et al.* (2009) applied localized triangulation only to trimmed knot spans, refining preliminarily the trimmed zone with T-splines. Triangles affected by the trimming edge are adjusted by a NURBS curves to that edge in the parameter space. Schmidt, Wuchner and Bletzinger (2012) proposed local re-parametrization of the trimmed knot spans. Nagy and Benson (2015) generated specific quadrature for the trimmed elements, using optimisation techniques to locate the integration points in the vicinity of the trimmed edges. Jaxon and Qian (2014); and Xia, Wang and Qian (2015) proposed a decomposition of the whole surface into Bézier triangles.

The Finite Cell Method (Parvizian, Düster and Rank, 2007; Düster *et al.*, 2008) is a technique to integrate trimmed domains (surfaces or solids). It was used in IGA for structures (Schillinger *et al.*, 2012; Schillinger and Ruess, 2015) and for fluids (F. Xu *et al.*, 2016; Varduhn *et al.*, 2016). The strategy of this technique is to detect which elements are trimmed and divide them into octants. After a few iterations finer element sizes are obtained where the trimming entity lies. Only the elements inside the domain are integrated. Kudela *et al.* (2016) increased the accuracy of the octants sub-division approach by moving the octrees near the trimming surface.

The discretization of solids into tetrahedrons has been proposed by different authors. Xia and Qian (2017) used tetrahedrons for both integration and parametrization of the solid. Previous to the parametrization a conversion of solid surfaces to Bézier triangles is done. This conversion brings a significant number of control points because for each Bézier rectangle of the surface that has degree pxq, the degree of the triangle needs to be $p + q$ (Goldman and Filip, 1987). Since these triangles are used as facets of tetrahedrons, the latter inherits the high number of control points. Therefore, the number of control points resultant from this discretization is typically high. Scholz and Jüttler (2019) proposed discretization with linear tetrahedrons, each one with its own parametrization and its own Gauss points. The integration rule is corrected to reduce the error due to linear approximation. Antolin, Buffa and Martinelli (2019) used higher order tetrahedrons. Firstly the domain is divided into the Bézier elements and then only the trimmed elements are approximated by tetrahedrons with degree according to shape of the element. To approximate the tetrahedral surfaces they used the *Gmsh software* (Geuzaine and Remacle, 2009).

In this work we discretize the domain into mixed-degree tetrahedrons. The approach followed in this work has similarities with the NURBS enhanced finite element method (NEFEM), proposed by Sevilla *et al.* (2008) and Sevilla *et al.* (2011) for surfaces and volumes respectively. The NEFEM uses NURBS entities as boundaries of the domain, being the facets of the triangles / tetrahedrons that lie on those boundaries approximated by Lagrangian polynomials. In this work we approximate also those facets with the same type of polynomials but there are remarkable differences. First, we introduce a method to estimate the number of required nodes of the Lagrangian polynomials to control the error (distance between the Lagrange approximation and the actual NURBS entity). Second, we use mixed-degree tetrahedrons, meanwhile Sevilla *et al.* (2011) used standard tetrahedrons (i.e. linear, quadratic, cubic, etc.). In addition we perform the approximation in the parameter space and not in the parent space.

It is worth to mention that there are alternatives to Galerkin's method in IGA, which require different approaches for the integration. See for example collocation methods by Auricchio *et al.* (2010); and Kiendl *et al.* (2015).

2.5. Constraints to trimmed solids

Constraints (prescribed displacements or couplings) can be strongly imposed, by control point's restraint, or weakly imposed. The strong approach requires specific locations of the control points at the boundary, contouring the zone where the constraint is applied. For coupling, in addition, both patches must be conformal at the interface. The weak imposition of constraints allows coupling regions independently on the control points and the parametrization of the patches. It forces the displacements to be the same in both domains over the whole interface. There are two manners of imposing weakly the condition: by Lagrange multipliers and by penalty method.

Coupling was first weakly imposed in FEA by Bernardi, Debit and Maday (1990), and then improvements were developed, see for example works by Belgacem and Maday (1997); or Belgacem (1999).

The implementation of Lagrange multipliers approach in the Finite Cell Method to impose essential boundary conditions in has been applied by Ruess *et al.* (2013) and for coupling not conformal domains discretized with NURBS by Ruess *et al*. (2014). Brivadis *et al.* (2015) proposed three different spaces for the Lagrange multipliers discretization. Coox *et al.* (2017) coupled non-conformal untrimmed 2D-patches by *virtual uncommon knot insertion*, that equates control points at the common edges of both patches.

The weak imposition of constraints, using both Lagrange and penalty approaches, has been used for three-dimensional surfaces extracted directly from CAD by Breitenberger *et al.* (2015). They coined the term Isogeometric B-rep analysis (IBRA). Teschemacher *et al.* (2018) integrated IBRA in a process that involves CAD design.

The weak imposition of constraints in IGA contact problems has been also implemented successfully, see for example the works from Temizer, Wriggers and Hughes (2012); and Kim and Youn (2012).

For solids, Zuo *et al.* (2015) used also the weakly imposed constraints. They introduced the concept of Constructive Solid Isogeometric Analysis (CSIGA), where IGA is applied to constructed solids geometry (CSG) models.

In this work we apply the constraints using the Lagrange multipliers approach similarly to Apostolatos *et al.* (2014), but extending it to solids.

2.6. Point projection

The mapping inversion from the physical space to the parameter space in NURBS entities does not have exact solution. The alternative is the so-called point projection technique, which approximates the parameter coordinates iteratively. According to Ko and Sakkalis (2014), point projection techniques may be classified in iteration-based and subdivision-based methods, the latter ends up with iterations.

2.6.1. Iteration-based methods

The most common iteration-based method is Newton-Raphson, that approximates the solution of a non-linear system of equations (Piegl and Tiller, 1996). These equations need the initial trial point sufficiently close to the solution to converge, *i.e.* the Newton-Raphson approach might have a poor robustness. There are other options of iteration-based methods as presented here.

Limaiem and Trochu (1995) provided an approach based on Kriging method (Matheron, 1980) to find intersections between two surfaces. Hoschek, Lasser and Schumaker (1993) tackled the problem for surfaces by iterative parameter adjustment in what they called the marching method.

Other variation consists of approximating a known geometry to the actual curve or surface. Hu and Wallner (2005) approximated a circle locally to the curve and find the solution on such circle. Local circles are iteratively used until the difference between the circle and the curve is less than a tolerance. For surfaces, they used spheres. Liu *et al.* (2009) varied the method using a torus instead, which improves the speed and stability.

2.6.2. Subdivision-based methods

Subdivision-based methods divide the domain selecting in each step the portion that contains the solution. After a few divisions, the iteration approach (section 2.6.1) is used to obtain the final answer. There is a variety of procedures to divide the domain as shown below.

Piegl and Tiller (2001) subdivided surfaces into quadrilaterals, with their sizes depending on the curvature. Ma and Hewitt (2003) proposed a division of the domain into Bézier elements and find the initial estimation using their control points. Selimovic (2006) subdivided the domain based on the control polygon. Chen *et al.* (2008) proposed a method based on circular clipping algorithm, which sets a circle centred at the point and reduces its radius until does not contact the curve. Oh *et al.* (2012) extended this clipping idea to surfaces, using spheres.

In this work we extend the marching approach, used by Hoscheck *et al.* (1993) for surfaces intersection, to curves and solids, as presented in appendix 10A.

2.7. Triangulation of surfaces

Triangulation of NURBS surfaces represents an open problem which is continuously evolving (Baker, 2005). The triangulation techniques can be classified in three types (Shimada, 2011): direct, purely parametric and hybrid approaches.

2.7.1. Direct approaches

Direct approaches compute the vertexes of triangles on the surface physical space. The three main methods within this type are Delaunay triangulation (Du, Faber and Gunzburger, 1999; Frey, Borouchaki and George, 1998), advancing front technique (Löhner and Parikh, 1988; Tristano, Owen and Canann, 1998; Frey, Borouchaki and George, 1998) and octree division (Yerry and Shephard, 1984; Shephard and Georges, 1991). Collision of two different fronts may appear in advancing front methods, which generates conflicts in the computation of the vertexes.

2.7.2. Parametric approaches

Parametric approaches compute the triangulation in the parametric domain. Then, they are projected into the physical domain (Sheng and Hirsch, 1992; Borouchaki, Laug and George, 2000; Cripps and Parwana, 2011). These methods lack uniformity on the resultant triangles in case the parametrization of the surface is not uniform.

2.7.3. Hybrid approaches

Hybrid approaches, which mix the two other types, cover most of the publications within the last two decades. For example, Béchet, Cuilliere and Trochu (2002); and Wang *et al.* (2006) tessellated surfaces by primary coarse triangulation in the parameter space and then increase the quality by Delaunay methods. Yang and Choi (2010) developed sequential triangulation to reduce the computational cost. Initial mesh is generated by Delaunay triangulation and then extra vertexes are added where curvature is more pronounced in the physical space. Marchandise, Remacle and Geuzaine (2014) presented three different linear parametrization techniques for refining one initial triangulation and increase its quality. Aubry *et al.* (2015) triangulated surfaces with the initial state assuming the edges already discretized. Then vertexes are added in the interior of the surface according to the curvature of the surface. Guo *et al.* (2019) presented the automatic triangulation which starts from a preliminary coarse triangulation that is refined and improved in two sequential stages.

2.7.4. Isotropic meshes

Isotropic mesh does not apply only to triangulation, but to other types such as quads or hexahedral, see for example the works done by Li *et al.* (2011); and Ebeida *et al.* (2011). For triangulations, the ideal isotropic mesh achieves vertexes valences equal to six and angles of 60 degrees. One ideal regular mesh has all triangles of the same size[14].

Surazhsky, Alliez and Gotsman (2003) developed a remeshing technique to gain isotropy. The technique is to be applied to an initial mesh in three stages: generation of vertexes, initial vertex partition and modification based on density function. The error diffusion algorithm was used for initial geometry sampling and then modifying that mesh to approximate to isotropic arrangement. Yan *et al.* (2009) applied the restricted Voronoi diagram (RVD) repeatedly to improve the isotropy of a given mesh.

In this work we introduce a new algorithm called *Quasi-isotropic initial triangulation* presented in appendix 10B.

[14] Refer to Appendix 5A for further information on isotropic surfaces.

3. **Algorithm overview**

This chapter sets the concepts and conventions used within the rest of the thesis and provides a general view of the algorithm. In addition, the generation of the geometry in CAD to be analysed in the algorithm is explained. This geometry is an input but not part of the algorithm itself. However, it is not a trivial process that is subject to rules and, therefore, requires explanation.

There are two appendixes directly related to this chapter. Appendix 3A contains schemes of the code itself, showing the most relevant routines and variables. This appendix might be seen as a developer hand book. Appendix 3B describes how the patches and surfaces are referenced. These references are used to allocate boundary conditions and apply refinement to the patches (see Appendix 8A).

This chapter is structured as follows. Section 3.1 presents the main definitions and conventions. Section 3.2 outlines the main stages of the algorithm. Section 3.3 describes the generation of the CAD geometry. In section 3.4 the relation of this chapter with the code is briefed. Finally, a summary is given in section 3.5.

3.1. Terms and conventions

3.1.1. Definitions

Definitions of basic concepts are listed below. **Fig. 3.1** to **Fig. 3.3** illustrate examples for clarity.

- **Patch**: solid geometry with tri-variate parametrization.
- **Domain**: set of patches that form the geometry to be analysed.

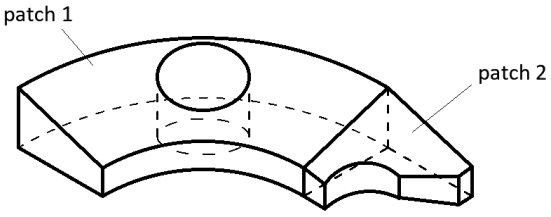


Fig. 3.1 Domain composed of two patches.

- **Gross patch**: patch bounded by six faces that coincide with the parametrization limits.
- **Trimming surface**: surface that trims one gross patch and contributes to its final shape.
- **Computable side**: opposite side to the normal vector of one trimming surface (the normal vector is assumed to point outwards the computable domain).
- **Computable domain**: portion of the gross patch that lies at the computable side of the trimming surfaces. The non-computable domain is the rest of the patch.

- **Trimmed patch**: patch that has trimming surfaces.
- **Bounded face**: remaining face of gross patch within the computable domain after trimming.

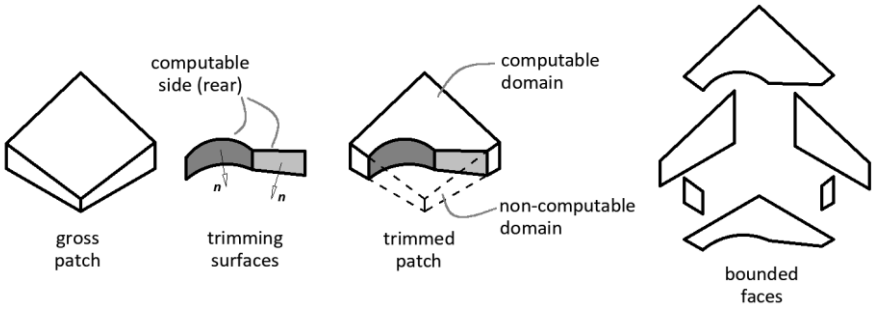


Fig. 3.2 Gross patch trimmed by two surfaces.

- **Boundary surface**: surface used to apply boundary conditions: Dirichlet, Neumann or coupling.
- **Coupling surface**: it is a particular case of boundary surface, which is used to couple two patches.
- **Bounded patch**: trimmed patch plus boundary surfaces on it.

The example illustrated in **Fig. 3.3** shows two patches, and their trimmed and bounded versions. Boundary surfaces are marked by numbers. Note that surface number 2 is a coupling surface. Boundary surfaces might coincide with trimming surfaces (surface 4), with bounded faces (surface 2) or with none of them (surfaces 1 and 3). In this example, one could apply Dirichlet boundary conditions to surface 1 and Neumann boundary conditions to surfaces 3 and 4.

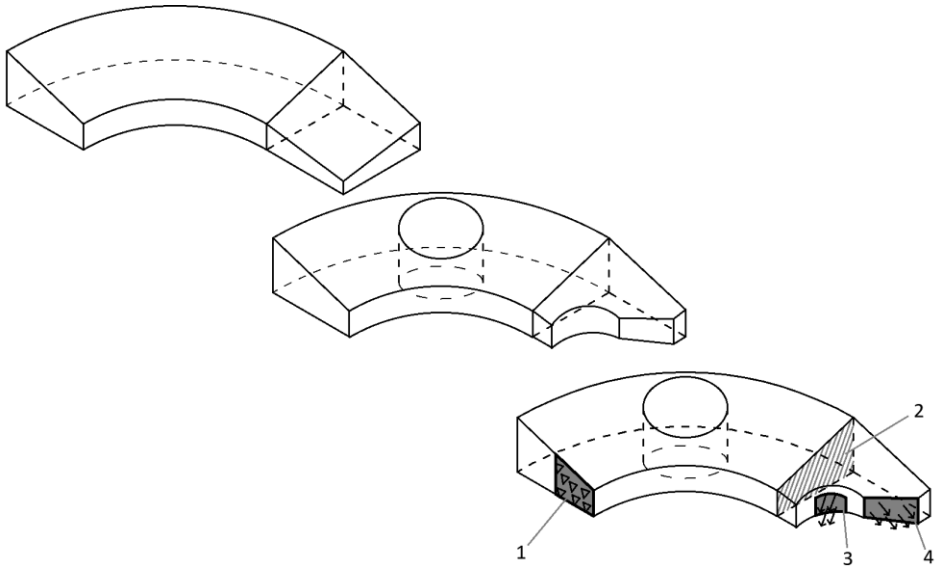


Fig. 3.3 Gross, trimmed and bounded patches.

- **Contour loop**: set of curves enclosing a region of one surface. The contour may coincide with the surface edges[15] or may trim the surface.

[15] They were called *edging contours* in section 2.2.3.

- **Background surface**: surface without considering the contour loops.
- **Computable surface**: portion of background surface that lies at the left-hand side of the curves of the contour loop (curves sense is given by their parametrization).

These three last items reflect the B-rep arrangement explained in section 2.2.3. **Fig. 3.4** illustrates one surface contoured by two loops, where part of the first loop is not trimming. The parameter sense of the curves is represented to highlight the computable surface is at the left-hand side.

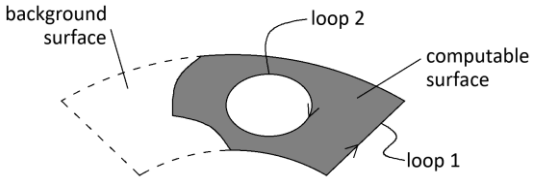


Fig. 3.4 Surface trimmed by contour loops.

3.1.2. Conventions for NURBS

The conventions listed below apply to NURBS. Some of them have been already mentioned in section 3.1.1, but they are stated here for clarity.

- The term *NURBS features refers* all the data necessary to define a NURBS object: number of control points, degree, knot vector and control points coordinates and weights. In addition, indicators of closed and planar domain might be included.

- All knot vectors span from 0 to 1.

- Control points might be referred with multi-indexes (one per parameter direction) or with a single index. In the latter case, the numbering follows the *hierarchical order* of parameter directions: numbers are allocated in rows along the last parameter direction, and the rows advance according to the previous parameter directions. One example is given in **Fig. 3.5**, for a volume with 3, 2 and 3 control points in parameter directions 1, 2 and 3 respectively. The control point 8 has multi-index (2,1,2).

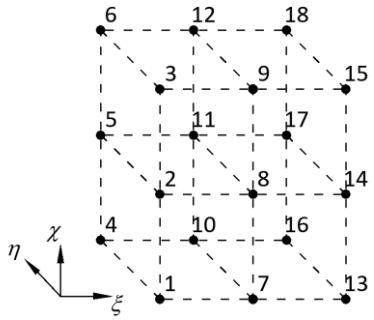


Fig. 3.5 Numbering of control points in *hierarchical order*.

- The positive side of a surface is the one where its normal vector commences, i.e. the negative side is the opposite to the normal vector.
- Surfaces that enclose a volume have its normal vectors opposite to the volume.
- Curves that contour one surface, trimming it or not, leave the computable portion of the surface at the left-hand side, looking at the surface from the positive side.
- *NURBS consistency* of one contour of curves (for surfaces) or one shell of surfaces (for volumes) is achieved if in each parameter direction the number of control points, degree and knot vector are the same. This concept only applies to surfaces contoured by four curves and volumes bounded by six faces. **Fig. 3.6** shows one example of non-consistent contour (left), where the number of control points is not the same in curves 3 and 4. At the right-hand side the consistency is achieved (it is assumed the same degree and knot vectors for curves 1-2 and for curves 3-4).

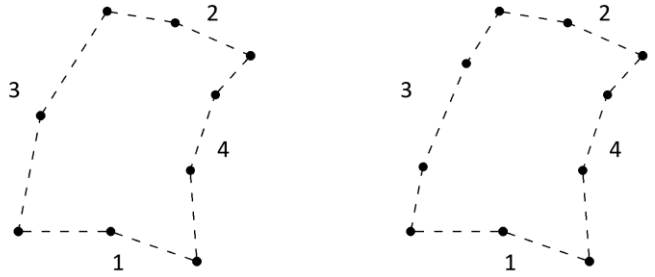


Fig. 3.6 Non-consistent and consistent contours.

3.2. Algorithm stages

The algorithm is composed by a sequence of five well-defined stages as shown in **Fig. 3.7**. The figure also indicates the chapters of this thesis involved in each stage. This section briefs the stages A to E to facilitate the comprehension of the rest of the thesis.

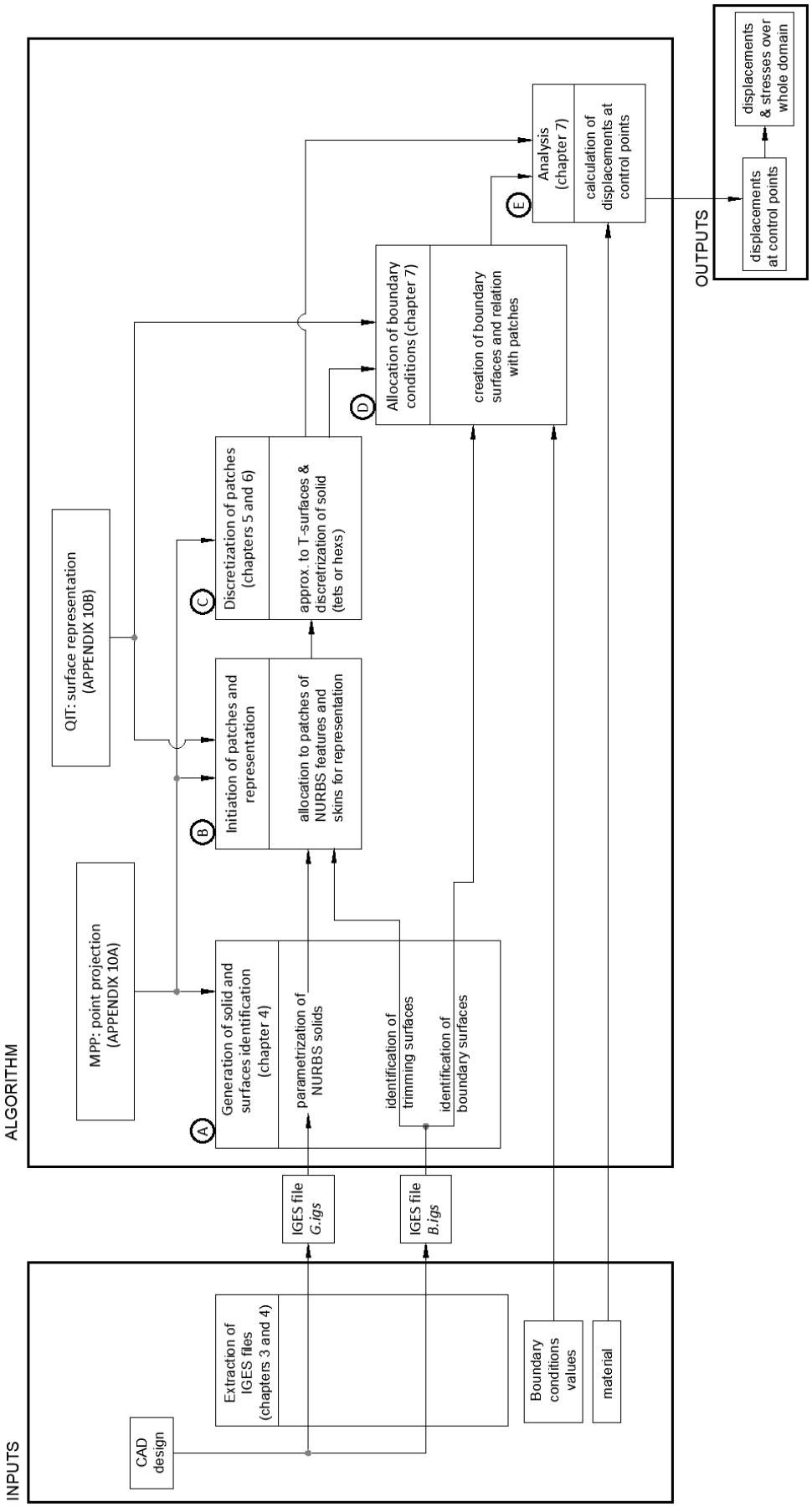


Fig. 3.7 Algorithm scheme.

There is a preliminary stage: the CAD design and extraction of IGES files, which is required to define the geometry inputs. Both operations (design and IGES extraction) are detailed in section 3.3, but we provide a summary here to understand the rest of the present section. The user is to draw the gross patches (restrained to have six faces) and extract their IGES file (called *G.igs*). Then, the user trims the gross domain to achieve the final shape, and attach any other boundary surfaces to the trimmed patches. The IGES file for the bounded patch is then extracted (called *B.igs*). This process is depicted in **Fig. 3.8**.

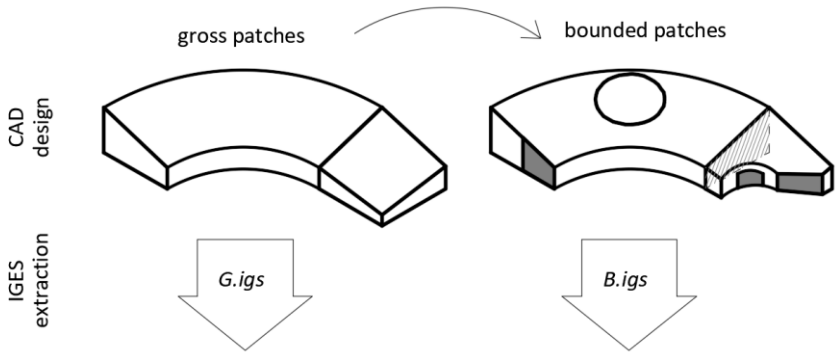


Fig. 3.8 CAD design and IGES extraction.

Sections 3.2.1 to 3.2.5 brief the stages of the algorithm itself. Each section corresponds to one of the stages, from A to E, indicated in **Fig 3.7**. In all the stages the point projection technique is required in multiple occasions. The used method, called *Marching point projection* (*MPP*), is presented in Appendix 10A.

3.2.1. Stage A - Generation of solid NURBS

With the IGES files at hand the solid patches and the surfaces (trimming and boundary) are generated in terms of NURBS. This stage is described in Chapter 4.

The solid information is taken from the *G.igs* file, and is generated from its faces in a process called solid parametrization, that is of special interest in this work. **Fig. 3.9** provides one example of solid parametrization. The starting point are the six faces, each one is a NURBS surface. The result is the NURBS solid: a three-variate parametric domain called gross patch.

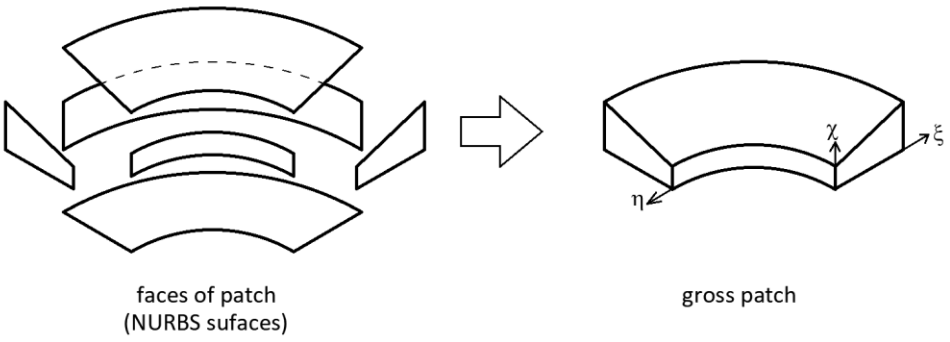


Fig. 3.9 Solid parametrization.

The *B.igs* file is used to define the trimming surfaces (and hence the trimmed patches) and boundary surfaces.

3.2.2. Stage B - Initiation of patches and representation

The NURBS features of both, the gross patches and the trimming surfaces, are allocated to the corresponding variables of the code (refer to Appendix 3A). These NURBS features will allow discretizing the trimmed patches in the next stage. This transference of information is not relevant from the theoretical point of view and that is why there is not chapter devised to it.

The solid parametrization corresponds to the gross patch regardless if it is trimmed or not. However the solid discretization applies only to the computable domain. One example is shown in **Fig. 3.10**.

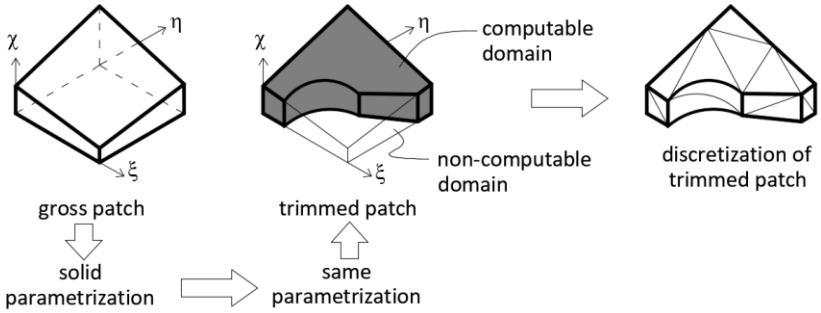


Fig. 3.10 Parameterization and discretization of a trimmed patch.

Visualization of the domain is necessary for displaying the results after the analysis: deformed shape and stresses. Bounded faces and trimming surfaces are triangulated to allow this representation. The triangles are linear, i.e. three nodes per triangle, and their sizes are pre-established by the user. The triangulation for each surface is called *skin*. **Fig. 3.11** illustrates one example of this triangulation procedure.

To allow for a high-quality triangulation a new scheme has been develop in this thesis, which is called *Quasi-isotropic initial triangulation* (*QIT*) and is briefed in Appendix 10B.

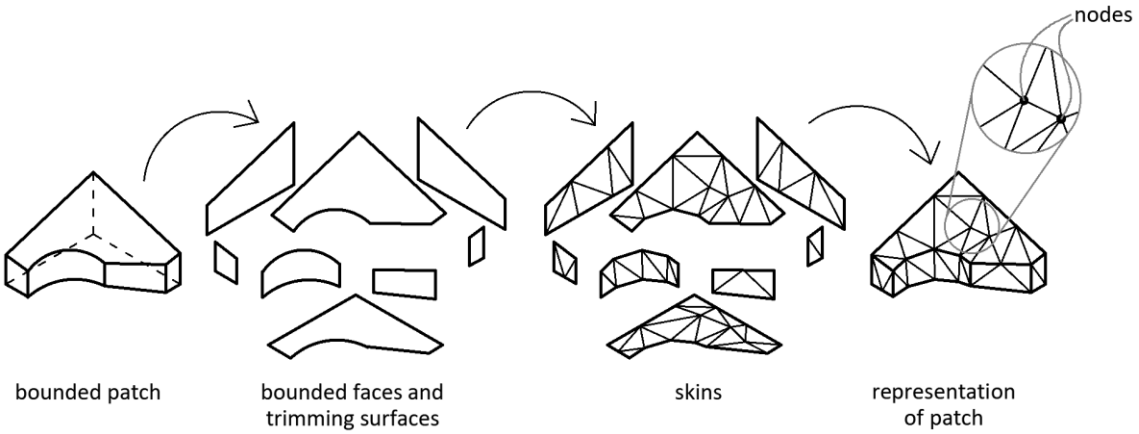


Fig. 3.11 Triangulation of bounded faces and trimming surfaces.

3.2.3. Stage C - Discretization of patches

In order to compute the stiffness matrix, Gauss points are needed within the computable domain. Therefore, patches need to be discretized into elements to place within them the corresponding Gauss points. We use different approaches for non-trimmed and trimmed patches. The former are discretized into hexahedrons according to the parametrization (recall section 2.1.2). The latter are discretized into tetrahedrons which have inherent adaptability to any shape of the trimming surfaces. Recall that this discretization, although it has similarities with the tretrahedralization proposed by Sevilla *et al.* (2011), it presents important differences as explained in section 2.4.

Let us highlight that this discretization is not the same as the triangulation of the skins presented in section 3.2.2, indeed both are independent. Skins triangulation is devised merely for representation meanwhile the solid discretization is used for analysis.

Trimming surfaces are unknown in the parameter space of the patch. Hence they are approximated (Chapter 5) prior to the discretization (Chapter 6). The tetrahedrons facets that lie at trimming surfaces coincide with them. One example of trimmed patch discretization is provided in **Fig. 3.12**.

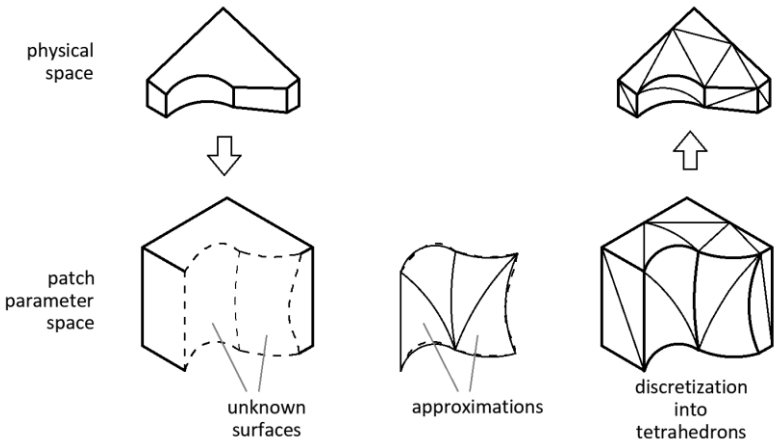


Fig. 3.12 Trimmed patch discretization.

3.2.4. Stage D - Allocation of boundary conditions

Boundary conditions and coupling between patches are applied onto surfaces. To consider these constraints in the analysis, the basis functions from the affected patches are integrated over these surfaces[16]. Therefore we need to define Gauss points on the boundary surfaces to carry out the integration. These surfaces are discretized into linear triangles where Gauss points are placed. The process is shown in Chapter 7.

The triangulation is done with the *Quasi-isotropic initial triangulation* (*QIT*) algorithm, presented in Appendix 10B. The sizes of the triangles are defined by the user. This triangulation is independent from the representation skins (section 3.2.2) and the tetrahedral discretization of the solid (section 3.2.3). It is devised exclusively to apply constraints.

In this stage, the values of the boundary conditions (defined previously by the user) are inserted in the algorithm. These values may be prescribed displacements (Dirichlet boundary conditions) or tractions (Neumann boundary conditions). **Fig. 3.13** provides one example, where one coupling surface and two boundary surfaces are discretized. Tractions are allocated to the boundary surfaces.

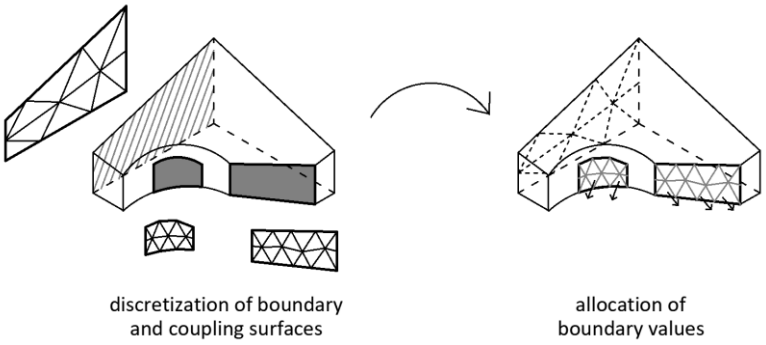


Fig. 3.13 Discretization of boundary and coupling surfaces, and allocation of tractions.

3.2.5. Stage E - Analysis

Once the patches are discretized and their Gauss points defined, their stiffness matrices can be computed. The material of the domain is considered in this stage. Boundary conditions can be added to the system because the boundary and coupling surfaces have their own Gauss points also defined. The aggregated linear system[17] is solved obtaining the displacements at each control point of the patches. Chapter 7 details the construction of the aggregated system of equations.

To represent the results, the positions of the skins nodes are recomputed with the deformed configuration of the control points. The stresses are derived from these displacements (see

[16] See Appendix 2B.
[17] Equation (2B.54) of Appendix 2B.

Appendix 7A). Chapter 8 presents a set of examples that validates the algorithm proposed in this thesis.

3.3. CAD stage and IGES files extraction

The geometry is passed to the algorithm in form of IGES files, extracted directly from CAD, which is aligned with the main idea of IGA (recall section 2.1). Two IGES files are to extract from CAD. The first one, called *G.igs*, contains the gross patches that will be used for solid parametrization. The second file, called *B.igs*, contains the bounded patches that will establish the computable domain and the boundary surfaces.

To illustrate the process, the example shown in **Fig. 3.14** will be mentioned along with the explanations. In addition, Appendix 8A details this CAD stage for the examples given in Chapter 8 of this thesis.

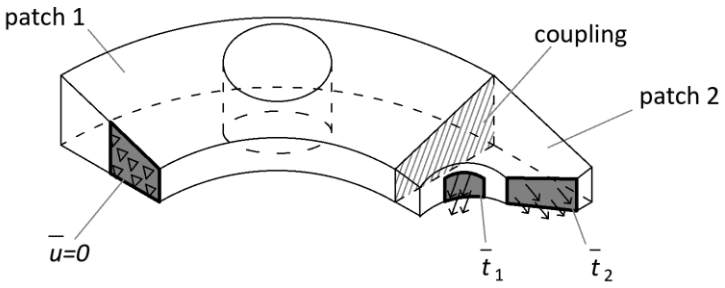


Fig. 3.14 Bounded domain to achieve.

3.3.1. Gross patches

The gross patches are created by common CAD techniques[18] but considering next restrictions:

- Each gross patch must have six faces.
- Angles between faces must be greater than 0 degrees and less than 180 degrees.
- Each face must be smooth with minimum continuity C^1.
- In case of two or more patches, they must be in contact by a surface (not a single line) or have their volumes intersecting.

Non-valid gross patches are shown in **Fig. 3.15** where these restrictions are violated as explained here (from left to right). The first has more than six faces (eight faces). The second possesses one angle greater than 180 degrees. The third has one non-smooth surface, it can be seen also as a patch with eight faces, which is neither valid. The fourth is a gross domain composed by two valid patches, however they are not in contact or intersecting.

[18] The most common CAD techniques used for solids generation are extrusion, revolving, sweep and loft.

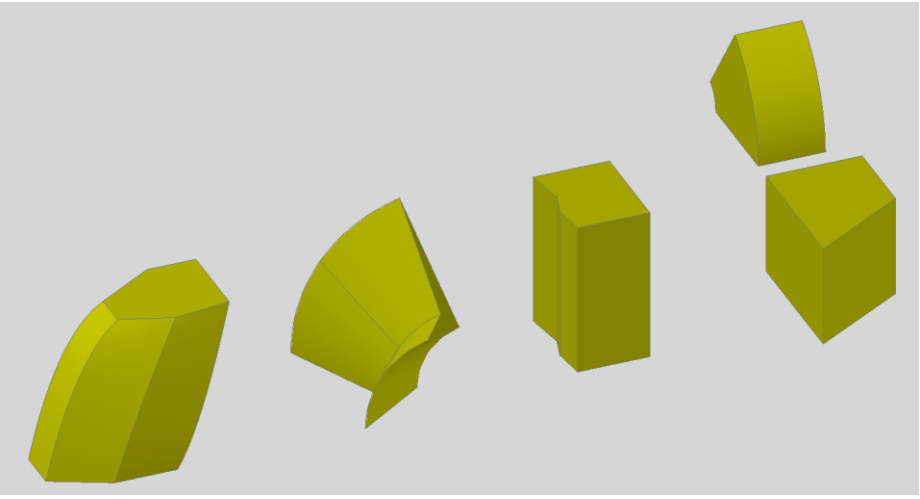


Fig. 3.15 Non-valid gross patches.

Ideally, the final shape should be achieved by gross patches, without trimming surfaces. These un-trimmed patches present faster analysis since there are no trimming surfaces to deal with. However that is not possible in the majority of the cases. Indeed, the restrictions of the gross patches listed above make them very rigid to achieve any shape and trimming surfaces are normally needed.

There are multiple valid gross patches for a certain final shape. **Fig. 3.16** shows two valid gross patches for the patch 1 of the example. Above, the gross patch is a cube that needs to be fully trimmed to achieve the final shape. Below, the gross patch is almost the same as the final shape because it needs only one trimming surface. The second option is computationally cheaper since there is only one trimming surface to deal with. In general terms, the more similar the gross patch to the trimmed patch the less trimming surfaces are required and therefore the lower computational cost.

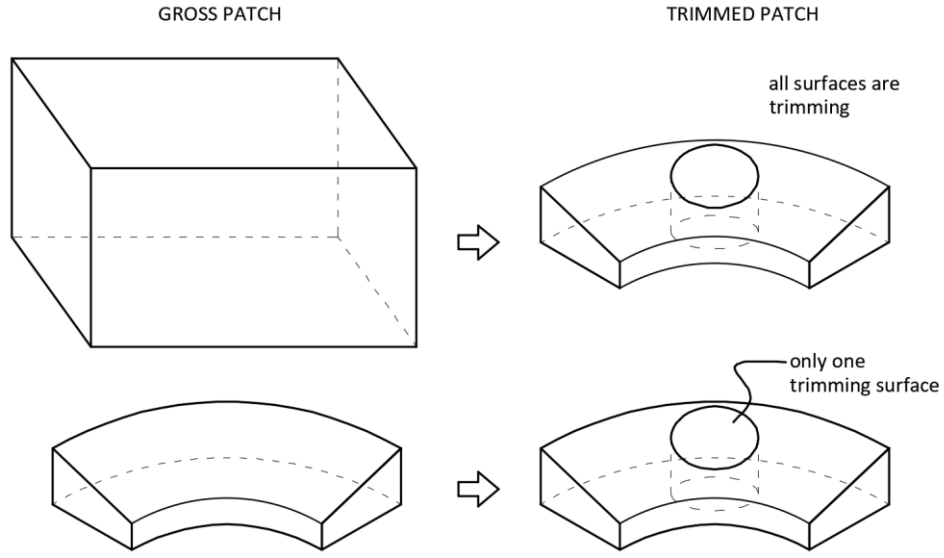


Fig. 3.16 Two valid options for gross patch 1.

The gross patch must envelope the volume of the trimmed patch because trimming surface will remove non-computable portions, but never provide additional volume. In **Fig. 3.17** the proposed gross patches used for the example are shown. Note that they are in contact at one face that will become a coupling surface.

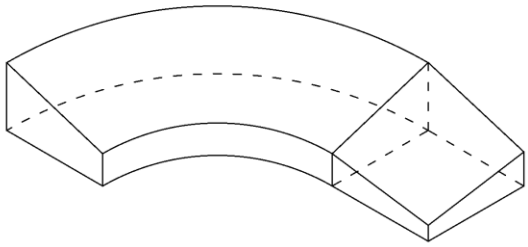


Fig. 3.17 Proposed gross patches for the example.

Once the gross patches are designed, their IGES file is extracted. In this work we call to this file *G.igs*, however the name may be different as long as it coincides with the algorithm input data. Before presenting the construction of the bounded patches, let us remark that once the *G.igs* is extracted, the gross patches cannot be moved. This means that trimming and adding boundary surfaces must be done to the gross patches placed at the same location. This is required because the surfaces identification checks their location with respect to the gross patches (see section 4.3).

3.3.2. Bounded patches

Bounded patches are constructed in CAD from gross patches in two steps:

- Trimming to achieve final shape. The so-called trimming surfaces are used here.
- Attaching surfaces onto the final shape to apply boundary conditions or coupling. These are the boundary surfaces.

To trim the gross patches any technique available in CAD is valid, as long as the final shape is obtained. For example, in **Fig. 3.18** substraction (Boolean operation with CAD volumes) is applied. In **Fig. 3.18** the trimmed patches are also shown. These patches achieve the final shape, but the process is not yet finished since we need the boundary surfaces.

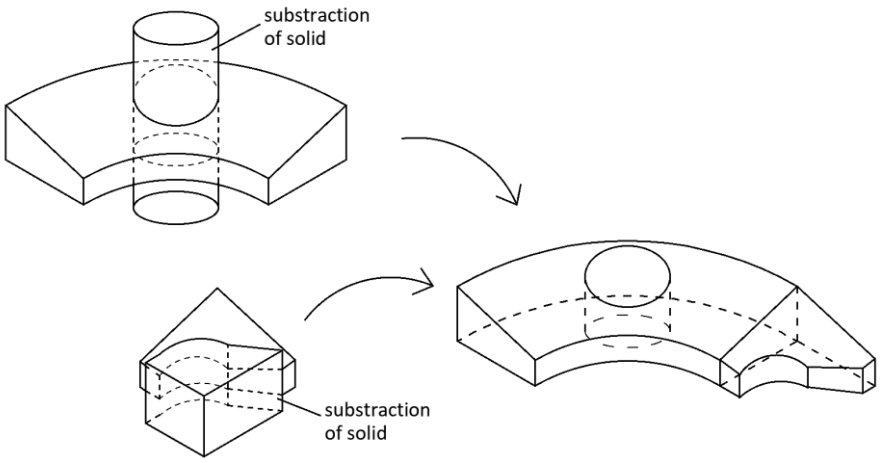


Fig. 3.18 Trimming of patches and resultant shape.

Boundary surfaces are placed onto trimmed patches. Again, there is not a unique manner to generate boundary surfaces. In this work the procedure used is: copy and extract a patch surface, trim if required and placed back onto the patch[19]. The process is shown in **Fig. 3.19**.

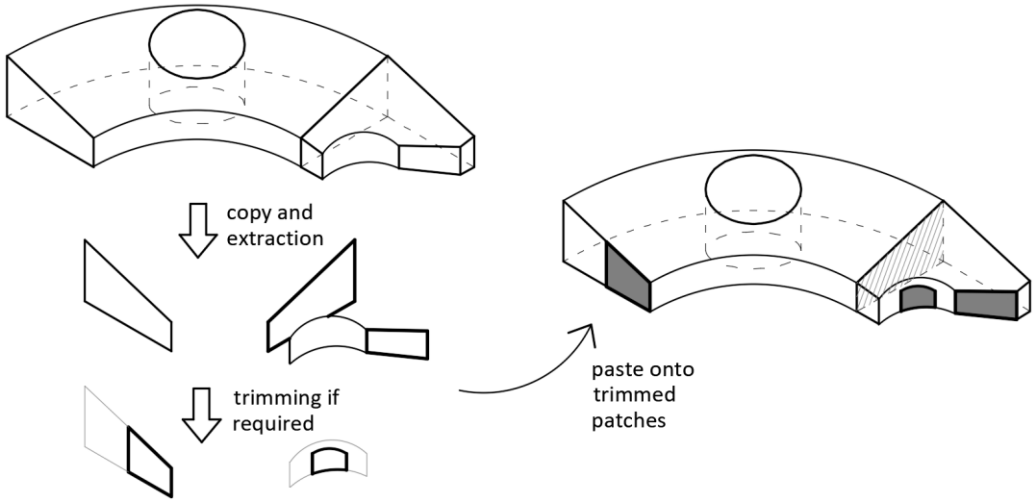


Fig. 3.19 Generation of boundary surfaces and allocation to the domain.

Once the bounded patches are finished, the IGES file is extracted. In this work we call to this file *B.igs*, however the name may be different as long as it is reflected in the algorithm input data.

Coupling surfaces will be detected automatically by the algorithm (see section 4.3). Boundary conditions are attached to the corresponding boundary surfaces by the user as shown in Appendix 8A. Boundary surfaces and patches are referred by numbers that are allocated as detailed in Appendix 3B.

[19] In some CAD packages (as in AutoCAD®) the surface generated by extraction of a solid is initially not a NURBS surface. If that is the case, the surface need to be converted to NURBS by the user.

3.4. Relation with the code

The code is structured in five main functions, each one corresponding to one main stage (recall **Fig. 3.7**). The first stage is the **generation of solid NURBS** from the IGES files, the routine `i3100_IGES` hosts this procedure. The other four stages are all related to the buttons of a user form created for this thesis. **Initiation of patches and representation** stage lies within the `B_Generate_Skins` button, the **discretization of patches** is under the `B_DomainIni` button, **boundary conditions allocation** happens under the `B_BCs` button and the **analysis** stage under the `B_Analysis` button. Within each button there are multiple routines to carry out the computations defined in this thesis. The routines related to each chapter are further explained in them.

To start running the code, once the IGES files are transferred to the 'TXT' folder of Matlab®, the user is to type `FirstRoutine` in the command window. Then the `i3100_IGES` is executed and the user form is shown in the screen.

3.5. Summary of the chapter

In this chapter the concepts and conventions used along the rest of the thesis are defined. The algorithm main stages are established such that the reader knows in advance how the chapters are connected. This general view will enhance understanding of the subsequent chapters. In addition, the construction of geometry in CAD, that is necessary to create the geometry input for the algorithm, is presented. In this CAD design process there are two distinguished steps, one for gross patches and one for bounded patches. This distinction is necessary because gross patches will be used for solid parametrization, as shown in Chapter 4, and bounded patches will define trimming and boundary surfaces. The trimming surfaces are used to discretize the computable domain, as shown in Chapters 5 and 6, meanwhile the boundary surfaces are used to apply constraints as explained in Chapter 7.

4. **Parametrization of solids and identification of surfaces**

This chapter describes the parametrization of the solid domain and the identification of surfaces, both trimming and boundary surfaces. **Fig. 4.1** shows one portion of the main diagram, extracted from section 3.2. In view of **Fig. 4.1**, the *G.igs* file is used for solid parametrization and the *B.igs* file for identification of surfaces.

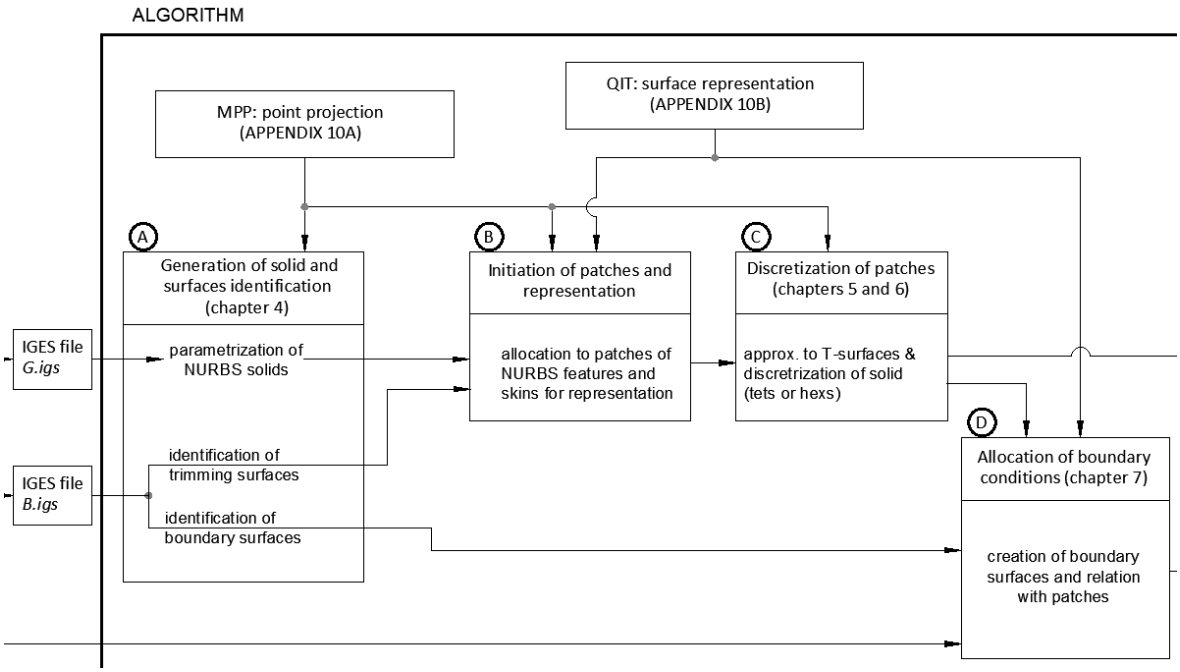


Fig. 4.1 Extraction from main scheme.

The *G.igs* file contains ***manifold solid B-Rep*** objects (section 2.2.1) with the gross patches, which is used to parametrize the solid. *B.igs* file contains ***manifold solid B-Rep*** objects with the trimmed patches, and ***bounded surfaces*** (section 2.2.2) with the boundary surfaces. **Fig. 4.2** presents a scheme of this information.

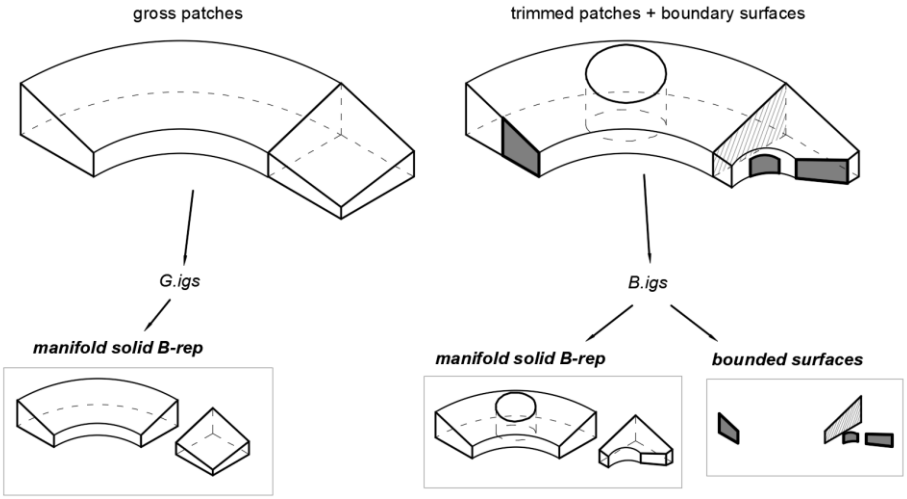


Fig. 4.2 Content of *G.igs* and *B.igs*.

This chapter is organized as follows. Section 4.1 describes the key points of the transformation of IGES files into numerical arrays. Section 4.2 details the solid parametrization of gross patches. Surfaces identification is explained in section 4.3. The relation of the presented procedures with the code is briefed in section 4.4. Finally section 4.5 summarizes the chapter content.

4.1. Transfer of IGES information to the algorithm

The transfer of information from the IGES files to the code has three key points explained here. For further details of the code and variables involved in this process refer to Appendix 3A.

The first point affects to the format of the information. The IGES information, that are text format (see Appendix 2G), must be moved to numerical arrays readable by the algorithm. This information is stored in numerical arrays with similar structure to the IGES.

The second point regards to the arrangement of NURBS features in surfaces: parameter directions in the IGES files and in this work are swapped. Due to this discrepancy, knot vectors, degrees and numbering of control points need to be interchanged between the parameter directions to match the desired arrangement. **Fig. 4.3** shows one surface as it comes from IGES and as it is used in this work where the parameter directions are swapped.

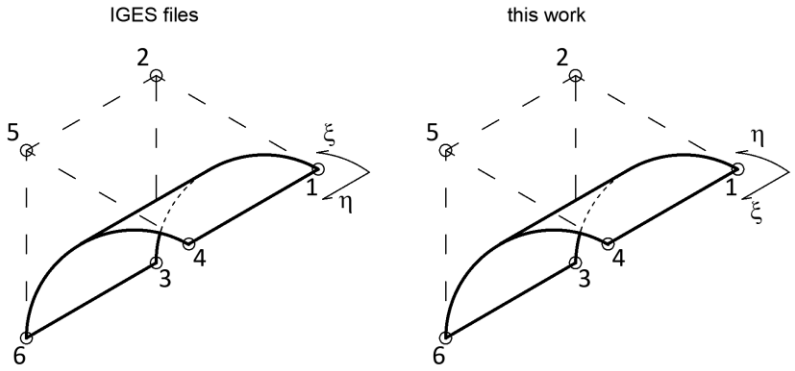


Fig. 4.3 Arrangement of surfaces in IGES files (left) and in this work (right).

The third point is about knot vector span. In IGES files, the span limits of the knot vectors vary. In order to meet this work conventions (section 3.1.2), they need to be scaled to span from 0 to 1.

4.2. Parametrization of gross patches

Gross patches are parametrized using their enveloping faces, which are given as NURBS surfaces and NURBS contour curves in the *G.igs* file. The content of this file needs processing since it is not devised for the solid parametrization as shown in section 4.2.1. In this work we create the object *Gross Patch Shell* (GPS). Each patch has its own GPS, which forms a shell of surfaces arranged to facilitate the ulterior solid parametrization. GPS is obtained from the *G.igs* file, by modification to

the NURBS surfaces and curves as explained here. This section details the generation of the GPS (section 4.2.1) and the solid parametrization (section 4.2.2). In section 4.2.3 the specific algorithm called *lattice fitting*, used in some cases for solid parametrization, is presented.

4.2.1. Gross patches shells (GPS)

Each GPS contains the next information of the corresponding gross patch:

- Coordinates of the eight vertexes.
- NURBS features of the 12 curves that form the edges.
- NURBS features of the six surfaces (gross patch faces).

Differences between GPS and IGES information

The GPS is not a mere copy of the surfaces contained in *G.igs*, but an improved version. Each GPS is a collection of six NURBS surfaces forming a watertight regular boundary[20], with genus-zero topology (Bauer, Catanese et al. 2011), that bounds the gross patch. Each of those surfaces is contoured by four curves which are the intersection with the contiguous surfaces.

The surfaces contained in *G.igs* are defined independently from each other. In GPS, by contrast, the opposite surfaces have the same degree, number of control points and knot vector in each parameter direction. This consistency enables an easier parametrization of the enclosed solid as shown in section 4.2.2.

Fig. 4.4 outstands the differences between *G.igs* and the GPS. In IGES files (left) the NURBS surfaces are not in general ending where the contiguous surface starts, i.e. it is not a regular boundary. This is due to the B-rep system: some CAD surfaces are trimmed by the contour loop that encloses the visible portion (section 2.2.3). However the GPS (the right-hand side of **Fig. 4.4**) removes the non-computable portions of surfaces to achieve regularity. In addition, opposite surfaces do not have necessarily the same parametrization in *G.igs*, meanwhile in GPS they do. This latter difference is represented in **Fig. 4.4** for the top and bottom surfaces, where one can see the orientations of the parameter coordinates coincide in GPS but not in the IGES file.

[20] A regular boundary is made of surfaces that do not intersect pairwise, except at their contour curves, and they have no self-intersection points.

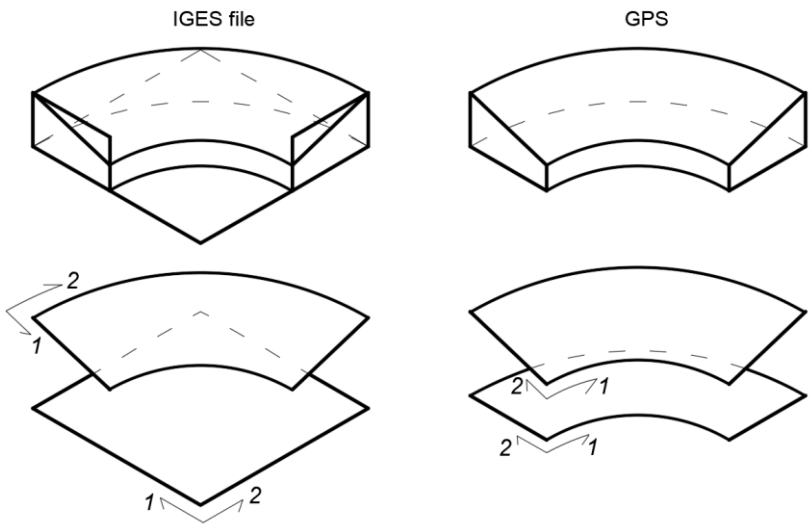


Fig. 4.4 Comparison between *G.igs* and GPS.

Features of patches stored in G.igs

In the case of *G.igs*, due to the restrictions established when drawing gross patches (section 3.3.1) and the structure of IGES files (sections 2.2.1 and 2.2.2) next conditions are always fulfilled:

- The reference of face, NURBS surface and loop coincide (there is one loop of four curves per face).
- Edge list indexes coincide with NURBS curves references.
- The number of faces is six.
- The number of vertexes is eight.
- The number of contour curves is 24 (four curves per face).

Generation of GPS

The allocation of face 1 and vertex 1 in the GPS object obeys to the next arbitrary rule[21]: face 1 of GPS is the surface of *G.igs* whose average of its four vertexes coordinates is the closest of the origin of the physical space. The closest vertex to the origin among those four is defined as vertex number 1. The other faces and parameter directions are designated by their relative position to face 1 and vertex 1 by the next rules:

- Origin of parameter directions coincides with vertex 1.
- Face 1 is perpendicular to the third parameter direction (χ).
- First (ξ) and second (η) parameter directions are orientated w.r.t. χ according the right hand side rule ($\chi = \xi \times \eta$).
- Faces 2 and 3 are perpendicular to η and ξ respectively and share vertex 1 with face 1.
- Faces 4, 5 and 6 are opposite to 1, 2 and 3 respectively.

[21] Other rules may have been applied.

The references of the faces result as illustrated at the left-hand side of **Fig. 4.5**. At the right-hand side the local parameter orientations that each face must have are also represented, as well as the contour curves (with orientation and references local to the surface). Contour curves for each face of GPS are referred locally by two indexes, the first one relates the parameter direction of the curve within the face and the second the position.

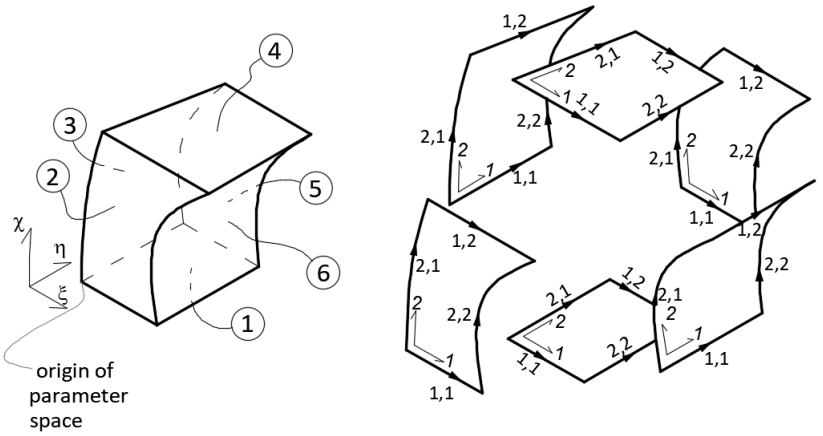


Fig. 4.5 Faces references in GPS and local orientation of faces and contour curves.

The rest of the vertexes references are allocated by the parameter directions hierarchy (recall section 3.1.2) and their coordinates are transferred directly from *G.igs*. Edge curves are grouped in three sets, one per parameter direction with four curves each. Curves 1 to 4 are parallel to ξ direction, 5 to 8 to η and 9 to 12 to χ. Within each set the curves are numbered according to the parameter directions hierarchy (see **Fig 4.6**).

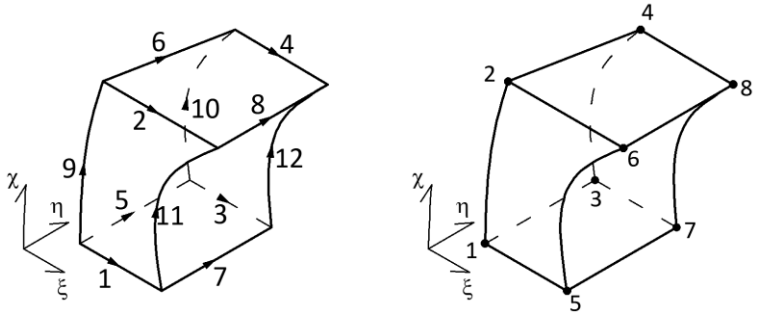


Fig. 4.6 Curves and vertexes references in GPS.

Number of control points, degree and knot vector of edge curves must be the same within each of the three groups, i.e. they must be consistent (recall NURBS consistency concept in section 3.1.2). However, in general it is not the case in the IGES file. Therefore, the four curves within each group are compared and refined, if required, to achieve consistency as listed below in a homogenization process:

- Degree: the target degree is the maximum among the four curves. Any of them with a lower degree needs *p*-refinement to meet the target degree (see **Fig. 4.7**). If one curve is

p-refined, knots removal is applied afterwards to remove the repeated internal knots due to the degree elevation (Piegl, Tiller 1996).

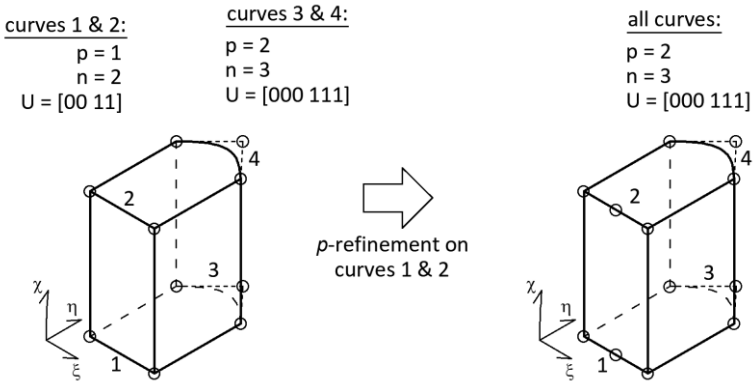


Fig. 4.7 Homogenization of degree for set of curves in ξ direction.

- Number of control points: the target number of control points is the maximum among the four curves. Any of them with a lower number of control points needs *h*-refinement to equal the target number (see **Fig. 4.8**). New knots inserted are selected among the envelope knot vector (see next step).

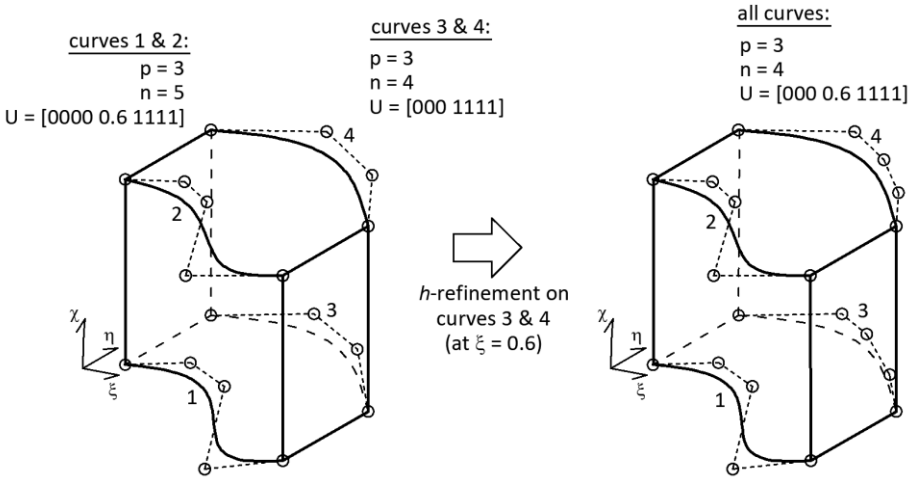


Fig. 4.8 Homogenization of number of control points for set of curves in ξ direction.

- Knot vector: the envelope knot vector possesses all knots from the four curves. If one curve lacks of any of the envelope knots it needs *h*-refinement with knots inserted where required to match the envelope (see **Fig. 4.9**).

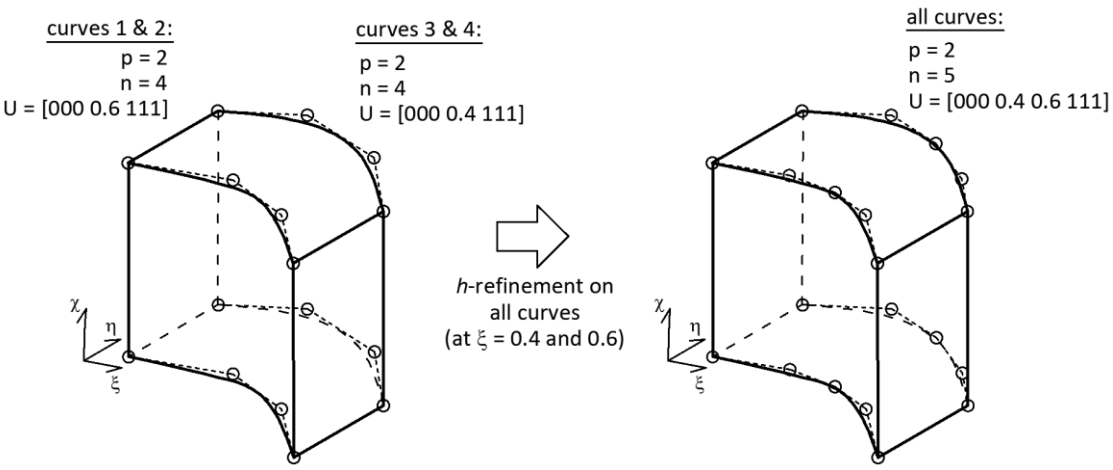


Fig. 4.9 Homogenization of knot vector for set of curves in ξ direction.

Each of the six faces must have its local orientation as per **Fig. 4.5** right. One face has four contour curves, referred as (1,1), (1,2), (2,1) and (2,2) as per same figure, and a NURBS surface. The features of the surface need to be consistent with the contiguous surfaces, i.e. same degree, number of control points and knot spans at the intersection curves. To achieve this consistency the surface is compared to its contour curves (already homogenized). There are three cases depending on the surface characteristics:

- Case A: non-planar surface. This type of surface is untrimmed in gross patches (recall section 2.2.3). The surface features are directly transferred from *G.igs* with the precaution of flipping the orientation (recall **Fig. 4.5** criterion). When correct orientation is achieved, degree, control points and knot vector are compared to its edge curves. In case of discrepancy, the surface needs refinement to match the curves features. **Fig. 4.10** illustrates one example: (a) the surface is highlighted in the gross patch with the required parameter local orientations for GPS; (b) the surface comes from IGES file with different parameter orientations; (c) the orientation of first parameter is inverted and the node insertion is done in the second parameter direction to match the edge curve (drawn adjacent to the surface); (d) the surface obtained is oriented following the criterion of this work and it is consistent with the edge curves.

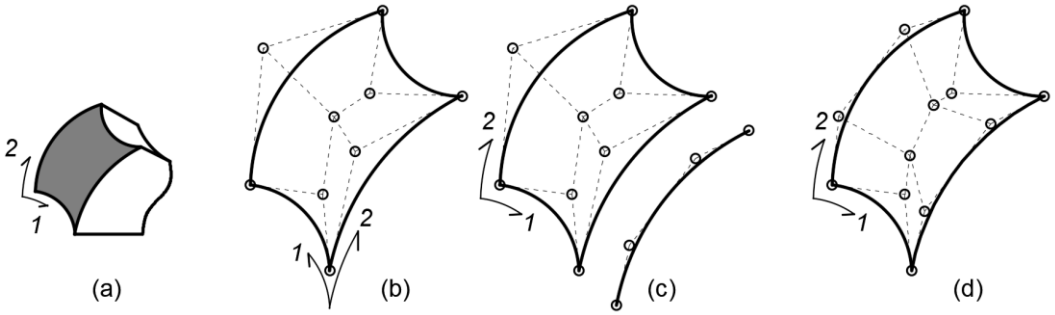


Fig. 4.10 Face generation. Case A.

- Case B: planar surface with both directions non-linear. As mentioned in section 2.2.3, these NURBS surfaces are stored in IGES files as a plane rectangle trimmed by contour curves. Since the GPS object requires non-trimmed surface, that trimmed arrangement would not be valid. Then the surface is created by the procedure that is called *plane lattice fitting*. This routine consist of creating a quadrilateral initial grid of control points and then apply displacements to its contours to match the contour control points of the desired surface in a 2D lattice structure fashion. The procedure is detailed in section 4.2.3. **Fig. 4.11** illustrates one example: (a) the surface is highlighted in the gross patch; (b) the background surface is a rectangle with trimming contour curves; (c) the initial lattice of control points is set (independent from the IGES rectangular contour); (d) the final lattice is consistent with the edge curves.

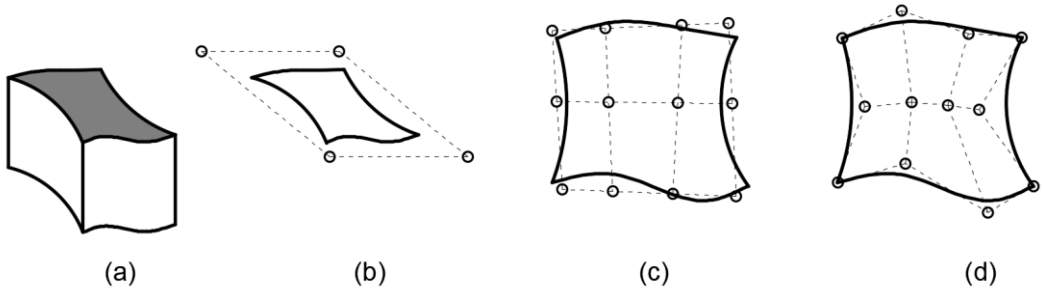


Fig. 4.11 Face generation. Case B.

- Case C: there is at least one linear direction. The surface features are read from the pair of contour curves that are not linear (degree >1), with previous flipping of orientation if required (recall **Fig. 4.5** criterion). Knot vector in the linear direction is {0011}. **Fig. 4.12** illustrates one example: (a) the surface is highlighted in the gross patch with the parameter required orientations for GPS; (b) the non-linear edge curves are prepared and jointed to form the surface (c).

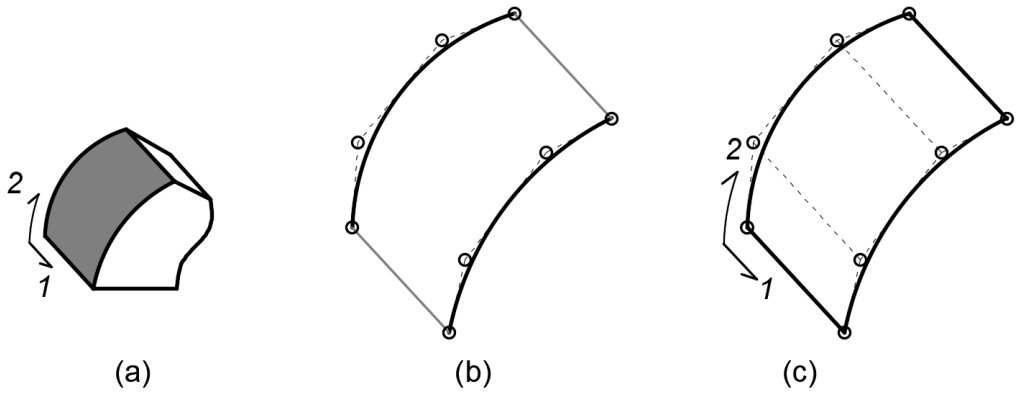


Fig. 4.12 Face generation. Case C.

Note that GPS does not follow some of the conventions presented in section 3.1.2: not all the contour curves leave the surface to the left-hand side and not all the surfaces have their normal pointing outwards the enclosed volume. This special arrangement of GPS is convenient for the solid parametrization (as shown in section 4.2.2), and is the only exception in the whole thesis to the NURBS conventions.

4.2.2. Parametrization of solid patches

This section describes the solid parametrization using the GPS (section 4.2.1). The number of control points, degree and knot vector in the solid parameter directions 1, 2 and 3 can be directly transferred from GPS curves 1, 4 and 9 respectively. The coordinates of control points that do not belong to the edges have different treatment depending on the type of solid as follows. According to the number of parameter directions that are linear (NLD) we establish four cases:

- Case A: NLD = 3. The gross patch is a tri-linear cuboid. Control points coordinates are transferred directly from the GPS vertexes, and all weights are all equal to 1.

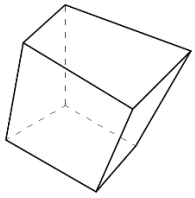
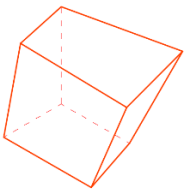
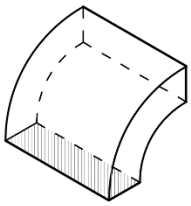
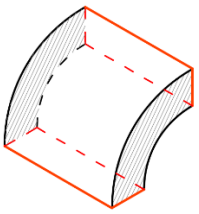
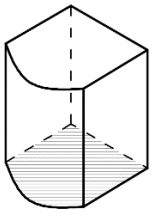
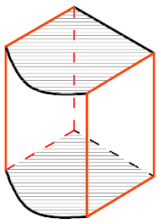
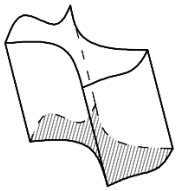
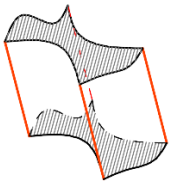
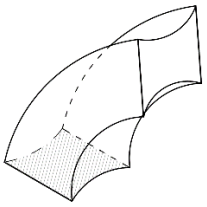
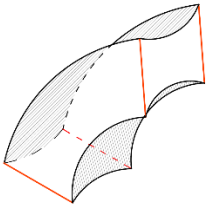
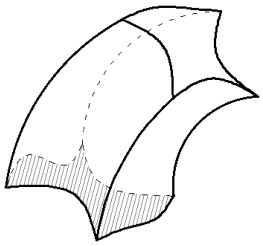
- Case B: NLD = 2. The parametrization is achieved by the *sandwich algorithm:* GPS is linear between two faces called *generation faces*. The solid is achieved by re-arrangement of NURBS features of those faces. The *sandwich algorithm* is explained below.

- Case C: NLD = 1. Same procedure that case B, therefore the *sandwich algorithm* is also used.

- Case D: NLD = 0. Control points coordinates are computed by the *volume lattice fitting* algorithm described in section 4.2.3.

These cases are summarized in **Table 4.1**, which includes also comments on the potential construction techniques.

Table 4.1 Cases considered for solid parametrization.

Case	Examples		Construction comments	NLD
	CAD construction*	Solid parametrization**		
A			Formed by any 6 plane sides achieved by linear extrusion. Admits plane trimming as long at the tri-linear cuboid is kept.	3
B			Constructed by revolution or sweep of basis face. Basis face has two linear directions.	2
			Constructed by extrusion of basis face. Basis face has one non-linear direction.	
C			Constructed by extrusion of basis face. Basis face has two non-linear directions.	1
			Constructed by revolution, sweep or loft of basis face. Basis face has one linear direction.	
D			Constructed by revolution, sweep or loft of basis face.	0

* Hatched face indicates basis face for construction

** Hatched face indicates generation face. Red line indicates linear edge (or linear parameter direction).

Sandwich algorithm:

To apply the sandwich algorithm, the GPS must have one or two parameter directions with degree=1 (cases B and C in **Table 4.1**). Hence, the GPS has one pair of faces that are perpendicular

to the linear direction, referred here as *generation faces*. In case B there are two pairs of *generation faces*, selecting one or another is not relevant. The solid is linear between the *generation faces*: no control points are needed between them. Therefore the control points and other NURBS features from *generation faces* are re-arranged to attain the parametrized solid. Knot vector in the linear direction is {0011}. See **Fig. 4.13** with one example, where GPS is linear between faces 2 and 5.

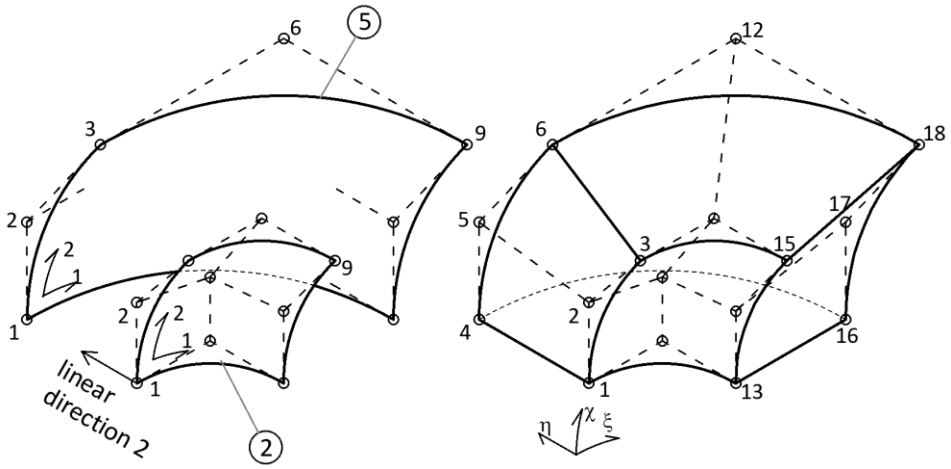


Fig.5.13 Parametrization by the *sandwich algorithm*.

4.2.3. Lattice fitting algorithms

These algorithms parametrize one target NURBS entity (plane surface or solid) given the NURBS features of the *boundary entities*. The algorithm is called *plane lattice fitting* for plane surfaces and *volume lattice fitting* for solids, being the boundary entities curves and surfaces respectively.

Let us designate the target NURBS entity as Ω (unknown) and the NURBS boundary entities as Γ, which are known and have NURBS consistency (see section 3.1.2). Due to this consistency, knot vectors and degrees of Ω are directly inherited from Γ. Control points of Ω are unknown except those ones at the boundary entities, called in this section external control points. The remaining unknowns of Ω are the inner control points coordinates (the weights of the inner control points are all assumed to be 1). **Fig. 4.14** illustrates a bi-dimensional example.

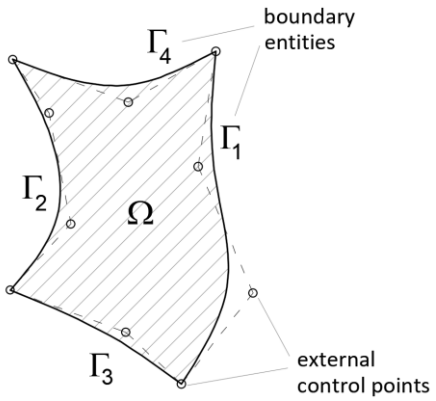


Fig. 4.14 2D NURBS surface and its boundary entities.

The strategy to find the coordinates of the inner control points is to treat the control net as a lattice structure. Prescribed displacements are applied to external nodes of one initial truss, called Λ_0, resulting in a final configuration, called Λ_1, with its external nodes at the same position as Γ. After application of these displacements, the inner nodes of Λ_1 have been moved accordingly. Nodal displacements, from Λ_0 to Λ_1, are computed by direct stiffness method (DSM) which is outlined in Appendix 4A (for further information of DSM refer to Nagarajan (2019)).

Let us define the *cell* as the space confined by four nodes (surfaces) or eight nodes (volumes). We introduce two types of bars in the lattice: rods, that joint the nodes to form the cells edges, and diagonals, that pass across the cells joining diagonally opposite nodes. **Fig. 4.15** illustrates examples in 2D and 3D. In the 3D case the diagonals run on the cell faces, but not internally. Therefore one 3D cell has 12 diagonals (6 faces x 2 diagonals/face). The 2D cells have 2 diagonals.

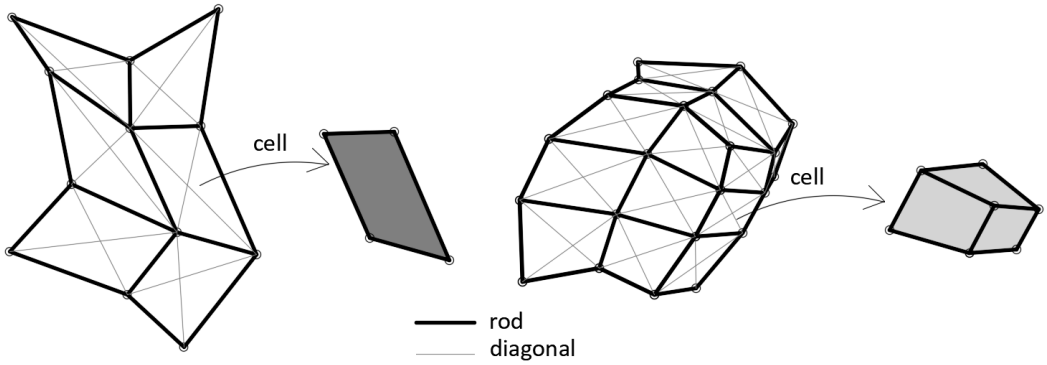


Fig. 4.15 Cells, rods and diagonals, in 2D and 3D.

Let us remark the role of the rods and diagonals within the lattice fitting algorithm. The rods restraint mostly the *volume* change, i.e. the change in lengths of the cells without varying the corner angles. By contrast, the diagonals restraint mostly the *distortion* change, i.e. the variation of the corner angles. **Fig. 4.16** (a) illustrates a volumetric change in the cell, where the rods vary their lengths but the diagonals barely change. **Fig. 4.16** (b) illustrates a distortion of the cell, where the diagonals change their lengths while the rods lengths remain the same.

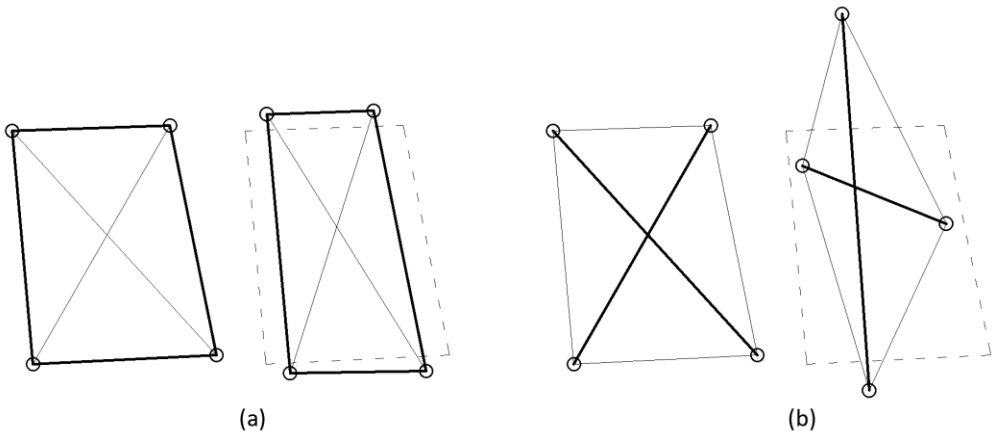


Fig. 4.16 Constraints produced by rods and diagonals.

The ratio between the stiffness of the rods and the diagonals dictates the prevalent variation in the lattice after applying the DSM: the stiffer the diagonals the larger volumetric change and lesser distortion. Conversely, the stiffer the rods the larger distortion and lesser volumetric change. Hence depending on that ratio, the shape of the lattice varies, achieving different qualities (the quality is measured by the Jacobian of the obtained mesh, as show in afterwards in this section).

Previous to the detailed explanation of the lattice fitting algorithm, a couple of examples are provided in **Fig. 4.17** and **Fig. 4.18**, which illustrate 2D and 3D cases respectively. The sequence shown in these figures is:

(a) region confined by boundary entities;
(b) quadrilateral/cuboid fitted to control points of boundary entities (thick lines);
(c) initial truss (Λ_0) with rods in thick and diagonals in thin lines;
(d) prescribed displacements (arrows);
(e) final truss (Λ_1) to use as Ω control points.

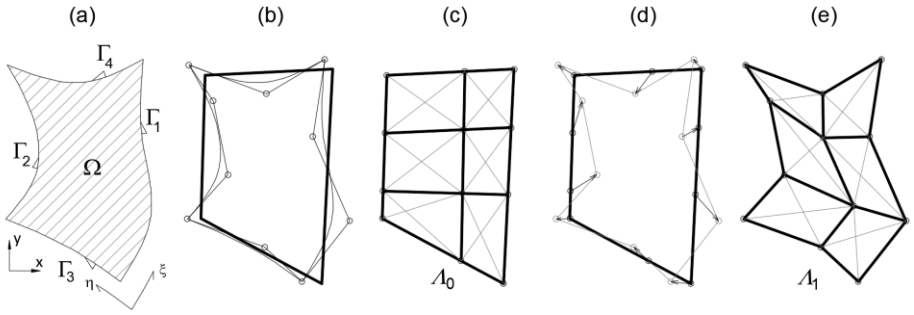


Fig. 4.17 Lattice fitting algorithm in 2D.

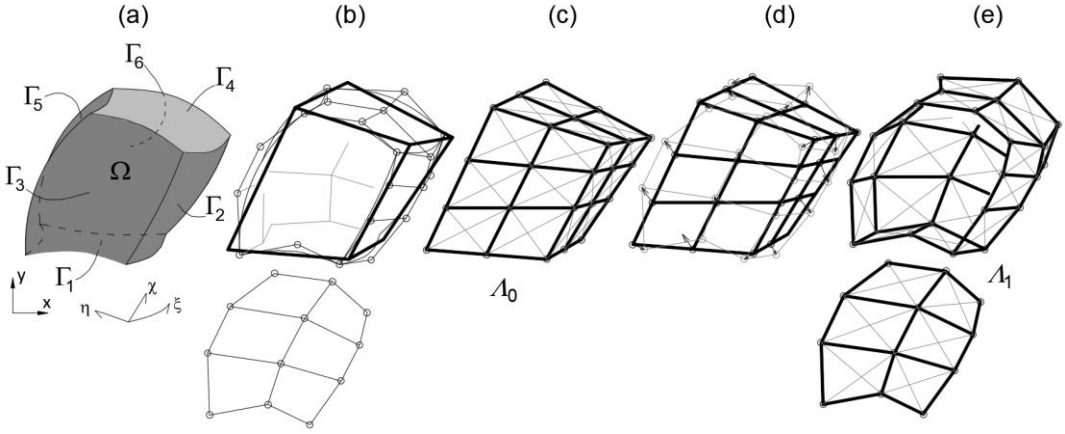


Fig. 4.18 Lattice fitting algorithm in 3D. Control points of Γ_3 are moved to visualize the interior of the truss.

In view of the examples, two questions arise:

- How to define the initial lattice (Λ_0)?
- How to select the stiffness of rods and diagonals to arrive to the best parameterization?

Next sub-sections detail the algorithm and provide the answers for these two questions.

Arrangement of outer entities Γ

For *plane lattice fitting* there are four boundary curves (Γ) arranged by parameter directions in pairs 1,2 and 3,4 as per **Fig. 4.17** (a). Recall from 4.2.1 that those curves achieved NURBS consistency in the GPS generation.

For *volume lattice fitting* there are six boundary surfaces (Γ) arranged as per the GPS, and also have NURBS consistency. See **Fig. 4.18** (a).

Initial lattice (Λ_0)

For surfaces case, one quadrilateral with one straight line per Γ curve is computed (**Fig. 4.17** (b)). Each quadrilateral line is fitted by the Least Square Method (LSM)[22] to the control points of each Γ curve. The average of the Γ control points $\boldsymbol{x}_{avg}$ is to belong to the fitted line and the function to minimize is the normal distance to each Γ control point. The LSM delivers the direction of the line $\boldsymbol{v}$, that is defined as (4.1) where is t a free parameter.

$$\boldsymbol{l}(t) = \boldsymbol{x}_{avg} + \boldsymbol{v}\, t \tag{4.1}$$

To set the initial truss (Λ_0), nodes are placed first on edge lines of the quadrilateral lines, they are the so-called **external nodes**. The number of nodes in each direction equals to the number of control points of Γ curves in each direction. They are spaced proportionally to the corresponding control points spacing, measured on the control polygon. Then, the distribution of the control points is reflected in the initial lattice. Internal nodes of the quadrilateral are obtained by intersection of lines between opposite external nodes. Once all the nodes are placed, the rods and diagonals are set (**Fig. 4.17** (c)).

For 3D case the process is analogous. The initial lattice is a cuboid grid of nodes that equals the number of control points of Γ surfaces in each direction. The face of each cuboid is fitted by LSM[23] using the average of the Γ surface control points $\boldsymbol{x}_{avg}$ as a point of the plane, and the function to minimize is the normal distance to each point of Γ (**Fig. 4.18** (b)). The LSM delivers the normal direction of the plane $\boldsymbol{n}$, defined as parametric equation (4.2), where $\boldsymbol{a}$ is any location in the plane.

$$\boldsymbol{n} \cdot (\boldsymbol{x}_{avg} - \boldsymbol{a}) = 0 \tag{4.2}$$

[22] See Appendix 4B.
[23] See Appendix 4B.

Nodes on the cuboid edges are posed proportionally to the spacing of the control points of the Γ surface edges, considering lengths measured on the control polygon. Nodes on the cuboid faces are obtained by intersection of lines between opposite edge nodes. Nodes within the cuboid are obtained by location of closest point to three lines between opposite face nodes. These points are computed by LSM. With all the nodes localized the rods and diagonals can be defined (**Fig. 4.18** (c)).

Computation of control points location for final configuration (Λ_1)

Once the initial lattice Λ_0 is set, the vectors from its **external nodes** to the Γ control points are imposed displacements in the DSM (**Fig. 4.17** (d) and **Fig. 4.18** (d)). The output of the DSM is the vector of displacements of all the nodes. The final lattice configuration Λ_1 is obtained after adding to the nodes of Λ_0 the computed displacements (**Fig. 4.17** (e) and **Fig. 4.18** (e)). These displaced nodes will became the control points of Ω.

The result depends on the bars properties: length, cross sectional area and elastic modulus (E). Length is given by initial configuration, area is assumed equal to 1 throughout and elastic modulus is to be different between rods (E_r) and diagonals (E_d). The ratio E_r/E_d is varied within certain range to achieve the maximum quality for the parametrization of Ω.

Measurement of quality of parametrization

The condition for a valid parametrization is that the domain has no self-intersection, i.e. there is no point where the Jacobian determinant becomes zero. Let us define the condition number of one Jacobian J as follows:

$$f_{cond} = \|J\|_F \, \|J^{-1}\|_F \tag{4.3}$$

Where $\|J\|_F$ is the Frobenius norm of the Jacobian[24]. Condition number is equal to d for an equilateral-orthogonal parametrization[25], which is the best case, otherwise is greater. The higher condition number the less quality: more distortion and/or stretching.

To measure the quality of the resultant parametrization the condition number is measured on the cells formed by the control net and not on the NURBS entity itself. This simplification enhances the efficiency and still measures the quality. Jacobian within the ath cell is computed considering the cells as linear quadrilaterals (**Fig. 4.19**) or linear hexahedrons (**Fig. 4.20**), with equations (4.4) and (4.5) for surfaces an volumes respectively. The shape functions and the derivatives of these elements are detailed in Appendix 2E. The Jacobians are calculated at parent coordinate $\tilde{\boldsymbol{\xi}} = \mathbf{0}$.

[24] See Appendix 2F.
[25] d is the number of dimensions 2 and 3 for surfaces and volumes respectively.

$$J^a = \begin{bmatrix} x^a{}_{,\tilde{\xi}} & x^a{}_{,\tilde{\eta}} \\ y^a{}_{,\tilde{\xi}} & y^a{}_{,\tilde{\eta}} \end{bmatrix} \tag{4.4}$$

$$J^a = \begin{bmatrix} x^a{}_{,\tilde{\xi}} & x^a{}_{,\tilde{\eta}} & x^a{}_{,\tilde{\chi}} \\ y^a{}_{,\tilde{\xi}} & y^a{}_{,\tilde{\eta}} & y^a{}_{,\tilde{\chi}} \\ z^a{}_{,\tilde{\xi}} & z^a{}_{,\tilde{\eta}} & z^a{}_{,\tilde{\chi}} \end{bmatrix} \tag{4.5}$$

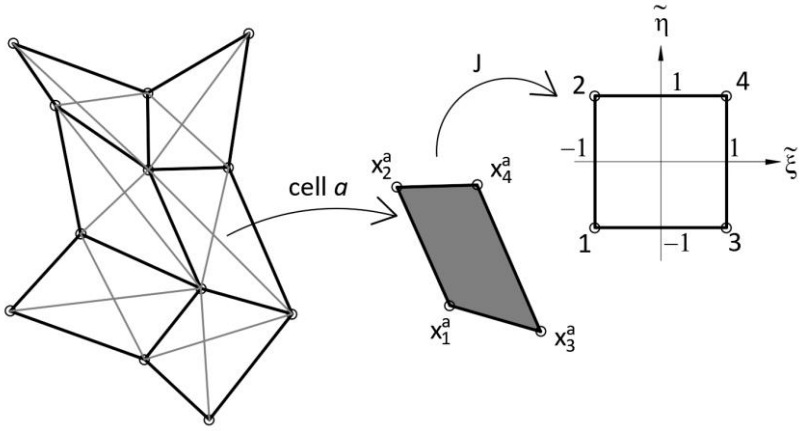


Fig. 4.19 Cell formed from 2D-truss and its parent space (linear quadrilateral).

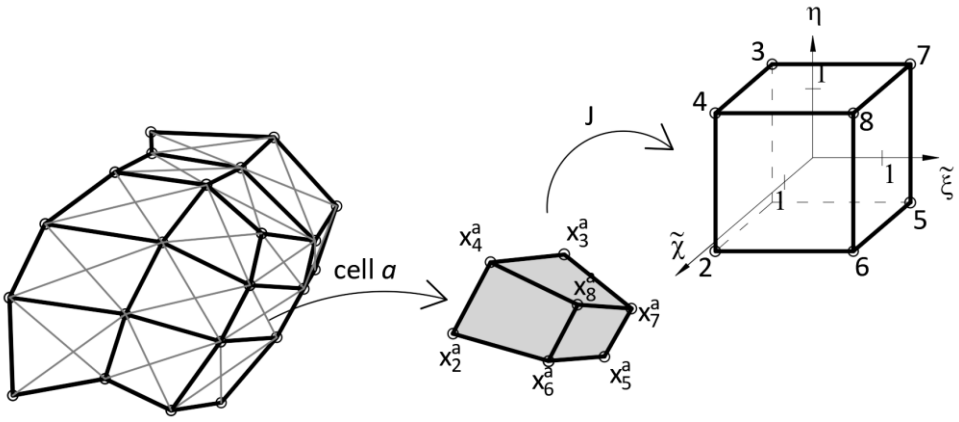


Fig. 4.20 Cell formed from 3D-truss and its parent space (linear hexahedron).

In practise, a number of initial lattices (Λ_0) are analysed varying the ratio E_r/E_d from 5 to 0.5. For each resultant truss configuration the Jacobian and condition numbers are calculated in each cell. The quality of the lattice is measured by two parameters:

- local average of condition number (f_{cond}^{ave2}), that averages the two worst cells (the two greatest condition numbers);
- global average that computes condition number of all cells of the truss (f_{cond}^{ave}).

The candidate configurations must have the Jacobian determinant positive for all the cells. The selected configuration possesses the lowest local average (f_{cond}^{ave2}). Global average (f_{cond}^{ave}) is used in case two or more configurations have the same f_{cond}^{ave2}.

Examples

Plane lattice fitting:

Fig. 4.21 left illustrates the control points of boundary curves Γ, each pair of one colour, and the fitted quadrilateral by LSM. On the right-hand side the truss initial configuration Λ_0 is shown, diagonals are not drawn for clarity. The nodal spacing at contours is not regular since it reflects the contours control points relative positions.

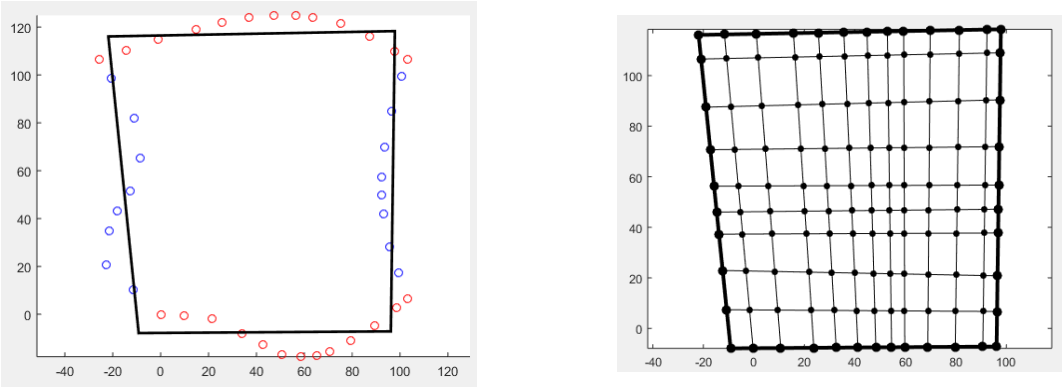


Fig. 4.21 Quadrilateral fitted to contours (left) and initial lattice (right).

Fig. 4.22 left shows all the resultant lattices by varying the rods/diagonals ratio of elastic modulus as per **Table 4.2**. Selected truss is shown at the right hand side of that figure, which corresponds to the hatched row in **Table 4.2**.

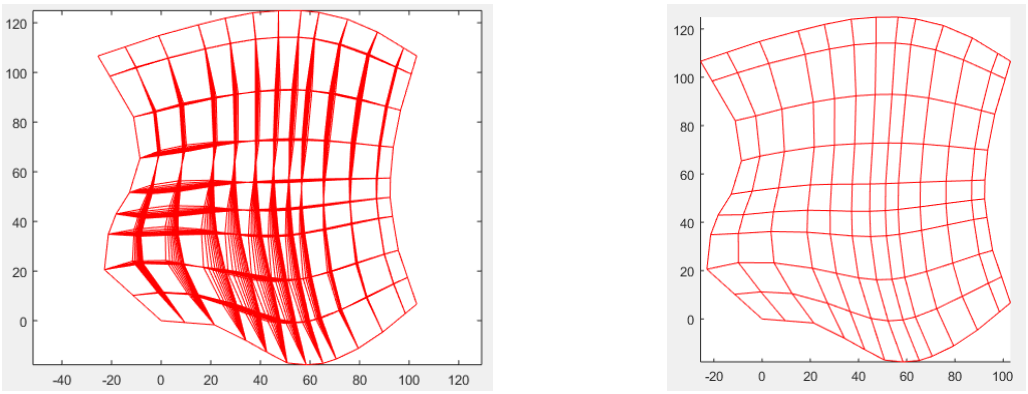


Fig. 4.22 Superposition of all the solutions (left) and selected configuration (right).

Table 4.2 Local and total average of condition number varying ratios of E.

Er	Ed	Er/Ed	f_{cond}^{ave2}	f_{cond}^{ave}
100	20	5.00	3.84	2.50
100	40	2.50	3.84	2.49
100	60	1.67	3.84	2.48
100	80	1.25	3.96	2.49
100	100	1.00	4.03	2.49
100	120	0.83	4.11	2.50
100	140	0.71	4.18	2.51
100	160	0.63	4.25	2.51
100	180	0.56	4.32	2.52
100	200	0.50	4.42	2.53

Volume lattice fitting:

Fig. 4.23 left illustrates the control net of boundary surfaces Γ and the fitted cuboid by LSM (only 3 faces are depicted for clarity). At the right-hand side the nodes at the edges are shown alongside with the edges control points. These nodes spacing reflect the distribution of the control points.

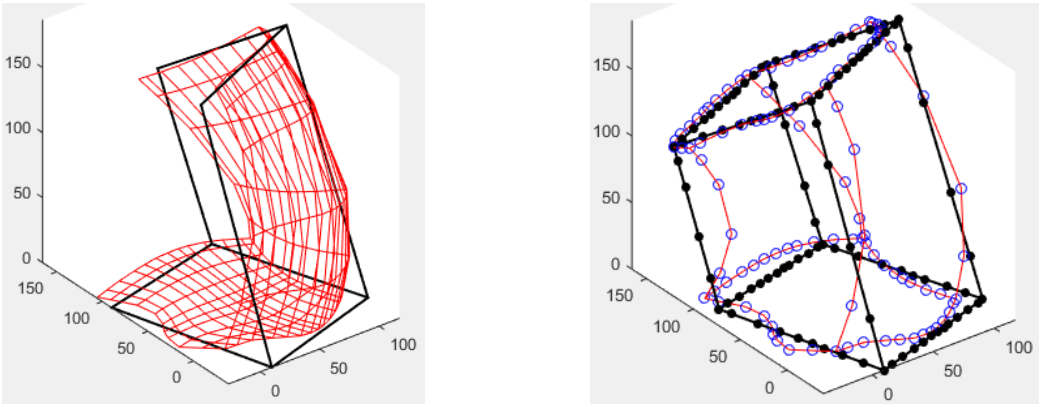


Fig. 4.23 Three planes of the cuboid fitted to Γ surfaces (left) and nodes at edges (right).

The rods of one face of the initial truss Λ_0 are illustrated in **Fig. 4.24** left, and the whole lattice at the right-hand side. The diagonals are not shown for clarity.

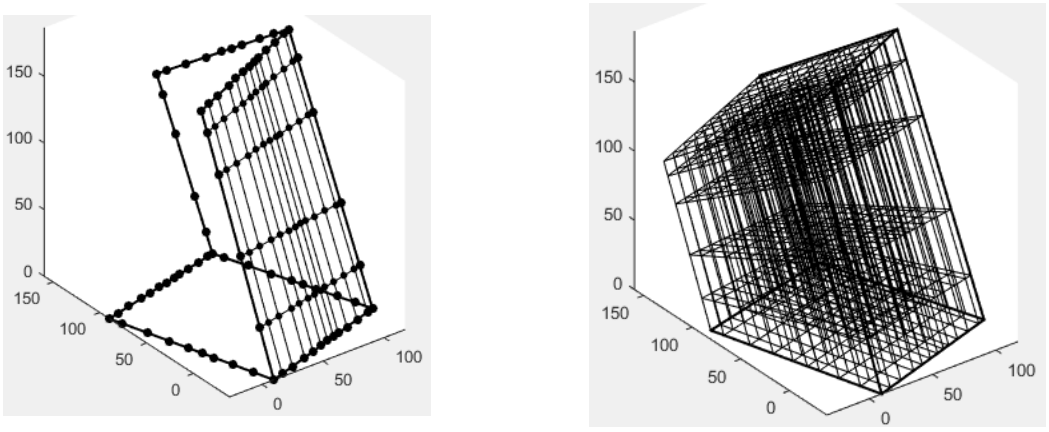


Fig. 4.24 Three planes of the fitted cuboid (left) and whole initial lattice.

Fig. 4.25 left shows all the resultant trusses by varying the rods/diagonals ratio of elastic modulus as per **Table 4.3**. Only one internal set of bars is shown (located at third node in the parameter direction 1). At the centre the worst case and on the right-hand side the selected lattice, that corresponds to the hatched row of **Table 4.3**. The resultant lattice is depicted in **Fig. 4.26** (diagonals are not drawn for clarity).

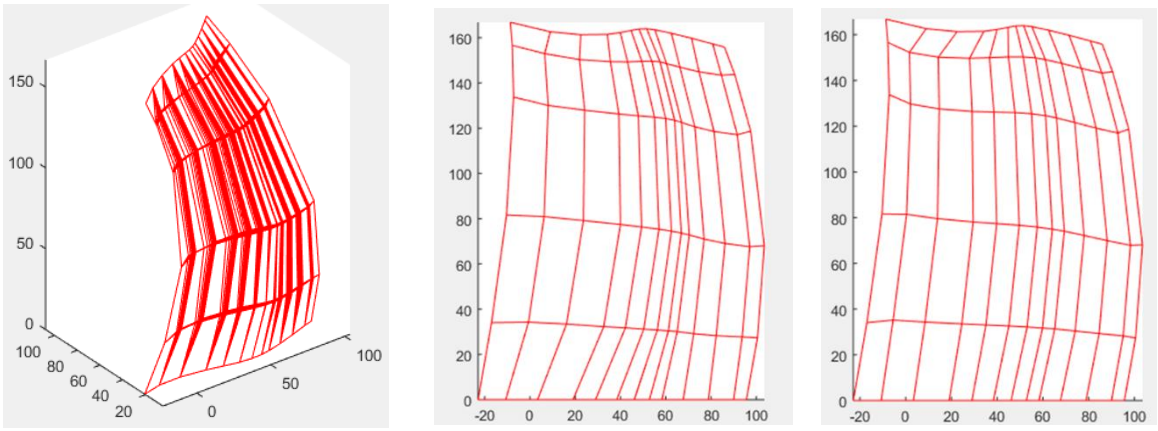


Fig. 4.25 Superposition of all the solutions (left), worst configuration (centre) and selected configuration (right), for the lattice at third node in ξ direction.

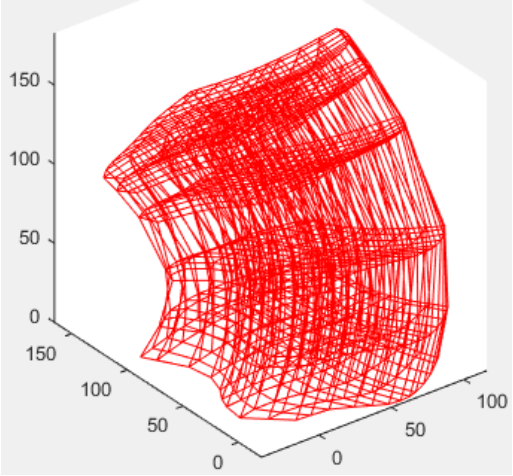


Fig. 4.26 Resultant control points of parametrized Ω.

Table 4.3 Local and total average of condition number varying ratios of E.

Er	Ed	Er/Ed	f_{cond}^{ave2}	f_{cond}^{ave}
100	20	5.00	12.58	5.80
100	40	2.50	12.84	5.78
100	60	1.67	12.99	5.78
100	80	1.25	13.07	5.78
100	100	1.00	13.54	5.79
100	120	0.83	13.98	5.81
100	140	0.71	14.40	5.82
100	160	0.63	14.81	5.83
100	180	0.56	14.87	5.85
100	200	0.50	15.35	5.86

4.3. Identification of surfaces

The information of trimming and boundary surfaces is retrieved from *B.igs*. On one hand, the *B.igs* contains the trimmed patch, on the other hand the boundary surfaces. For the discretization of the trimmed patches (recall stage C of **Fig. 4.1**) the trimming surfaces need to be identified and extracted. For the imposition of constraints (recall stage D of **Fig. 4.1**) the patches where each boundary surface lies on need to be identified, i.e. the boundary surfaces must be allocated to one patch. In addition, those boundary surfaces that are shared by two patches need to be identified since they will become coupling surfaces.

Let us call the *trimmed patch shells* (TPS) to the object that contains the IGES information for the trimmed patch. There is one TPS per patch. Differently to GPS, the TPS inherits the surfaces features as they come from the IGES file without improving them. This is because TPS is not used for parametrization as GPS was. No further details on TPS are given since the process merely transfers the information contained in *B.igs*.

For the identification of trimming surfaces, both, the GPS (section 4.2.1) and the TPS are used. For the identification of boundary surfaces only the TPS is used. This identification process requires one sample point that must lie within the computable surface. The selection of the sample point is not trivial since the surface may be trimmed by contour loops, being necessary a trial searching. **Fig. 4.27** shows one example with trial points valid and non-valid to become the sample point. At left-hand side the surface parameter space is represented. At the right-hand side of **Fig. 4.26** the physical space shows only the valid points.

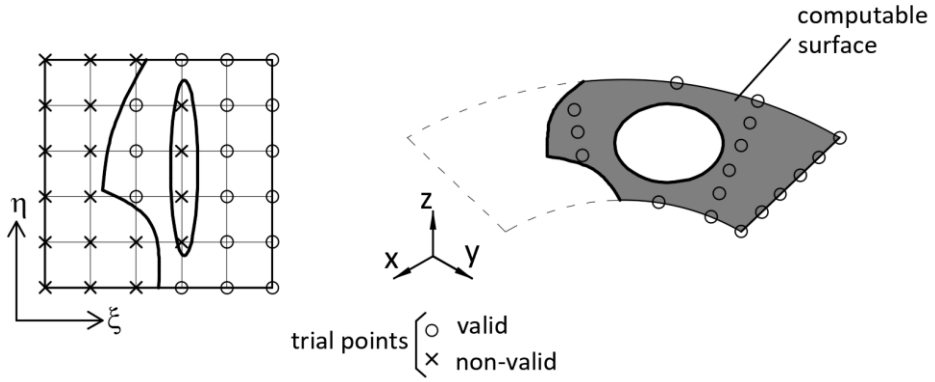


Fig. 4.27 Trial points on a surface.

The trial points are set in a net on the surface background (in this work the step of the net in the parameter space is initially 0.2). If one of those trial points lies within the computable surface it is chosen as sample point and the search stop. If none of the net trial points lie within the computable surface, the net is re-calculated halving the step.

4.3.1. Identification of trimming surfaces

To find out if one surface of the TPS is trimming, the sample point of the surface is projected to the GPS. If the distance from the sample point to the projection is zero, i.e. both locations coincide, the surface lies on the GPS and therefore it is not trimming surface. Otherwise it is a trimming surface and is identified to form part of the trimmed solid discretization in stage C (recall **Fig. 4.1**). One example is provided in **Fig. 4.28**, where surface 1 is not trimming since its projection on GPS coincides with the sample point. By contrast, the projection from the sample point of surface 2 is not coincident with such point and therefore it is trimming surface.

Here we bring the importance of the construction sequence explained in section 3.3, where it was stated that the trimmed patch must hold the same position as the gross patch in CAD. This spatial coincidence allows the identification described.

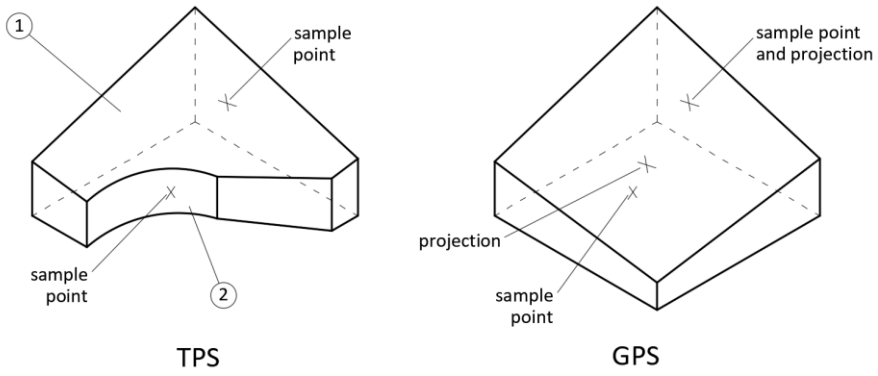


Fig. 4.28 Identification of trimming surfaces.

4.3.2. Identification of boundary surfaces

In the the *B.igs* there is no relationship between the boundary surfaces and the patches. However the former are used to apply constraints to the patches. Therefore the patch where each boundary surface lies needs to be identified. The sample point of each boundary surface is projected to each face of the TPS. The face where the projection coincides with the sample point is the one attached to the surface, and hence its patch.

Those boundary surfaces attached to two patches are coupling surfaces. By convention, the first patch detected will be the master patch. **Fig. 4.29** shows the boundary surface 1, which is coupling, and the boundary surface 2 that is attached to patch 2.

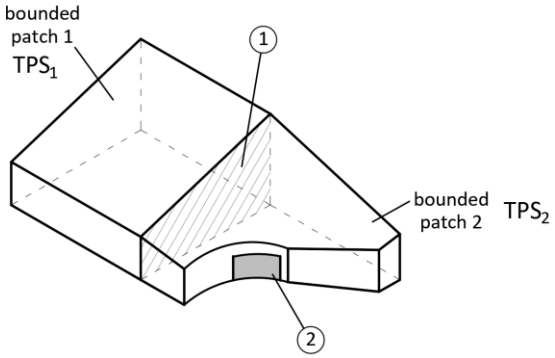


Fig. 4.29 Identification of boundary surfaces.

4.4. Relation with the code

The generation of solid NURBS from the IGES files is contained in the routine called `i3100_IGES`, which has two routines inside:

- The `gp3110_readGP` routine generates the parametrized solid from *G.igs*. The solid is stored in the variable `FgPatch`. The number of patches is stored in `Npa`. There are other auxiliary variables generated:

- `gFace`: contains the information of the faces of the GPS.
- `gShell`: is the GPS object.
- `gbShell`: is a copy of the GPS object, but with its structure modified to facilitate the comparison against surfaces.

The most relevant functions within `gp3110_readGP` are:

- `i3150_IGEStoMat`: transfers IGES text information to numerical arrays.
- `gp3171_curvesHomog`: homogenizes GPS curves.
- `gp3175_surfHomog`: homogenizes GPS surfaces.
- `gp3184_soSandwich`: parametrizes solid by the sandwich algorithm.
- `gp3140_soLattFitt`: parametrizes solid by the lattice fitting algorithm.

- The `gp3120_readBE` routine identifies the trimming and boundary surfaces using the *B.igs* file and the GPS. This information is stored in the variable `bEnti`. The number of boundary surfaces is stored in `Nbe`. `tFace` and `tCurv` contain the information of the trimming surfaces.

  To identify the surfaces the `gbShell` (obtained in `gp3110_readGP` and corresponding to the GPS) and `bShell`, corresponding to TPS, are used. The most relevant functions within `gp3120_readBE` are:

  - `i3150_IGEStoMat`: transfers IGES text information to numerical arrays.
  - `be3122_bShell`: creates the `bShell`.
  - `i3170_sVSshell`: compares surfaces against `gbShell` or `bShell`.

4.5. Summary of the chapter

Solid parametrization is the objective ***c*** of this thesis. This chapter addresses this objective since the techniques for solid parametrization are presented, including the novel lattice fitting approach. The objective ***b*** of the thesis, translation of IGES files, is also introduced.

In addition, trimming and boundary surfaces are identified. Trimming surfaces will be used in stage C (Chapters 5 and 6) to determine the limit of the solid discretization. Boundary surfaces are used to prescribed constraints in stage D (Chapter 7).

5. **Approximation to trimming surfaces**

This chapter describes how the trimming surfaces, which contribute to the shape of the trimmed domain, are approximated. **Fig. 5.1** shows one portion of the main diagram, extracted from section 3.2. This chapter covers part of stage C, where the patches are discretized. The inputs for stage C are the trimming surfaces and the solid, both coming from stage A (Chapter 4). In stage B the features of the solid and trimming surfaces were allocated (this task is not relevant in the theoretical approach) and the skins for representation were created by *QIT* algorithm (refer to Appendix 10B).

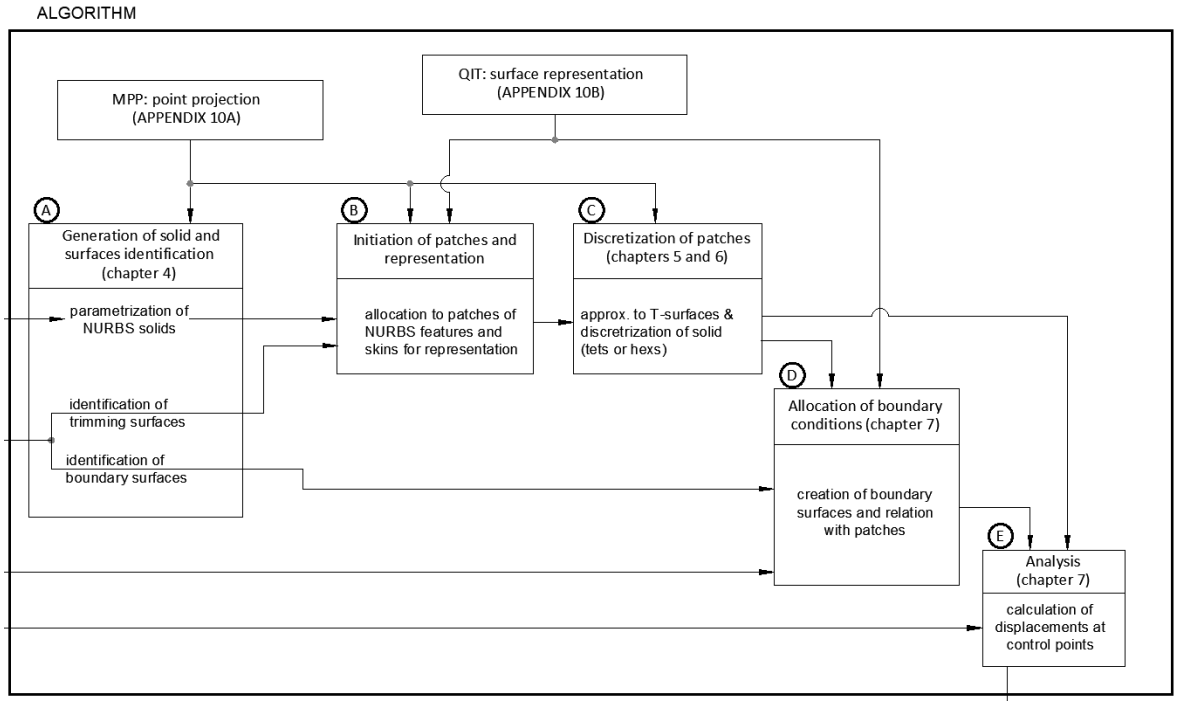


Fig. 5.1 Extraction of main scheme.

When computing the stiffness matrix of a trimmed patch, the integration limits are of two types: trimming surfaces and patch bounded faces (see **Fig. 5.2**). The latter are the patch limits themselves, trimmed by contours (recall section 3.1.1). The trimming surfaces, which lie within the gross patch, have an image in the patch parameter space that is unknown and need to be approximated. The better the approximation the closer to the exact integration limits, and the more accurate the integration. Therefore, achieving accuracy in this approximation is crucial for getting acceptable results in the analysis.

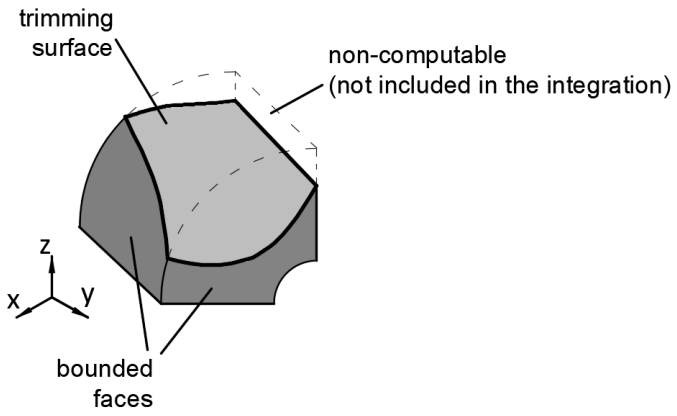


Fig. 5.2 Integration limits of trimmed patch.

This chapter is structured as follows: section 5.1 defines the general concepts and overviews the approximation to trimming surfaces. Section 5.2 justifies why the surface must be approximated in the patch parameter space and not in the physical space. Sections 5.3 and 5.4 explain the nodal insertion to approximate the surface by triangles. Section 5.6 shows the triangulation of the inserted nodes. One example is presented in section 5.5. The main routines involved in the code are mentioned in section 5.7. Finally, section 5.8 summarizes the most important ideas of the chapter.

5.1. Overview

The term *patch* will refer to the solid domain that may be trimmed by a set of surfaces called T-*surfaces*. These surfaces are contoured by curves that form loops enclosing the computable T-surface region. These curves are called C-curves and may trim the T-surface or lie on its parameter limits[26]. One surface has at least one loop of C-curves, but may have more to define internal regions to exclude. This configuration of surfaces and contour loops is inherited from the B-rep arrangement (section 2.2.3).

The patch, its T-surfaces and C-curves meet at the physical space, but each entity has its own parameter space. **Fig. 5.3** illustrates one example with two T-surfaces trimming the patch, being surface 1 with two loops of C-curves.

[26] Those curves where referred as *edging contours* in section 2.2.3.

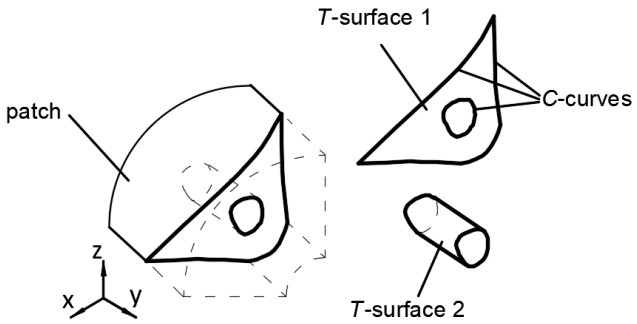


Fig. 5.3 Entities involved in trimmed patches, represented in the physical space.

The image of one T-surface in the patch parameter space is called Q-surface and is unknown. The Q-surface is assumed a NURBS surface with the same parameter space than T-surface. Both parameter spaces, $\hat{T}$ and $\hat{Q}$, are assumed coincident[27]. The images of C-curves in the patch parameter space are called K-*curves* and in the surface parameter space L-*curves*. Both curve images are unknown but are assumed NURBS curves with parameter spaces $\hat{K}$ and $\hat{L}$ coincident with the curve parameter space $\hat{C}$.

The involved spaces are listed below and **Fig. 5.4** illustrates one example with two trimming surfaces. The parameter spaces for surface 1 and curve 1 are shown. Note the T and Q-surfaces parameter spaces coincide. Equally, the C, K and L curves share the same parameter space.

Patch physical space:	P
Patch parameter space:	$\hat{P}$
T-surface physical space:	T
T-surface parameter space:	$\hat{T}$
Q-surface physical space:	Q (image of T-surface in $\hat{P}$)
Q-surface parameter space:	$\hat{Q}$
C-curve physical space:	C
C-curve parameter space:	$\hat{C}$
K-curve physical space:	K (image of C-curve in $\hat{P}$)
K-curve parameter space:	$\hat{K}$
L-curve physical space:	L (image of C-curve in $\hat{T}$)
L-curve parameter space:	$\hat{L}$

[27] This coincidence will make sense in sections 5.3 and 5.4.

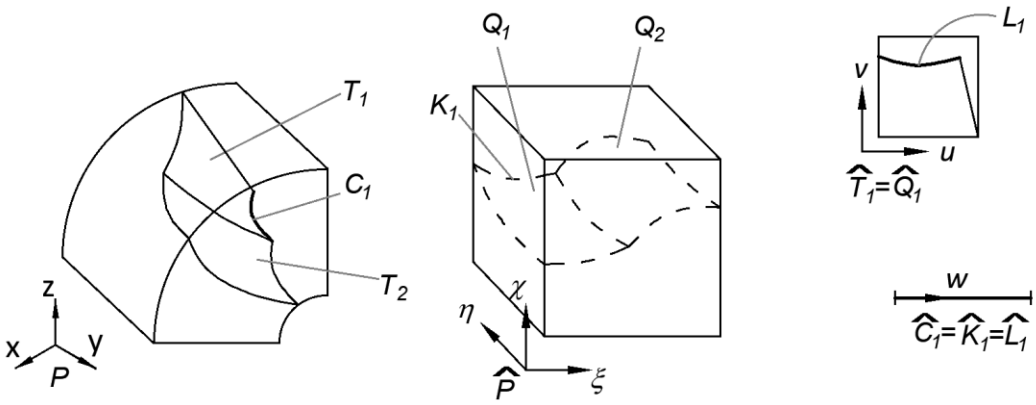


Fig. 5.4 Spaces involved in a trimmed patch.

5.1.1. Nomenclature used

To reduce the nomenclature, the surface parameter space will be referred as $\hat{Q}$ and the curve parameter space as $\hat{K}$. The coordinates in each space are referred differently as shown in **Table 5.1** and **Fig. 5.4**. The object of this chapter is to find accurate approximations to Q-surfaces and K-curves.

Table 5.1 Coordinates names in the different spaces.

Space	***Symbol***	***Coordinates***
Physical	P	x, y, z
Patch parameter	$\hat{P}$	ξ, η, χ
Surface parameter	$\hat{Q}$	u, v
Curve parameter	$\hat{K}$	w

5.1.2. Quadratic approximation to Q-surfaces and K-curves

The approximations to Q-surfaces and K-curves are called $\bar{Q}$-surfaces and $\bar{K}$-curves respectively. The left-hand side of **Fig. 5.5** shows the mapping from the patch parameter space to the physical space (Q to T direction), that has analytical equation. The right-hand side represents the opposite direction (T to Q), where we most we can do is estimating isolated points of Q-surface by point projection[28]. The second is the case that this chapter deals with.

[28] The point projection technique used in this thesis is in Appendix 10A.

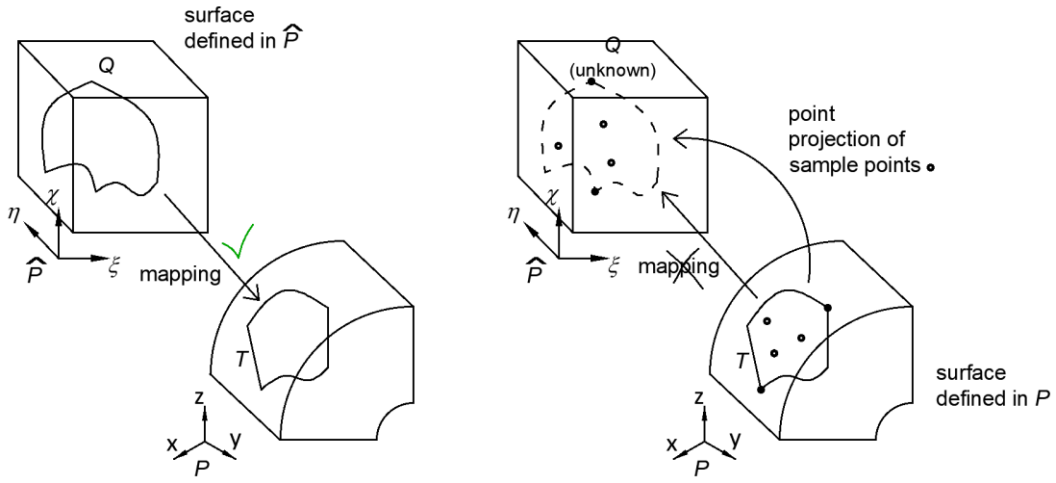


Fig. 5.5 Mapping from Q to T and its inverse, that is carried out by point projection.

The approximation to Q uses a set of points located on T-surface, called nodes, and projects them back to the patch parameter space. The surface between the nodes is interpolated by quadratic triangles becoming the $\overline{Q}$-surface. The same explanation applies to curves, where $\overline{K}$ is the approximation with quadratic segments to K-curve.

Quadratic approximation is selected because it keeps the number of required nodes of triangles lower than linear approximation, as shown in Appendix 5C. Cubic and higher order approximations lead to more complex mappings from parent space that, in the author´s opinion, increase in excess the complexity of the algorithm.

5.1.3. Types of nodes

In $\overline{K}$-curves, the nodes at extremes of segments are called end-nodes and the central ones mid-nodes. In $\overline{Q}$-surfaces the edges of the triangles are called lines. Nodes at vertexes of triangles are called end-nodes and at mid location of lines are called mid-nodes. Since surfaces are contoured by $\overline{K}$-curves, the nodes that lie on these curves are called contour nodes and the rest inner nodes. **Fig. 5.6** illustrates this nomenclature, where a $\overline{Q}$-surface is represented and one of its contour curves is extracted at the left-hand side.

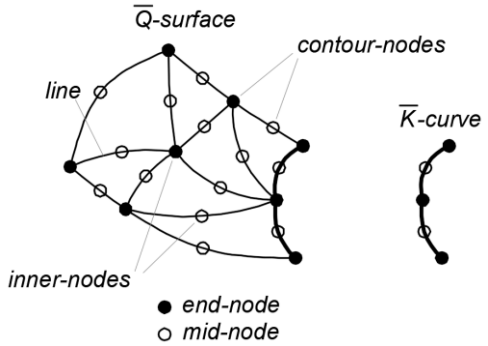


Fig. 5.6 Types of nodes for the quadratic approximations.

5.1.4. Error measurement

Let $\boldsymbol{E}_2$ be the error vector from one point from $\bar{Q}$ to its closest location on Q, i.e. the distance between both surfaces in the patch parameter space (see **Fig. 5.7** left). We define the error as the norm $E_2 = \|\boldsymbol{E}_2\|$. The sub-index 2 refers to quadratic approximation, which is used in this work. The error at any location must be equal or less than the admissible error (E_{adm}) which is computed as:

$$E_{adm} = \frac{tol_S}{100} D \tag{5.1}$$

Where tol_S is a percentage and D is the length of the cube diagonal of the patch parameter space. In this work, typical values of tol_S vary between 0.1 and 1.0 %. **Fig. 5.7** illustrates the error at one location and the diagonal in the patch parameter space.

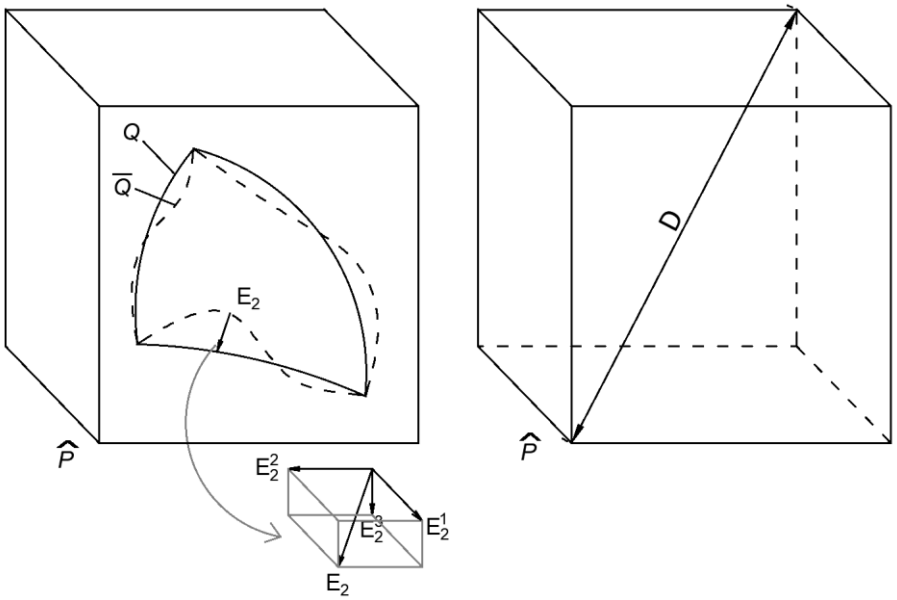


Fig. 5.7 Error in the patch parameter space (left) and diagonal of this space (right).

5.2. Surface to approximate

One may doubt which is the best surface to approximate, T or Q? This section justifies why the surface to fit is Q arguing two reasons.

First, if we approximate in the parameter space to the Q-surface, the shape in the physical space is controlled by the connection from $\bar{Q}$ to $\bar{T}$ via NURBS mapping. By contrast, if we approximate in the physical space to the T-surface, the shape in the parameter space is out of control because there is no direct connection from $\bar{T}$ to $\bar{Q}$ (recall **Fig. 5.5**).

Second, when approximating one surface with triangles, the number of required nodes to keep the error below E_{adm} depends on the surface shape[29], since T and Q do not have the same shape the required number of nodes and location for each of them is different. Approximating to T, instead Q, can bring situations where the number of nodes is insufficient or redundant.

Next two examples illustrate the convenience of approximating to Q-surface instead T-surface. Bi-dimensional domains and linear interpolation are used for simplicity, but the idea is the same for three-dimensional domains with quadratic interpolation.

In the first example T-surface is flat and Q-surface is curved (see **Fig. 5.8**). If we approximate to T-surface, only two nodes are required. Therefore, during the integration, the volume considered is way different from the exact (**Fig. 5.8** top), introducing a large error. By contrast, approximating to Q-surface inserts as many nodes as needed to control the error and the approximated shape will be mapped to the physical space (**Fig. 5.8** bottom).

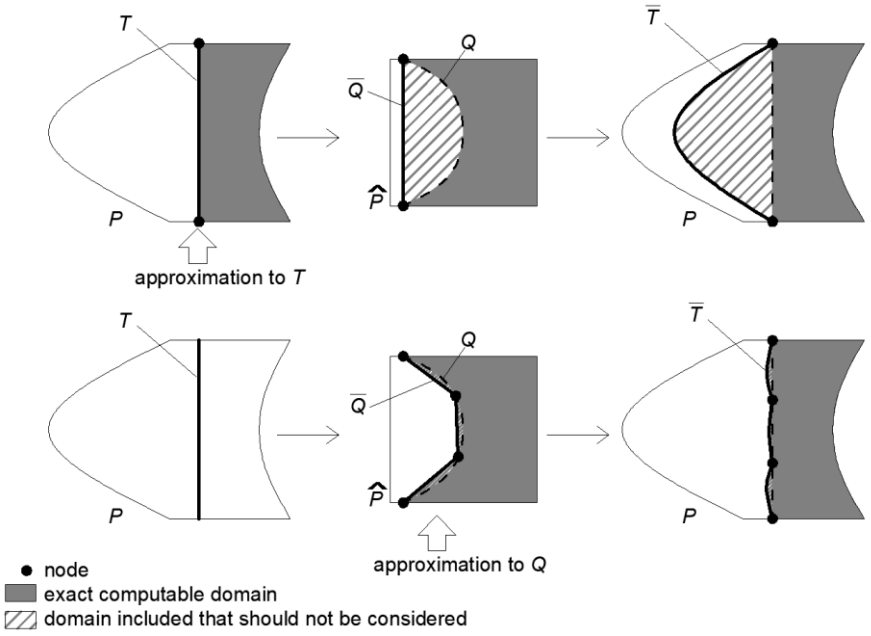


Fig. 5.8 Curved Q-surface: approximation to *T* (top) and to *Q* (bottom).

In the second example T-surface is curved and Q-surface is flat (see **Fig. 5.9**). If we approximate to T-surface, a greater number of nodes are required than for Q-surface (see **Fig. 5.9** top). The volume to integrate will be the exact (because Q is a straight line, linear approximation fits with no error), but many of the inserted nodes are redundant. By contrast, approximating to Q-surface inserts less nodes and still the volume to integrate is the exact (see **Fig. 5.9** bottom).

[29] In particular, for quadratic triangles, number of nodes depends on the third derivatives of the surface (see section 5.3).

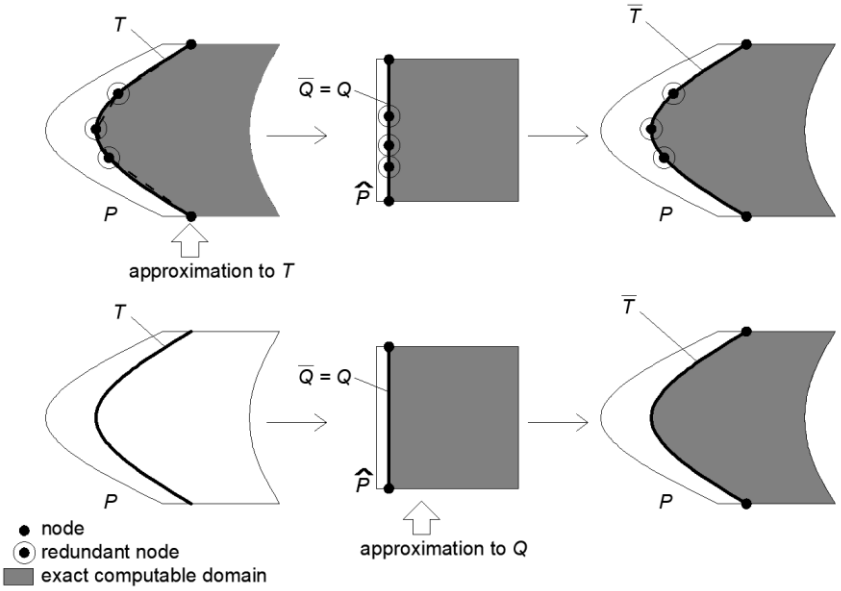


Fig. 5.9 Flat Q-surface: approximation to *T* (top) and to *Q* (bottom).

5.3. Error estimation and admissible nodal distances

As mentioned earlier, the approximations to trimming surfaces and their contour curves are done by piece-wise entities spanning between nodes. This section explains how to insert those nodes.

5.3.1. Curves

The K-curve, which maps from its parameter space $\widehat{K}$ to the patch parameter space $\hat{P}$, is approximated by quadratic segments. The error vector at location a, whose coordinate in $\widehat{K}$-space is w_a, is estimated as equation (5.2), that is taken from Appendix 2A.

$$E_{2\,j} \leq \frac{K_{,www}\,(w_\beta)_j}{3!}\,(w_a - w_0)(w_a - w_1)(w_a - w_2) \tag{5.2}$$

Where the sub-index j indicates one component of the error vector (recall section 5.1.4). w_0, w_1, w_2 are locations of the nodes in the curve parameter space. The third derivative is calculated at w_β that lies between w_0 and w_2 such that maximises the error. **Fig. 5.10** shows one example. The distance between nodes in $\widehat{K}$-space is called nodal step and is represented by s.

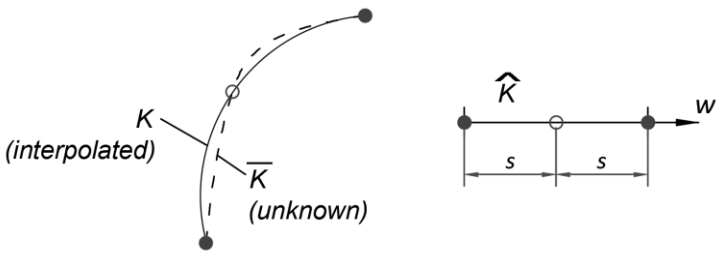


Fig. 5.10 Interpolation of NURBS curve with quadratic polynomial.

To handle with (5.2) two simplifications are done:

- w_β lies at the mid-point of the segment, then the derivative is referred as $K^m{}_{,www\,j}$
- w_a lies at $0.211\,L$, with $L = w_2 - w_0$, therefore the product of nodal increments is equal to $0.385\,s^3$ (this assumption maximizes the error as demonstrated in Appendix 2A).

Then, the error vector is re-expressed as (5.3) and its norm calculated as (5.4).

$$E_{2\,j} \leq \frac{1}{15.6}\; s^3\; K^m{}_{,www\,j} \tag{5.3}$$

$$E_2 \leq \frac{1}{15.6}\; s^3\; \left\| K^m{}_{,www\,j} \right\| \tag{5.4}$$

The maximum nodal step to keep the error equal or less than E_{adm} is called s_{adm} and is calculated as (5.5), that is obtained from (5.4) equating E_2 to E_{adm}.

$$s_{adm} = \left(\frac{15.6\; E_{adm}}{\left\| K^m{}_{,\xi\xi\xi\,j} \right\|} \right)^{\frac{1}{3}} \tag{5.5}$$

With this distance the curve can be divided into segments of length equal or less than $2s_{adm}$ in $\widehat{K}$-space. Therefore, considering the mid-nodes, the nodal space remains equal or less than s_{adm} and the error is equal or less than E_{adm}.

The third derivative $K^m{}_{,\xi\xi\xi}$ has no analytical expression since K is unknown, hence it is approximated by finite divided differences (FDD) as equation (5.6), where k_{aj} is the jth component of the coordinate in the $\widehat{P}$-space for the ath-sample point and h the step used for FDD (see Appendix 2A for more details on FDD).

$$K^m{}_{,\xi\xi\xi\,j} \cong \frac{k_{3\,j} - 3\;k_{2\,j} + 3\;k_{1\,j} - k_{0\,j}}{h^3} \tag{5.6}$$

To use (5.6) we need four sample points on the K-curve (in the $\widehat{P}$-space), whose spacing in $\widehat{K}$-space is h and are centred about w_m. These points are set in $\widehat{K}$-space, then mapped to the physical space, and their $\widehat{P}$ coordinates are estimated by point projection. **Fig. 5.11** shows the process.

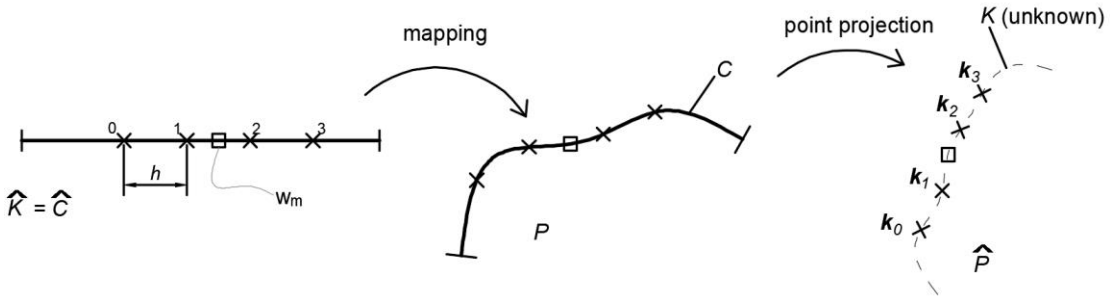


Fig. 5.11 Calculation of coordinates in $\hat{P}$-space for sample points used for FDD in curves.

5.3.2. Surfaces

For Q-surfaces, the error is measured in both parameter directions. Each direction is treated as a curve applying the same procedure shown in section 5.3.1. The third derivative is also estimated by FDD, therefore a set of points are to be mapped from $\hat{Q}$ to the physical space, and then point projected to $\hat{P}$, as illustrated in **Fig. 5.12**.

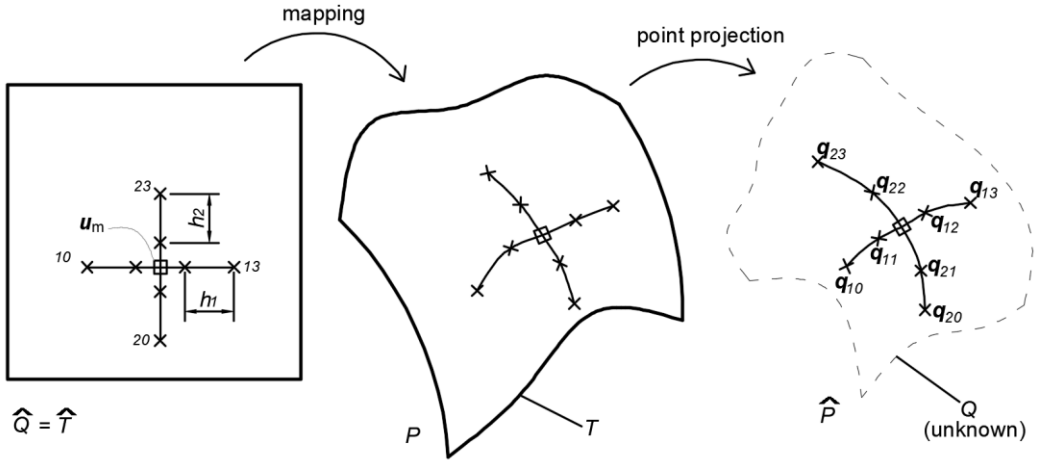


Fig. 5.12 Calculation of coordinates in $\hat{P}$ space for sample points used for FDD in surfaces.

FDD in jth direction is computed as equation (5.7), where c may be 1 or 2 for surface parameter directions u or v respectively.

$$Q^{m}{}_{,ccc\,j} \cong \frac{q_{c3\,j} - 3\,q_{c2\,j} + 3\,q_{c1\,j} - q_{c0\,j}}{h^3} \tag{5.7}$$

where $\boldsymbol{q}_{ca}$ are the coordinates of the ath sample points in the $\hat{P}$-space obtained as **Fig. 5.12.** Analogously to curves, the maximum admissible distance in the cth direction between nodes is estimated as follows:

$$s^{c}{}_{adm} = \left(\frac{15.6\,E_{adm}}{\left\| Q^{m}{}_{,ccc\,j} \right\|} \right)^{\frac{1}{3}} \tag{5.8}$$

5.4. Nodes insertion

Nodes are inserted in K-curves and Q-surfaces to attain a separation between them equal or less than the admissible distance s_{adm}. The value of s_{adm} varies from point to point, *i.e.* it is a continuous field on the domain ($\widehat{K}$ or $\widehat{Q}$). In this work the s_{adm} field is estimated by linear interpolation between a set of *test* points, where s_{adm} is calculated with (5.5) and (5.8).

The strategy to insert nodes into the domain follows a recursive process, where the stretch between two nodes, called *division*, is split if the minimum value of $2s_{adm}$ within such division is less that the division itself. Next two sections detail the process for K-curves and Q-surfaces.

The considered admissible nodal distance is $2s_{adm}$ since nodes inserted will be end-nodes. Later on the mid-nodes will be inserted (Chapter 6) achieving the admissible distance and therefore the accuracy required.

5.4.1. Nodal insertion in curves

Field of admissible distances:

The test points[30] are equally spaced at h in $\widehat{K}$-space. Each test point is mapped onto the physical space and then its $\hat{P}$-space coordinate is estimated by point projection (recall **Fig. 5.11**).

The admissible distance s_{adm} at each test point in $\widehat{K}$-space is computed using equation (5.5). The third derivative required for s_{adm} calculation is estimated by FDD, as equation (5.6), where we select the adjacent test points as FDD sample points. Although these adjacent points brings speed to the process (calculation of extra points is avoided) the derivative is not centred at the test point. Indeed, at ith test point, the FDD is centred at half division backward or forward (for FDD sample points starting at $i-2$ or $i-1$ respectively). To overcome this pitfall, the derivative is calculated as the average of both FDDs. **Fig. 5.13** shows the sample points for FDDs centred at previous and next mid-segments of one test point.

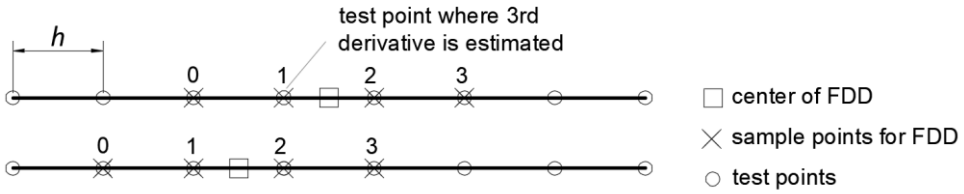


Fig. 5.13 FDD using adjacent test points for curves.

After the calculation, s_{adm} is known at each test point and the field may be plotted as shown in **Fig. 5.14** left.

[30] We use 21 test point for curves in this work. Therefore there are 20 equal divisions of length h.

Nodal insertion:

The nodal insertion starts with nodes at the end of the curve[31]. Within each division of the domain the minimum admissible distance ($s_{adm,min}$) is obtained from the s_{adm} field. If the division length, i.e. distance between its nodes in the curve parameter space (Δw), is greater than $2s_{adm,min}$ a new node is inserted at the mid-location, splitting into two divisions. The process carries on until no division has inside $2s_{adm,min}$ smaller than the division length. One example is given in **Fig. 5.14**, where four iterations are required. At the right-hand side, the values of $2s_{adm,min}$ for divisions *a* and *b* (second iteration) are drawn.

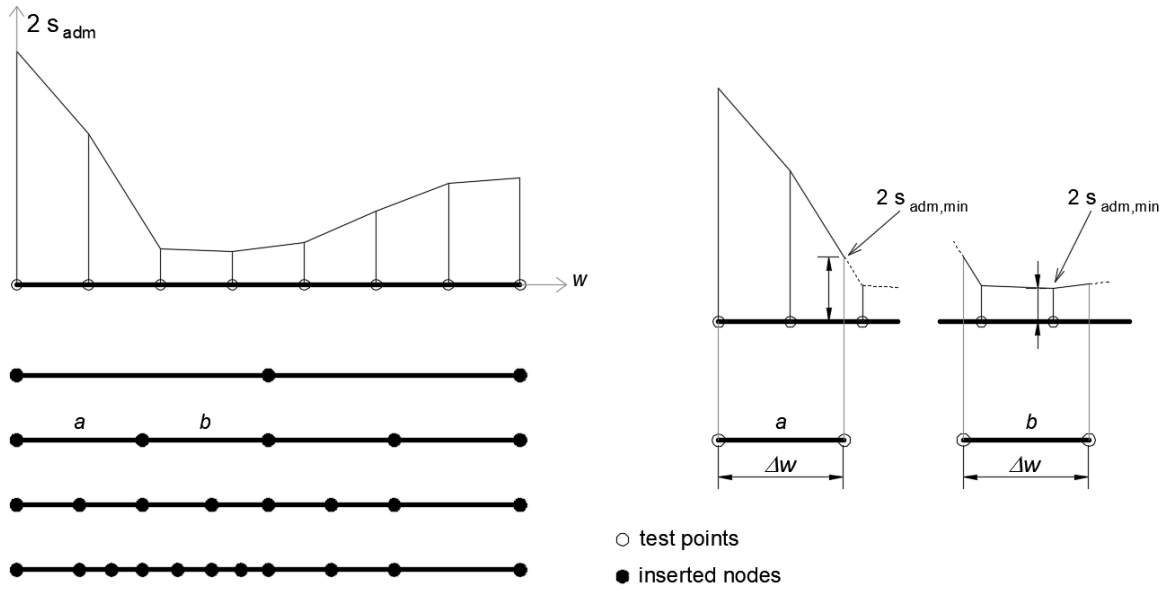


Fig. 5.14 Node insertion in curves.

5.4.2. Nodal insertion in surfaces

Field of admissible distance:

Test points are set in a rectangular net that encompasses the computable surface in its parameter space (see **Fig. 5.15**). Their number in each parameter direction is 11, i.e. ten divisions of equal length, h_1 and h_2 for u and v directions respectively. In each parameter direction same procedure as curves is used: adjacent test points are used as FDD sample points (5.7). Hence two FDDs are computed and the average is assumed as the third derivative. One illustration is given in **Fig. 5.16.** The maximum admissible step in each direction is then calculated a per equation (5.8).

[31] At the beginning the whole curve is composed by a single division.

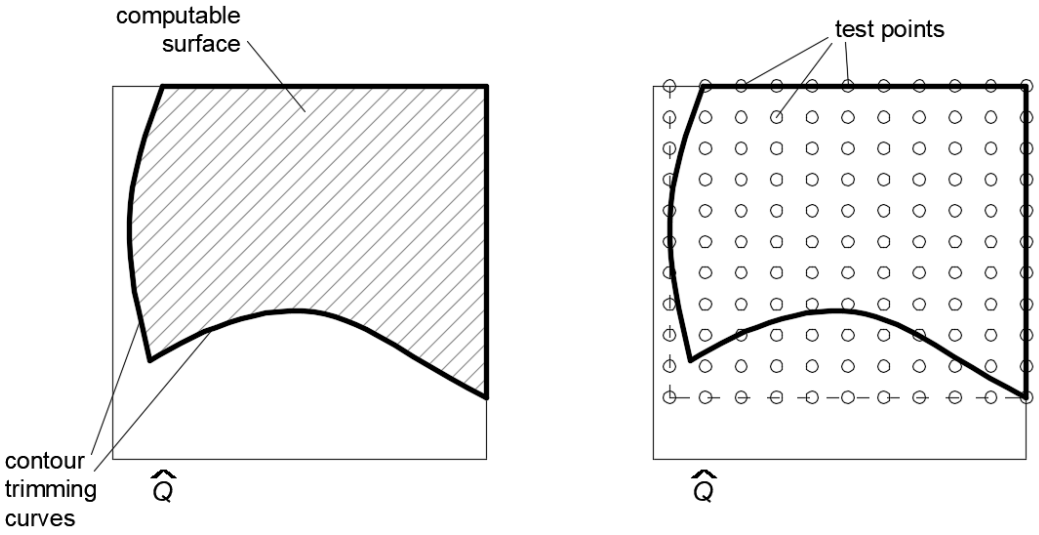


Fig. 5.15 Sample points in surfaces.

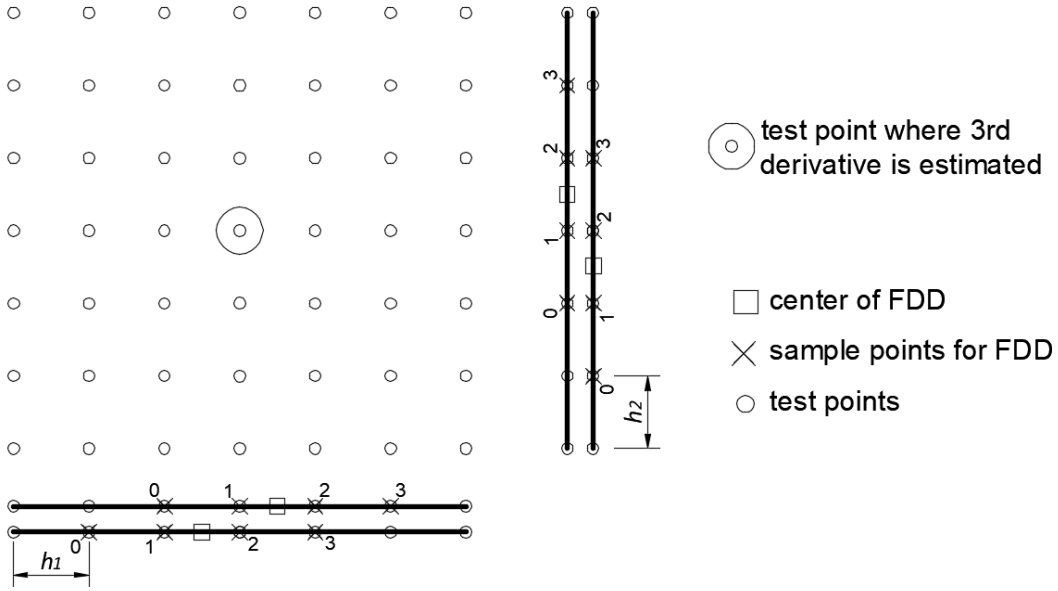


Fig. 5.16 FDD using adjacent test points for surfaces in both parameter directions.

After these calculations, $s^c{}_{adm}$ is known at each test point and the field may be plotted as shown in **Fig. 5.17** left. The super-index c is 1 and 2 for u and v parameter directions respectively.

Nodal insertion:

The nodal insertion starts with four nodes at the corners of the background surface, either if it is trimmed or not. Within each division[32] the minimum admissible distance ($s^c{}_{adm,min}$) is obtained from the $s^c{}_{adm}$ field, in both parameter directions. If the division side length (Δu or Δv), is greater than $2s^c{}_{adm,min}$ the division is split into four quadrilaterals. The process carries on until no division has inside $2s^c{}_{adm,min}$ smaller than the division side length. The check is done

[32] At the beginning the whole surface has a single division.

separately for both parameter direction, and if any of them requires division the rectangle is split into four sub-rectangles. One example is provided in **Fig. 5.17**.

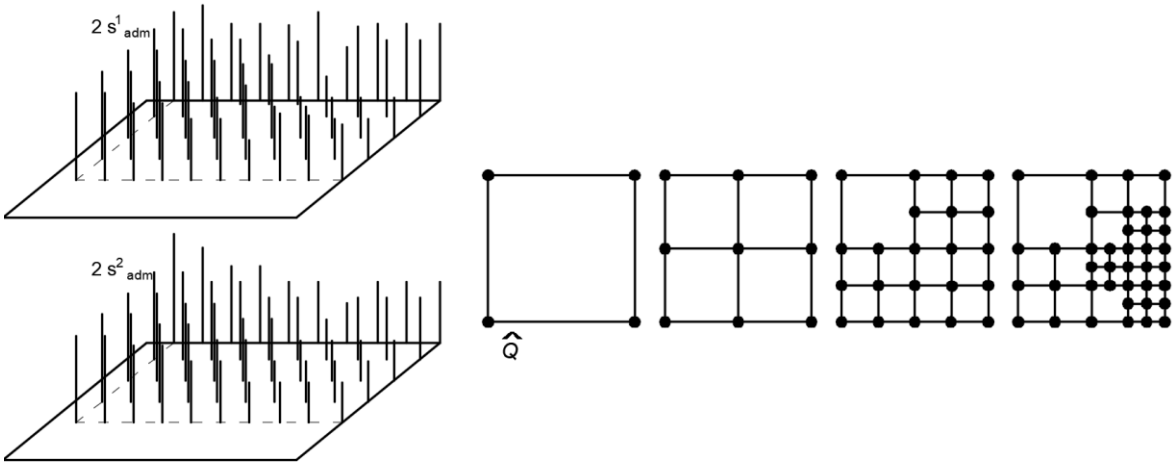


Fig. 5.17 Nodal insertion in surfaces.

After nodal insertion, only those ones within the computable surface and separated from the contour lines more than a pre-established threshold are selected (see **Fig. 5.18**). The contour is given by the piece-wise line that joints the nodes of the curves computed in section 5.4.1.

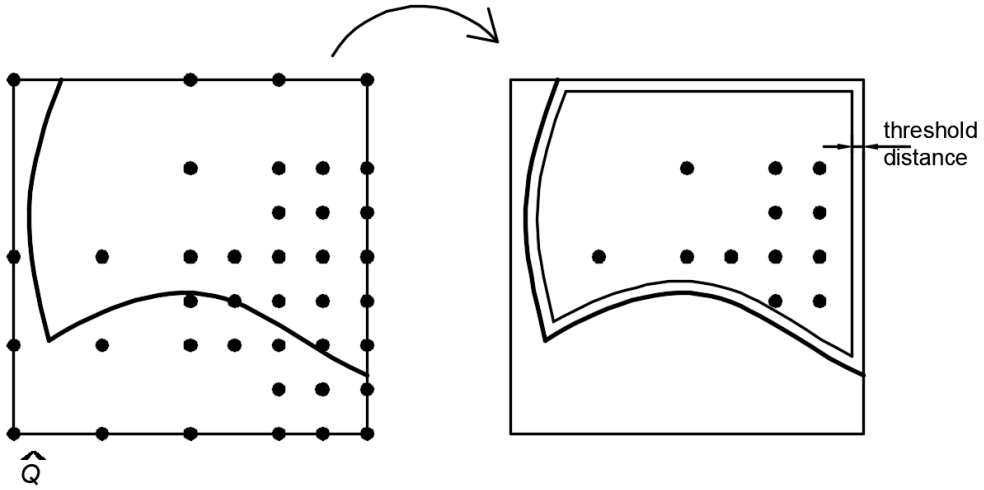


Fig. 5.18 Nodes removal.

5.5. Surface triangulation

5.5.1. Triangulation of end-nodes

For each trimming surface, its end-nodes are triangulated in the $\hat{Q}$-space using Delaunay technique. The resultant mesh forms a convex triangulation where some of its triangles may be outside the contours. Such triangles need to be removed to achieve the final linear triangulation as shown in **Fig. 5.19**.

To check if one triangle is inside or outside of the contours two vectors are compared. One from the centre of gravity of the triangle to the closest contour segment[33], called $\boldsymbol{v}_1$, and another along the segment orientation, called $\boldsymbol{v}_2$. Given that the contours leave the computable domain at their left-hand side, if the third component of $\boldsymbol{v}_1 \times \boldsymbol{v}_2$ is negative the triangle is outside, if it is positive is inside.

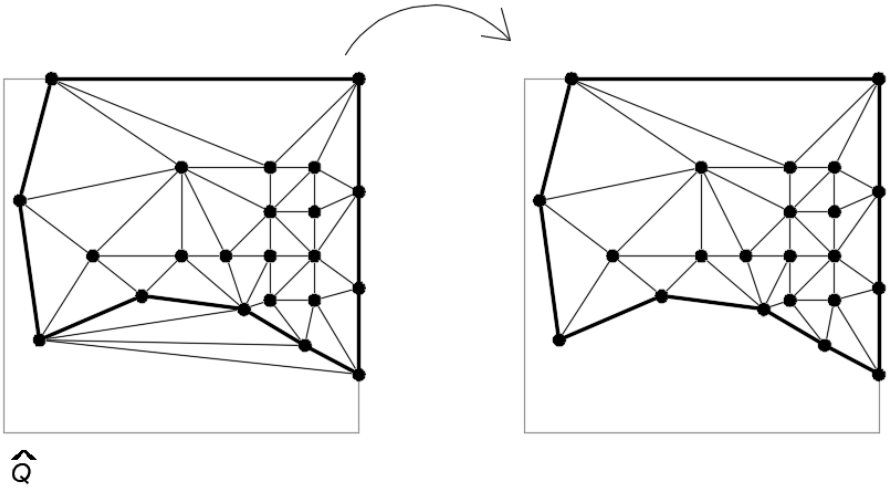


Fig. 5.19 Triangulation and removal of outside triangles in the surface parameter space.

5.5.2. Improvement in the patch parameter space

The solid will be discretized into tetrahedrons in the $\hat{P}$-space as will be explained in Chapter 6. These tetrahedrons use the $\bar{Q}$-surfaces triangles as the facets that lie on the trimmed boundaries. Therefore the quality of the triangulation in $\hat{P}$-space must be as higher as possible, i.e. as closer as possible to isotropic triangulation (Appendix 5A), to improve the quality in the tetrahedralization of the solid.

Initially, the triangulation has not the best quality in $\hat{P}$-space since it has been carried out in $\hat{Q}$-space (section 5.5.1). To improve such quality the triangles are checked in pairs, comparing the current configuration to the alternative configuration by flipping the common line of the pair (see **Fig. 5.20**). The common line is flipped if two conditions are met: the quality of the triangles increases in the $\hat{P}$-space and the resultant triangles are not embedded in the $\hat{Q}$-space. These two conditions are further developed in this section.

[33] Contours have been discretized into segments that joint the nodes computed as section 5.4.1.

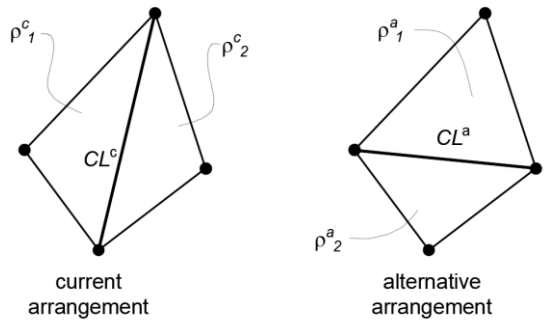


Fig. 5.20 Comparison of current and alternative arrangements.

Increment of triangles quality: the quality condition

The two triangles in the current and in the alternative arrangements are measured in terms of common line length (CL) and triangles distortion (ρ). **Fig. 5.20** illustrates both arrangements. Super-indexes c and a refer to current and alternative situations respectively.

The shorter the common line the higher quality. The distortion of one linear triangle is measured by comparison with the equilateral triangle by the ρ number. The more distorted the triangle the lower value of ρ. The maximum ρ happens for the equilateral triangle, where $\rho = 1$ (refer to Appendix 5B for further details).

The quality of the alternative arrangement is considered to raise if one of the two scenarios given in **Table 5.2** occurs. Each scenario corresponds to improvement in one of the mentioned parameters (CL or ρ) by more than 20 % in the alternative arrangement:

- The ρ number is increased by more than 20 % while the common line remains equal or shorter; or
- The common line gets shorter by more than 20 % while the ρ number remains equal or greater.

Sub-indexes av and max in **Table 5.2** indicate average and maximum of both triangles.

Table 5.2 Conditions to assume increment of quality.

Scenario	***Common line length***	***ρ number (distortion)***
1.- Distortion is reduced	$CL^a \leq CL^c$	$1.2\ \rho^c{}_{av} < \rho^a{}_{av}$ and $1.2\ \rho^c{}_{max} < \rho^a{}_{max}$
2.- Common line gets shorter	$1.2\ CL^a < CL^c$	$\rho^c{}_{av} < \rho^a{}_{av}$ and $\rho^c{}_{max} < \rho^a{}_{max}$

Triangles must not be embedded in $\hat{Q}$-space: the embedded condition

Some alternative arrangements may lead to one triangle embedded within the other in $\hat{Q}$-space. These triangles in $\hat{P}$-space apparently may improve the quality (CL or ρ parameters) but when they are upgraded to quadratic degree (Chapter 6) they result highly distorted or even overlapping.

Fig. 5.21 illustrates one example in $\hat{P}$-space. The surface (**Fig. 5.21** (a)) is initially triangulated (b) and one edge is flipped according exclusively to the quality condition (CL or ρ parameters) resulting in **Fig. 5.21** (c). For each configuration a section profile is depicted, where one can see one side of the alternative configuration becomes quasi-perpendicular to the surface line (**Fig. 5.21** (c)). When the degree is increased to quadratic, the current configuration (**Fig. 5.21** (d)) leads to a better triangulation than the alternative (**Fig. 5.23** (e)).

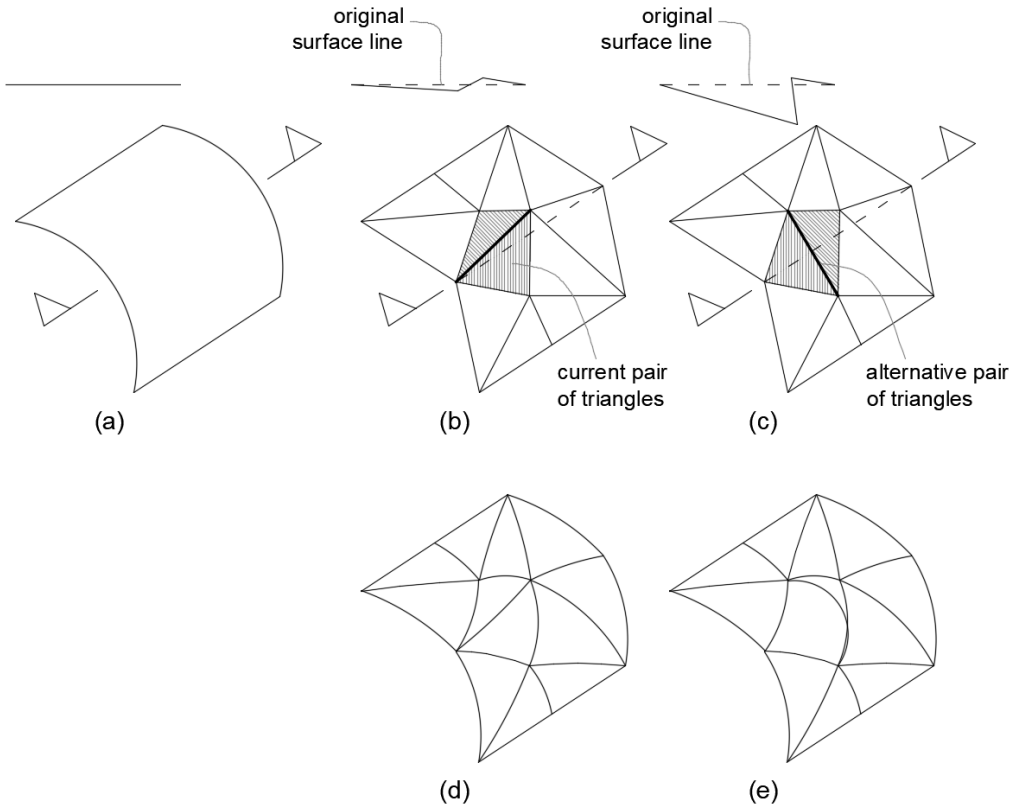


Fig. 5.21 Modification of triangulation in $\hat{P}$-space attending exclusively to the quality condition.

This situation is not detected by the quality condition itself, hence it is necessary to introduce the embedded condition, which states that no flipping is allowed if the alternative arrangement leads to embedded triangles in the $\hat{Q}$-space. The flipping leads to embedded triangles if the external angle of one of the end-nodes of the line to flip is smaller than 180 degrees (see **Fig 5.22** left). In practise we set this angle to 170 degrees, to avoid not only the embedded triangles but the highly distorted quadratic triangles (recall (**Fig. 5.21** (e)).

The image of the pair of hatched triangles of **Fig. 5.21** is represented in **Fig. 5.22** in $\hat{Q}$-space. At the left-hand side the current pair of triangles, where the angle of one of the end-nodes of the common line is indicated. At the right-hand side the alternative arrangement, where triangle 2 is embedded within triangle 1.

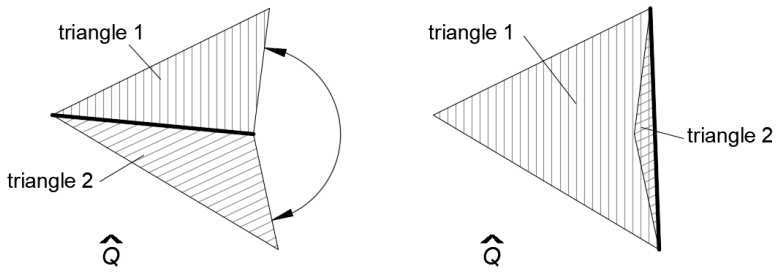


Fig. 5.22 Flipping the common line in $\hat{Q}$-space with resulting embedded triangles.

5.6. Example

This section provides one example that goes throughout the process. Full data of the example is given in Appendix 5E. The tolerance is set to 0.289 %, therefore the admissible error is $E_{adm} = \sqrt{3}\,\frac{0.289}{100} = 0.0050$. **Fig. 5.23** shows the geometry in CAD of the gross and the trimmed patch, alongside with the trimming surface. Note the trimming surface itself is trimmed by contour curves.

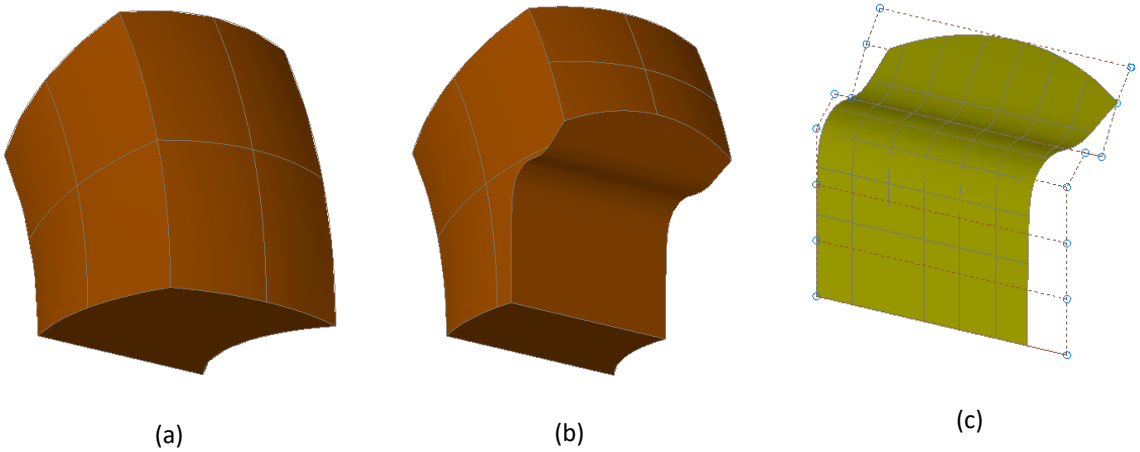


Fig. 5.23 Gross patch (a), trimmed patch (b) and trimming surface (c). All in the physical space.

The nodes are inserted in contour curves and surface to keep the estimated error below E_{adm}. At the left-hand side of **Fig. 5.24** the maximum step allowed and the end-nodes inserted are illustrated $\hat{K}$-space, for one of the curves of the trimming surface. At the right-hand side all the

end-nodes inserted are represented in $\hat{P}$-space. The minimum nodal distance is computed as section 5.3.

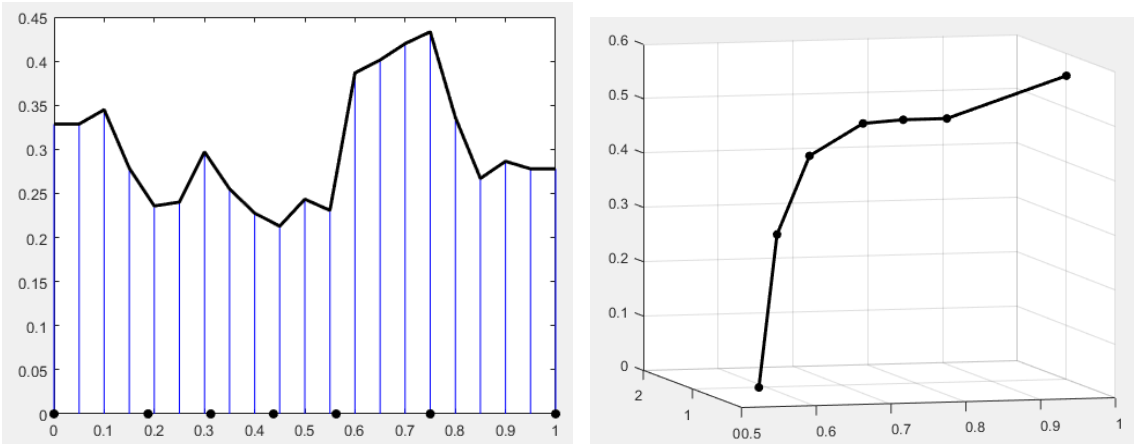


Fig. 5.24 Left: profile of $2s_{adm}$ and inserted end-nodes in $\hat{K}$-space (dots at the bottom line). Right: nodes inserted in $\hat{P}$-space.

For surfaces the insertion of end-nodes is similar to curves, as explained in section 5.4.2. The resultant end-nodes inserted in the surface plus the nodes of the contour curves are depicted in **Fig. 5.25** in $\hat{P}$-space and $\hat{Q}$-space.

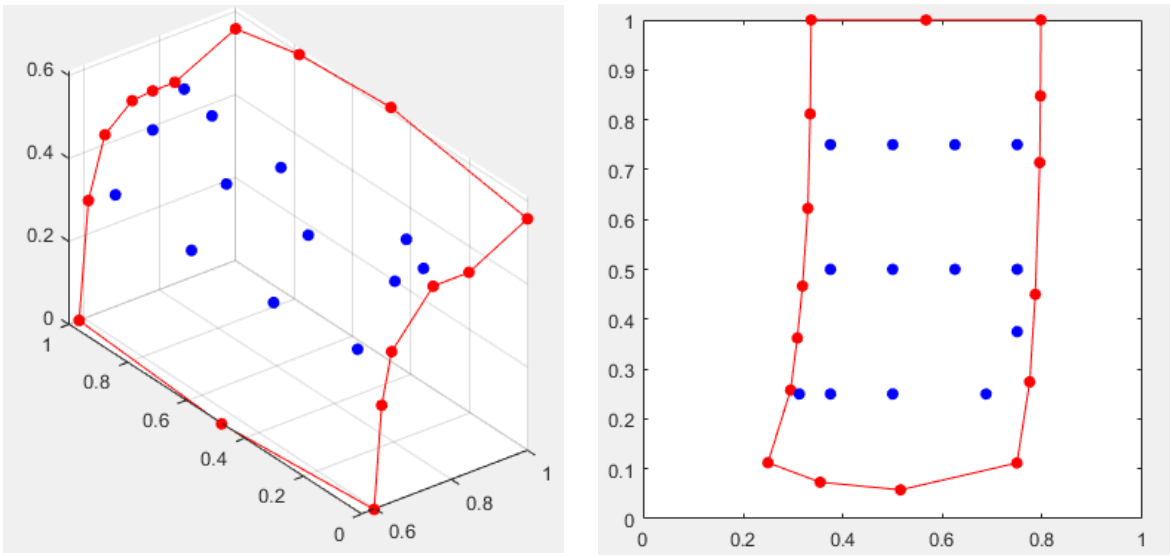


Fig. 5.25 Contour and inner end-nodes, in $\hat{P}$-space and $\hat{Q}$-space.

The initial triangulation of the end-nodes is shown in **Fig. 5.26** (left), where triangles outside the contours are observed. After the removal of these triangles (section 5.5.1) the mesh looks like the right-hand side of **Fig. 5.26**.

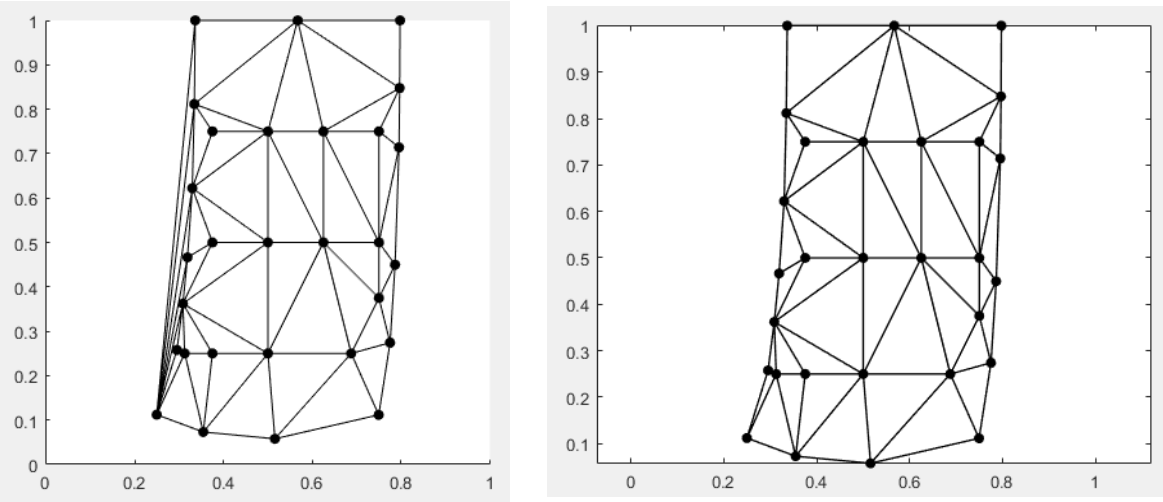


Fig. 5.26 Initial triangulation of surface in $\hat{Q}$-space (left) and without outside triangles (right).

The triangulation in $\hat{P}$-space is plotted in **Fig. 5.27**, at the left-hand side the initial arrangement and at the right-hand side the triangulation after the improvement described in section 5.5.2.

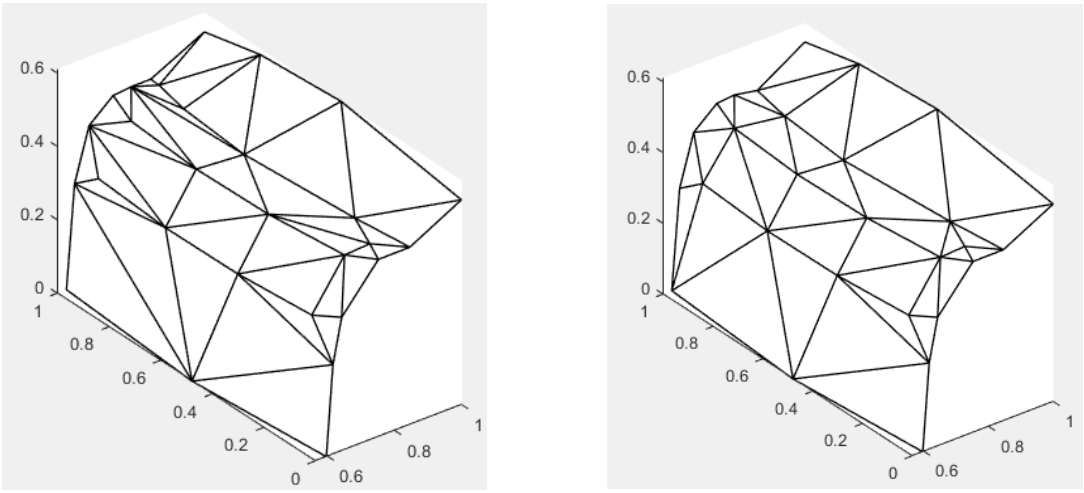


Fig. 5.27 Triangulation of surface in $\hat{P}$-space, before (left) and after (right) improvement.

Mid–nodes are inserted after the tetrahedralization is done, as will be explained in Chapter 6. However, with illustrative purpose, the mid-nodes are inserted in this example resulting the quadratic triangulation shown in **Fig. 5.28**.

To compare the approximated surface against the real surface, a number of sample points forming a triangular mesh are represented in $\hat{P}$-space as per **Fig. 5.29**. Dashed lines indicate exact Q-surface, generated by point projection from the physical space. Continuous lines indicate the approximation $\bar{Q}$-surface mapped from the quadratic mesh obtained in the previous steps. It can be observed that both surfaces are fairly similar.

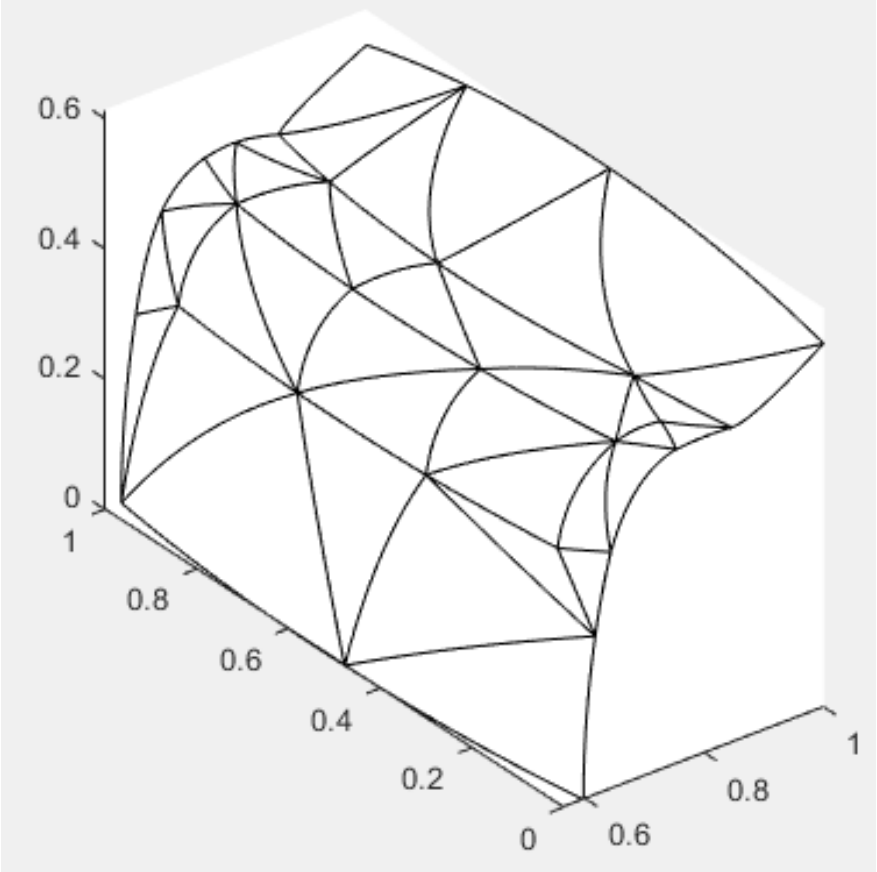


Fig. 5.28 Quadratic triangulation in $\hat{P}$-space.

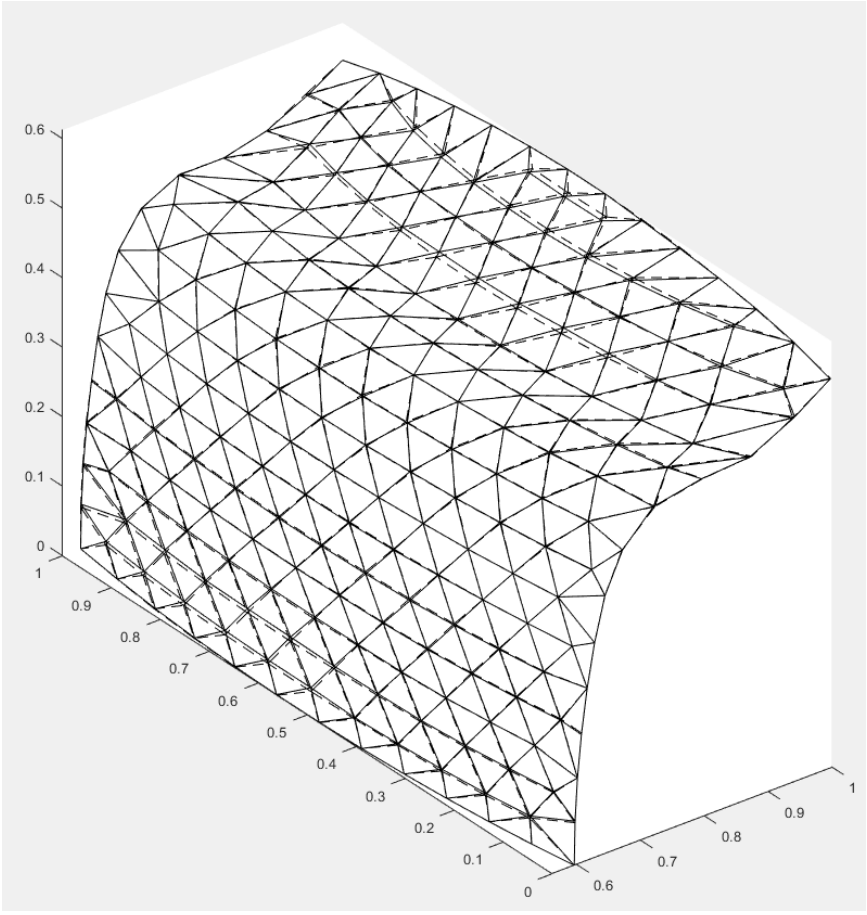


Fig. 5.29 Comparison between Q (dashed lines) and $\bar{Q}$ surface (continuous) in $\hat{P}$-space.

The errors committed in the approximations, i.e. difference between Q and $\bar{Q}$ are shown in **Fig. 5.30**, scaled by 100. Note that at most locations the error is smaller than $E_{adm} = 0.0050$, locally at some areas the error reaches about 0.0080.

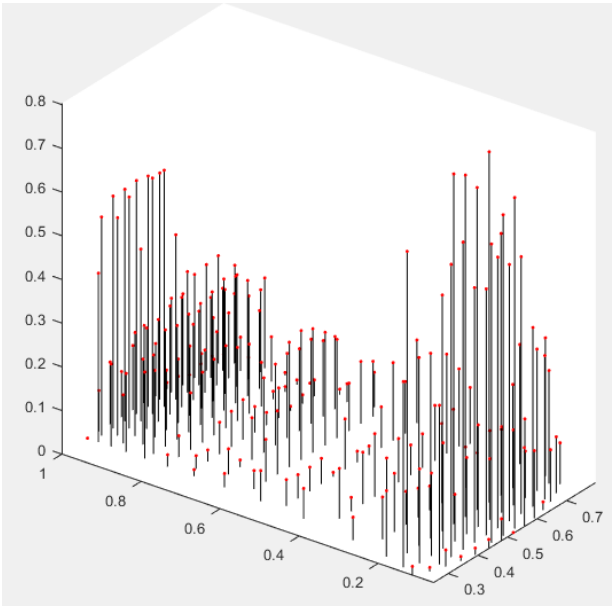


Fig. 5.30 Errors of $\bar{Q}$-surface represented in $\hat{Q}$-space and scaled by 100.

5.7. Relation with the code

The approximation of trimming surfaces is carried out by the routine called `st0301_rTskin`. This routine is within a more general one called `i0100_IniPatches`, that also contains the solid discretization itself (detailed in Chapter 6).

The main inputs of `st0301_rTskin` are:

- `tFace`: contains the trimming surfaces features.
- `Spatches`: contains the patches parametrization.
- `ccTp`: contains the information of curves that contour the trimming surfaces.

Within `st0301_rTskin` the most important steps are:

- Identification and arrangement of trimming surfaces for each patch. This task is done by `st0305_tsR` and `st0308_tcR` routines.
- Construction of $\bar{Q}$-surfaces, done in `fs0700_fit2surf`. Within this function, the most relevant routines are:
  - `fs_0710_fit2conto`: the contour nodes are defined.
  - `fs_0730_fit2inner`: the surface inner nodes are defined.
  - `fs_0750_trianSkin`: triangulation is carried out, including improvement in $\hat{P}$-space.

The output of `st0301_rTskin` is the variable `rTskin` that contains all the $\bar{Q}$-surfaces nodes and their connectivity matrices.

5.8. Summary of the chapter

In this chapter the trimming surfaces are estimated in the patch parameter space to allow accurate integration of the trimmed solid. The approximation is carried out by quadratic triangles, whose size guarantees the error below a pre-established tolerance. Trimming surfaces in general are trimmed by contour curves, therefore these curves need also to be approximated by quadratic segments.

The output of this chapter is a set of triangles in the patch parameter space that approximate the trimming surfaces. The nodes of these triangles are inserted considering quadratic degree, however the output is a linear triangulation. The upgrade to quadratic triangulation (by insertion of mid-nodes on the triangles lines) will be done once the tetrahedral mesh for the solid is finished, as explained in Chapter 6. We postpone this mid-nodes insertion because during the tetrahedralization process extra nodes may be inserted within the triangulation, i.e. the trimming surface obtained here is the minimum required to control the error, but may be refined during the tetrahedralization.

These triangulations will form the limit of the integration of the patches at the trimming surface. Chapter 5 and Chapter 6 cover the thesis objective ***d***.

6. **Discretization of solid domains**

This chapter covers the discretization of solid domains, both trimmed and non-trimmed. The discretization allows the location of the Gauss points that are used to calculate the stiffness matrix of the patch and the body forces[34].

The term *patch* will refer to the solid domain to integrate. The T*-surfaces* and Q*-surfaces* trim the patch in physical and parameter spaces respectively. In this chapter we distinguish three types of patches (see **Fig. 6.1**):

- Non-trimmed patch, with no T-surfaces.
- Trimmed patch, obtained from trimming with T-surfaces the corresponding gross patch.
- Gross-trimmed patch, which is the gross patch before trimming, i.e. non-trimmed version of the trimmed patch.

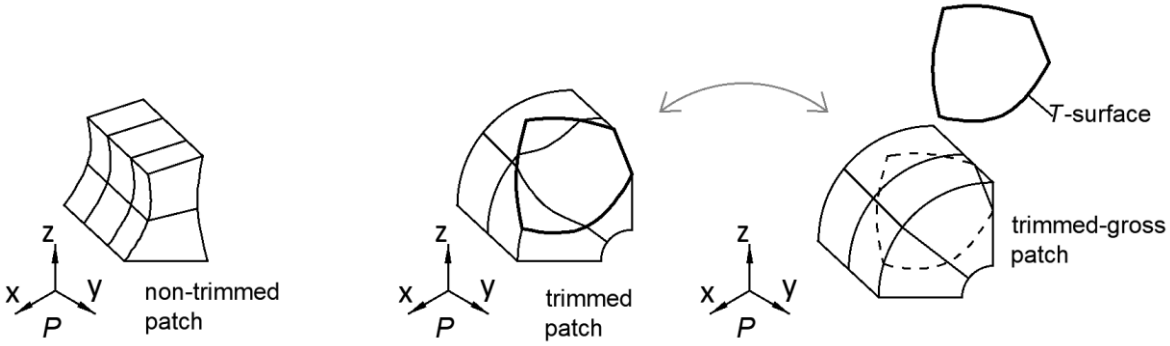


Fig. 6.1 Non-trimmed patch, trimmed patch and its gross-trimmed patch.

The physical space of Q-surface, and their contours K-curves, lie in the parameter space of the patch $\hat{P}$. Since both are unknown, they are approximated with quadratic polynomials $\bar{Q}$ and $\overline{K}$ (recall Chapter 5). Some of the spaces introduced in Chapter 5 are also used here as listed below. **Fig. 6.2** illustrates them.

Physical space:	P
Patch parameter space:	$\hat{P}$
Q-surface physical space:	Q
Q-surface approximation:	$\bar{Q}$
Q-surface parameter space:	$\hat{Q}$
K-curve physical space:	K
K-curve approximation :	$\overline{K}$
K-curve parameter space:	$\hat{K}$

[34] See Appendix 2B.

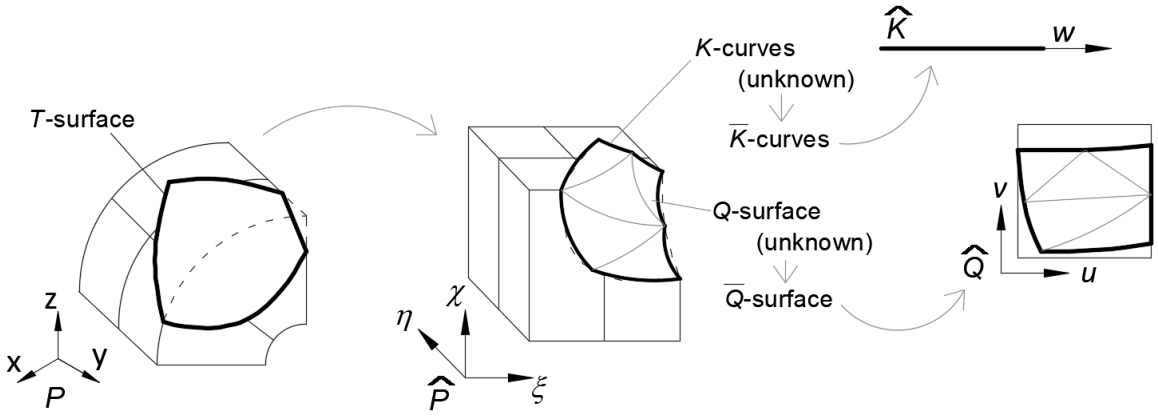


Fig. 6.2 Relevant spaces in the solid discretization.

The Gauss quadrature approximates the integral of a function φ as equation (6.1), recall section 2.1.2, where the functions are evaluated at each Gauss point.

$$\mathbb{I}_{\varphi} \approx \sum_{g=1}^{N_g} \varphi_g \, \mathcal{J}_{1g} \, \mathcal{J}_{2g} \, w_g \tag{6.1}$$

Hence the Gauss points locations are needed for the integration. The number and location of Gauss points depends on the discretization of the patch and NURBS degree. For non-trimmed domains, the Gauss points are arranged by knot spans in FEA fashion with hexahedral elements. **Fig. 6.3** shows the Gauss points for one hexahedron in P-space, $\hat{P}$-space and hexahedron parent space.

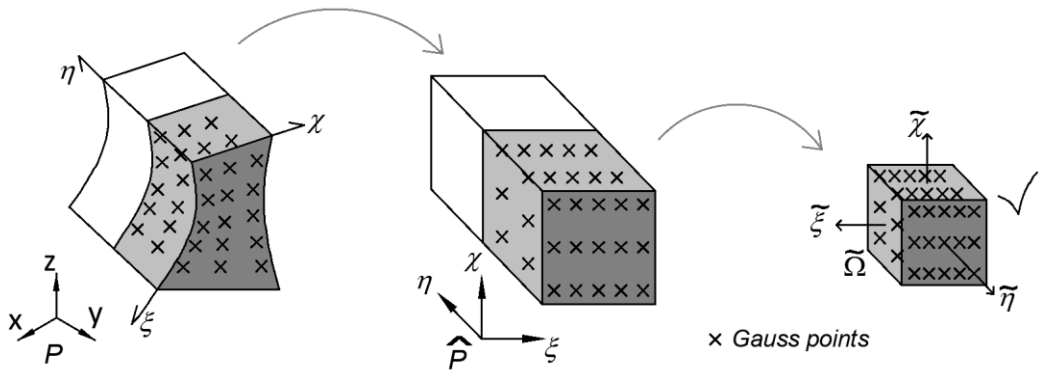


Fig. 6.3 Discretization and Gauss points (crosses) scheme for non-trimmed patches.

For trimmed patches, the scheme shown previously is not valid since it includes the trimmed portion (non-computable domain), as illustrated in **Fig. 6.4** (top). Therefore, trimmed patches need different discretization scheme. This work uses tetrahedrons as shown in **Fig. 6.4** (bottom). Tetrahedrons occupy easily volumes different from knot spans and hence provide flexibility to adapt to trimming surfaces. Once the tetrahedral mesh is set, integration points are located

according to Gauss quadrature rules[35]. The tetrahedrons used in this work may be linear or mixed-degree[36].

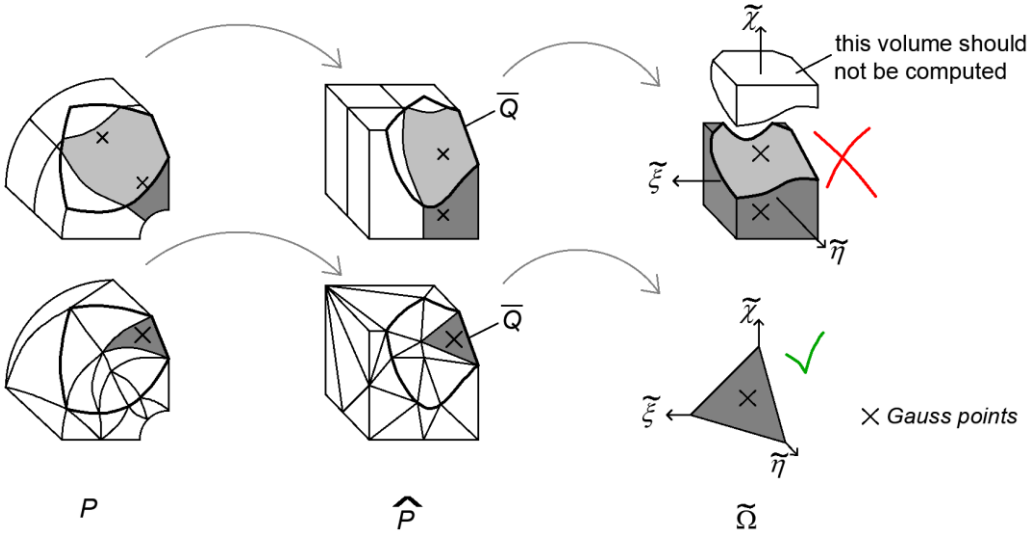


Fig. 6.4 Trimmed patches discretization with hexahedrons (top) and tetrahedrons (bottom).

The tetrahedralization flowchart for trimmed patches is shown in **Fig. 6.5**. First the trimming surfaces must be approximated by $\bar{Q}$-surfaces (Chapter 5). Then a preliminary tetrahedralization is done for the gross-trimmed patch. The nodes of these tetrahedrons that lie within the computable domain, together with the nodes from the $\bar{Q}$-surfaces are used to obtain the tetrahedral mesh of the trimmed patch.

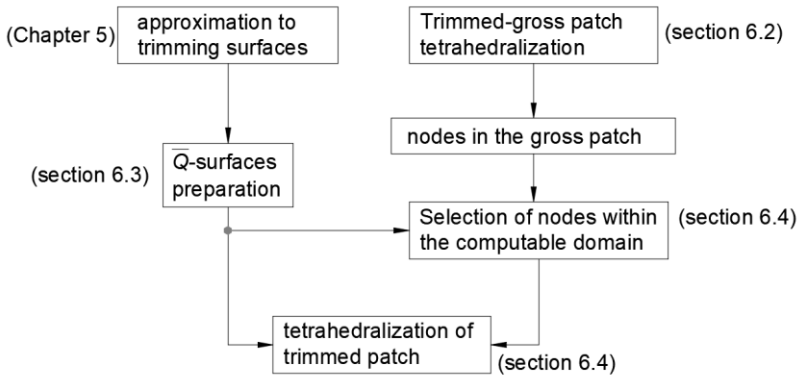


Fig. 6.5 Flowchart for generation of tetrahedral mesh for trimmed patches.

This chapter is structured as follows. Section 6.1 outlines the discretization of non-trimmed patches using the hexahedral scheme. The rest of the chapter is dedicated to trimmed patches. Section 6.2 describes the tetrahedral discretization of gross-trimmed patches. In section 6.3 the treatment to the $\bar{Q}$-surfaces prior to its use in the trimmed patch is explained. Section 6.4

[35] Appendix 6A.
[36] Appendix 6E.

describes the tetrahedralization of trimmed patches. Section 6.5 explains the deactivation of control points. Section 6.6 provides one example of trimmed patch discretization. The most relevant routines in the code related are outlined in section 6.7 and section 6.8 summarizes the content of this chapter.

6.1. Hexahedral discretization of non-trimmed patches

In this work we keep the traditional approach with hexahedral discretization similar to FEA: each non-void knot span is treated as one hexahedral element. Within each element we place a number of Gauss points in each direction according to the degree of the NURBS basis functions as follows.

The number of Gauss points required to provide exact integration of a polynomial of s degree is:

$$n = \begin{cases} \dfrac{s+1}{2} & if\ s\ is\ odd \\ \dfrac{s+2}{2} & if\ s\ is\ even \end{cases} \tag{6.2}$$

NURBS are rational function instead polynomial, but we accept the approximation with a polynomial of equal or greater order than the NURBS function. The integrand for the stiffness matrix computation is of the form[37]:

$$\varphi = R_{i,\alpha}\, D\, R_{j,\beta} \tag{6.3}$$

Where D is one constant of the constitutive matrix $\boldsymbol{D}$, and α and β indicate x, y or z. Let us focus on one parameter direction with the degree of the basis functions (R_i) equal to p. Then order of the integrand φ is $s = (p-1) + (p-1) = 2p - 2$, which can be the input to (6.2) obtaining the number of required Gauss points. However, to avoid spurious modes (see Oñate et al. (2008)) the degree is increased by three being s always an odd number, which leads to the number of Gauss points provided by (6.4). This expression (6.4) is suitable also for body forces integration since the its order is lower than $2p - 2$ (see Appendix 2B).

$$n = p + 1 \tag{6.4}$$

6.2. Tetrahedral discretization of gross-trimmed patches

The gross-trimmed patch is discretized prior to the trimmed patch as the nodes of the former will be used in the discretization of the latter. The number of linear tetrahedrons[38] to discretize the

[37] See Appendix 2B.
[38] There is one Gauss point per linear tetrahedron.

gross-trimmed patch must be *similar* to the number of Gauss points in the standard hexahedral discretization (section 6.1) to achieve *similar* accuracy.

The discretization of gross-trimmed patches has two main steps. Firstly, we determine the number of tetrahedrons in each parameter direction (section 6.2.2). Secondly, the resultant nodes are re-located to reflect the parametrization (section 6.2.3). Prior to these explanations, some definitions are introduced in section 6.2.1.

6.2.1. Preliminary definitions

The patch parameter space is a cube that might be divided into *sub-cube*s called *cells*[39]. The corners of the cells will become nodes of the tetrahedral mesh. Each cell can be split by one of its diagonal planes into two wedges. These terms are illustrated in **Fig. 6.6**.

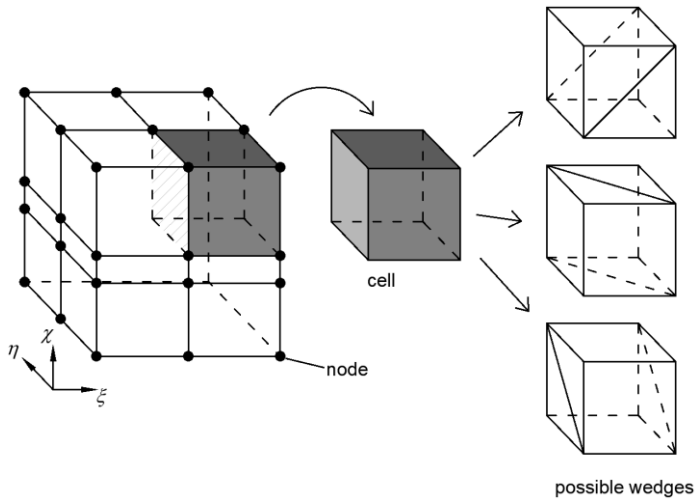


Fig 6.6 Nodes, cells and wedges.

The number of required Gauss points (NR) in each parameter direction is calculated as equation (6.5), where NK is the number of non-void spans and n the number of Gauss points per knot span computed with equation (6.4).

$$NR = NK\ n \tag{6.5}$$

Each cell is assumed to contain six linear tetrahedrons (see **Fig. 6.7**) *i.e*. six Gauss points, then the number of Gauss points added by one cell in each parameter direction is $6^{1/3} \cong 1.8$.

[39] These cells are different from the cells used in the lattice fitting algorithms, section 4.2.3.

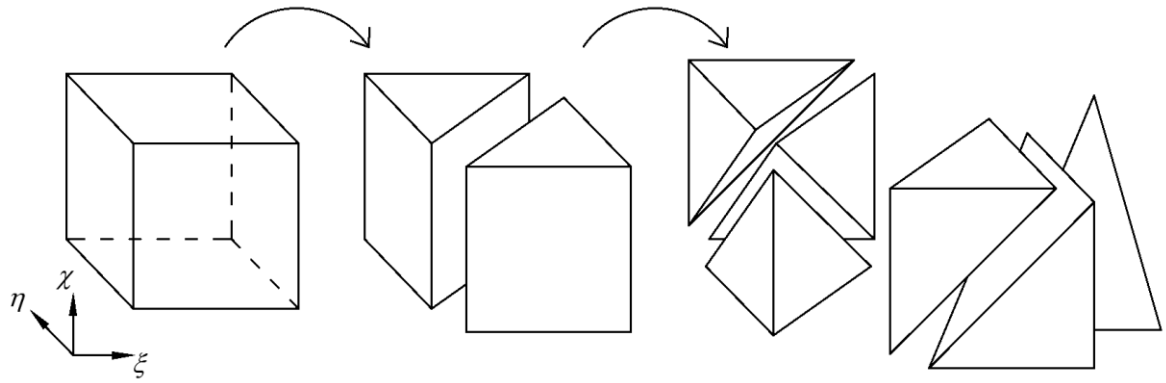


Fig 6.7 Decomposition of one cell into six tetrahedrons.

Initially the patch consists of one single cell with eight nodes. The number of required cells to add in each direction is given by equation (6.6). This number is in fact the number of nodes to add in that direction. One example is illustrated in **Fig. 6.8** with $NC = \{2\ 1\ 1\}$.

$$NC = \frac{NR - 1.8}{1.8} \tag{6.6}$$

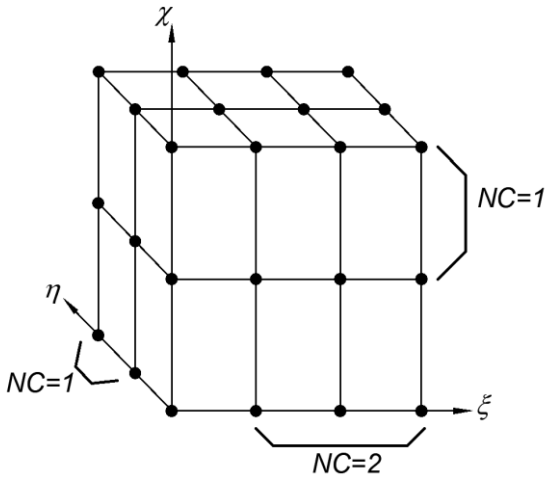


Fig. 6.8 Number cells to add in each parameter direction.

One wedge contributes with $1.80/2 = 0.90$ Gauss points in each parameter direction. The number of wedges to add NW is computed as equation (6.7). NW is more accurate than NC since the increments of added Gauss points are 0.9 instead 1.8.

$$NW = \frac{NR - 1.8}{0.9} \tag{6.7}$$

Fig. 6.9 shows one patch with $NW = 3$ in the first parameter direction, note the initial cell provides already two wedges.

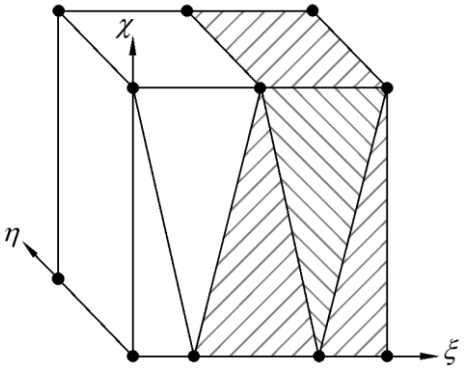


Fig. 6.9 Wedges to add (hatched).

We call *alternating direction* to the parameter direction with the largest odd NW. If none of the three directions have odd NW, the *alternating direction* has the largest NC. The other two directions are called *perpendicular directions*.

6.2.2. Nodes addition

The flowchart of adding nodes to the gross-trimmed patch is illustrated in **Fig. 6.10**.

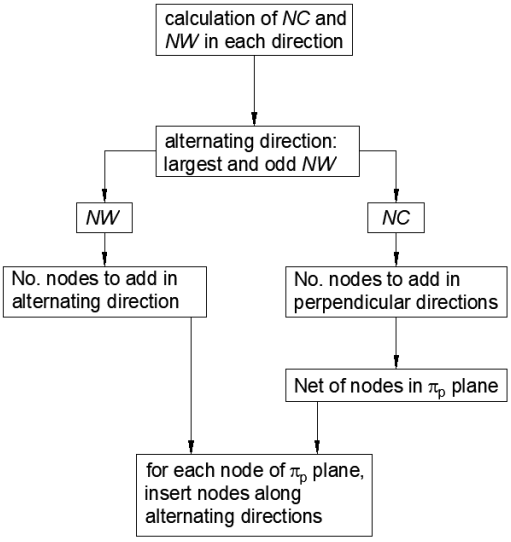


Fig. 6.10 Flowchart of nodes insertion.

First step is to calculate the number of cells (NC) and wedges (NW) in each parameter direction, which allows the define the alternating and perpendicular directions. With NC of the perpendicular directions a net of nodes is set in the plane that both directions form. This plane is called π_p and is located at the start of the alternating direction. **Fig. 6.11** shows one net of nodes, in the π_p plane at $\chi = 0$, with NC equal to three and one in directions ξ and η.

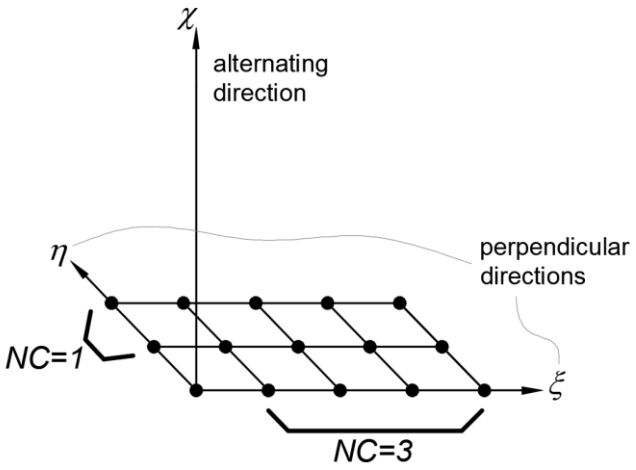


Fig. 6.11 Net of nodes in π_p plane.

From each node of the π_p plane, a perpendicular line is created parallel to the alternating direction to add nodes on it. The number of nodes to add in each perpendicular line varies alternatively from n_1 to n_2, which are computed with (6.8) and (6.9) respectively[40]. Finally, the net of nodes of the π_p plane is copied at the end of the alternating direction.

$$n_1 = floor\left(\frac{NW}{2}\right) \tag{6.8}$$

$$n_2 = ceil\left(\frac{NW}{2}\right) \tag{6.9}$$

One example is illustrated in **Fig. 6.12**, where $NW = 5$, therefore the number of nodes to add are $n_1 = 2$ and $n_2 = 3$.

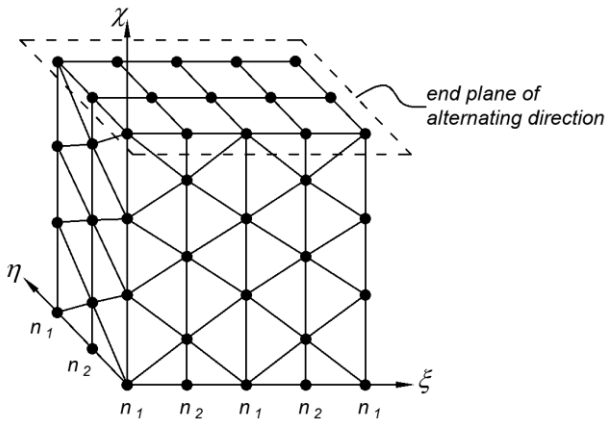


Fig. 6.12 Nodal insertion along alternating direction.

[40] Floor and ceiling functions are presented in Appendix 2A.

The setting out of the nodes along the lines is also alternating from one line to the next. Let us define the basic nodal distance (ℓ_b) as equation (6.10), where L is the span of the alternating direction[41].

$$\ell_b = \frac{L}{NW + 1} \tag{6.10}$$

The distance from the π_p plane to the first inserted node along the perpendicular lines alternates between ℓ_b, and $2\ell_b$ as illustrated in **Fig. 6.13** (only two lines are shown for clarity). This alternating arrangement increases the isotropy of the triangulation at the faces (Appendix 5A) and increases the tetrahedrons regularity (see Appendix 6B).

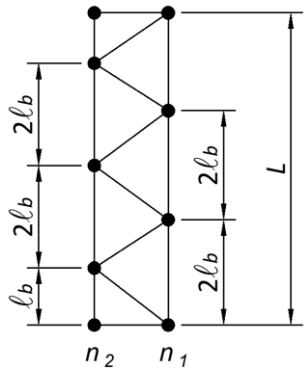


Fig. 6.13 Nodal spacing along lines in alternating direction.

6.2.3. Redistribution of nodes

Nodes are not to be uniformly spaced but according to the knot spans distribution. To reflect this distribution we introduce the mapping $\mathbb{R}^1 \to \mathbb{R}^1$ from the γ-space to the ξ-space. Both spaces correspond to one parameter direction, are piece-wise linear, with a number of divisions equal to the number of non-void knot spans, and spanning from zero to one. The divisions are equal to knot spans in the ξ-space and uniformly spaced in the γ-space. Both spaces are connected by the piece-wise line in the Cartesian plane. **Fig. 6.14** shows one example of both spaces, with four divisions and knot spans $[0 \;\; 0.1 \; 0.2 \;\; 0.5 \;\; 1]$.

[41] The span of parameter space goes from 0 to 1 (recall section 3.1.2).

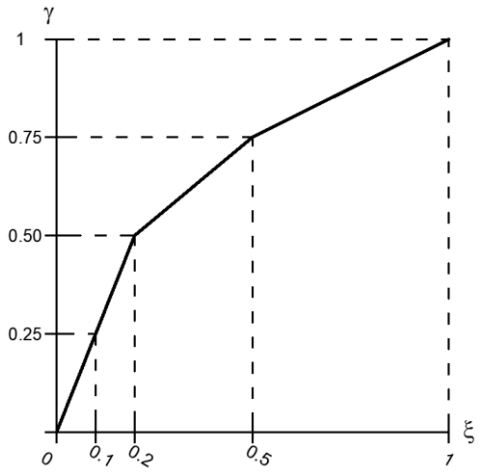


Fig 6.14 ξ and γ-spaces in the Cartesian plane.

The number of nodes in one parameter direction (NN) are computed adding to n_1 and n_2, equations (6.8) and (6.9), two more nodes corresponding to the initial cell. In the γ-space these nodes are equally spaced at $L_g = 1/(NN-1)$ and are projected to the ξ-space as follows. Let us consider one node with coordinate in the γ-space $\gamma_N = L_g(N-1)$, where *N* indicates the position of the node. The division in the γ-space where γ_N lies is referred as *j* division. The node coordinate is mapped to the ξ-space by the linear interpolation (6.11).

$$\xi_N = \xi_j + \frac{(\xi_{j+1} - \xi_j)}{(\gamma_{j+1} - \gamma_j)} (\gamma_N - \gamma_j) \tag{6.11}$$

One example is illustrated in **Fig. 6.15**, with six nodes to insert. The third vertex, whose γ-coordinate is $\gamma_3 = 0.4$, is mapped into the ξ-coordinate resulting $\xi_3 = 0.16$. This mapping is applied to each parameter direction, using the ξ-coordinates in the discretization.

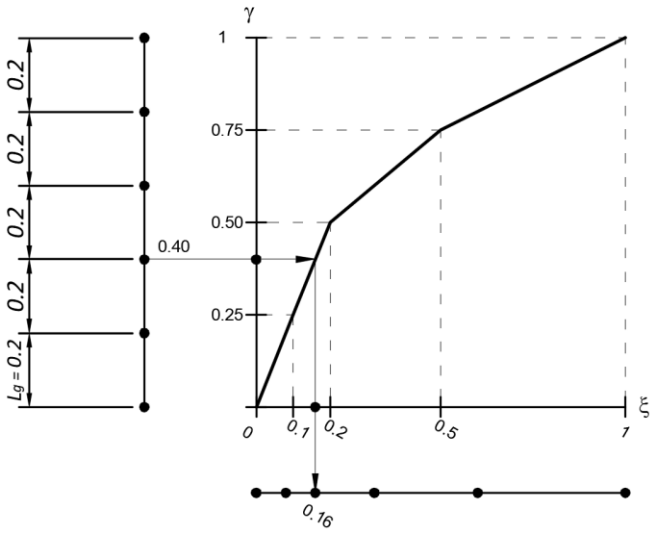


Fig 6.15 Mapping γ to ξ space.

6.3. Treatment of trimming surfaces: the Merged Linear Triangulation (MLT)

The approximation to the trimming surfaces (the $\bar{Q}$-surfaces) obtained in Chapter 5 will be used to delimit the trimming boundaries of the trimmed patches. Prior their use, they are amended to improve their suitability for the tetrahedralization as shown here.

6.3.1. Adaptation of triangles to the tetrahedrons sizes

We have on one hand the spacing of the nodes of the $\bar{Q}$-surfaces that were based on the surface shape (section 5.4.2). On the other hand the spacing of the tetrahedrons nodes of the gross-trimmed patches that are based on the number of knot spans and degree of the patch. Hence both sizes are not related so far, which may lead to highly distorted tetrahedrons. **Fig. 6.16** shows this event in 2D.

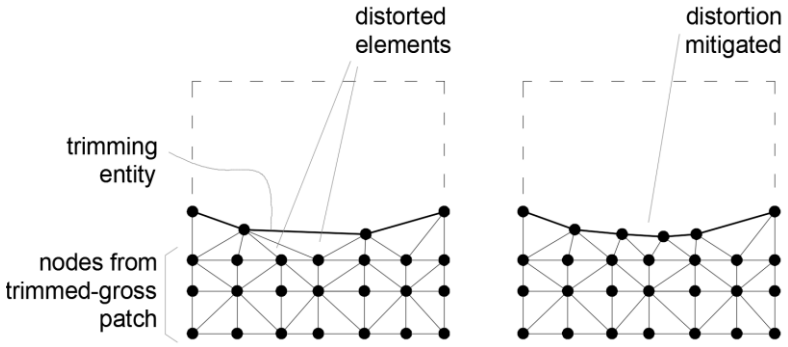


Fig 6.16 Left: tetrahedralization when spacing of the nodes of the trimming entities and tetrahedrons are widely different. Right: improvement by nodes insertion in the trimming entity.

In case the spacing of the triangle nodes is smaller than the spacing of the tetrahedron nodes, the nodes of the trimming surface cannot be removed since geometrical accuracy would be lost. The only valid procedure is to refine the patch to achieve smaller tetrahedrons. This case is not common though.

In case the spacing of the triangle nodes is larger than the tetrahedrons, additional nodes are required in the triangulation of the trimming surface to avoid distorted tetrahedrons in the final discretization (section 6.4.3). This nodal insertion process, which occurs entirely in the patch parameter space, is outlined in this section.

Each line of the $\bar{Q}$-surface triangulation is checked separately, using its mid-point to locate the tetrahedron where it lies in. Let us call L_{av} to the average length of the lines of such tetrahedron, and L_t to the length of the line of the $\bar{Q}$-surface triangulation being checked. If $L_t > L_{av}$ the triangle line is divided into a number of segments (ns) as follows:

$$ns = max\begin{cases} 2 \\ \dfrac{L_t}{L_{av}} \end{cases} \tag{6.12}$$

After revising all the lines, the $\bar{Q}$-surface is triangulated again, and their lines are checked. The process is repeated until no insertion is done.

6.3.2. The merged linear triangulation

All the adjacent $\bar{Q}$-surfaces are merged in a resultant linear mesh is called *MLT*, acronym for *merged linear triangulation*. One example with three surfaces is shown in **Fig. 6.17**. The *MLT* can be composed by separated parts.

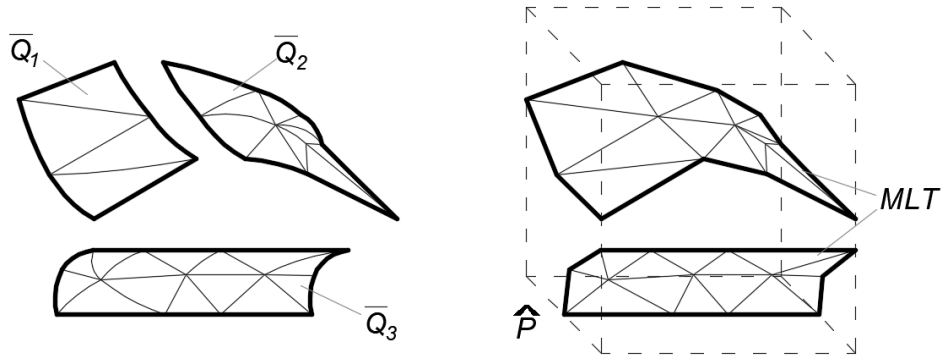


Fig. 6.17 *MLT* with two parts generated from three $\bar{Q}$-surfaces.

Normal vectors of *MLT*, that point outwards the computable domain, will be used to select the nodes of the gross-trimmed patch that are inside computable domain. The normal vectors to the triangles are called $\boldsymbol{n}_t$. The normal vectors to the *MLT* nodes ($\boldsymbol{n}_n$) are computed from the adjacent triangles normal vectors as follows:

$$\boldsymbol{n}_n = \frac{\sum_{i=1}^{m} A^i {\boldsymbol{n}_t}^i}{\sum_{i=1}^{m} A^i} \tag{6.13}$$

Where A^i is the area of the ith triangle attached to the node, ${\boldsymbol{n}_t}^i$ its normal (with unit norm), and m is the number of triangles attached to the node. The normal to one line of *MLT* at x location is computed as follows:

$$\boldsymbol{n}_l = \frac{l^1 \boldsymbol{n}_n^1 + l^2 \boldsymbol{n}_n^2}{l} \tag{6.14}$$

Where $\boldsymbol{n}_n^i$ are the normal vectors of the end-nodes of the line, both with unit norm, l is the length of the line, and l^i are the distances from x to the opposite end-node. The *MLT* allows to calculate the normal vector at nodes that lie at one common edge between two $\bar{Q}$-surfaces. **Fig. 6.18** shows this case when *MTL* is used (left) and when $\bar{Q}$-surfaces are used instead (right). With separated $\bar{Q}$-surfaces the normal vector at the shared node is not uniquely defined, leading to potential errors in the nodes removal (section 6.4.1).

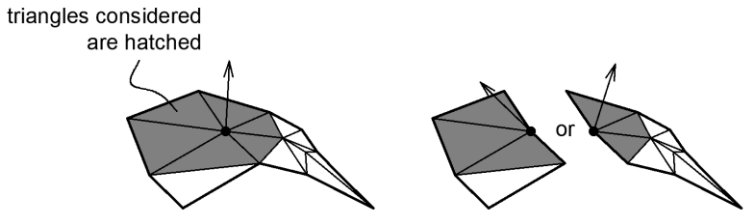


Fig. 6.18 Normal at one surface node with and without *MLT.*

6.3.3. Optimal vertex location and optimal distance

For each triangle of the *MLT* we define the optimal vertex as follows. Let us assume one tetrahedron with one of the faces coincident with a pre-established triangle, whose centre of gravity is denoted by $\boldsymbol{G}_t$. The fourth vertex of the tetrahedron lies on the line that passes through $\boldsymbol{G}_t$ and is perpendicular to the triangle. The position of the fourth vertex, along such perpendicular line, that optimizes the tetrahedron in terms of regularity[42] is called the optimal vertex location. The distance from one triangle to its optimal vertex is called optimal distance. Both, the optimal vertex location and the optimal distance are illustrated in **Fig. 6.19**. For further details of the calculation of the optimal vertex location refer to Appendix 6C.

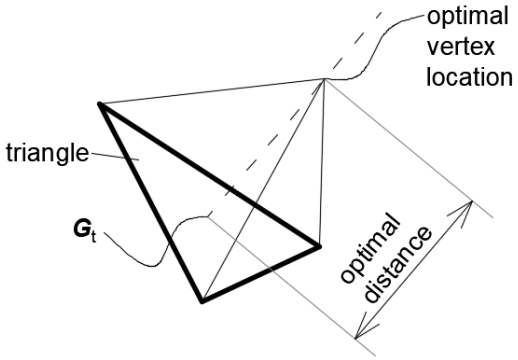


Fig. 6.19 Optimal vertex location and the distance to the triangle.

[42] See Appendix 6B for the definition of the regularity for tetrahedrons.

6.4. Tetrahedral discretization of trimmed patches

To facilitate the comprehension of the tetrahedralization of trimmed patches a summary of the whole process is outlined previously. The figures used in this brief are in 2D for the shake of clarity. All the operations are done in the $\hat{P}$-space.

The *MLT* is brought to the parameter space of the patch (**Fig. 6.20** (a)) to remove the nodes of the gross-trimmed patch that lie outside the computable domain (**Fig 6.20** (b)). This removal process is presented in section 6.4.1.

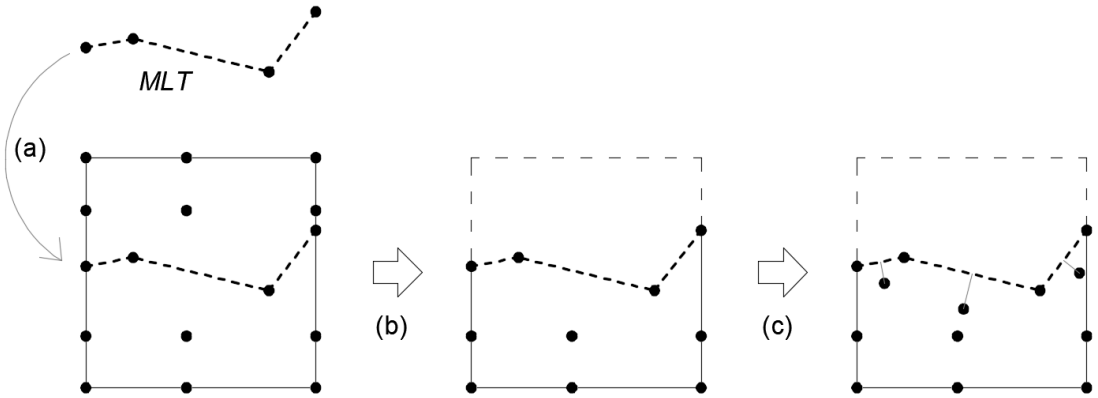


Fig 6.20 Preparation of nodes for tetrahedralization.

With the purpose of enhancing the quality of the tetrahedralization, new nodes are added at 0.4 times the optimal distance from the *MLT* triangles as shown in **Fig 6.20** (c). This insertion is explained in section 6.4.2.

With all the nodes in the computable domain, including *MLT* nodes, the tetrahedralization is carried out (**Fig. 6.21** (d)). Recall that the lines of the $\bar{Q}$-surfaces will define the trimmed limits of the domain (Chapter 5), hence all the lines of the *MLT* must be present in the resultant tetrahedrons. However the tetrahedralization might miss out some of those lines.

This problem, called constrained tetrahedralization (Lee and Lin, 1986), may require addition of nodes and redefinition of the connectivity of some tetrahedrons to comply with the constraints[43]. **Fig. 6.21** (e) shows the amendment of the tetrahedrons to include all the *MLT* lines. The tetrahedralization and its adaptation to *MLT* are presented in section 6.4.3.

[43] The constraints are the lines of the *MLT*.

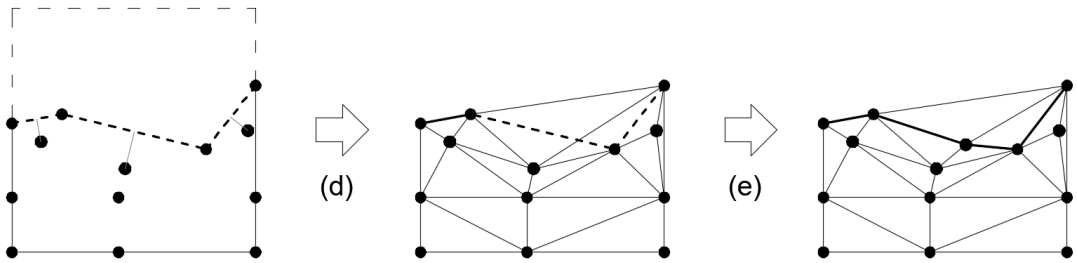


Fig 6.21 Tetrahedralization and adaption to include all the lines from $\bar{Q}$-surfaces.

Some tetrahedrons might be outside the computable domain and other might be quasi-plane. They are removed from the mesh as detailed in section 6.4.4. This process is illustrated in **Fig. 6.22.**

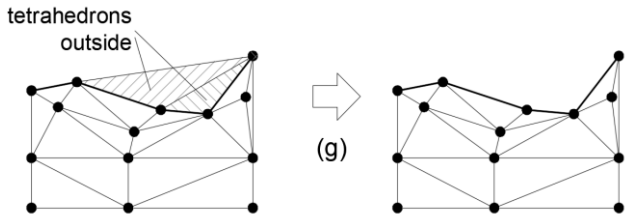


Fig 6.22 Removal of non-valid tetrahedrons.

Mid-nodes are finally inserted at the lines of the tetrahedral mesh that lie on the *MLT*, achieving the quadratic degree for the approximation of the trimming surfaces as shown in **Fig. 6.23** and explained in section 6.4.5. Tetrahedrons in contact with *MLT* result mixed-degree and the rest are linear. Basis functions of mixed-degree tetrahedrons are detailed in Appendix 6E.

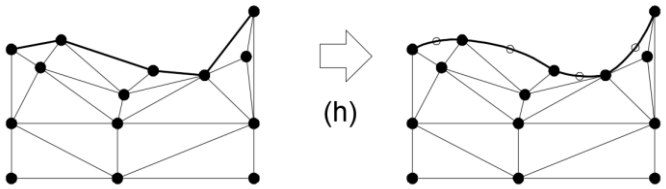


Fig 6.23 Insertion of mid-nodes at lines that lie on *MLT*.

These steps are detailed in section 6.4.1 to 6.4.5.

6.4.1. Removal of non-computable nodes that come from gross-trimmed patch

Each node is evaluated separately. Let us call $\boldsymbol{\xi}_a$ to the position in $\hat{P}$-space of the node to evaluate. The projection of $\boldsymbol{\xi}_a$ in *MLT* is called $\boldsymbol{\xi}_b$, which may lie within a triangle, line or node (see **Fig. 6.24**).

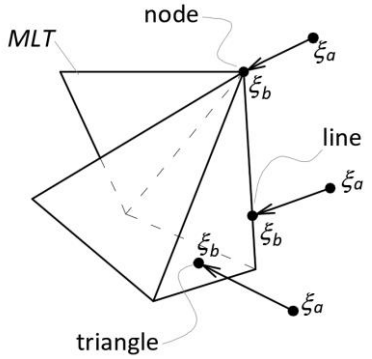


Fig. 6.24 Three cases of location of $\boldsymbol{\xi}_b$.

The vector that goes from $\boldsymbol{\xi}_a$ to $\boldsymbol{\xi}_b$ is called $\boldsymbol{d}$ and the normal vector of *MLT* at $\boldsymbol{\xi}_b$ is $\boldsymbol{n}$. The normal vector is calculated as explained in section 6.3.2.

To assess the position of one node we compute the projection of $\boldsymbol{d}$ on $\boldsymbol{n}$ (6.15). If that projection is positive both vectors have *similar* sense, then the node is inside, otherwise it is outside. If $\boldsymbol{\xi}_b$ lies in a triangle both vectors, $\boldsymbol{d}$ and $\boldsymbol{n}$, are parallel. **Fig. 6.25** shows examples.

$$proj = \boldsymbol{d} \cdot \boldsymbol{n} \tag{6.15}$$

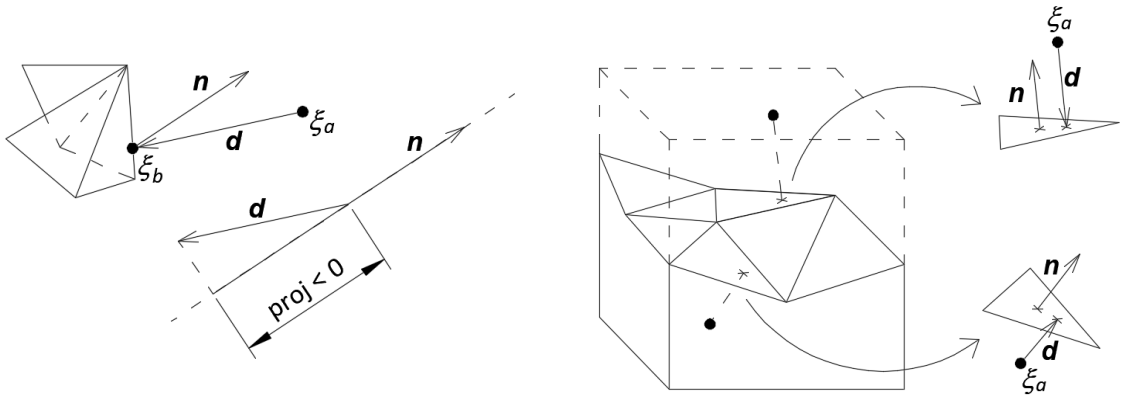


Fig. 6.25 Evaluation of nodes position in general case (left) and with the projection in a triangle (right)

Apart from the mentioned rule, if the norm of $\boldsymbol{d}$ is less than 1.3 times the optimal distance (recall section 6.3.3), the node is removed. This last rule avoids nodes too close to *MLT* and, in conjunction with the additional nodes (section 6.4.2), improves the quality of the tetrahedrons.

6.4.2. Addition of nodes offset from trimming triangles

Due to the removal of nodes close to the *MLT* (end of previous section), a zone without nodes appear in the vicinity of the *MLT*. Additional nodes are added 0.40 times the optimal distance projected from the centre of gravity of each *MLT* triangle. **Fig. 6.26** illustrates the process (in 2D for clarity): the outside nodes and those closer than 1.3 times the optimal distance are removed

(a); additional nodes are added at 0.40 times the optimal distance from the *MLT* (b); the resultant nodes arrangement is prone to produce a tetrahedralization of better quality than without the added nodes (c).

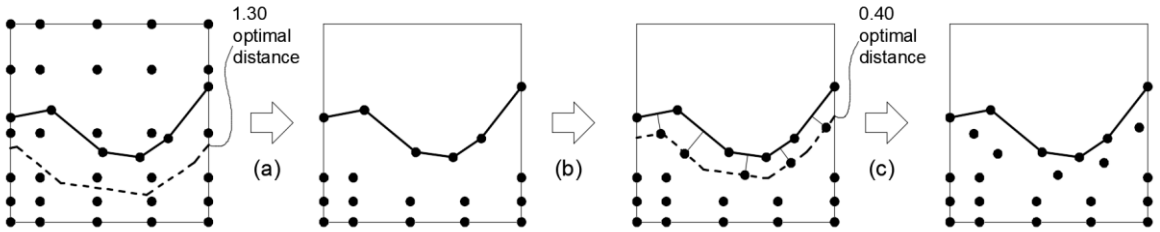


Fig. 6.26 Removal and addition of nodes.

This strategy might be seen as a substitution of the nodes that were too close to the *MLT* by another set of nodes whose location is customized to achieve a better quality mesh.

6.4.3. Tetrahedralization and mesh amendment

All the nodes inside the computable domain plus the nodes from the *MLT* are used for the tetrahedralization by Delaunay technique. As mentioned at the beginning of section 6.4, all the lines from the *MLT* must be present in the tetrahedralization. However, the tetrahedralization might not include some *MLT* lines. Therefore the tetrahedralization needs amendments to include those missing lines, which can involve the insertion of new nodes in both, the tetrahedral mesh and the *MLT*.

Let us call r_m to the line of the *MLT* that is missing in the tetrahedral mesh. We use four different techniques to include r_m in the tetrahedral mesh which are briefed in this section. They are arranged by preference of use in the next list:

- flipping of tetrahedrons;
- flipping of intersected tetrahedron line;
- guided insertion of node in tetrahedron line;
- insertion of node in tetrahedrons.

The preference of use is based on the simplicity to include the r_m line within the tetrahedrons, the first technique incorporates the r_m line in the simplest manner, the fourth is the most intricate.

These transformations are well settled in the field, see for example work by Si and Shewchuk (2014). Flipping involves only changes in the connectivity of the tetrahedral mesh, while the *MLT* remains untouched. Regarding nodal insertion, the new node obligates to updated both, the tetrahedral mesh and the *MLT*.

The new node to insert must lie on the trimming Q-surface. The preliminary position of the new location, called $\boldsymbol{\xi}_p$, is place on r_m. Then the new node position is calculated as Appendix 5D to obtain $\boldsymbol{\xi}_i$. Once the node is inserted, the affected tetrahedrons must be divided to accommodate it, as well as the triangles of the *MLT*. The particularity of this work, compare with the previous works (Si and Shewchuk 2014), is that the new node ($\boldsymbol{\xi}_i$) is at an unknown distance from the preliminary estimation ($\boldsymbol{\xi}_p$) because the Q-surface is unknown.

Flipping of tetrahedrons

Flipping of tetrahedrons is used in two cases:

- If one line r_m intersects one single facet of the tetrahedral mesh, 2-3 flipping includes such line within the mesh (see **Fig. 6.27**).
- In case there are three tetrahedrons whose common line passes across one triangle of the *MLT* and the tetrahedron lines that do not share nodes with that common line coincide with such triangle, 3-2 flipping is used (see **Fig 6.28**). In this case there is not *MLT* missing line, but one tetrahedron line passes across one *MLT* triangle, which is removed.

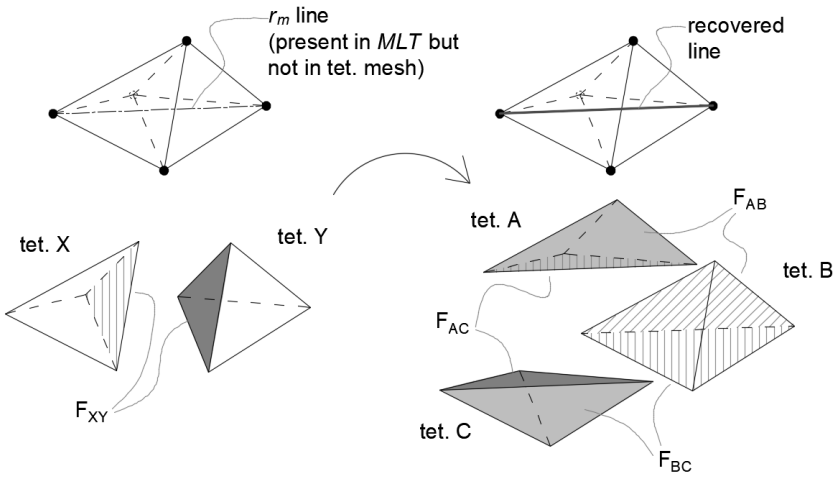


Fig. 6.27 2-3 flipping.

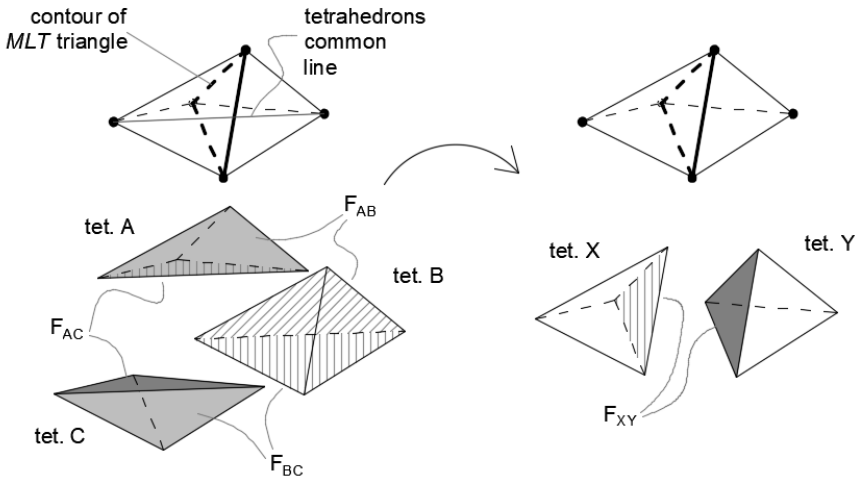


Fig. 6.28 3-2 flipping.

Flipping of Intersected tetrahedron line

When one common line of four tetrahedrons is intersected the r_m line and the next two conditions are met:

- there are four common facets that share the common line;
- each common facet has two tetrahedrons attached;

then, the common line can be flipped to become the r_m line, integrating such line in the tetrahedral mesh. **Fig. 6.29** illustrates the line flipping technique. The procedure can be applied also when there are two tetrahedrons involved (see the right-hand side of **Fig. 6.29**).

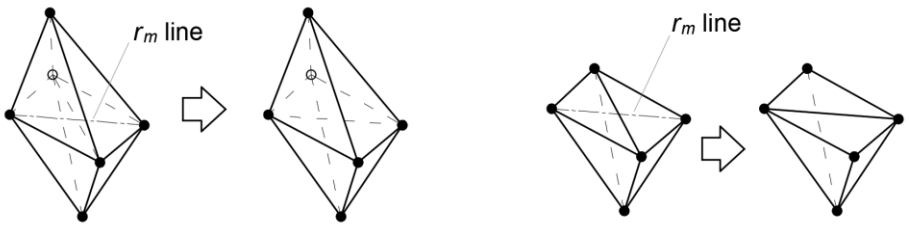


Fig. 6.29 Flipping of tetrahedral line for 4 and 2 tetrahedrons.

Guided Insertion of new node at tetrahedron line

The r_m line is assumed to cross at least two facets of the tetrahedral mesh. Let us call n to the number of intersected facets, then there are $n-1$ *hinge* lines between those facets called r_1, r_2, … r_{n-1} (see **Fig. 6.30** where $n=3$). The interest lies only in the first and last *hinge* lines: r_1 and r_{n-1}, the latter is called r_e for simplicity.

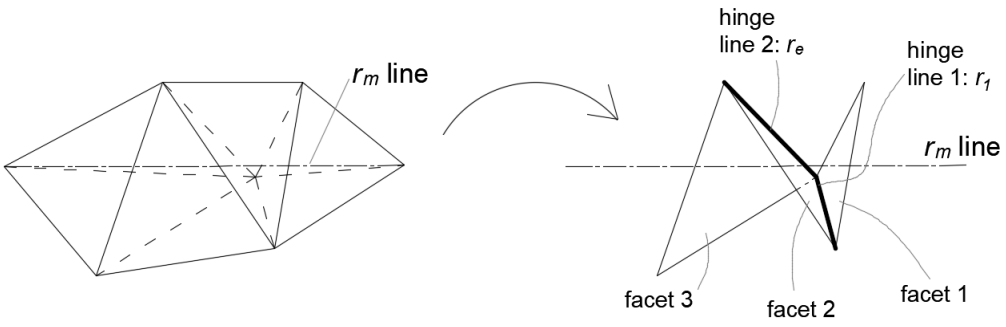


Fig. 6.30 Facets intersected by the r_m line and their hinge lines.

The closest locations from the r_m line to the *hinge* lines r_1 and r_e are computed[44]. Both locations, called $\boldsymbol{\xi}_p^1$ and $\boldsymbol{\xi}_p^e$, are used as preliminary coordinates to calculate the potential new insertions, that are called $\boldsymbol{\xi}_i^1$ and $\boldsymbol{\xi}_i^e$ (see Appendix 5D for this calculation). The selected candidate point, $\boldsymbol{\xi}_i^1$ or $\boldsymbol{\xi}_i^e$, is the closest to its corresponding *hinge* line. The candidate is designated by $\boldsymbol{\xi}_i^c$, and its *hinge* line by r_c (see **Fig. 6.31**).

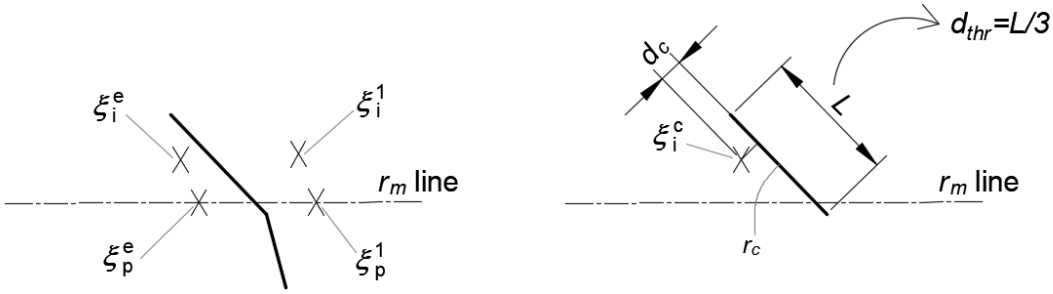


Fig. 6.31 Selection of candidate node and distance to its hinge line.

Let us define as d_c the distance from $\boldsymbol{\xi}_i^c$ to r_c, and d_{thr} the threshold distance, computed as one third of the length of r_c (**Fig. 6.31** right). If $d_c < d_{thr}$ the new node is inserted into r_m and r_c, that are deviated from their original trajectory (see **Fig. 6.32**). The tetrahedrons attached to such *hinge* line are split by the inserted node.

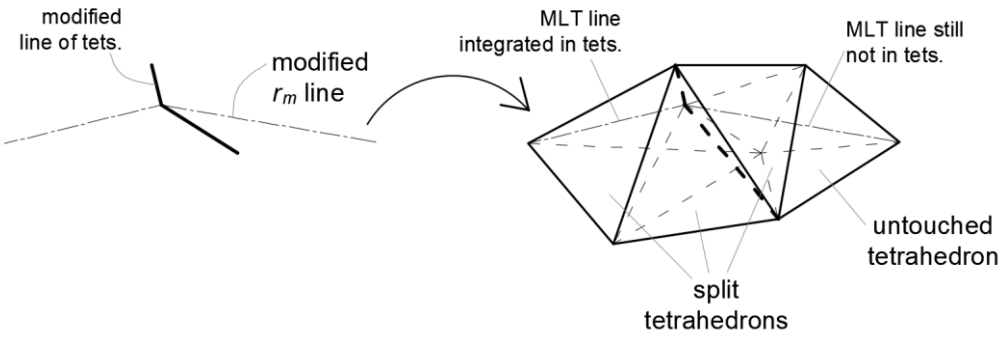


Fig. 6.32 Insertion at tetrahedron and *MLT* lines.

[44] See Appendix 2A for computation of closest point from one line to another line.

In view of **Fig. 6.32** one can see why the insertion must occur on first or last *hinge* lines (r_1 or r_e). By inserting in one of them we guarantee that at least one of the new *MLT* lines is included in the tetrahedrons after the node is inserted.

Insertion of new node in tetrahedrons

In case the guided insertion is not possible, the new node is inserted using the preliminary location ($\boldsymbol{\xi}_p$) at the mid-point of r_m. The coordinates of the new node $\boldsymbol{\xi}_i$ (calculated as Appendix 5D) may lie far away from $\boldsymbol{\xi}_p$, even within tetrahedrons different from those intersected by the r_m line. The tetrahedron where $\boldsymbol{\xi}_i$ lies is identified and three insertions are preliminarily simulated:

- $\boldsymbol{\xi}_i$ is inserted within the tetrahedron, dividing it into four;
- $\boldsymbol{\xi}_i$ is inserted within the closest facet of the tetrahedron, dividing each of the attached tetrahedrons to that facet into 3;
- $\boldsymbol{\xi}_i$ is inserted within the closest line of the tetrahedron, dividing each of the attached tetrahedrons to that line into 2.

The regularity (see Appendix 6B) of the resultant tetrahedrons in each scenario is measured, storing the lowest value for each case[45]. The three regularities are compared and the scenario with higher regularity is chosen, inserting the node in the tetrahedral mesh. The r_m line of the *MLT* is split by that new node.

This procedure does not guarantee that any stretch of the new *MLT* line becomes part of the updated tetrahedralization. In addition, it is prone to generate small or distorted tetrahedrons. For these two reasons, this procedure is the latest in the list to use.

6.4.4. Removal of non-valid tetrahedrons

One tetrahedron is non-valid if it lies outside the computable domain, or it is quasi-plane. **Fig 6.33** illustrates outside tetrahedrons (references 1 and 2) and quasi-plane tetrahedrons (references 3 and 4) represented by grey thick lines. 2D view is shown for clarity. Outside tetrahedrons appear on concave sides of *MLT*. Plane tetrahedrons appear because their four nodes should belong to the same plane but numerical truncation errors moved them slightly, therefore Delaunay algorithm interprets them as not in plane.

[45] The regularity is $R \leq 1$, being equal to 1 for the regular tetrahedron (the best case). The more distorted the tetrahedron the lower value of R.

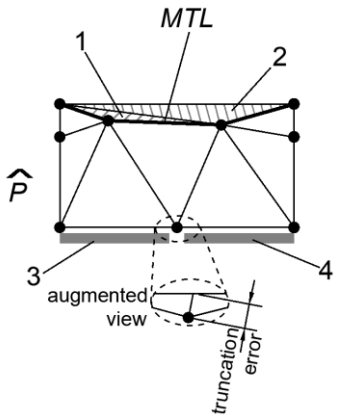


Fig 6.33 Non-valid tetrahedrons.

Outside tetrahedrons are detected similarly to outside nodes (see section 6.4.1). The $\boldsymbol{\xi}_a$ location used in this case is the centroid of the tetrahedron. Plane tetrahedrons are detected because they regularity is smaller than a threshold (0.01 in this work).

6.4.5. Insertion of mid-nodes and correction of self-intersections

Let us use the prefix Q for nodes and lines that lie on Q-surfaces. The rest of nodes / lines may be preceded by the term *inner*. By the mid-nodes insertion in the Q-lines we recover the accuracy of the quadratic $\bar{Q}$-surfaces (recall Chapter 5). The insertion of new nodes is according to Appendix 5D. The preliminary position in $\hat{P}$-space ($\boldsymbol{\xi}_p$) is the mid-location of the line to upgrade.

After mid-nodes insertion, self-intersections may occur between inner lines and Q-lines, **Fig. 6.34** illustrates one scheme in 2D.

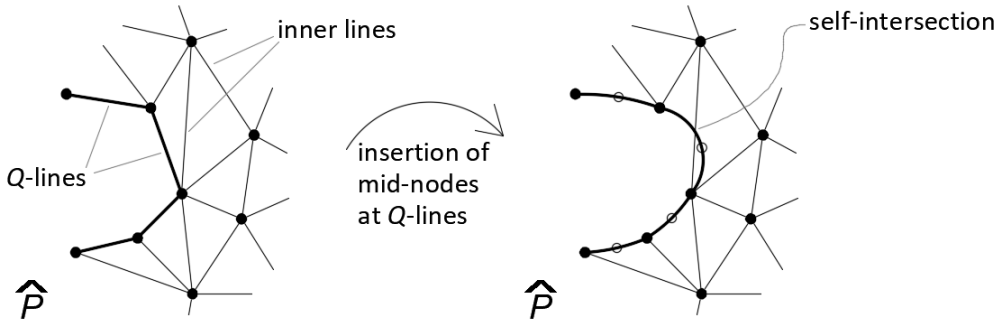


Fig 6.34 Self-intersection due to degree elevation of Q-lines.

To check the self-intersections at each Q-node, the tangent vectors for the inner lines (inner tangents) and for the Q-lines (Q-tangents) are calculated. The philosophy is that those inner tangents that lie in the outer side of the $\bar{Q}$-surface are corrected, i.e. reoriented, and move back to the computable domain. The process to check each Q-node is described here.

With the Q-tangents we form a *crown* of triangles convergent to the Q-node as shown in **Fig. 6.35**. Each Q-tangent forms a line between two triangles (**Fig. 6.35** (b)). The normal vector ($\boldsymbol{n}_i$) to each triangle is computed such that it is oriented to the outside of the $\bar{Q}$-surface (**Fig. 6.35** (c)).

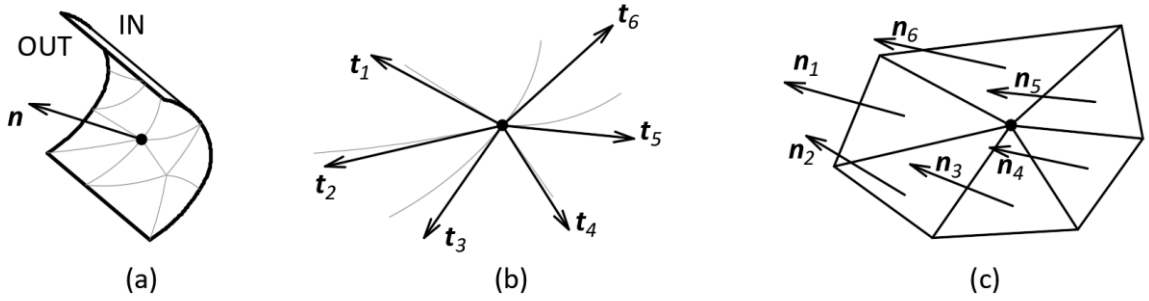


Fig 6.35 Generation of crown of triangles at one Q-node.

Each inner tangent lies on one triangle of the crown[46]. Let us assume the ith-inner tangent $\boldsymbol{t}_i$, lies on the jth-triangle with normal $\boldsymbol{n}_j$. The angle α_i is defined as the difference $\alpha_i = \hat{\boldsymbol{n}}_j - \hat{\boldsymbol{t}}_i$. We define lower and upper limit angles $\alpha_{L1} = 95$ degrees[47] and $\alpha_{L2} = 120$ degrees. One example is illustrated in **Fig. 6.36**, where three inner lines lie on the triangle 6. The α angles of those lines and the limit angles are depicted in **Fig. 6.36** (b). We define Δ angles of each inner tangent as the difference $\Delta_i = \hat{\boldsymbol{t}}_i - \alpha_{L2}$ and the limits increment angle as $\Delta_L = \alpha_{L1} - \alpha_{L2}$ (see **Fig. 6.36** (c)).

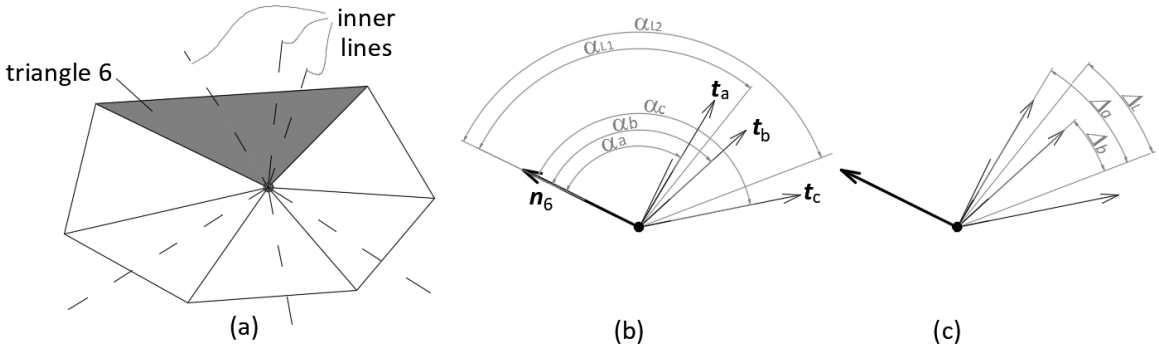


Fig 6.36 Angles relative to the crown triangle normal vector and Δ angles used for correction.

The condition to re-orientate a set of inner tangents attached to a Q-node is that $\alpha_i < \alpha_{L1}$ for at least one of the inner tangents. In that case only the inner lines whose tangent angle is $\alpha_i < \alpha_{L2}$ are corrected, these ones are called *deviated tangents*. The rest of the inner lines remain untouched. To carry out the re-orientation, the Δ angles are used. We define the maximum increment as (6.16), where α_{imin} is the closest angle to $\boldsymbol{n}_j$ of the *deviated tangents*.

[46] Precisely speaking, the projection of the inner tangent lie on one triangle.
[47] Note there is an allowance of 5 degrees.

$$\Delta_{max} = \alpha_{L2} - \alpha_{imin} \tag{6.16}$$

Fig. 6.37 shows three examples. In **Fig. 6.37** (a) there is no correction to do. In **Fig. 6.37** (b) tangent $\boldsymbol{t}_a$ is closer to $\boldsymbol{n}_j$ than the limit α_{L1}, hence all tangents closer than α_{L2} are to be corrected ($\boldsymbol{t}_a$ and $\boldsymbol{t}_b$). In **Fig. 6.37** (c) there are two tangents closer than α_{L1}, choosing the closest to $\boldsymbol{n}_j$ to obtain Δ_{max}. Tangents to correct are $\boldsymbol{t}_a$, $\boldsymbol{t}_b$ and $\boldsymbol{t}_c$ in the latest case.

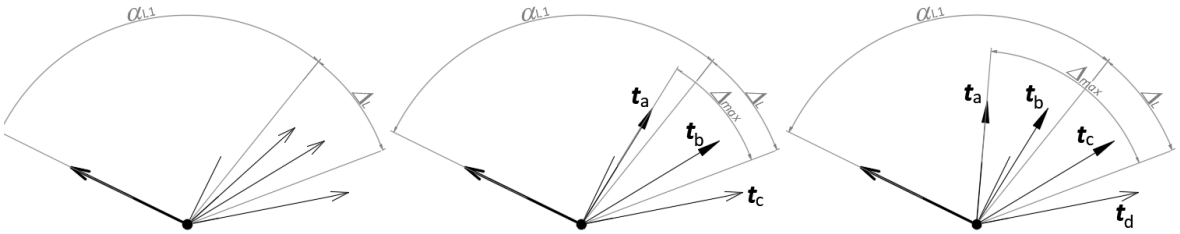


Fig 6.37 Different cases of inner lines tangents.

The corrected angles (Δ_i^c) are linearly scaled from (Δ_i) as (6.17), which moves them inside the computable domain. Then the corrected angle w.r.t. the normal $\boldsymbol{n}_j$ is obtained as (6.18).

$$\Delta_i^c = \Delta_i \frac{\Delta_L}{\Delta_{max}} \tag{6.17}$$

$$\alpha_i^c = \alpha_{L2} - \Delta_i^c \tag{6.18}$$

The deviated tangent vectors are rotated to achieve the corrected orientation (α_i^c) but keeping the same norm. The mid-node is inserted in each of deviated lines to attain the corrected tangent, which is the first derivative of the curve formed by de inner line. For each inner line, since the end-nodes are known (they remain in the same position) as well as the derivative at the initial position (the corrected tangent), the mid-node coordinates may be obtained from (6.19), that is deducted from the quadratic lines derivative detailed in Appendix 6D.

$$\boldsymbol{\xi}_2^i = \frac{1}{2}\left(\boldsymbol{t}_i^c + \frac{3}{2}\boldsymbol{\xi}_1^i + \frac{1}{2}\boldsymbol{\xi}_3^i\right) \tag{6.19}$$

In **Fig. 6.38** one bi-dimensional version of the process is illustrated. In (a) the mesh previous the correction is shown, (b) shows the normal vectors to the triangles of the crown and the inner tangents prior to correction. Since tangent $\boldsymbol{t}_a$ is closer to the normal vector of the triangle than

α_{L1}, the tangents $\boldsymbol{t}_a$, $\boldsymbol{t}_b$ and $\boldsymbol{t}_d$ need to be reoriented, as shown in (c). Finally, the corrected mesh is given in (d), where the self-intersection has disappeared.

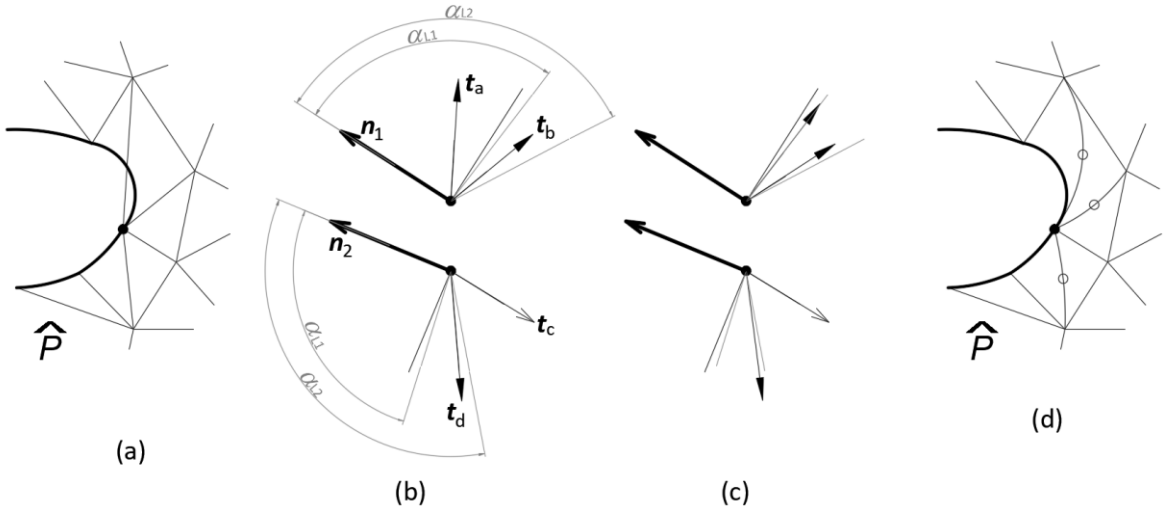


Fig 6.38 Correction of self-intersecting inner lines (2D version).

6.5. Control points considered in the patch integration

Once the tetrahedralization is finished, the Gauss points are allocated: one Gauss point to the linear tetrahedron and four Gauss points to the mixed degree tetrahedrons. Those control points of the gross patch whose influential volume does not enclose any Gauss point are non-active, being not considered in the stiffness matrix calculation.

Fig. 6.39 shows one example in two dimensions and two degrees of freedom, with twelve control points. The first case is non-trimmed, hence all the control points are included in the stiffness matrix, whose size is 24 x 24. Second case is trimmed and some control points are removed for the stiffness matrix calculation, which have lesser number of components (16 x 16).

It is necessary to remove the non-active control points because they introduced zero rows that would make the stiffness matrix singular and therefore a non-solvable problem.

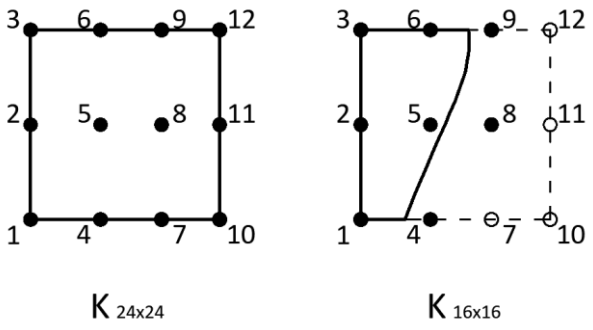


Fig. 6.39 Active control points (filled dots) for untrimmed and trimmed patches.

6.6. Example

This section provides one example of the discretization of trimmed patches. Full data of the example, which is the same as section 5.6, is given in Appendix 5E. **Fig. 6.40** illustrates the gross-trimmed patch with the knot spans and the final trimmed patch.

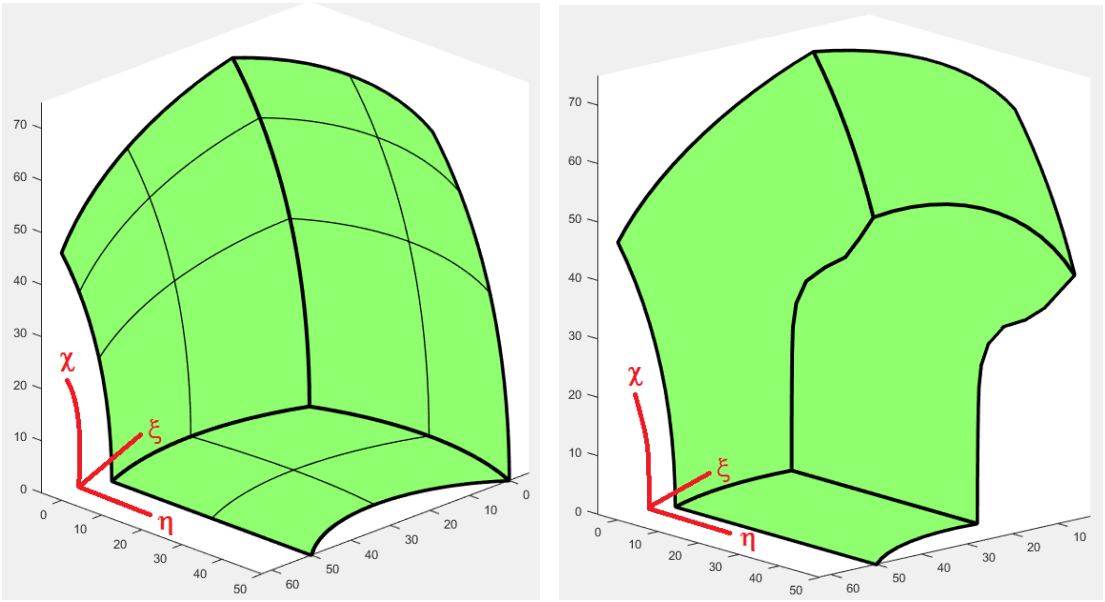


Fig. 6.40 Gross-trimmed patch and trimmed patch.

6.6.1. Gross-trimmed patch tetrahedralization

The cells and wedges to insert are shown in **Table 6.3**, where references of equations used are indicated.

Table 6.3 Insertion of cells and wedges.

Parameter direction	p	n (6.4)	Non-void knot spans	NR (6.5)	NC (6.6)	NW (6.7)
1	2	3	0 0.5 1	6	2	5
2	2	3	0 0.5 1	6	2	5
3	2	3	0 0.5 0.8 1	9	4	8

The alternating direction is the first, meanwhile the second and third are the perpendicular directions (the alternating direction could have been the second also). The nodes at the π_p plane and the alternating direction (dashed line) are shown at the left-hand side of the **Fig. 6.41**. The resulting tetrahedrons are depicted at the right-hand side of the figure.

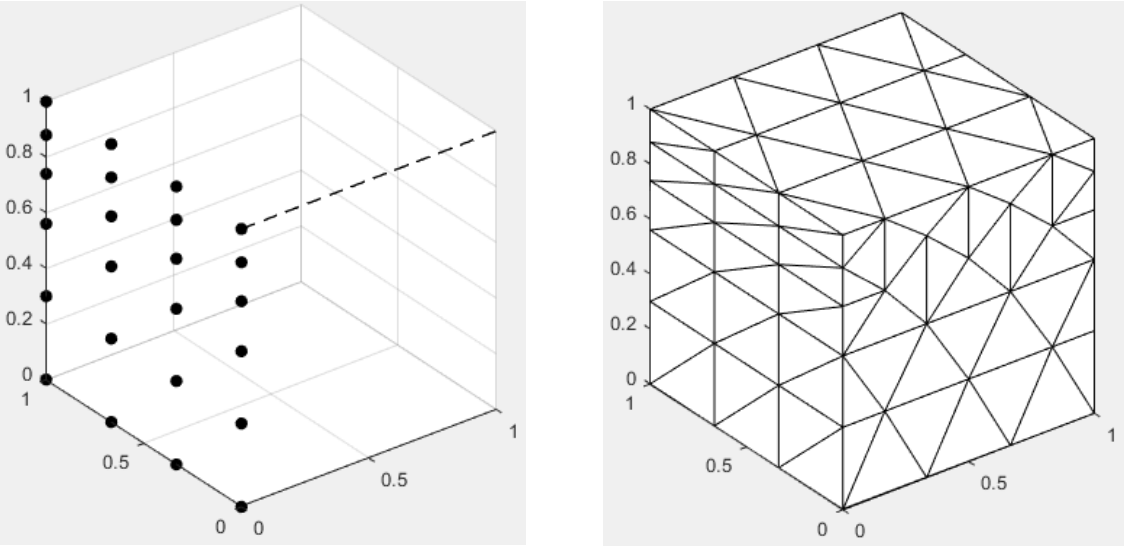


Fig. 6.41 Left: nodes in the π_p plane and the alternating direction (dashed line). Right: tetrahedral mesh.

6.6.2. Merged linear triangulations and selection of nodes

The $\bar{Q}$-surfaces are merged producing the *MLT*, as described in section 6.3.2 (see **Fig. 6.42**). The nodes from the previous stage that lie outside the computable domain or too close to the *MLT* are removed (hollow dots in **Fig. 6.43** left). The resulting arrangement lacks of nodes in the vicinity of the *MLT* (**Fig. 6.43** centre), hence new nodes are added at 0.40 the optimal distance from each triangle of the *MLT* (**Fig. 6.43** right).

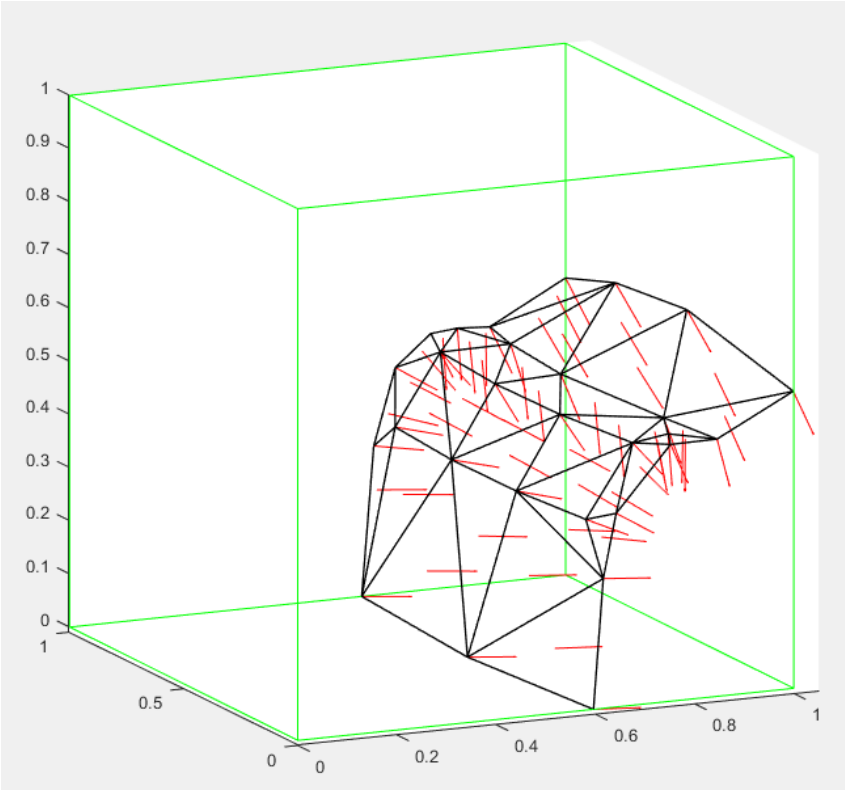


Fig. 6.42 *MLT* with normal to triangles and nodes.

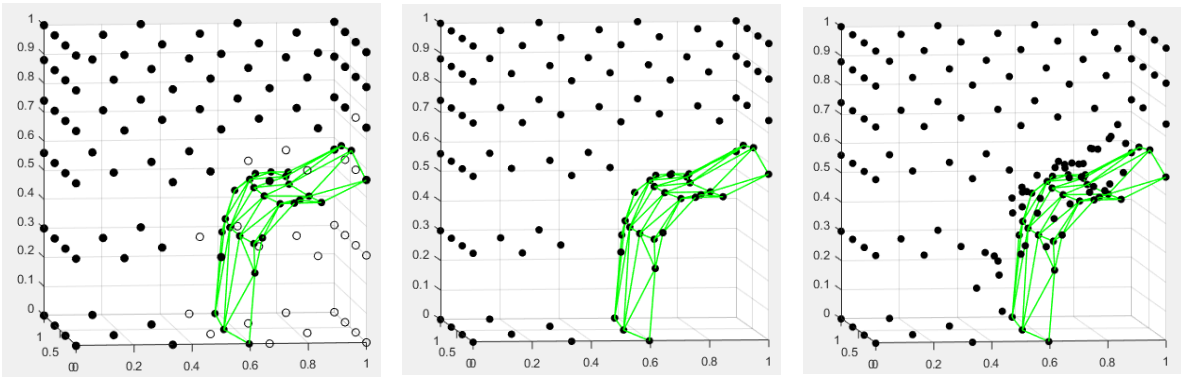


Fig. 6.43 Setting of nodes prior to the tetrahedralization.

6.6.3. Tetrahedralization

With all the nodes in, including the nodes from *MLT*, the tetrahedralization is done. Recall that to include all the *MLT* lines in the tetrahedrons new nodes might be inserted, being necessary the amendment of the tetrahedralization and the *MLT* (section 6.4.3). **Fig. 6.44** shows the original *MLT* and the updated version after the new nodes are inserted during the amendment.

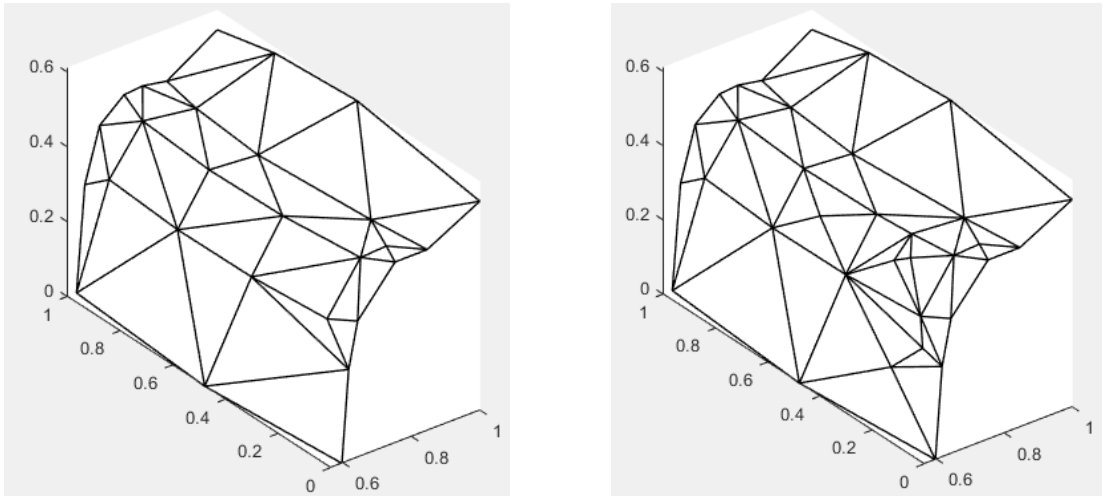


Fig. 6.44 *MLT* before and after the amendment process.

The left-hand side of **Fig 6.45** illustrates the tetrahedral mesh prior to the removal of non-valid tetrahedrons (section 6.4.4). At the right-hand side the final linear tetrahedral mesh is shown.

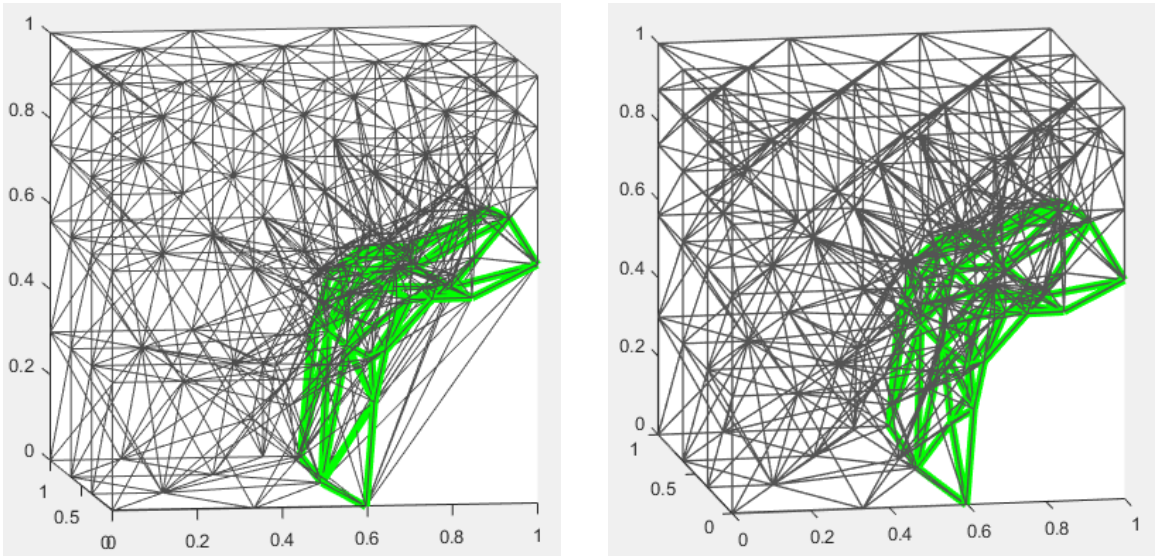


Fig. 6.45 Tetrahedral mesh before and after removal of non-valid tetrahedrons. The *MLT* is in thick lines.

The lines of the tetrahedral mesh that lie on the MLT are raised to degree two by insertion of a new node at the mid-location. Some self-intersections appear as a consequence of this upgrade (section 6.4.5). **Fig 6.46** shows the tetrahedral mesh indicating one node where self-intersection appears. In **Fig. 6.47** the lines at that node are detailed, being the Q-lines the thicker and the inner lines the thinner. Inner lines *a* and *b* form an angle smaller than 95 degrees with the normal vector of the corresponding triangle, therefore need to be re-orientated.

After the insertion of mid-nodes to the required inner lines, the self-intersections disappear. **Fig. 6.48** shows the inner lines *a* and *b* re-oriented and curved since they are now quadratic.

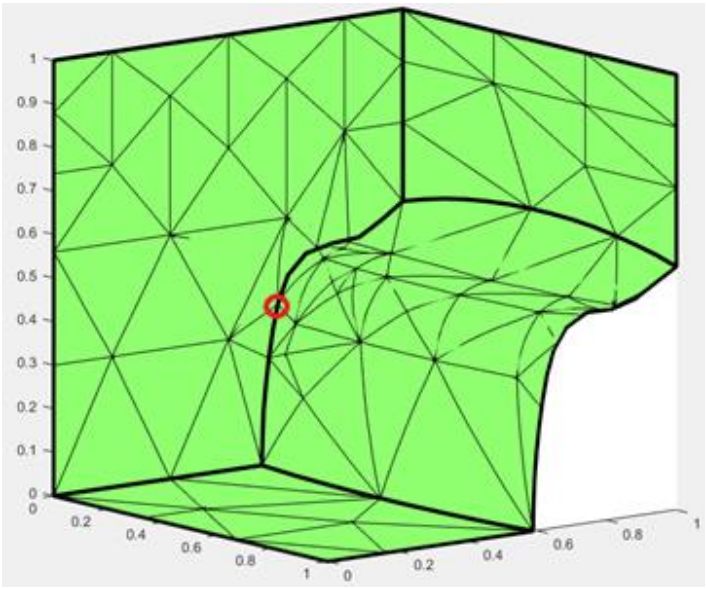


Fig. 6.46 Tetrahedral mesh and location of one self-intersection (red circle).

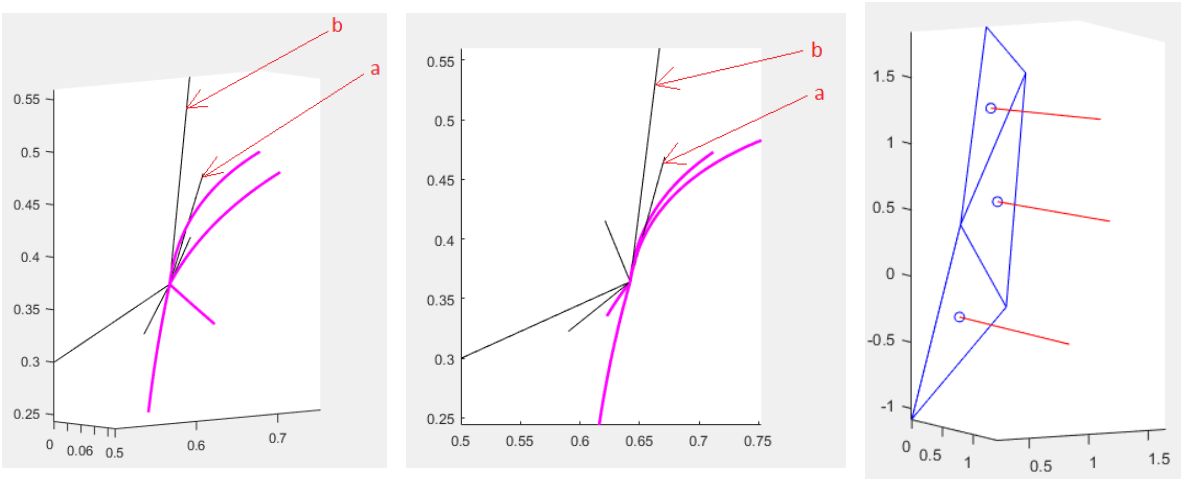


Fig. 6.47 Inner lines *a* and *b* need to be re-oriented. Left: view from bottom-side. Centre: lateral elevation. Right: crown of triangles and their normal vectors.

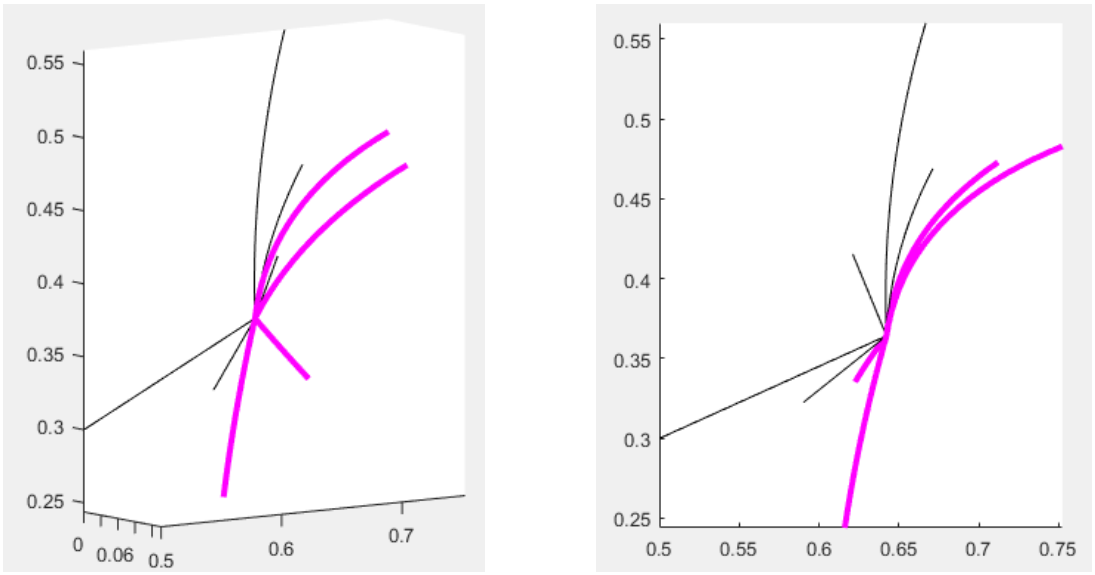


Fig. 6.48 Inner lines *a* and *b* re-oriented. Left: view from bottom-side. Centre: lateral elevation

The resultant tetrahedral mesh is represented in the patch parameter space and in the physical space, in **Fig. 6.49** and **Fig 6.50** respectively.

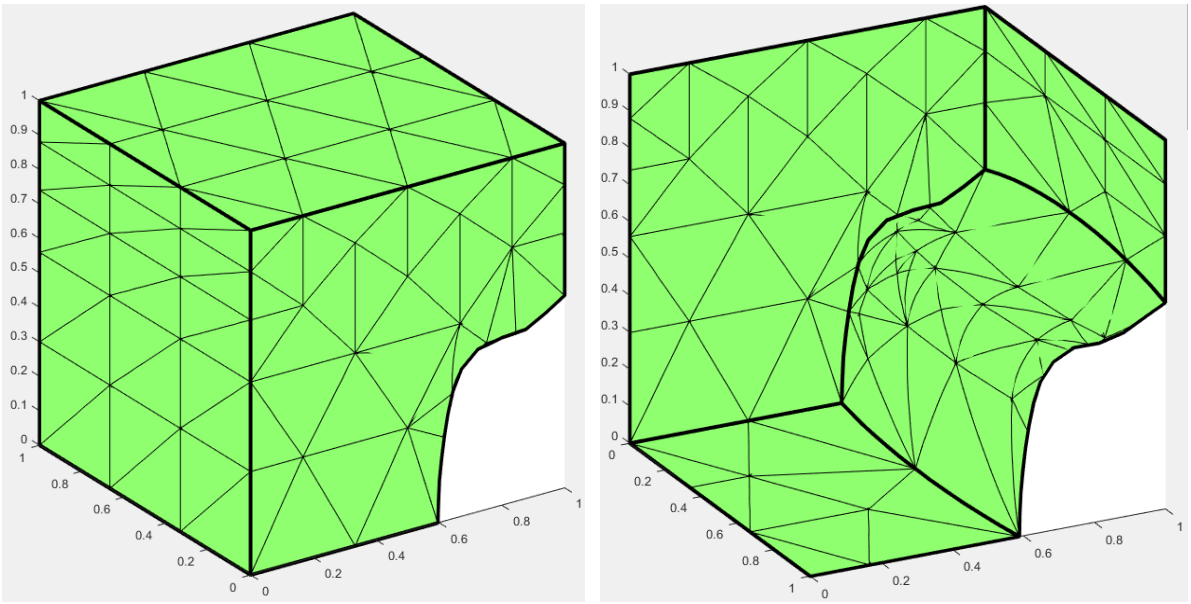


Fig. 6.49 Resultant tetrahedralization in $\hat{P}$-space.

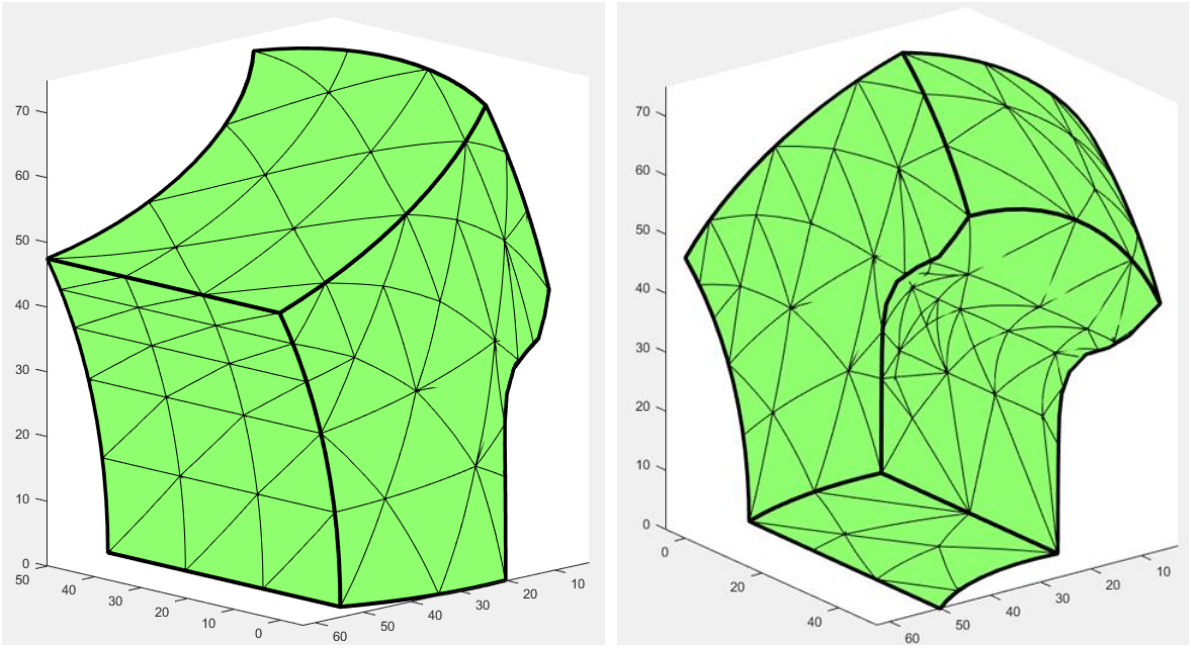


Fig. 6.50 Resultant tetrahedralization in physical space (P-space).

6.7. Relation with the code

The tetrahedralization of the patches is done within the `im0141_IMesh` routine, that includes the `rm0161_Rmesh` routine that discretizes the un-trimmed patches (section 6.1). The trimmed patches follow a different route as indicated here.

The `im0158_GPT` routine calculates the nodes for the gross-trimmed patch (section 6.2). The `mt0601_MLT` routine obtains the *MLT* (section 6.3). The removal of outside nodes (section 6.4.1)

is carried out in `im0146_SetAllNods`. Tetrahedralization itself (sections 6.4.3 to 6.4.5) is done in `im0147_TetMesh` routine. The Gauss points are set in `im0144_patchGPs`.

The main inputs of this tetrahedralization process in the code are:

- The trimming surfaces (`rTskin`), obtained as output of `st0301_rTskin` (Chapter 5).
- The number of required points in each parameter direction (`NGd`) that is the output of `rm0162_NRgp` routine.

The output of `im0141_IMesh` routine is the tetrahedral mesh used for integration. This mesh is stored in `IM`, which is a member of the patches object (`Spatches`).

6.8. Summary of the chapter

In this chapter the discretization of the solids is explained, for both non-trimmed and trimmed patches. The former is done as standard IGA, with knot spans acting as hexahedral elements. Trimmed domains are discretized into tetrahedrons instead. Tetrahedrons adapt to any shape achieving quadratic degree at their facets that lie on the trimming surfaces. As result some of the them become mixed-degree tetrahedrons. With the tetrahedral mesh at hand, the location of Gauss points can be set, which allows the calculation of the stiffness matrix and body forces. This chapter, together with Chapter 5, covers the thesis objective **d**.

7. **Constraints and analysis**

This chapter is focused on the construction of the *aggregated system of equations*[48], whose solution is the vector of the displacements at control points. With that vector the displacements and stresses at any location of the domain can be computed[49].

An extraction of the general scheme from section 3.2 is shown in **Fig. 7.1**. To construct the aggregated system, one needs on one hand the discretized solid patches, which comes from the stage C (Chapter 5 and Chapter 6). On the other hand, the boundary surfaces are also required. These surfaces, which were identified in stage A (Chapter 1), need to be discretized to allow the integration of the patch basis functions on them[50].

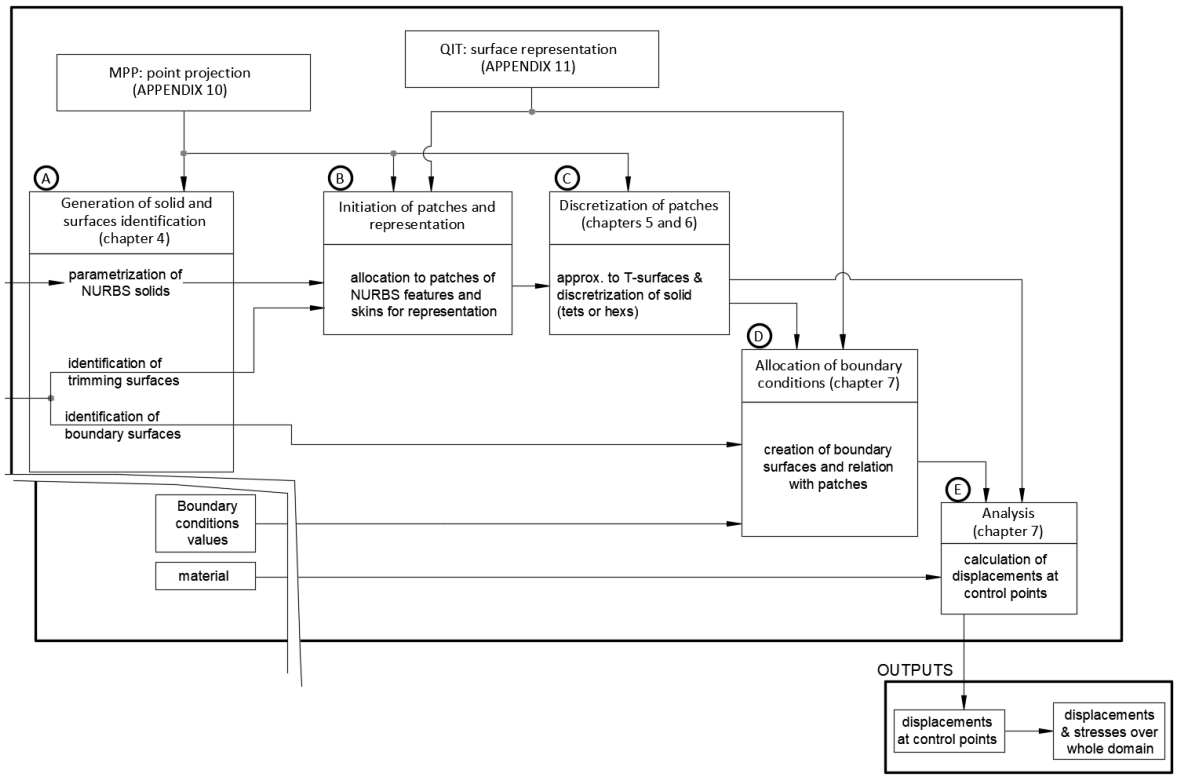


Fig. 7.1 Extraction of main scheme.

This chapter, which covers stages D and E, is structured as follows. Section 7.1 presents the integration of the patch basis functions on the boundary surfaces. Section 7.2 explains in detail the calculation of the aggregated system of equations. Sections 7.1 and 7.2 use the approach presented by Apostolatos *et al.* (2014), that is introduced in Appendix 2B. Section 7.3 relates this chapter to the main routines and variables of the code. Finally, section 7.4 summarizes the main items introduced by this chapter. No example is given since Chapter 8 provides three examples to validate the analysis outputs of this algorithm.

[48] See equation (2B.54) of Appendix 2B.
[49] Refer to Appendix 7A.
[50] See equations (2B.45) and (2B.53) in Appendix 2B.

7.1. Integration on boundary surfaces

7.1.1. Gauss points on boundary surfaces

To account for constraints the basis functions of the patches are to be integrated on the boundary surfaces. These integrals are estimated by Gauss quadrature, therefore a set of Gauss points must be defined on the boundary surfaces.

The boundary surfaces are in general trimmed by contour curves (recall section 2.2.3). Therefore, their discretization according to the knot spans is not valid[51]. In this work we discretize the computable boundary surfaces into triangles using the *QIT* algorithm (detailed in Appendix 10B). The size of the triangles and number of Gauss point per triangle is selected by the user. After *QIT* is applied, the Gauss points are located within each triangle[52]. **Fig. 7.2** illustrates one example.

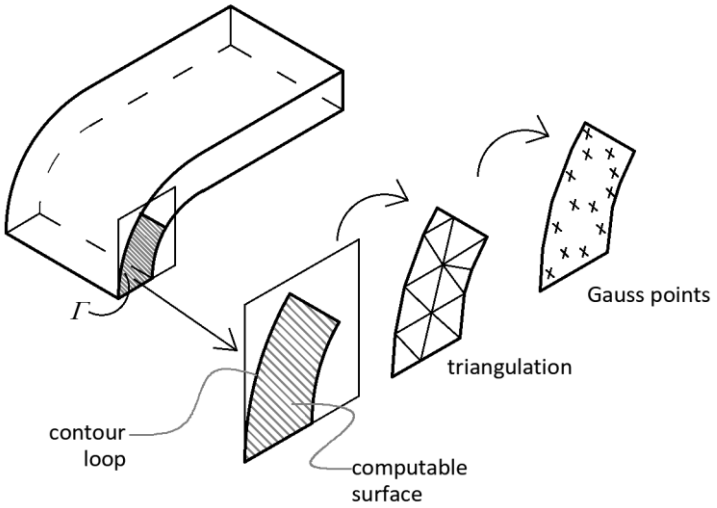


Fig. 7.2 Allocation of Gauss points to boundary surfaces.

7.1.2. Spaces involved

The integration on the boundary surfaces involves three spaces: the physical space, where the surface (Γ) meets the patch (Ω); the surface parameter space ($\hat{\Gamma}$) and the triangle parent space ($\tilde{\Gamma}$). **Fig. 7.3** shows these spaces for one surface.

[51] It is the same situation for trimmed solids (Chapter 6), but for surfaces.
[52] Appendix 6A details the arrangement of the Gauss points within the different parent spaces.

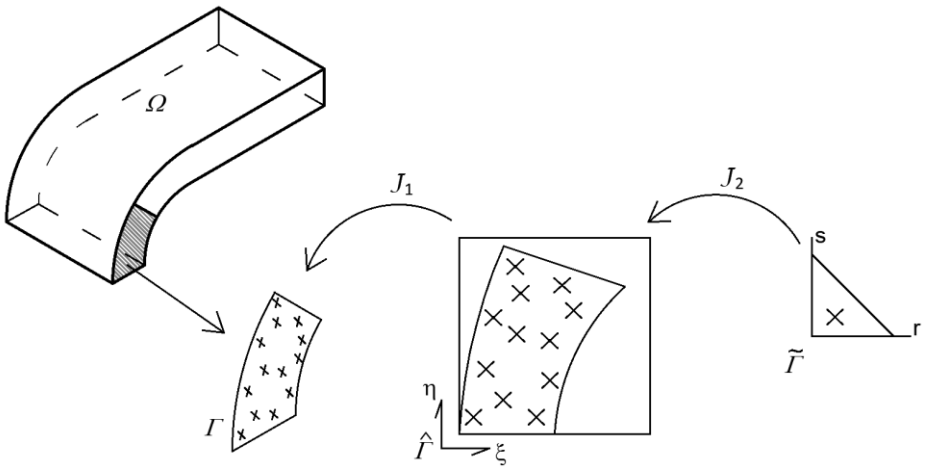


Fig. 7.3 Involved spaces in the integration of the boundary surfaces.

To transform the integral limits from the physical space to the parent space two Jacobians are needed as shown in **Fig. 7.3**: from physical to parameter ($\mathcal{J}_1$) and from parameter to parent ($\mathcal{J}_2$).

7.1.3. Integration on boundary surfaces

The function B to integrate over the boundary surfaces is R_i or $R_i R_j$, where R_i is the basis function of the ith control point of the patch[53]. This integral $\mathbb{I}_b$, whose domain is Γ, is shown in (7.1), where the sub-index b refers to one basis function i or two basis functions i and j.

$$\mathbb{I}_b = \int_{\Gamma} B_b \, d\Gamma \tag{7.1}$$

In order to facilitate the calculation of (7.1), the integration domain is changed to $\tilde{\Gamma}$, which introduces the Jacobian $\mathcal{J}_1$ and $\mathcal{J}_2$[54] as follows:

$$\mathbb{I}_b = \int_{\tilde{\Gamma}} B_b \, \mathcal{J}_1 \, \mathcal{J}_2 \; d\tilde{\Gamma} \tag{7.2}$$

The integral (7.2) is approximated by Gauss quadrature as follows:

$$\mathbb{I}_b \cong \sum_{g=1}^{{NG_b}^{\alpha}} {B_b}_g \, {\mathcal{J}_1}_g \, {\mathcal{J}_2}_g \, w_g \tag{7.3}$$

Where ${NG_b}^{\alpha}$ is the number of Gauss points of the αth boundary surface within the influential volume of the bth basis function; w_g the Gauss point weight; and the basis function and Jacobian are evaluated at the Gauss points. One example is illustrated in **Fig. 7.4**, that shows the Gauss points influenced by the bth basis function.

[53] See equations (2B.45) and (2B.53) of Appendix 2B.
[54] See Appendix 2F for calculation of Jacobian.

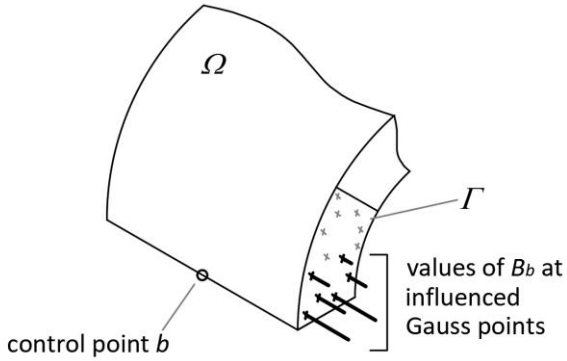


Fig. 7.4 Basis function values at influenced Gauss points.

7.1.4. Influenced Gauss points

The estimation of the integral of the bth basis function (7.3) requires the summation of the Gauss points that lie within the basis function limits. Since the limits of one basis function forms a cubic region in the patch parameter space, we use such space to identify influential Gauss points on one basis function.

The position of the Gauss point in the patch parameter space must be calculated by point projection[55]. One example is provided in **Fig. 7.5**, where the thicker crosses indicate those Gauss points within the influential volume of the bth control point. This example corresponds to **Fig. 7.4**, where the basis function values at Gauss points are depicted.

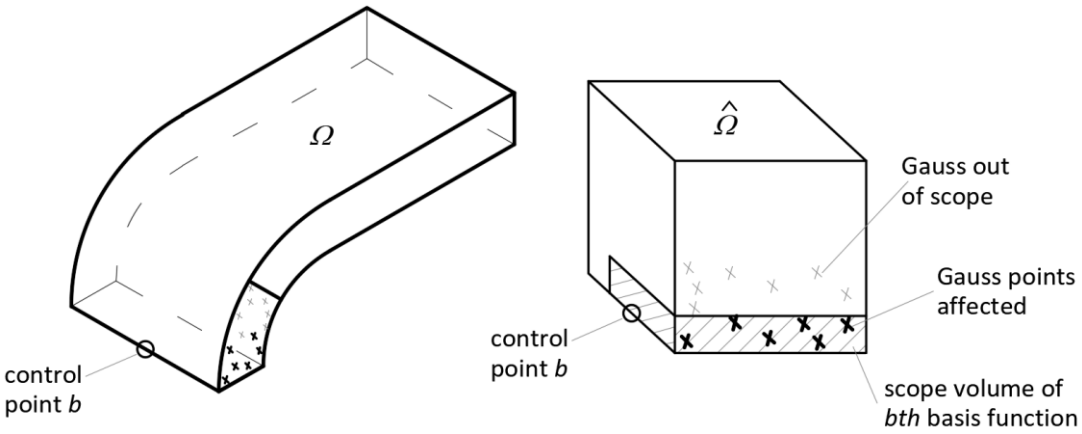


Fig. 7.5 Gauss points influenced by basis function b in the physical and parameter spaces.

[55] See Appendix 10A where the projection technique *MPP* is detailed.

7.2. Construction of the aggregated systems of equations

The *aggregated system* of equations (7.4) considers the constraints on the domain, being its solution the vector of displacements of the control points. The equation (7.4) is obtained from the discretization of the equilibrium equation as detailed in Appendix 2B[56].

$$\begin{bmatrix} \boldsymbol{K} & [\boldsymbol{H}_D \quad \boldsymbol{H}_C] \\ \begin{bmatrix} \boldsymbol{H}_D{}^T \\ \boldsymbol{H}_C{}^T \end{bmatrix} & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \hat{\boldsymbol{u}} \\ \hat{\boldsymbol{\lambda}}_D \\ \hat{\boldsymbol{\lambda}}_C \end{Bmatrix} = \begin{Bmatrix} \hat{\boldsymbol{f}} \\ \hat{\boldsymbol{u}}_D \\ \hat{\boldsymbol{u}}_C \end{Bmatrix} \tag{7.4}$$

where:

$\boldsymbol{K}$ is the patches stiffness matrix;

$\boldsymbol{H}_C$ is the matrix of coupling constraints;

$\boldsymbol{H}_D$ is the matrix of Dirichlet constraints;

$\hat{\boldsymbol{u}}$ is the displacement vector at control points, that is unknown;

$\hat{\boldsymbol{\lambda}}_D$ and $\hat{\boldsymbol{\lambda}}_C$ are the Lagrange multipliers vectors at control points for Dirichlet and coupling conditions respectively, that are unknown;

$\hat{\boldsymbol{f}}$ is the force vector at control points;

$\hat{\boldsymbol{u}}_D$ is the prescribed displacement vector at control points;

$\hat{\boldsymbol{u}}_C$ is the coupling relative displacement vector at control points, which is zero in this work.

The deactivated control points (section 6.5), for trimmed patches, are not considered in (7.4). The computation each item of (7.4) is explained in detail in this section. Previously some definitions are introduced.

7.2.1. Definitions

Given a constrained domain with multiple coupled patches, we set the next definitions:

NP: number of patches.

NC: number of couplings.

ND: number of Dirichlet boundary conditions.

NN: number of Neumann boundary conditions.

n^I: number of active control points of the Ith patch.

nc^{IJ}: number of control points of the Ith patch that are influential on the Jth coupling.

nd^{IJ}: number of control points of the Ith patch that are influential on the Jth Dirichlet constraint.

[56] Equation (7.4) is equivalent to equation (2B.54) from Appendix 2B.

nn^{IJ}: number of control points of the Ith patch that are influential on the Jth Neumann constraint.

NG^{I}: the number of Gauss points of the Ith patch.

Fig. 7.6 illustrates these definitions with a domain composed of three patches (2D version is shown for clarity). At the left-hand side the whole domain is represented. At the right-hand side the 3rd patch is detailed showing the control points and Gauss points. The influential control points on each boundary entity are filled.

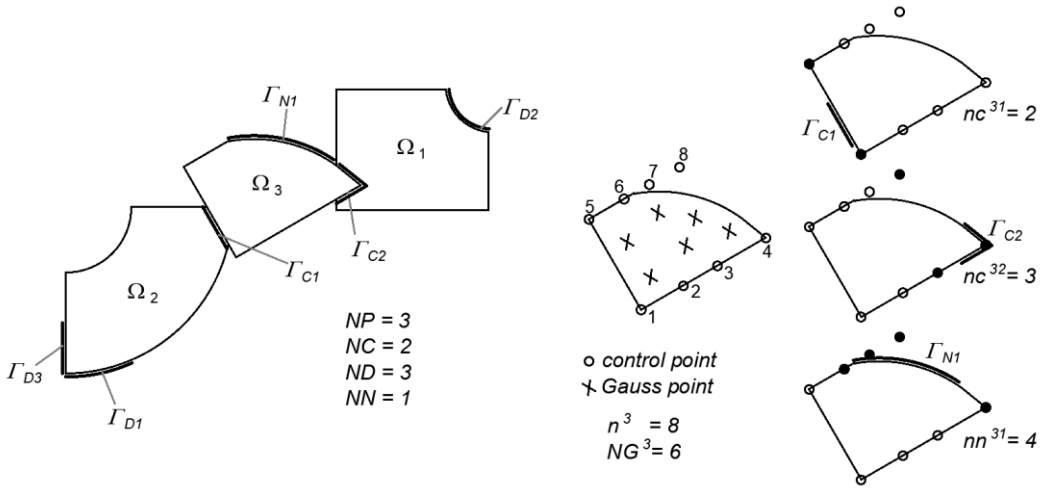


Fig. 7.6 Example of multi-patched coupled domain (2D version).

The matrices ($\boldsymbol{K}$, $\boldsymbol{H}_D$ and $\boldsymbol{H}_C$) and vectors ($\widehat{\boldsymbol{u}}$, $\widehat{\boldsymbol{\lambda}}_D$, $\widehat{\boldsymbol{\lambda}}_C$, $\widehat{\boldsymbol{f}}$, $\widehat{\boldsymbol{u}}_D$ and $\widehat{\boldsymbol{u}}_C$) shown in equation (7.4) are composed of blocks, components and scalars as listed here:

- One block corresponds to one patch and/or constraint. Each block is made of components.
- One component corresponds to one or two control points. Each component has $d\ x\ d$ scalars for matrices and $d\ x\ 1$ scalars for vectors[57].

Fig. 7.7 shows an example that includes one matrix $\boldsymbol{M}$ with $3x2$ blocks, and one vector $\boldsymbol{V}$ with $3x1$ blocks. Some blocks may be zeros (for example blocks 1,1 and 2,2 of $\boldsymbol{M}$) but their dimensions must be conformal with the surrounding blocks (see number of components of block 2,2).

[57] d is the number of degrees of freedom of each control point, $d = 3$ in this thesis.

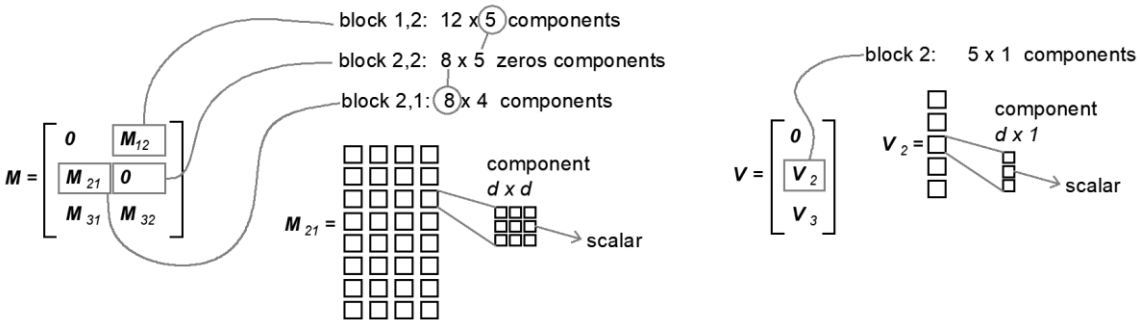


Fig. 7.7 Decomposition of matrices and vectors in blocks, components and scalars.

The components of the constraints matrices ($\boldsymbol{H}_D$ and $\boldsymbol{H}_C$) are designated as ${\boldsymbol{R}_{ij}}^{AB}$. Each of these components has the form (7.5), where A and B are patches references and i and j refer to basis functions of those patches.

$$ {\boldsymbol{R}_{ij}}^{AB} = \begin{bmatrix} {R_i}^A\, {R_j}^B & & \\ & {R_i}^A\, {R_j}^B & \\ & & {R_i}^A\, {R_j}^B \end{bmatrix} \tag{7.5} $$

7.2.2. Stiffness matrix $\boldsymbol{K}$

The stiffness matrix $\boldsymbol{K}$ of the domain is composed of $NP\ x\ NP$ blocks, which are all zeros except the diagonal blocks as shown in equation (7.6). The Ith diagonal block, called $\boldsymbol{K}^I$, corresponds to the Ith patch. The number of components of $\boldsymbol{K}^I$ is $n^I\ x\ n^I$.

$$ \boldsymbol{K} = \begin{bmatrix} \boldsymbol{K}^1 & & \\ & \ddots & \\ & & \boldsymbol{K}^{NP} \end{bmatrix}_{NP\ x\ NP} \tag{7.6} $$

For the Ith patch, the $i-jth$ component is calculated as follows:

$$ {\boldsymbol{K}_{ij}}^I = \int_\Omega {\boldsymbol{B}_i}^{I^T} \boldsymbol{D}\, {\boldsymbol{B}_j}^I \, \mathrm{d}\Omega \tag{7.7} $$

This integration is approximated by Gauss rule as equation (7.8), where ${NG_{ij}}^I$ is the number of Gauss points **of the patch** that lie within the influence of basis functions i and j. The strain-displacement matrix, constitutive matrix, Jacobian and weight are located at the gth Gauss point.

$$\boldsymbol{K}_{ij}{}^{I} \cong \sum_{g=1}^{NG_{ij}{}^{I}} \left(\boldsymbol{B}_i{}^{I^T} \boldsymbol{D}\, \boldsymbol{B}_j{}^{I} \right)_g \mathcal{J}_{1_g}\, \mathcal{J}_{2_g}\, w_g \tag{7.8}$$

7.2.3. Coupling constraints matrix $\boldsymbol{H}_C$

The matrix of coupling constraints $\boldsymbol{H}_C$ is composed of $NP\ x\ NC$ blocks.. Equation (7.9) shows the form of $\boldsymbol{H}_C$. The Ith row of $\boldsymbol{H}_C$ belongs to the Ith patch. If this patch has attached m couplings then all the blocks of the Ith row are zeros except the those one located at the m couplings locations. The calculation of these non-zeros blocks is explained here.

$$\boldsymbol{H}_C = \begin{bmatrix} \boldsymbol{H}_{C_{11}} & & \boldsymbol{H}_{C_{1\,NC}} \\ & \ddots & \\ & \boldsymbol{H}_{C_{IJ}} & \\ & \ddots & \\ \boldsymbol{H}_{C_{NP\,1}} & & \boldsymbol{H}_{C_{NP\,NC}} \end{bmatrix}_{NP\ x\ NC} \tag{7.9}$$

Let us focus on one patch A that is coupled to other patch B, being master and slave patches respectively. That coupling, referred as α, is applied on the boundary $\Gamma_{C\alpha}$. The domain is assumed to have more patches, up to NP, and more couplings, up to NC. The number of control points from both patches that are influential on $\Gamma_{C\alpha}$ are $nc^{A\alpha}$ and $nc^{B\alpha}$.

Let us define the block $\boldsymbol{H}_{C_{k\alpha}}$ (7.10) with $n^k\ x\ nc^{A\alpha}$ components, where k can refer to patch A or patch B.

$$\boldsymbol{H}_{C_{k\alpha}} = \begin{bmatrix} \vdots & & \\ \boldsymbol{0}_{1\ x\ nc^{A\alpha}} & & \\ \vdots & & \\ \boldsymbol{R}_{i1}{}^{Ak} & \dots & \boldsymbol{R}_{ic_{A\alpha}}{}^{Ak} \\ \vdots & & \\ \boldsymbol{0}_{1\ x\ nc^{A\alpha}} & & \\ \vdots & & \end{bmatrix}_{n^k\ x\ nc^{A\alpha}} \tag{7.10}$$

Each row of (7.10) has $nc^{A\alpha}$ components. The columns of (7.10) are referred locally, e.g. if the influential control points of one patch are 2, 3, 5 and 6 the columns references vary from 1 to 4 in (7.10). If one row of (7.10) corresponds to a non-influential control point on the coupling surface $\Gamma_{C\alpha}$, then all its components are zeros. When one row corresponds to one influential control point, its $nc^{A\alpha}$ components are calculated as follows:

$$\int_{\Gamma_{C\alpha}} \boldsymbol{R}_{ij}{}^{Ak}\, d\Gamma_{C\alpha} = \begin{bmatrix} \int_{\Gamma_{C\alpha}} R_i{}^{A}\, R_j{}^{k}\, d\Gamma_{C\alpha} & & \\ & \int_{\Gamma_{C\alpha}} R_i{}^{A}\, R_j{}^{k}\, d\Gamma_{C\alpha} & \\ & & \int_{\Gamma_{C\alpha}} R_i{}^{A}\, R_j{}^{k}\, d\Gamma_{C\alpha} \end{bmatrix} \tag{7.11}$$

The integral of each diagonal in (7.11) is approximated by the Gauss rule (7.3). In this case $B_b = R_i{}^{A}\, R_j{}^{k}$, therefore, the quadrature adopts the form (7.12), where $NG_{ij}{}^{\alpha}$ is the number of Gauss points on the coupling entity that lie within the influential volumes of both basis functions i and j.

$$R_{ij}{}^{Ak} \cong \sum_{g=1}^{NG_{ij}{}^{\alpha}} \left(R_i{}^{A}\, R_j{}^{k}\right)_g \mathcal{J}_{1_g}\, \mathcal{J}_{2_g}\, w_g \tag{7.12}$$

The number of columns for both patches, A and B, is equal to the number of influential control points of the master patch A since the $\hat{\boldsymbol{\lambda}}_{C\alpha}$ vector is applied to the control points of this patch.

Both blocks, $\boldsymbol{H}_{C_{A\alpha}}$ and $\boldsymbol{H}_{C_{B\alpha}}$ match the number of rows of their corresponding patches, therefore they can be inserted at kth block-row and αth block-column of the $\boldsymbol{H}_C$ matrix.

The example depicted in **Fig. 7.8** is used to clarify the $\boldsymbol{H}_C$ matrix construction. The master and slave patches are indicated for each coupling. The matrix $\boldsymbol{H}_C$ has $3x2$ blocks (three patches and two couplings).

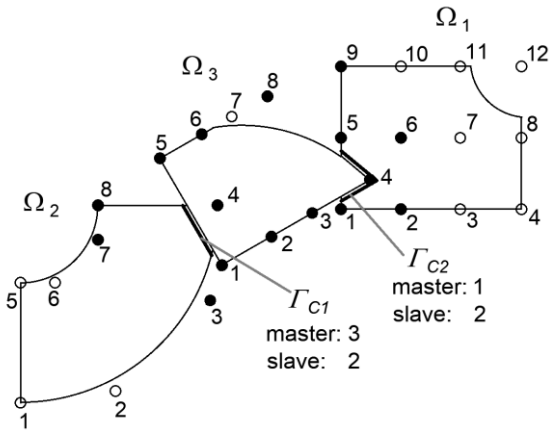


Fig. 7.8 Example of coupled patches (2D version).

The control points and sizes of the involved blocks are detailed in **Table 7.1**.

Table 7.1 Sizes of blocks of the example.

Coupling		Coupled patches		n^I	nc^{IJ}	Influential control points	$\boldsymbol{H}_{C_{k\alpha}}$ (Number of components)
J	boundary		I				
1	Γ_{C1}	Master	3	8	4	1,2,5,6	$\boldsymbol{H}_{C_{31}}$ (8x4)
		Slave	2	8	4	3,4,7,8	$\boldsymbol{H}_{C_{21}}$(8x4)
2	Γ_{C2}	Master	1	12	5	1,2,5,6,9	$\boldsymbol{H}_{C_{12}}$(12x5)
		Slave	3	8	3	3,4,8	$\boldsymbol{H}_{C_{32}}$(8x5)

Let us focus on block $\boldsymbol{H}_{C_{32}}$ that is computed as:

$$\boldsymbol{H}_{C_{32}} = \begin{bmatrix} & & \boldsymbol{0}_{1x5} & & \\ & & \boldsymbol{0}_{1x5} & & \\ \boldsymbol{R}_{3\,1}{}^{13} & \boldsymbol{R}_{3\,2}{}^{13} & \boldsymbol{R}_{3\,5}{}^{13} & \boldsymbol{R}_{3\,6}{}^{13} & \boldsymbol{R}_{3\,9}{}^{13} \\ \boldsymbol{R}_{4\,1}{}^{13} & \boldsymbol{R}_{4\,2}{}^{13} & \boldsymbol{R}_{4\,5}{}^{13} & \boldsymbol{R}_{4\,6}{}^{13} & \boldsymbol{R}_{4\,9}{}^{13} \\ & & \boldsymbol{0}_{1x5} & & \\ & & \boldsymbol{0}_{1x5} & & \\ & & \boldsymbol{0}_{1x5} & & \\ \boldsymbol{R}_{8\,1}{}^{13} & \boldsymbol{R}_{8\,2}{}^{13} & \boldsymbol{R}_{8\,5}{}^{13} & \boldsymbol{R}_{8\,6}{}^{13} & \boldsymbol{R}_{8\,9}{}^{13} \end{bmatrix}$$

The coupling matrix $\boldsymbol{H}_C$ is assembled from the blocks $\boldsymbol{H}_{C_{IJ}}$ as indicated below, where the noted size refers the number of components.

$$\boldsymbol{H}_C = \begin{bmatrix} 0 & \boldsymbol{H}_{C_{12}} \\ \boldsymbol{H}_{C_{21}} & 0 \\ \boldsymbol{H}_{C_{31}} & \boldsymbol{H}_{C_{32}} \end{bmatrix}_{28\,x\,9}$$

7.2.4. Dirichlet constraints matrix $\boldsymbol{H}_D$

The matrix of Dirichlet constraints $\boldsymbol{H}_D$ is composed of $NP\ x\ ND$ blocks. Equation (7.13) shows the form of $\boldsymbol{H}_D$. The Ith row of $\boldsymbol{H}_D$ corresponds to the Ith patch. For one patch, all the blocks of its corresponding row are zeros except those ones located at its constraints positions. The calculation of these non-zeros blocks is explained below.

$$\boldsymbol{H}_D = \begin{bmatrix} \boldsymbol{H}_{D_{11}} & & \boldsymbol{H}_{D_{1\,ND}} \\ & \ddots & \\ & \boldsymbol{H}_{D_{IJ}} & \\ & \ddots & \\ \boldsymbol{H}_{D_{NP\,1}} & & \boldsymbol{H}_{D_{NP\,ND}} \end{bmatrix}_{NP\,x\,ND} \tag{7.13}$$

Let us focus on one patch A subjected to the αth Dirichlet constraint, applied on the boundary $\Gamma_{D\alpha}$. We designate with $nd^{A\alpha}$ to number of control points of patch A that are influential on $\Gamma_{D\alpha}$.

The domain is assumed to have more patches, up to NP, and more Dirichlet constraints, up to ND. The block $\boldsymbol{H}_{D\,A\alpha}$ has $n^A\ x\ nd^{A\alpha}$ components as shown in (7.14).

$$\boldsymbol{H}_{D\,A\alpha} = \begin{bmatrix} \vdots & & \\ \mathbf{0}_{1\ x\ nc^{A\alpha}} & & \\ \vdots & & \\ \boldsymbol{R}_{i1}{}^{AA} & \dots & \boldsymbol{R}_{ic_{A\alpha}}{}^{AA} \\ \vdots & & \\ \mathbf{0}_{1\ x\ nc^{A\alpha}} & & \\ \vdots & & \end{bmatrix}_{n^A\ x\ nd^{A\alpha}} \tag{7.14}$$

Each row of (7.14) has $nd^{A\alpha}$ components. If the row corresponds to a non-influential control point on the boundary $\Gamma_{D\alpha}$, then all components are zeros. When the row refers to one influential control point, it is calculated by integration of $\boldsymbol{R}_{ij}{}^{AA}$ as follows:

$$\int_{\Gamma_{D\alpha}} \boldsymbol{R}_{ij}{}^{AA}\, d\Gamma_{D\alpha} = \begin{bmatrix} \int_{\Gamma_{D\alpha}} R_i{}^A\, R_j{}^A\, d\Gamma_{D\alpha} & & \\ & \int_{\Gamma_{D\alpha}} R_i{}^A\, R_j{}^A\, d\Gamma_{D\alpha} & \\ & & \int_{\Gamma_{D\alpha}} R_i{}^A\, R_j{}^A\, d\Gamma_{D\alpha} \end{bmatrix} \tag{7.15}$$

The integrals of (7.15) are approximated by Gauss rule (7.3). In this case $B_b = R_i{}^A\, R_j{}^A$, therefore, the quadrature adopts the form (7.16), where $NG_{ij}{}^{\alpha}$ is the number of Gauss points on the boundary surface $\Gamma_{D\alpha}$ that lie within the influence of basis functions i and j.

$$R_{ij}{}^{AA} \cong \sum_{g=1}^{NG_{ij}{}^{\alpha}} \left(R_i{}^A\, R_j{}^A\right)_g \mathcal{J}_{1_g}\, \mathcal{J}_{2_g}\, w_g \tag{7.16}$$

In equation (7.15) the columns are referred locally to the influential control points, e.g. if influential control points of one patch are 2, 3, 5 and 6 the columns references vary from 1 to 4. The block $\boldsymbol{H}_{D\,A\alpha}$ matches the number of rows of its corresponding patch, therefore it can be inserted at kth block-row and αth block-column the $\boldsymbol{H}_D$ matrix.

One example is depicted in **Fig. 7.9**, with three patches and Dirichlet constraints at Γ_{D1}, Γ_{D2} and Γ_{D3}. The matrix $\boldsymbol{H}_D$ has $3x3$ blocks (three patches and three constraints).

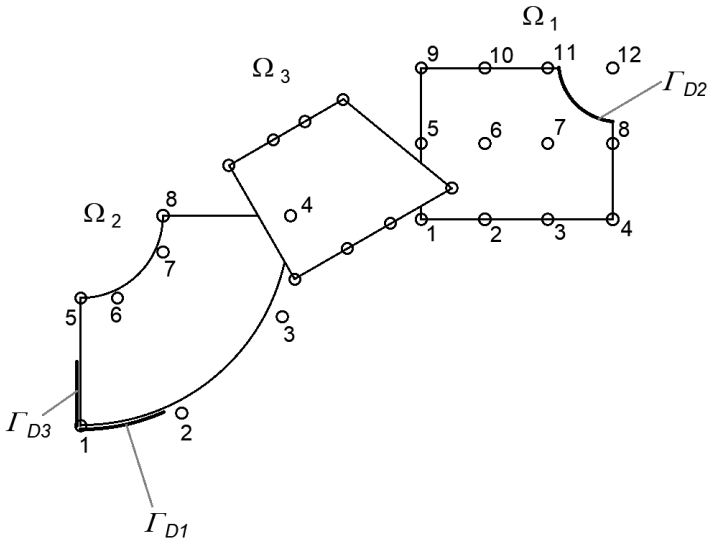


Fig. 7.9 Example of Dirichlet constraints (2D version).

The control points and sizes of the involved blocks are detailed in **Table 7.2**.

Table 7.2 Sizes of blocks of the example.

Constraint		Patch	n^I	nd^{IJ}	Influential control points	Matrix $\boldsymbol{H}_{D_{k\alpha}}$ (Number of components)
J	Boundary	I				
1	Γ_{D1}	2	8	2	1,2	$\boldsymbol{H}_{D_{21}}$ (8x2)
2	Γ_{D2}	1	12	4	7,8,11,12	$\boldsymbol{H}_{D_{12}}$ (8x4)
3	Γ_{D3}	2	8	2	1,5	$\boldsymbol{H}_{D_{13}}$ (8x2)

Let us focus on block $\boldsymbol{H}_{D_{13}}$ which is computed as:

$$\boldsymbol{H}_{D_{13}} = \begin{bmatrix} \boldsymbol{R}_{1\,1}{}^{22} & \boldsymbol{R}_{1\,5}{}^{22} \\ \boldsymbol{0}_{1x2} & \\ \boldsymbol{0}_{1x2} & \\ \boldsymbol{0}_{1x2} & \\ \boldsymbol{R}_{5\,1}{}^{22} & \boldsymbol{R}_{5\,5}{}^{22} \\ \boldsymbol{0}_{1x2} & \\ \boldsymbol{0}_{1x2} & \\ \boldsymbol{0}_{1x2} & \end{bmatrix}$$

The coupling matrix $\boldsymbol{H}_D$ is assembled from the blocks $\boldsymbol{H}_{D_{k\alpha}}$ as indicated below. The noted size of the matrix refers the number of components.

$$\boldsymbol{H}_D = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{H}_{D_{12}} & \boldsymbol{H}_{D_{13}} \\ \boldsymbol{H}_{D_{21}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}_{28\,x\,8}$$

7.2.5. Force vector at control points $\hat{\boldsymbol{f}}$

The vector of forces at control points ($\hat{\boldsymbol{f}}$) is obtained by summation of body forces and traction forces (Neumann constraints). The vector is composed of NP blocks, one per patch. Each block has a number of components equal to the number of active control points of the patch (n^I). The form of the $\hat{\boldsymbol{f}}$ vector is as follows:

$$\hat{\boldsymbol{f}} = \begin{Bmatrix} \hat{\boldsymbol{f}}_1 \\ \vdots \\ \hat{\boldsymbol{f}}_{NP} \end{Bmatrix} \tag{7.17}$$

For the Ith patch, the ith component (corresponding the ith control point) is computed as follows:

$$\hat{\boldsymbol{f}}_i^{\ I} = \int_{\Omega} \boldsymbol{R}^T \, \boldsymbol{b} \; d\Omega + \int_{\Gamma_N} \boldsymbol{R}^T \, \bar{\boldsymbol{t}} \, d\Gamma_N \tag{7.18}$$

Let us focus on patch A, with $\boldsymbol{b}^A$ body force and with attached Neumann boundary $\Gamma_{N\alpha}$ whose traction is $\bar{\boldsymbol{t}}_\alpha$. The corresponding body forces vector is:

$$\int_{\Omega} \boldsymbol{R}^{A^T} \boldsymbol{b}^A \; d\Omega = \int_{\Omega} \begin{bmatrix} \boldsymbol{R}_1^{\ A} \\ \vdots \\ \boldsymbol{R}_{n_A}^{\ A} \end{bmatrix} \; \boldsymbol{b}^A \; d\Omega \tag{7.19}$$

The resultant from (7.19) is a $n_A x 1$ vector that stores the forces applied at control points due to the body force. Each component of this vector can be calculated as:

$$\boldsymbol{b}_i^{\ A} = \int_{\Omega} \boldsymbol{R}_i^{\ A} \quad \boldsymbol{b}^A \; d\Omega \tag{7.20}$$

That integral is approximated by Gauss quadrature, using the Gauss points **of the patch** that lie within the influential volume of the ith control point:

$$\widehat{\boldsymbol{b}}_i^{\ A} \cong \sum_{g=1}^{NG_i^{\alpha}} \left(\boldsymbol{R}_i^{\ A}\right)_g \boldsymbol{b}^A \, \mathcal{J}_{1_g} \, \mathcal{J}_{2_g} \, w_g \tag{7.21}$$

The forces at the ith control point affected by the Neumann constraint are computed as:

$$\hat{\bar{\boldsymbol{t}}}_i^{\ A} = \int_{\Omega} \boldsymbol{R}_i^{\ A} \quad \bar{\boldsymbol{t}}_\alpha \; d\Omega \tag{7.22}$$

Which is approximated by Gauss quadrature (7.3). In this case $B_b = R_i$. Therefore, the quadrature adopts the form (7.23), where ${NG_i}^{\alpha}$ is the number of Gauss points **on the boundary surface** $\Gamma_{N\alpha}$ that lie within the influence of basis function i.

$$\hat{\bar{\boldsymbol{t}}}_i^{\,A} \cong \sum_{g=1}^{{NG_i}^{\alpha}} \left({\boldsymbol{R}_i}^{A}\right)_g \bar{\boldsymbol{t}}_\alpha \, \mathcal{J}_{1_g} \, \mathcal{J}_{2_g} \, w_g \tag{7.23}$$

The total force applied at the ith control point is the summation of both, the body and the traction force, as expressed in (7.24).

$$\hat{\boldsymbol{f}}_i^{\,A} = \hat{\bar{\boldsymbol{t}}}_i^{\,A} + \hat{\boldsymbol{b}}_i^{\,A} \tag{7.24}$$

The force vector of the Ath patch is then allocated to the Ath block of the force vector.

7.2.6. Prescribed displacements vector at control points $\hat{\boldsymbol{u}}_D$

The vector of prescribed displacements at control points is obtained by direct allocation of these displacements to the affected control points. The vector $\hat{\boldsymbol{u}}_D$ is composed of ND blocks, one per Dirichlet constraint. The number of components of each block is equal to the number of patch control points affected by the Dirichlet constraint. The form of the $\hat{\boldsymbol{u}}_D$ vector is shown in (7.25).

$$\hat{\boldsymbol{u}}_D = \begin{Bmatrix} \hat{\bar{\boldsymbol{u}}}_1 \\ \vdots \\ \hat{\bar{\boldsymbol{u}}}_{ND} \end{Bmatrix} \tag{7.25}$$

Let us focus on αth Dirichlet constraint, with prescribed displacement $\hat{\boldsymbol{u}}_\alpha$ applied on $\Gamma_{D\alpha}$, that lies on the Ith patch. The number of influential control points of this patch on $\Gamma_{D\alpha}$ is $nd^{I\alpha}$. Therefore the αth block of is composed of $nd^{I\alpha}$ components each one equal to $\hat{\boldsymbol{u}}_\alpha$.

7.2.7. Relative displacement at coupling vector $\hat{\boldsymbol{u}}_C$

The vector $\hat{\boldsymbol{u}}_C$ is composed of NC blocks, one per coupling constraint. The number of components of each block is equal to the number of control points of the master patch affected by the constraint. The form of the $\hat{\boldsymbol{u}}_C$ vector is shown in (7.26). Since in this work the relative displacement at coupling is to be zeros, all blocks are zeros.

$$\hat{\boldsymbol{u}}_C = \begin{Bmatrix} \boldsymbol{0}_1 \\ \vdots \\ \boldsymbol{0}_{NC} \end{Bmatrix} \tag{7.26}$$

7.3. Relation with the code

The discretization of the boundary surfaces (section 7.1) and the allocation of the boundary conditions (prescribed displacements and surface tractions), are done within the `b0801_defBE` routine. The main functions of `b0801_defBE` are:

- `bs0811_BSini`, where the surfaces discretization is done by *QIT* algorithm (Appendix 10B), as well as their Gauss points definition.
- `bs0812_BCvals`, where the prescribed displacement or surface tractions are allocated to the boundary surfaces.

The outputs of `b0801_defBE` routine are the discretized boundary surfaces with the attached boundary conditions, all stored in the variable `Bentities`.

In `b0802_BCcps` routine the influential control points from the patches on each Gauss point of the boundary surfaces are identified.

The calculation and assembly of the aggregated system of equations, the calculation of control point displacements and the stresses, are all carried out in the `c0900_Analysis` routine. The main inputs of such routine are:

- `Spatches`: that contains the discretization of the patches.
- `Bentities`: that contains the discretization of the boundary entities and the constraints allocated to them (prescribed displacements, tractions and couplings).

The `c0900_Analysis` routine is structure in a loop of number of steps equal to `Nts`. This loop arrangement will allow to implement in the future geometrical non-linearity, where the time is to be discretized in `Nts` steps. However in this thesis the number of steps is one (`Nts`=1).

Within each loop of `c0900_Analysis` there are two well distinguished stages:

- Preparation, where the constraints matrices (sections 7.2.3 and 7.2.4) are assembled in `b0821_BCkf` function; and the forces vector (section 7.2.5) are assembled in `b0904_FextTsep` function.
- Calculation of control point displacements and stresses, that happens in `c0911_DS` routine. Within this routine there are two main steps:

    - Computation of control point displacements, where the stiffness matrix (section 7.2.2) is calculated in `c0912_CT_Operator` and `s1011_KfromGP` routines. The stiffness matrix is assembled with the constraints matrices and the aggregated system is obtained. Therefore the displacements at control point can be calculated (we use the Matlab® solver). This step is coded as a loop to allow for

plastic behaviour, however in this work the number of loops is one since the analysis is linear elastic.

- Update of stresses and internal variables, where displacements and stresses at skins nodes are calculated to allow representation of results. Also internal variables within Gauss points are updated to be prepared for next time step, but as mentioned that will be used for geometrical non-linear problems, which is not the case of this work.

The displacements at control points, and the displacements and stresses at skin nodes are all stored in `Spatches` variable, which is used for representation of results.

7.4. Summary of the chapter

In this chapter the calculation of the aggregated system of equation is presented in detail. The solution to this system is the vector of the displacements at the control points $\widehat{\boldsymbol{u}}$. The displacement within the domain and stresses can be calculated from $\widehat{\boldsymbol{u}}$.

The chapter fulfils the thesis objective ***e***, since it explains how to take into account constraints applied to the domain whatever their shape and parametrization is.

The validation of the procedures explained in this chapter is carried out in Chapter 8, where a set of examples is simulated with the algorithm developed in this thesis. The results of the simulations (displacements and stresses derived from $\widehat{\boldsymbol{u}}$) are compared to reference models for validation purposes.

8. **Validation examples**

This chapter displays three examples to demonstrate the algorithm validity: a cantilever beam, a top loaded plateau and a twisted bracket. In all of them we present the information as follows:

- Geometry, boundary conditions and material of the problem.
- Parametrization of the gross patches.
- Discretization of the patches.
- Discretization of the boundary surfaces.
- Analysis of results and validation.
- Computation cost for each stage (stages A to E according to Fig. 3.7).

The computational cost shows the time spent in each stage (in seconds) and in percentage. The absolute values of the times depend indeed on the computer features where the code is run. In addition, we compare in a columns graph the time spent in the geometric calculation (stage A), in the discretization (stage C) and the analysis (stage E). It will we shown that the cost of geometric calculation is much lesser than the discretization or the analysis. The stages B and D involve the *QIT* algorithm, that is not devised for IGA itself but for representation purposes, that is the reason why they are not compared in the columns graph.

In each example, apart from demonstrating the algorithm delivers correct results, particular aspects are detailed. In the cantilever beam the lattice algorithm is used to generate one of the faces for one of the patches. The loaded plateau example proves that this algorithm can deal with arbitrary shapes of the trimming surface. The twisted bracket illustrates a three-patches domain and reveals the importance of the discretization of the boundary surfaces.

The validation is carried out by comparison with an alternative FEA model[58]. The purpose of this chapter is to demonstrate that the algorithm for IGA solids delivers results comparable to the alternative model in terms of displacements and stresses.

The results depend on the mesh size (in both FEA and IGA). Although guidance on the meshing is provided in Chapter 6 (tetrahedralization), this thesis is not focused on the meshing itself. The meshing process could be another research topic and it would exceed the extension of this work. Therefore sensitive analysis of mesh is not done in the examples, and the results are assumed valid as long as the compared displacements and stresses between this algorithm and the alternative FEA model present the similar tendencies.

[58] The used package is ABAQUS®

Appendix 8A firstly provides guidance for the usage of this algorithm. Then the preparation of each of the three examples is detailed: on one hand the CAD construction of the domain, on the other hand the preparation prior to the analysis itself via the user-interface generated in this thesis. The computation of displacements and stresses that are displayed on the domains skins is briefed in Appendix 7A.

8.1. Cantilever beam

8.1.1. *Geometry*

The problem consists of a cantilever beam with *T-section* that varies from a robust section at the clamped edge to a slender one at the end tip, as shown in **Fig. 8.1**. The profile of the curve that defines the underside of the flange is given by a set of 13 points as illustrated in **Fig. 8.2**. These points are to be approximated by a cubic spline. All units are given in mm.

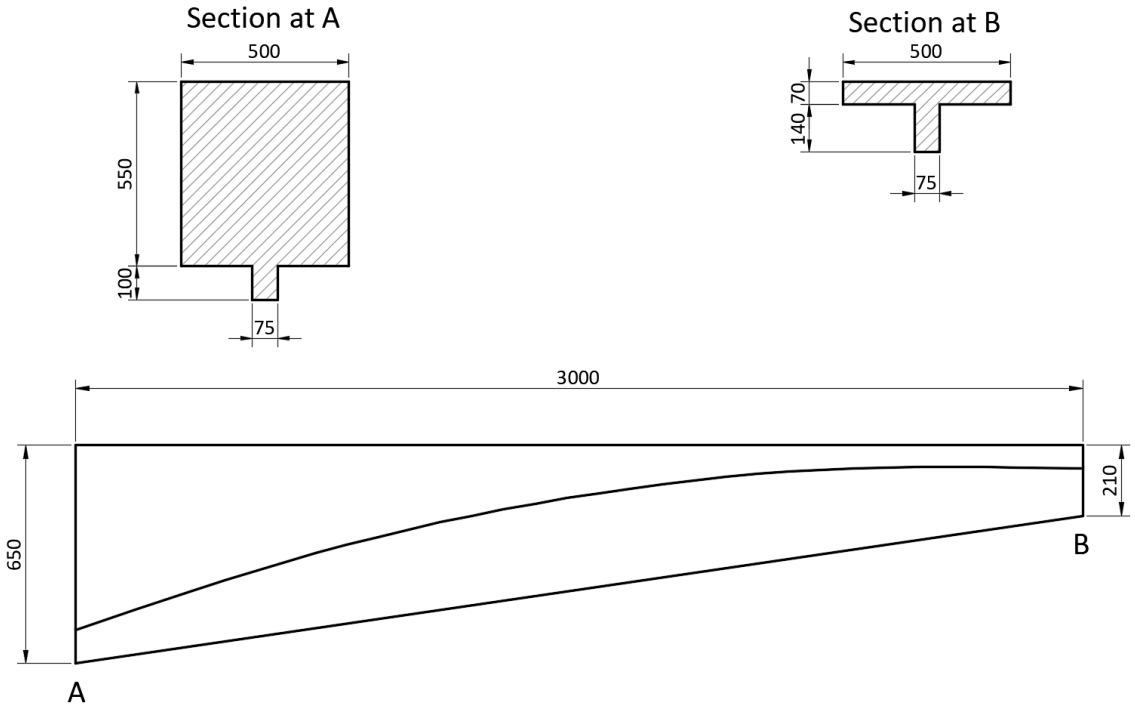


Fig. 8.1 Geometry of the cantilever beam (mm).

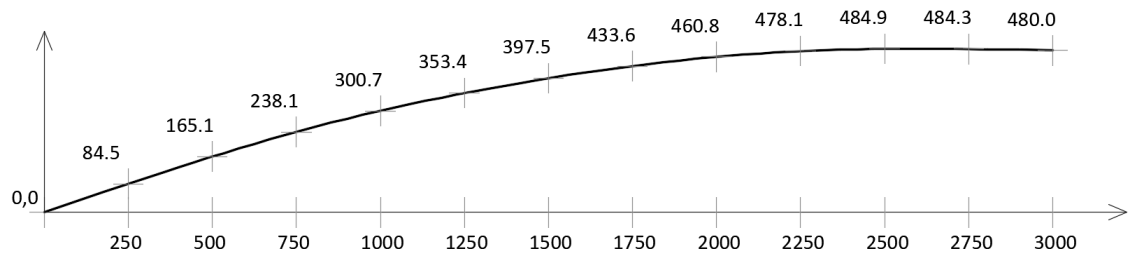


Fig. 8.2 Sample points to construct the profile for the underside of flange (mm).

The domain consists of two coupled patches: patch 1 for the flange and patch 2 for the web (see Appendix 8A for further details).

8.1.2. Boundary conditions and material

Boundary conditions are given in **Fig. 8.3**. End A is fully fixed. End B has traction of $t = 3.3\ MPa$ downwards. The area at B is 45500 mm^2, hence the force applied at this end is 150150 N. Material is to be isotropic steel with next elastic properties:

$E = 210\,000\ MPa$

$\nu = 0.3$

Self-weight is ignored.

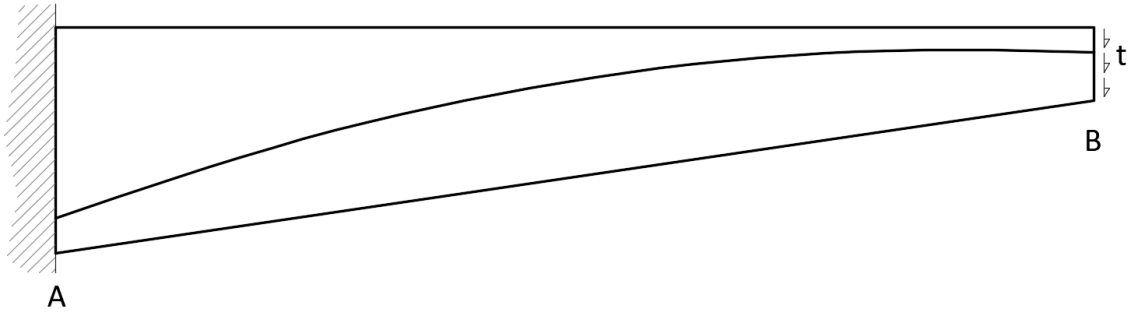


Fig. 8.3 Boundary conditions.

8.1.3. Parametrization of the gross patches

The parameter directions of the patches are indicated in **Fig. 8.4**. The gross patches parametrization features are provided in **Table 8.1**. To achieve such parametrization, refinement is required as detailed in Appendix 8A.

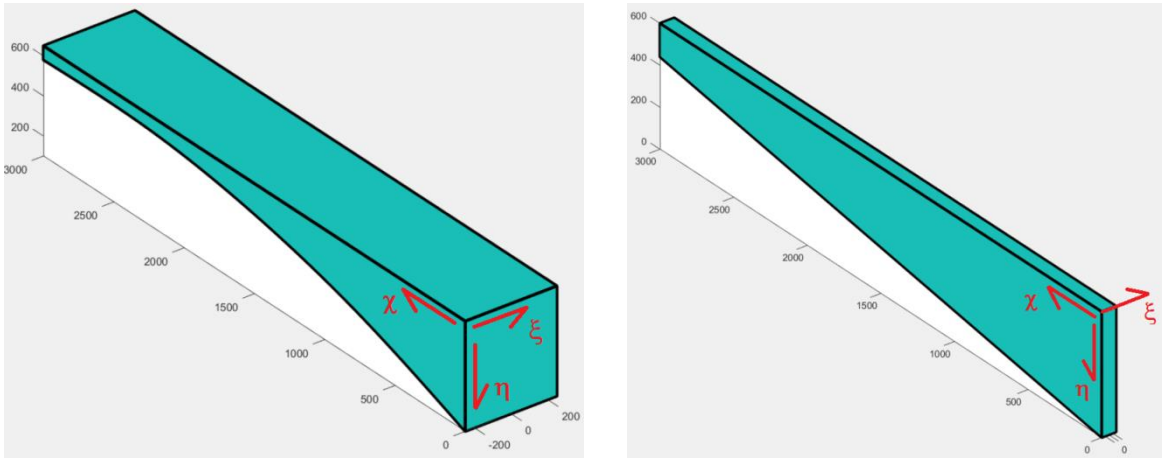


Fig. 8.4 Parameter directions for patch 1 and 2 (left and right respectively).

Table 8.1 NURBS features of the gross patches.

		Patch 1	Patch 2
Number of control points	n	3	3
	m	4	4
	l	7	6
Degree	p	2	2
	q	3	2
	r	3	2
Knot vectors	Ξ	000 111	000 111
	H	0000 1111	000 0.50 111
	X	0000 0.2573 0.5081 0.7547 1111	000 0.25 0.50 0.75 111

The parametrization of the patch 1 is generated by the sandwich algorithm (see section 4.2.2) being one of its faces created by the plane lattice fitting algorithm (recall section 4.2.3). **Fig. 8.5** shows the main steps for the lattice fitting: adjustment of lines to contour control points, initial lattice and final lattice, which is used as control net of the face. In the contour lines fitting small deviations are introduced in case they intersect. That deviation is shown at the left-hand side of the **Fig. 8.5**, where the vertical lines are deviated to avoid intersection between them. The resulting control net for the patch 1 after applying the sandwich algorithm is illustrated in **Fig. 8.6**.

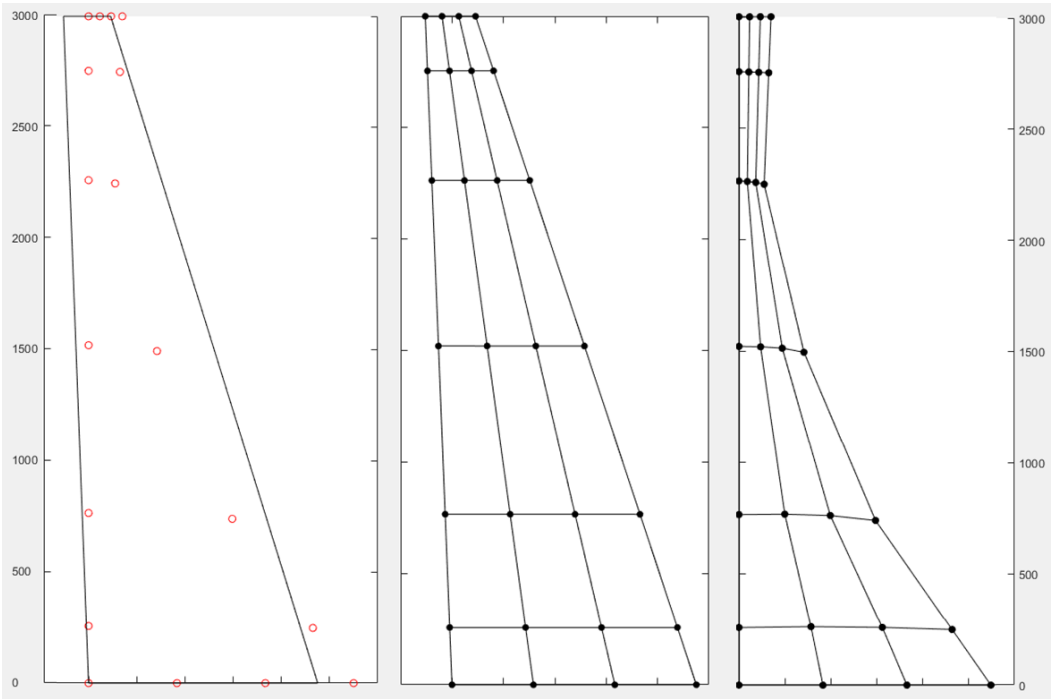


Fig. 8.5 Plane lattice fitting for one of the faces of the patch 1.

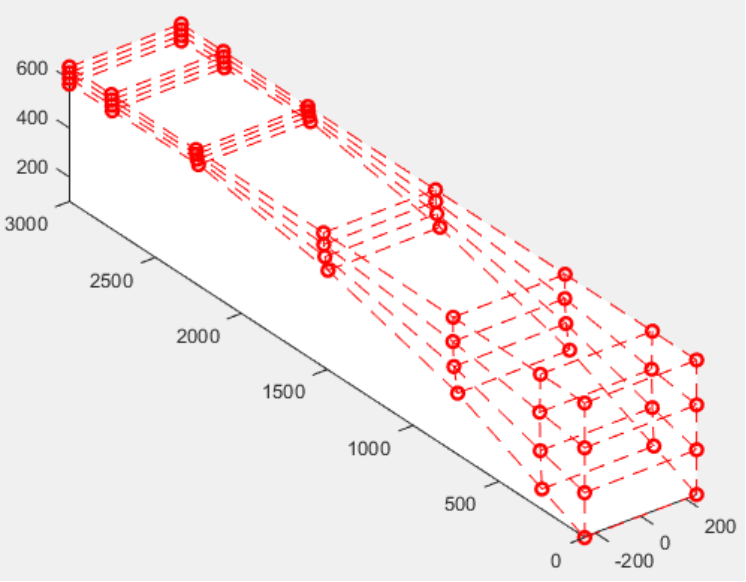


Fig. 8.6 Resulting control net after applying the sandwich algorithm to patch 1.

8.1.4. Discretization of patches

Since patch 1 is untrimmed, the discretization coincides with the non-void knot spans as shown in **Fig. 8.7**. By contrast patch 2 is trimmed, hence its discretization is achieved by tetrahedralization, as depicted in **Fig. 8.8**.

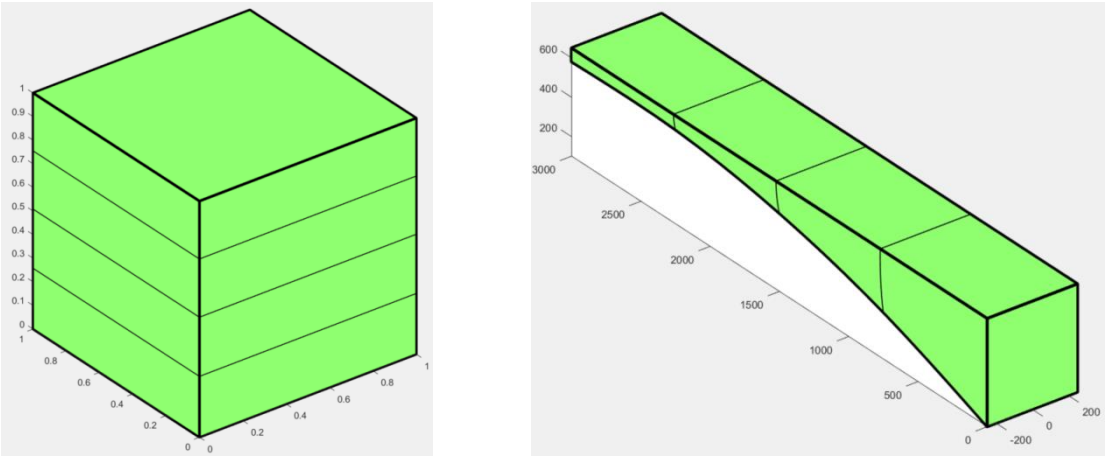


Fig. 8.7 Discretization of patch 1 in parameter and physical spaces.

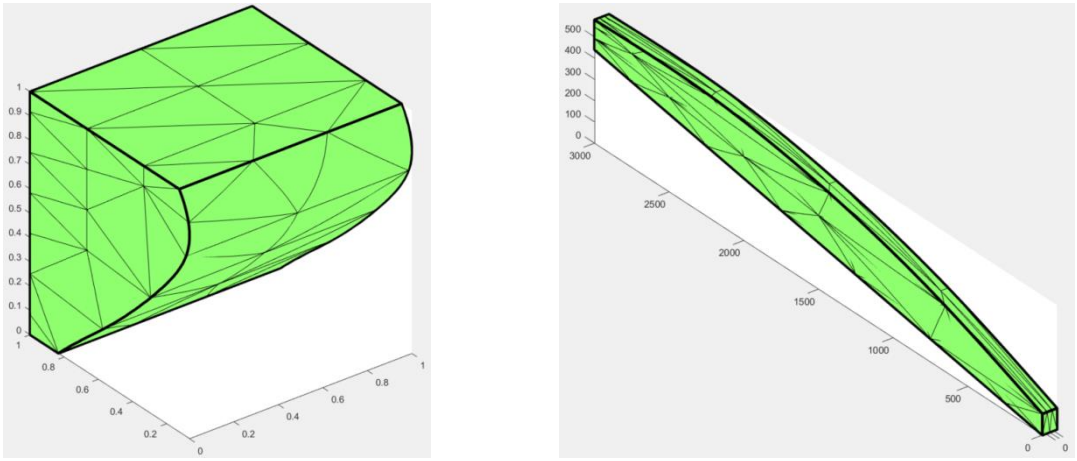


Fig. 8.8 Discretization of patch 2 in parameter and physical spaces.

8.1.5. *Discretization of boundary surfaces*

Boundary surfaces are discretized into triangles, whose size is given by the user (75 mm in this case). Within each triangle, one Gauss point is located as illustrated in **Fig. 8.9**.

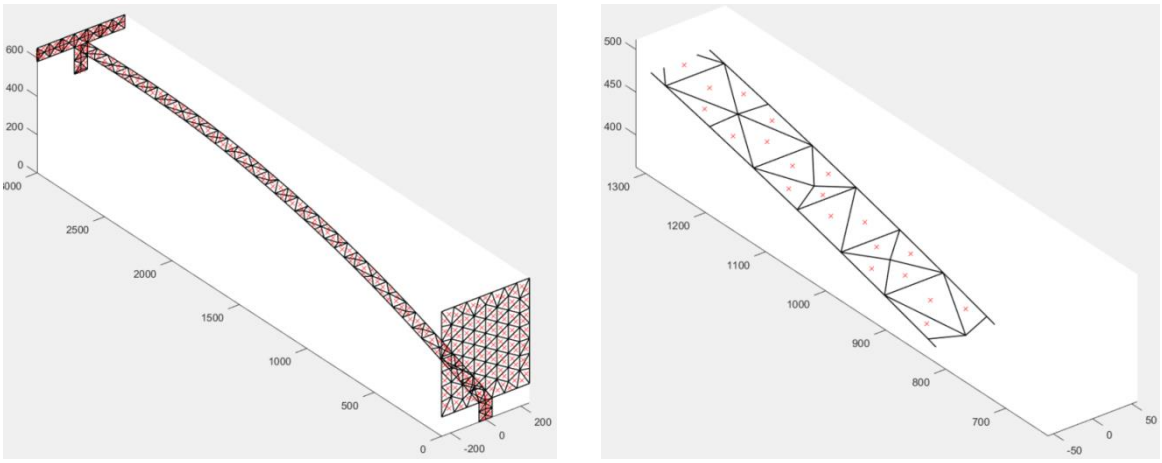


Fig. 8.9 Right: discretized boundary surfaces and Gauss points. Left: detail of one boundary surface.

8.1.6. Analysis and validation

The deformed shape scaled by 100 is shown in **Fig. 8.10**: above the outputs from our algorithm and below the output from FEA. Von Mises stresses are plotted on the domain.

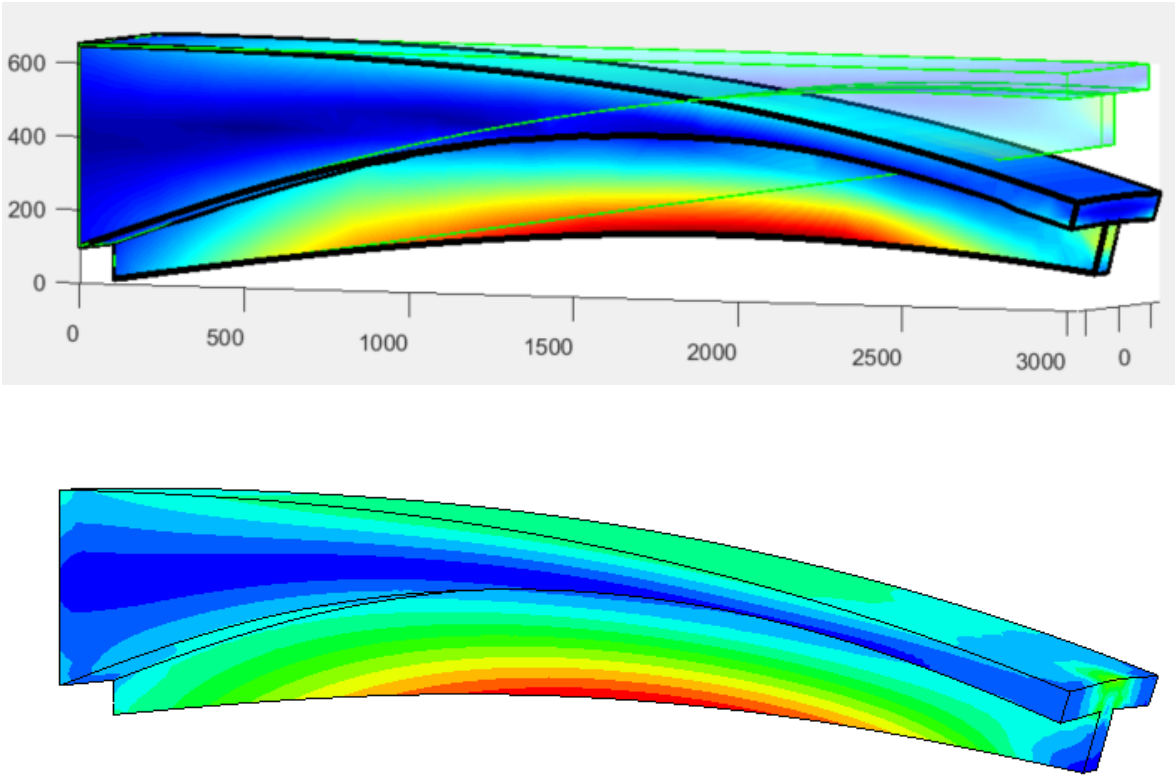


Fig. 8.10 Deformed shape (x100) with Von Mises stress. Above from IGA and below from FEA.

To compare the results quantitatively, stresses in the longitudinal direction (σ_y) and vertical displacements (u_z) are obtained along the path A, indicated in **Fig. 8.11**. There are seven equally spaced sample points along the path.

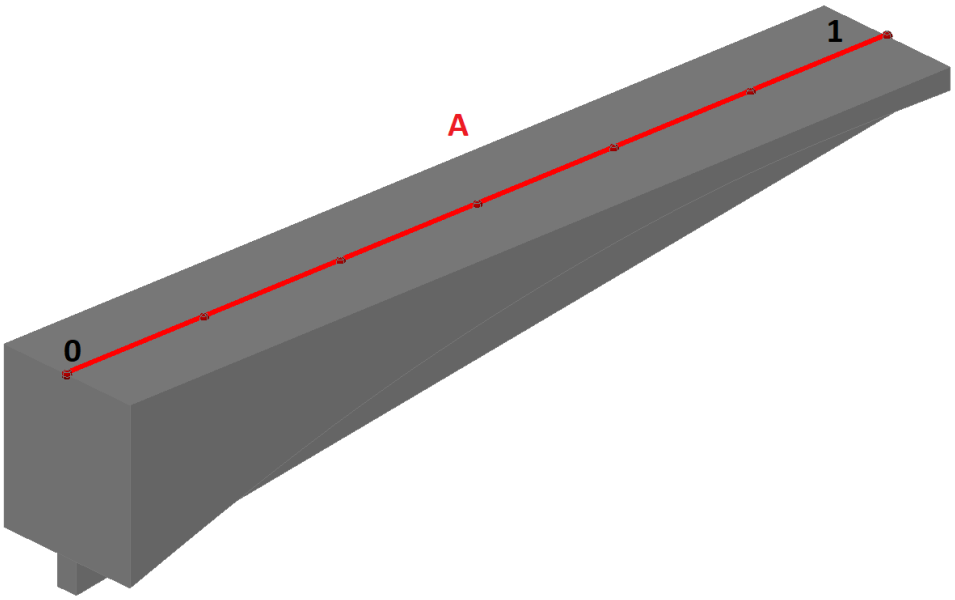


Fig. 8.11 Path of sample points.

Vertical displacements and longitudinal stresses are plotted in **Fig. 8.12** and **Fig. 8.13** respectively. The results from IGA are in solid line meanwhile the results from FEA in dashed line. The difference, in percentage with respect to the average value of IGA, is represented by a dotted line.

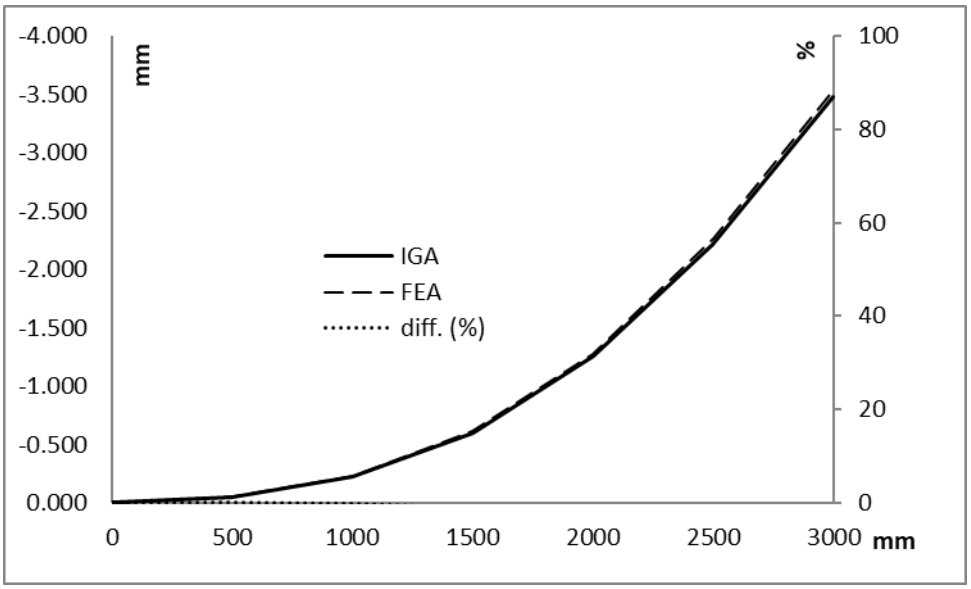


Fig. 8.12 Vertical displacements along the path A.

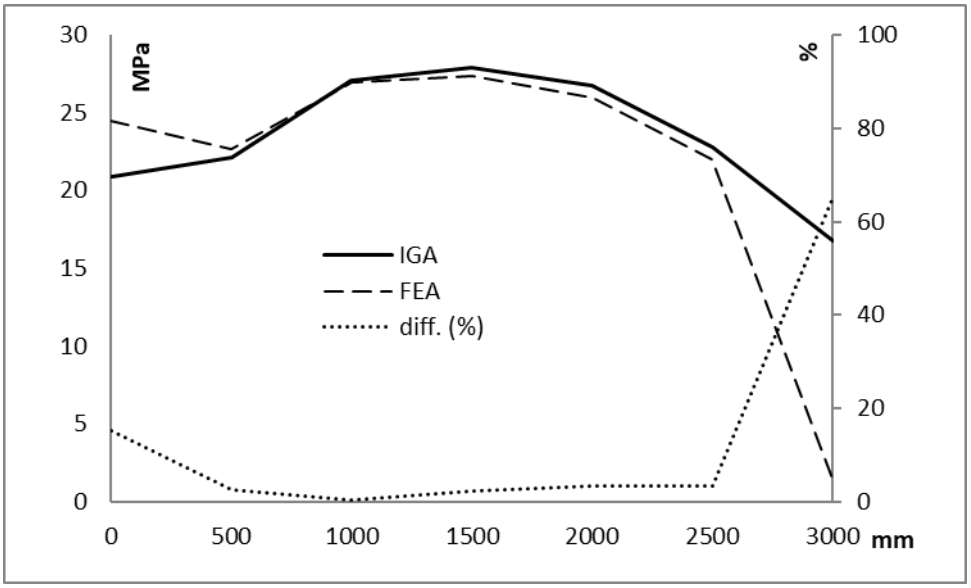


Fig. 8.13 Longitudinal stresses along the path A.

Vertical displacements in both models, FEA and IGA are practically the same (the difference line is even not seen in the graph). By contrast, the longitudinal stresses present differences, but they are assumed acceptable. These differences indicate that the post-processor of the algorithm is to be improved to achieve closer results.

8.1.7. Computational cost of each stage

The **Table 8.2** shows the time spent in each stage, in seconds and in percentage. In **Fig. 8.14** the costs of stages A, C and E are compared.

Table 8.2 Computational cost.

stage		t (s)	%
A	Generation of solid and surfaces identification	2.17	2.5
B	Initiation of patches and representation	46.14	52.6
C	Discretization of patches	13.22	15.1
D	Allocation of boundary conditions	9.61	11.0
E	Analysis	16.50	18.8
	TOTAL	87.64	100

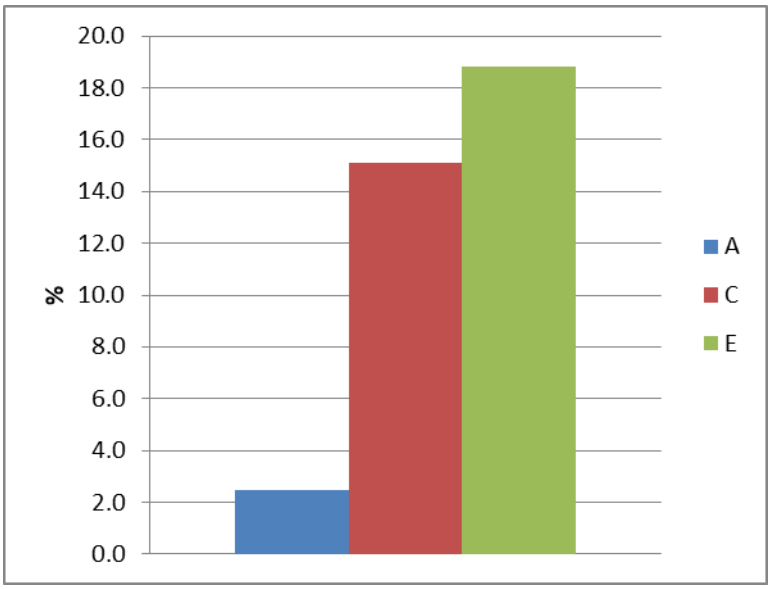


Fig. 8.14 Comparison of cost, in percentage, between geometrical definition (A), discretization (C) and analysis (E).

8.2. Loaded plateau

8.2.1. *Geometry*

The geometry corresponds to a plateau surrounded by an escarpment. It is assumed that the escarpment surface shown in **Fig. 8.15** fits the terrain profile, i.e. this surface is customized to reproduce the terrain shape. All units are given in m.

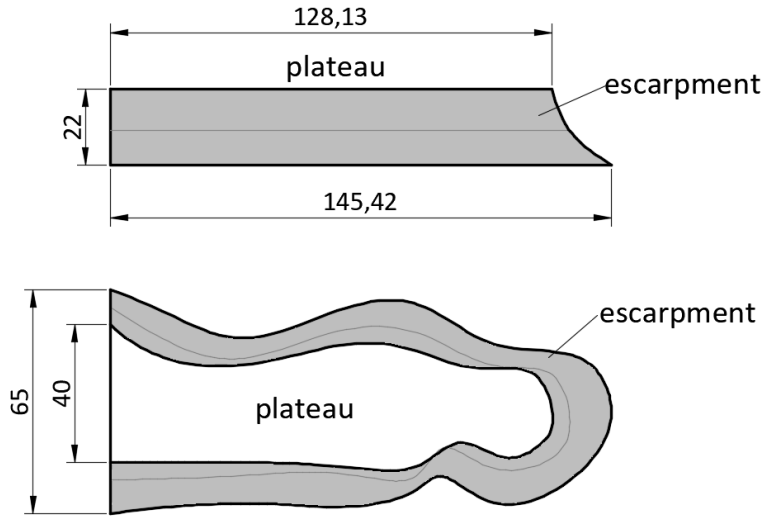


Fig. 8.15 Geometry of the plateau (m).

8.2.2. Boundary conditions and material

Boundary conditions are given in **Fig. 8.16**. Bottom surface is fully fixed and left end is only restrained in vertical direction. The loading corresponds to a building that exerts on plan $t = 400\ kPa$ vertically.

Material is to be isotropic with next elastic properties that correspond to a soft sandy clay:

$E = 20\,000\ kPa$

$\nu = 0.25$

$\gamma = 22\ kN/m^3$

Self-weight is considered.

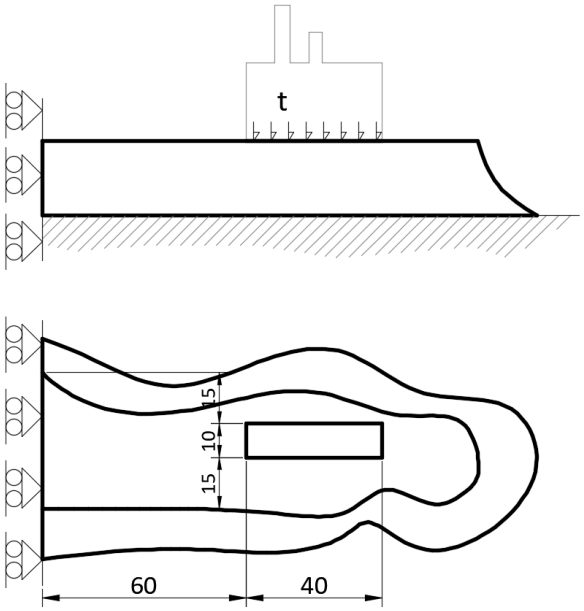


Fig. 8.16 Boundary conditions (units in m).

8.2.3. Parametrization of the gross patches

The parametrization of the patch is indicated in **Fig. 8.17** and detailed in **Table 8.3**. To achieve such parametrization, refinement is required as detailed in Appendix 8A.

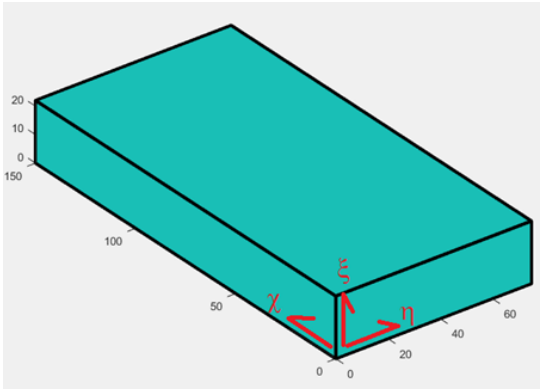


Fig. 8.17 Parameter directions of the patch

Table 8.3 NURBS features of the gross patch.

Number of control points	n	4
	m	4
	l	7
Degree	p	2
	q	2
	r	3
Knot vectors	Ξ	000 0.5 111
	H	000 0.5 111
	X	0000 0.25 0.50 0.75 1111

8.2.4. Discretization of patches

The trimmed patch is discretized in tetrahedrons, as depicted in **Fig. 8.18**.

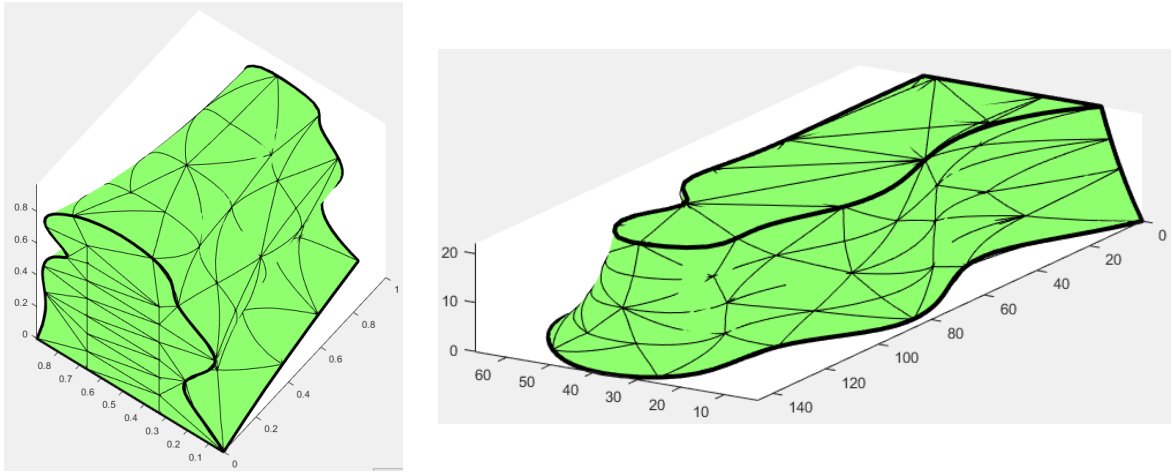


Fig. 8.18 Discretization of patch in parameter and physical spaces.

The discretization is different from the knot spans as shown in **Fig. 8.19**. The number of tetrahedrons is linked to the knots spans though (recall section 6.3.1).

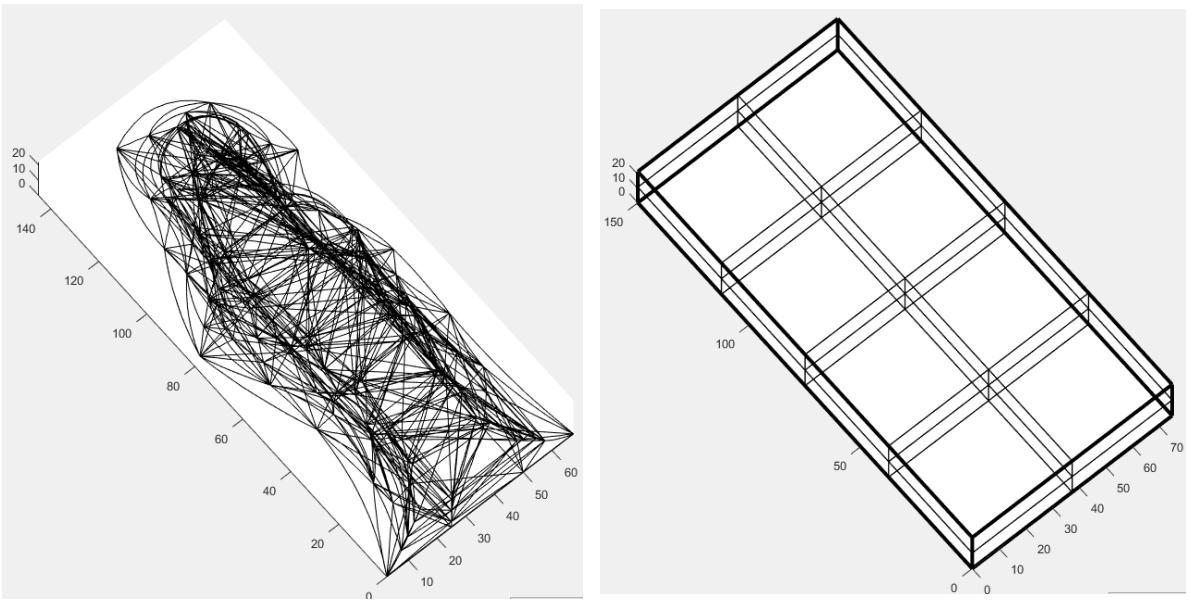


Fig. 8.19 Tetrahedralization and knot spans.

8.2.5. *Discretization of boundary surfaces*

Boundary surfaces are discretized into triangles, whose size is given by the user (4 m in this case). Within each triangle one Gauss point is located as illustrated in **Fig. 8.20**.

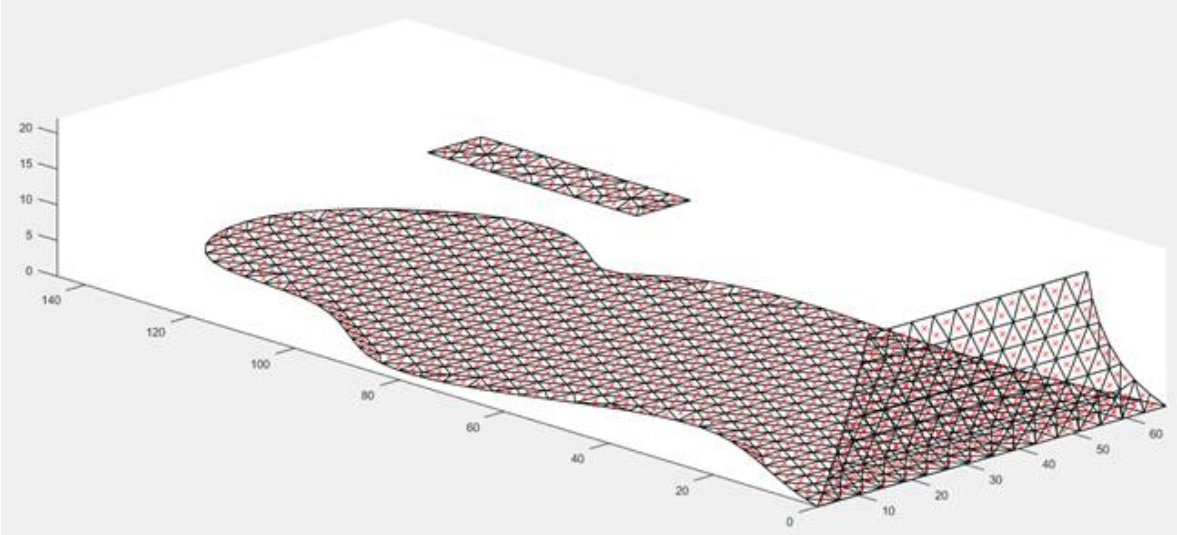


Fig. 8.20 Discretized boundary surfaces and Gauss points.

8.2.6. Analysis and validation

The deformed shape scaled by 25 is shown in **Fig. 8.21**, above the output from our algorithm and below the output from FEA.

The results are compared quantitatively along two paths, A and B (see **Fig. 8.22**), with 10 sample points each. The vertical displacements are recorded along path A and Von Mises stresses along path B. The ten sample points are equally spaced at 14 and 16 m for paths A and B.

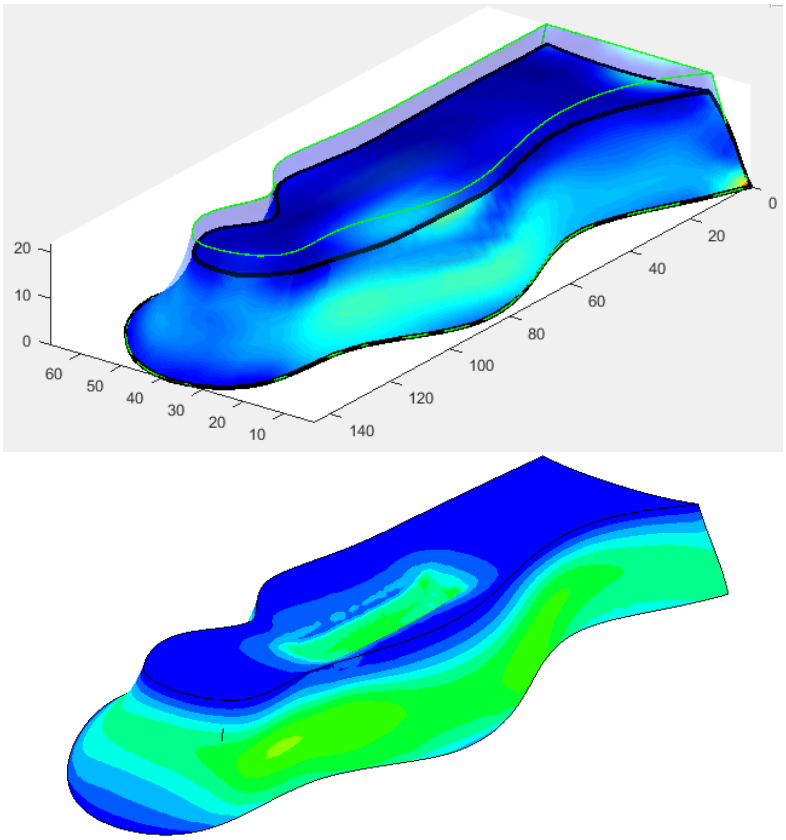


Fig. 8.21 Deformed shape (x25) with Von Mises stress. Above from IGA and below from FEA.

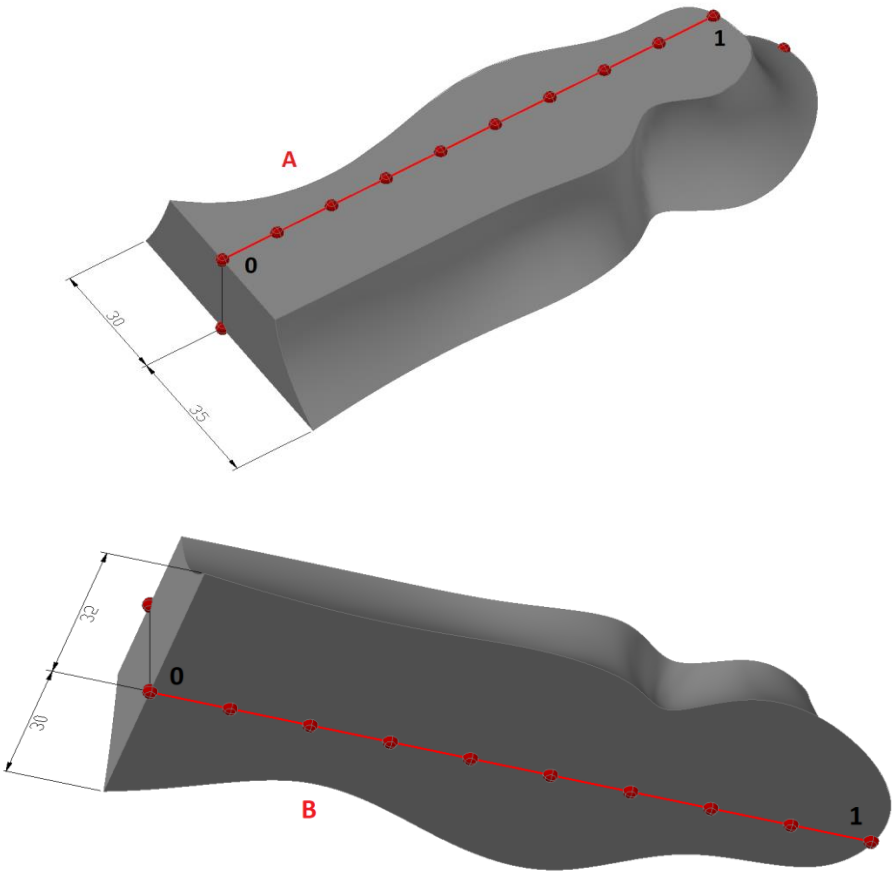


Fig. 8.22 Paths of sample points.

Vertical displacements (in m) and Von Mises stresses (in kPa) for paths A and B are plotted in **Fig. 8.23** and **Fig. 8.24** respectively. The tendency of the results from both models is similar. Mesh refinement could reduce the difference in the displacements plot.

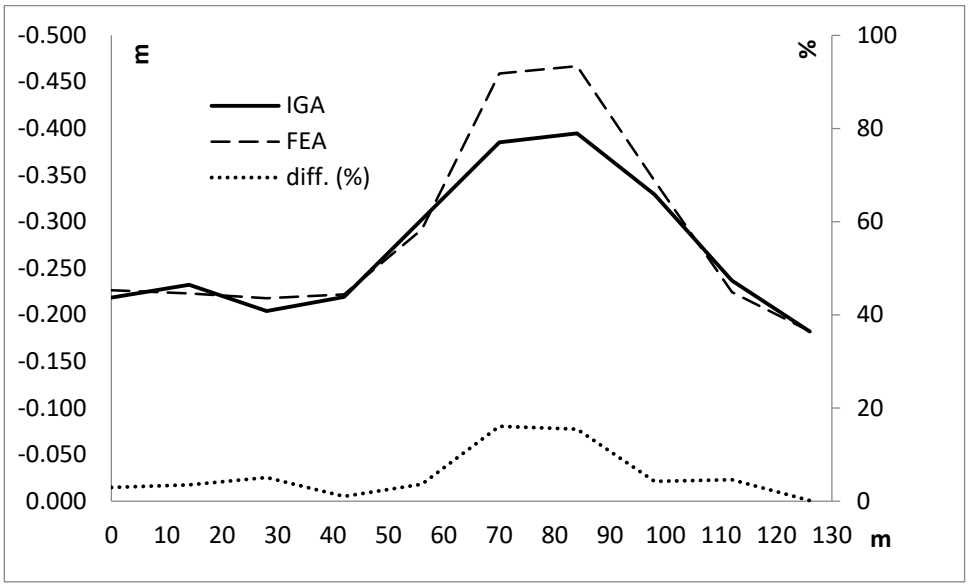


Fig. 8.23 Longitudinal stresses along the path A.

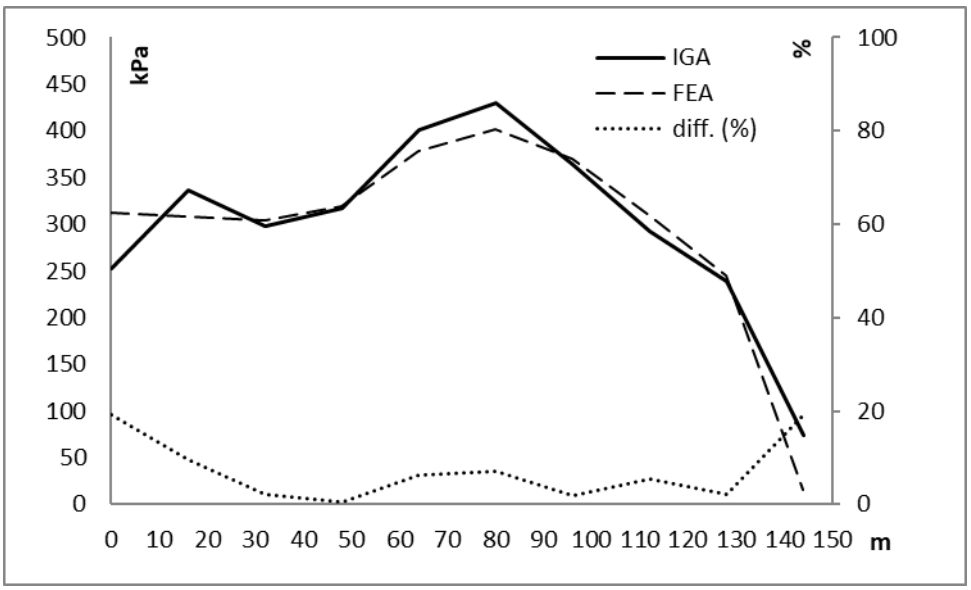


Fig. 8.24 Von Mises stresses along the path B.

8.2.7. Computational cost of each stage

The **Table 8.4** shows the time spent in each stage, in seconds and in percentage. In **Fig. 8.25** the cost of stages A, C and E are compared.

Table 8.4 Computational cost.

stage		t (s)	%
A	Generation of solid and surfaces identification	0.81	0.9
B	Initiation of patches and representation	40.16	42.3
C	Discretization of patches	19.22	20.2
D	Allocation of boundary conditions	11.86	12.5
E	Analysis	22.96	24.2
	TOTAL	95.01	100

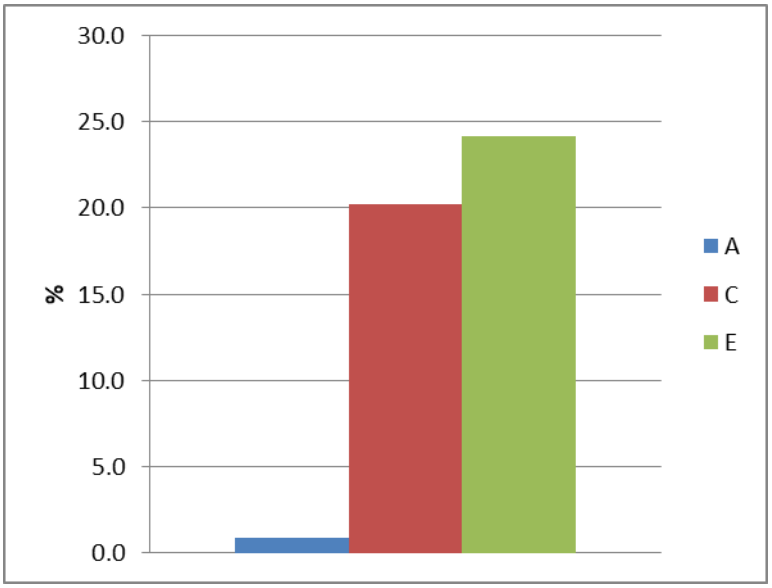


Fig. 8.25 Comparison of cost, in percentage, between geometrical definition (A), discretization (C) and analysis (E).

8.3. Twisted bracket

8.3.1. *Geometry*

The problem consists of the bracket illustrated in **Fig. 8.26**. The bracket has three parts: base, stem and head. The stem is generated by sweep of a squared base along the cubic spline shown in **Fig. 8.27**.

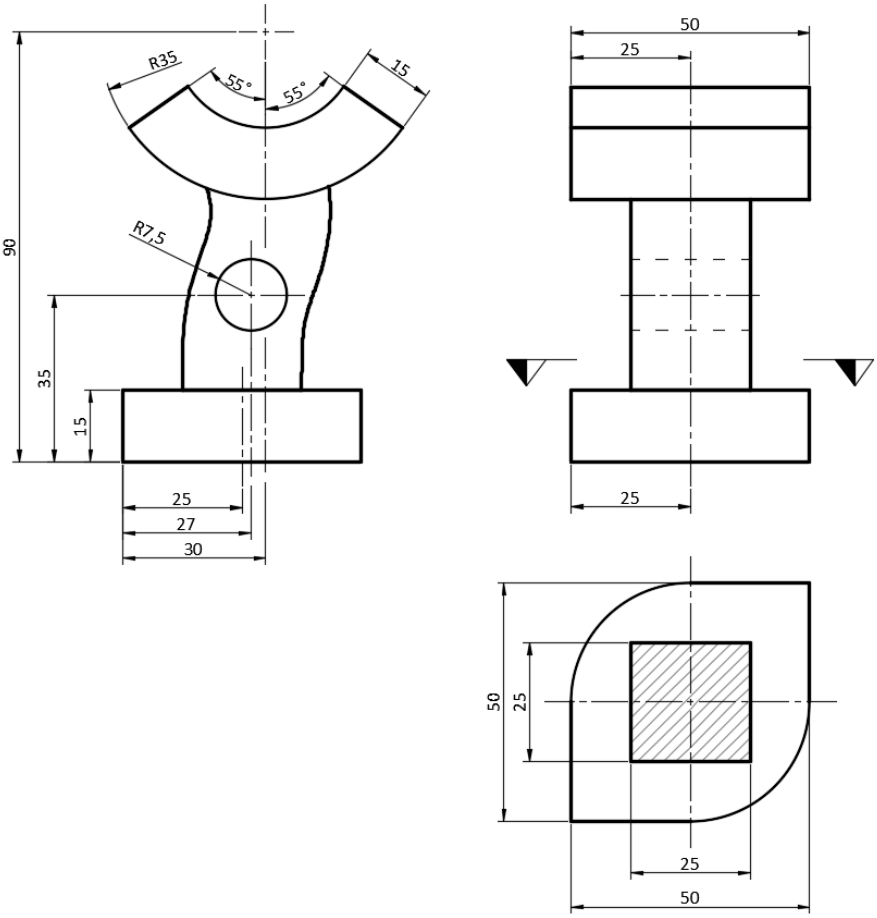


Fig. 8.26 Geometry of the bracket (mm).

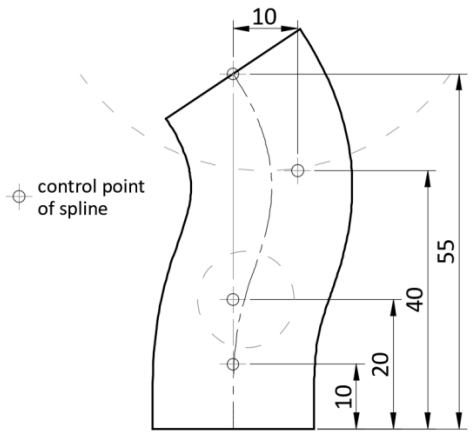


Fig. 8.27 Stem construction by sweep along cubic spline (mm).

8.3.2. Boundary conditions and material

The boundary conditions are given in **Fig. 8.28**. The bottom of base is fully fixed and a traction of $t = 1000\ MPa$ is applied perpendicular to the region indicated. The area of this region is 300 mm^2, hence the force applied is 300000 N.

Material is to be isotropic steel with next elastic properties:

$E = 210\,000\ MPa$

$\nu = 0.3$

Self-weight is ignored.

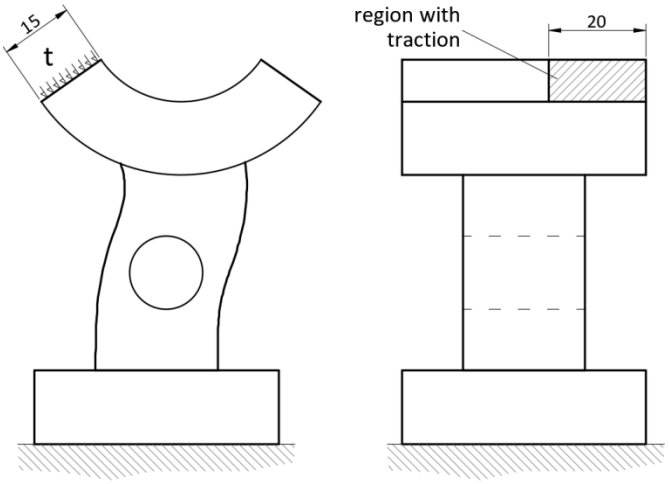


Fig. 8.28 Boundary conditions.

8.3.3. Parametrization of the gross patches

The parameter directions of the patches are indicated in **Fig. 8.29**. The gross patches parametrization features are provided in **Table 8.5**. To achieve such parametrization, refinement is required as shown in Appendix 8A.

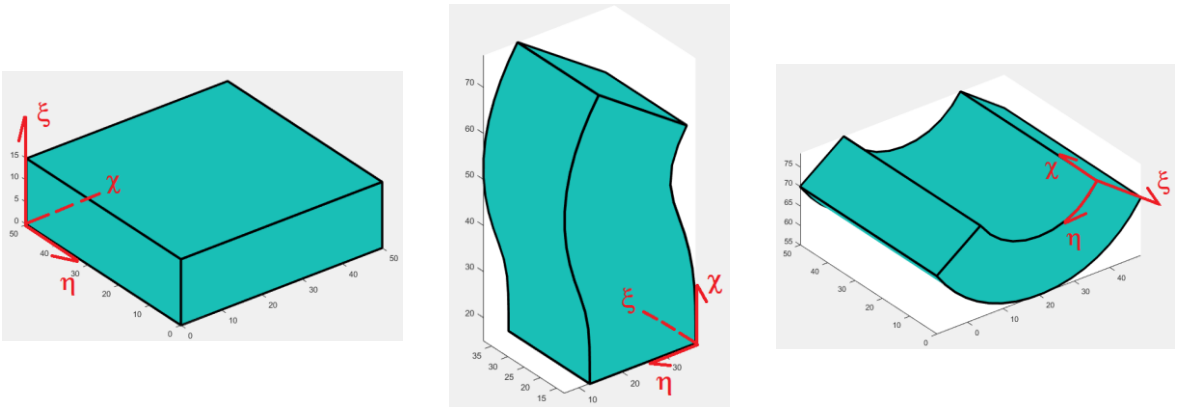


Fig. 8.29 Parameter directions of the patches

Table 8.5 NURBS features of the gross patches.

		Patch 1	Patch 2	Patch 3
Number of control points	n	4	4	4
	m	4	6	6
	l	4	13	6
Degree	p	2	2	2
	q	2	2	2
	r	2	3	2
Knot vectors	Ξ	000 0.5 111	000 0.5 111	000 0.5 111
	H	000 0.5 111	000 0.25 0.5 0.75 111	000 0.25 0.5 0.75 111
	X	000 0.5 111	0000 0.1 0.25 0.35 0.5 0.6 0.7 0.8 0.9 0.95 1111	000 0.25 0.5 0.75 111

8.3.4. Discretization of patches

Patches 1 and 2 are trimmed, therefore their discretization is achieved by tetrahedralization, as depicted in **Fig. 8.30** and **Fig. 8.31**. By contrast, in patch 3 the discretization coincides with the non-void knot spans as shown in **Fig. 8.32**, since it is untrimmed.

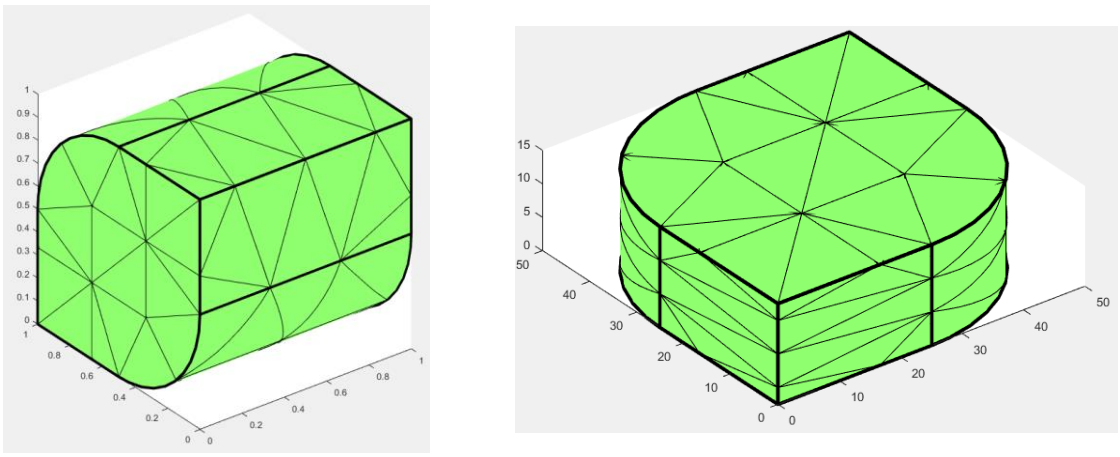


Fig. 8.30 Discretization of patch 1 in parameter and physical spaces.

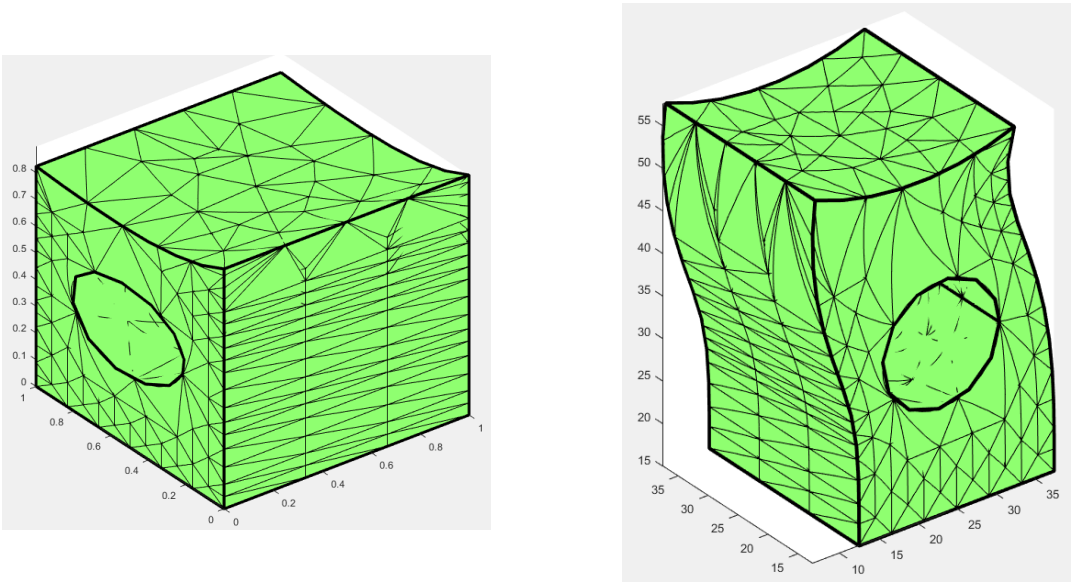


Fig. 8.31 Discretization of patch 2 in parameter and physical spaces.

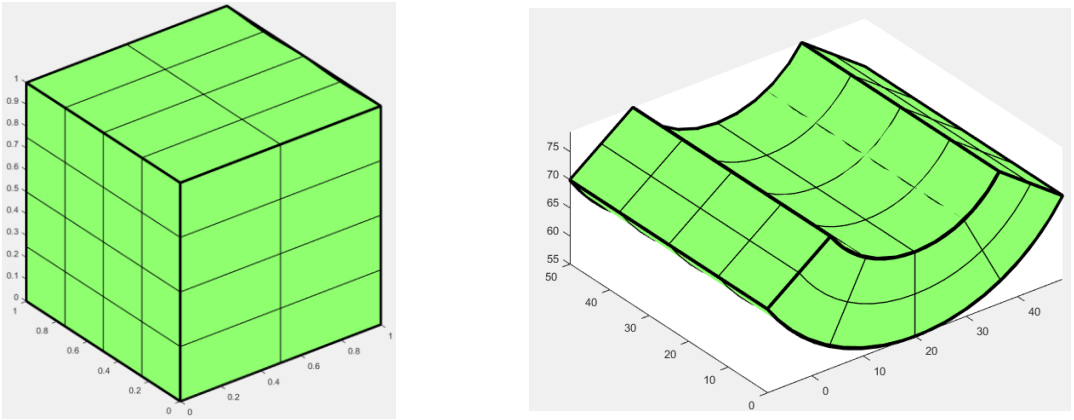


Fig. 8.32 Discretization of patch 3 in parameter and physical spaces.

8.3.5. *Discretization of boundary surfaces*

Boundary surfaces are discretized into triangles, whose size is given by the user (4 mm in this case). Within each triangle one Gauss point is located as illustrated in **Fig. 8.33**.

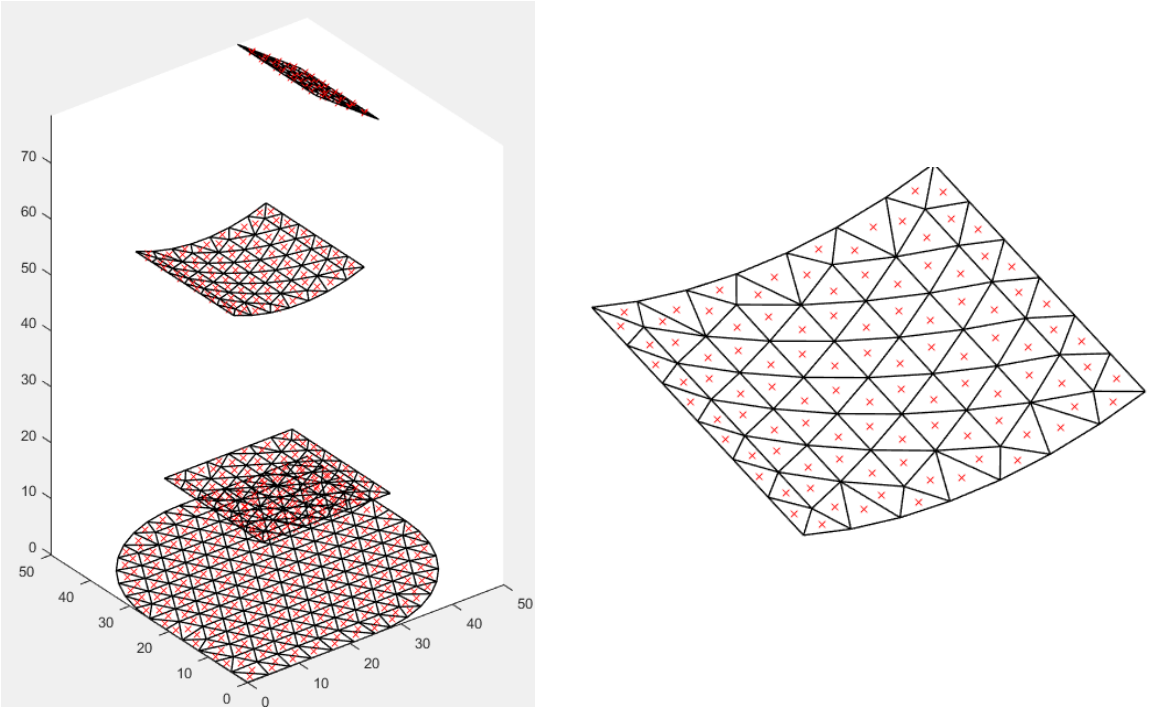


Fig. 8.33 Right: discretized boundary surfaces and Gauss points. Left: detail of boundary surface 3.

8.3.6. Analysis and validation

The deformed shape scaled by 5 is shown in **Fig. 8.34**, above the outputs from our algorithm and below the output from FEA. Von Mises stresses are plotted on the domain.

To compare the results quantitatively we use two paths on the domain that are shown in **Fig. 8.35**. Displacements are measured along path A. Von Mises stresses (σ_{VM}) are measured along the path B, that surrounds the stem at the level of the centre of the hole.

Displacements (in mm) in x, y and z directions along path A are plotted in **Fig. 8.36** to **Fig. 8.38**. Von Mises stresses (in MPa) along path B are plotted in **Fig. 8.39**. The difference between our algorithm and FEA is given by the dotted line at the bottom in percentage with respect to the average value of IGA.

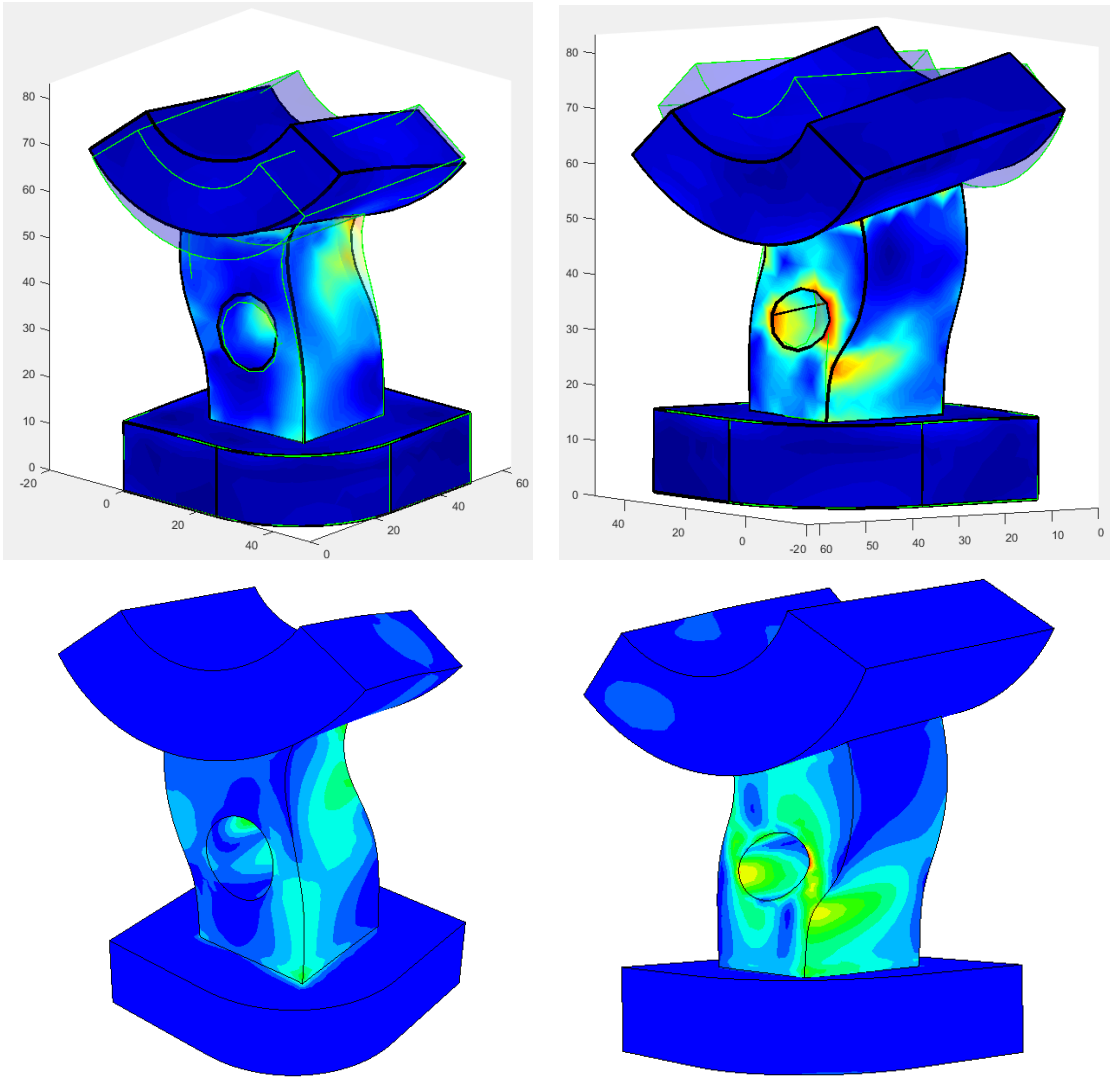


Fig. 8.34 Deformed shape (x5) with Von Mises stress, front and rear views, from IGA (above) and FEA (below).

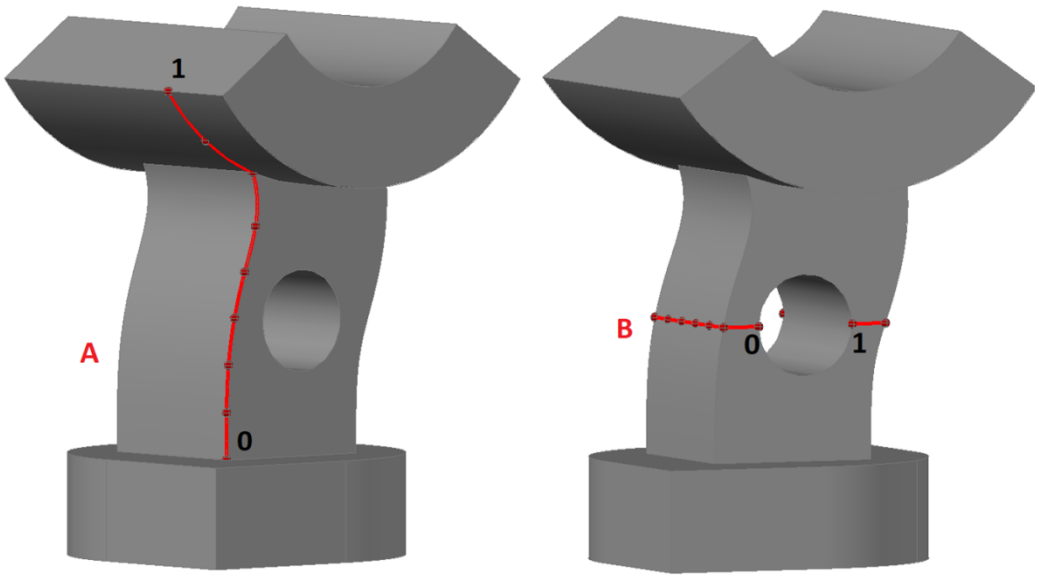


Fig. 8.35 Path of sample points.

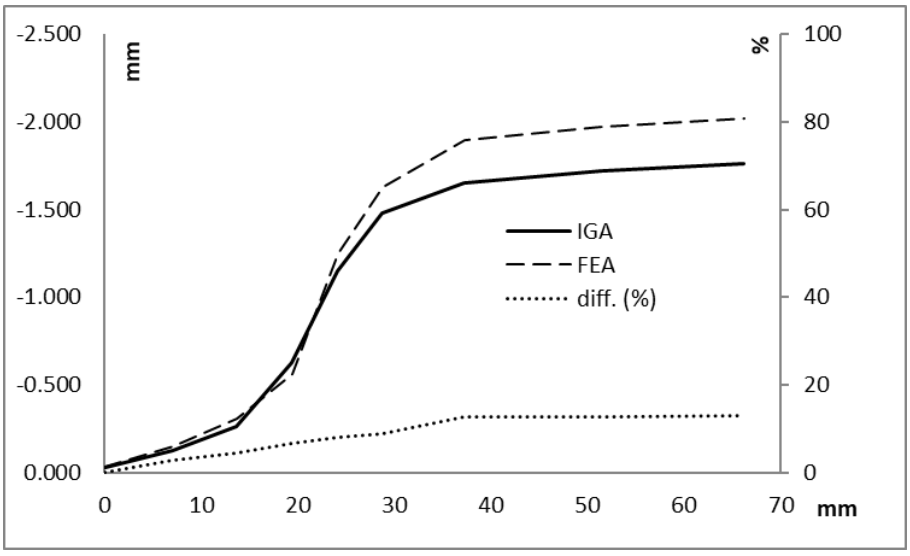


Fig. 8.36 Displacement in x direction along path A.

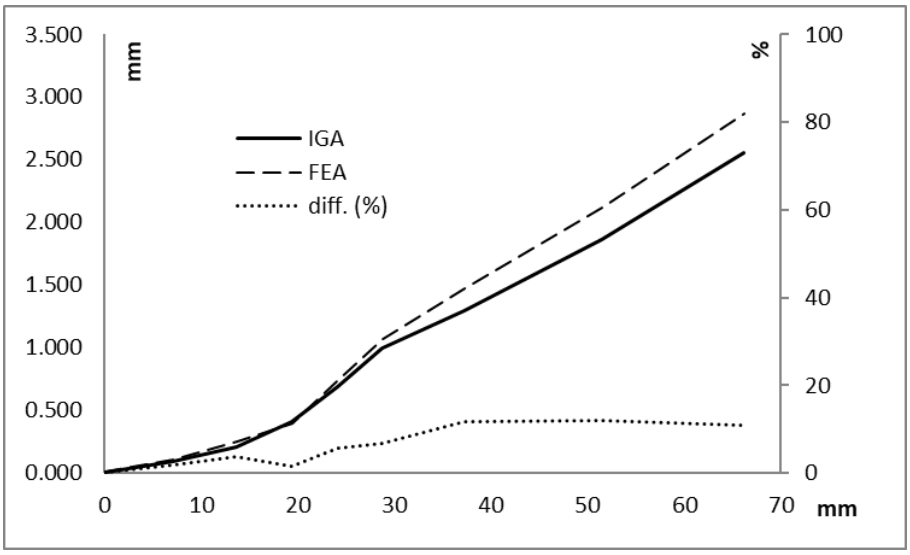


Fig. 8.37 Displacement in y direction along path A.

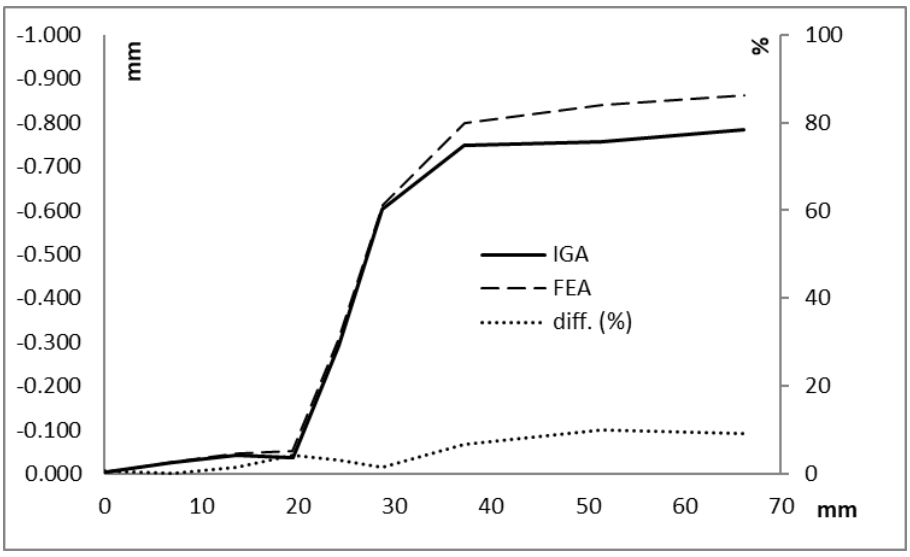


Fig. 8.38 Displacement in z direction along path A.

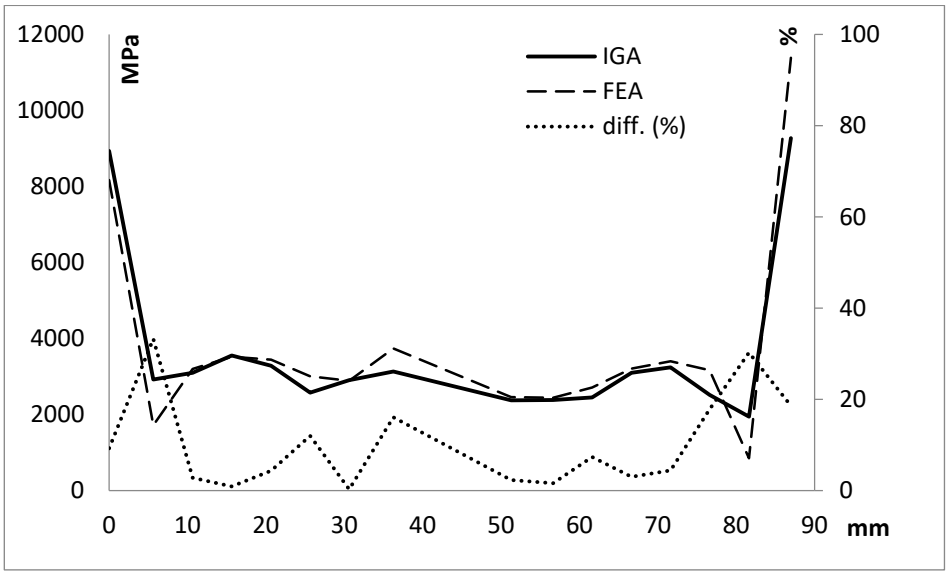


Fig. 8.39 Von Misses stresses along the path B.

8.3.7. Computational cost of each stage

The **Table 8.6** shows the time spent in each stage, in seconds and in percentage. In **Fig. 8.40** the cost of stages A, C and E are compared.

Table 8.6 Computational cost.

stage		t (s)	%
A	Generation of solid and surfaces identification	2.06	0.8
B	Initiation of patches and representation	47.31	19.5
C	Discretization of patches	86.98	35.8
D	Allocation of boundary conditions	12.14	5.0
E	Analysis	94.16	38.8
	TOTAL	242.65	100.0

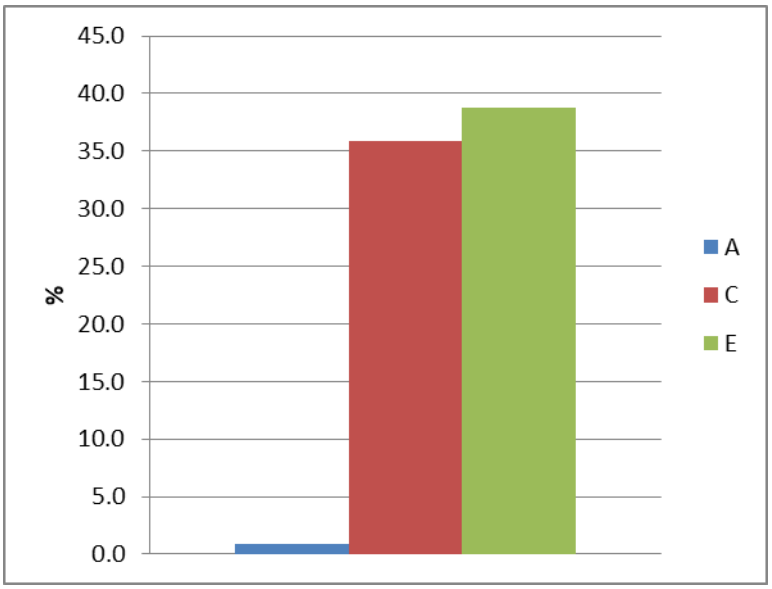


Fig. 8.40 Comparison of cost, in percentage, between geometrical definition (A), discretization (C) and analysis (E).

8.3.8. Coupling quality VS triangulation of coupling surface

The discretization of the coupling surface has an impact on the quality of the coupling. Indeed, if we set a path between patches 1 and 2, with nine sample points as shown in **Fig. 8.41** (path C), the distance between both patches along that path in the deformed configurations is not zero as theoretically it should be. The existence of this distance is due to the discretization of the coupling surface, because the integral of the patches basis functions on the coupling surface is done as summation of the Gauss points (recall section 7.2.3).

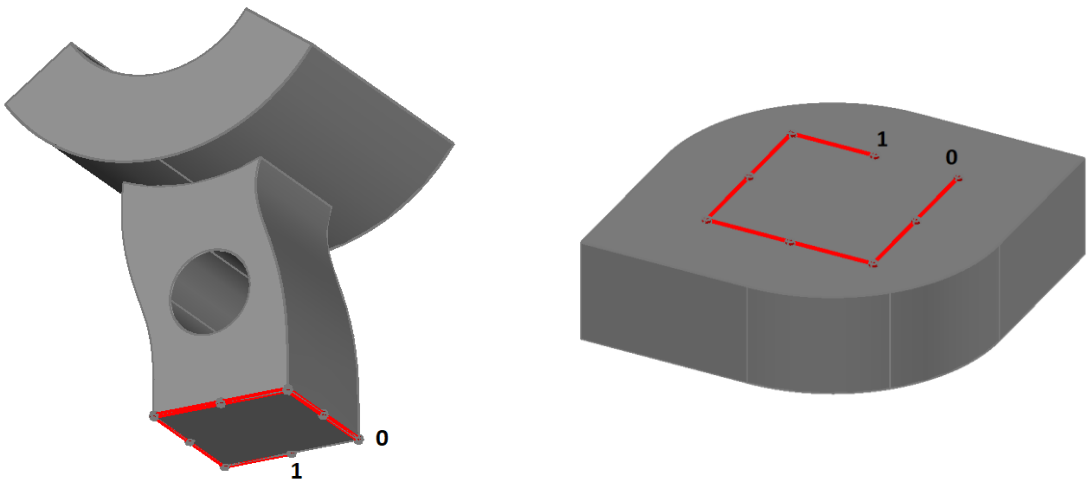


Fig. 8.41 Path C.

As a consequence, the more Gauss points on the coupling surface, the less the distance between both patches. To illustrate this effect, four triangulations have been done for the coupling surface 2 (that is contoured by the path C). The sizes of the triangles are shown in **Fig. 8.42**.

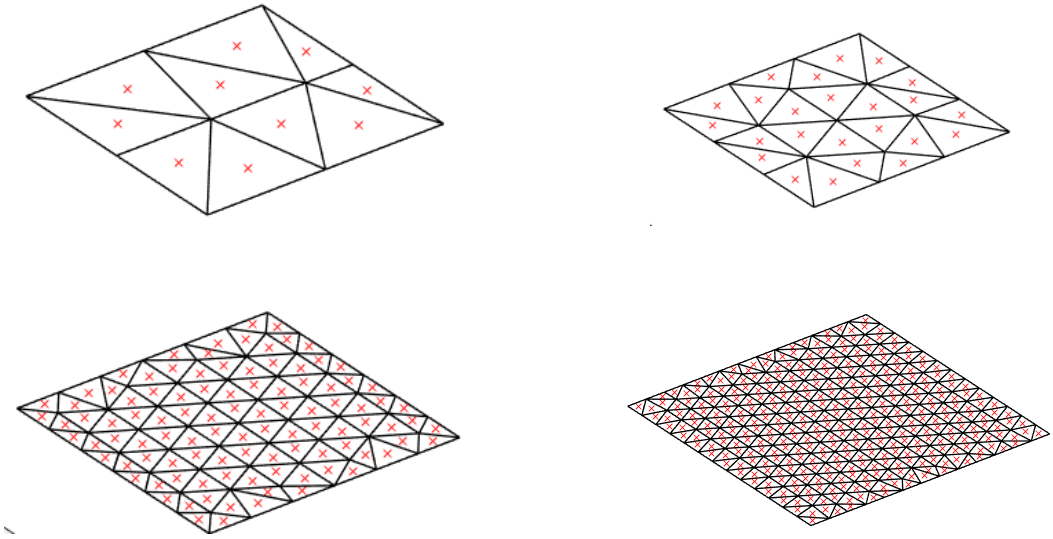


Fig. 8.42 Discretization of coupling surface 2 with triangle side equals to 16, 8, 4 and 2.

The deformed shape for those four cases is depicted in **Fig. 8.43** (scaled x 5). One can observe that the finer the mesh on the boundary surface, the less distance between both patches, i.e. the closer to the theoretical case.

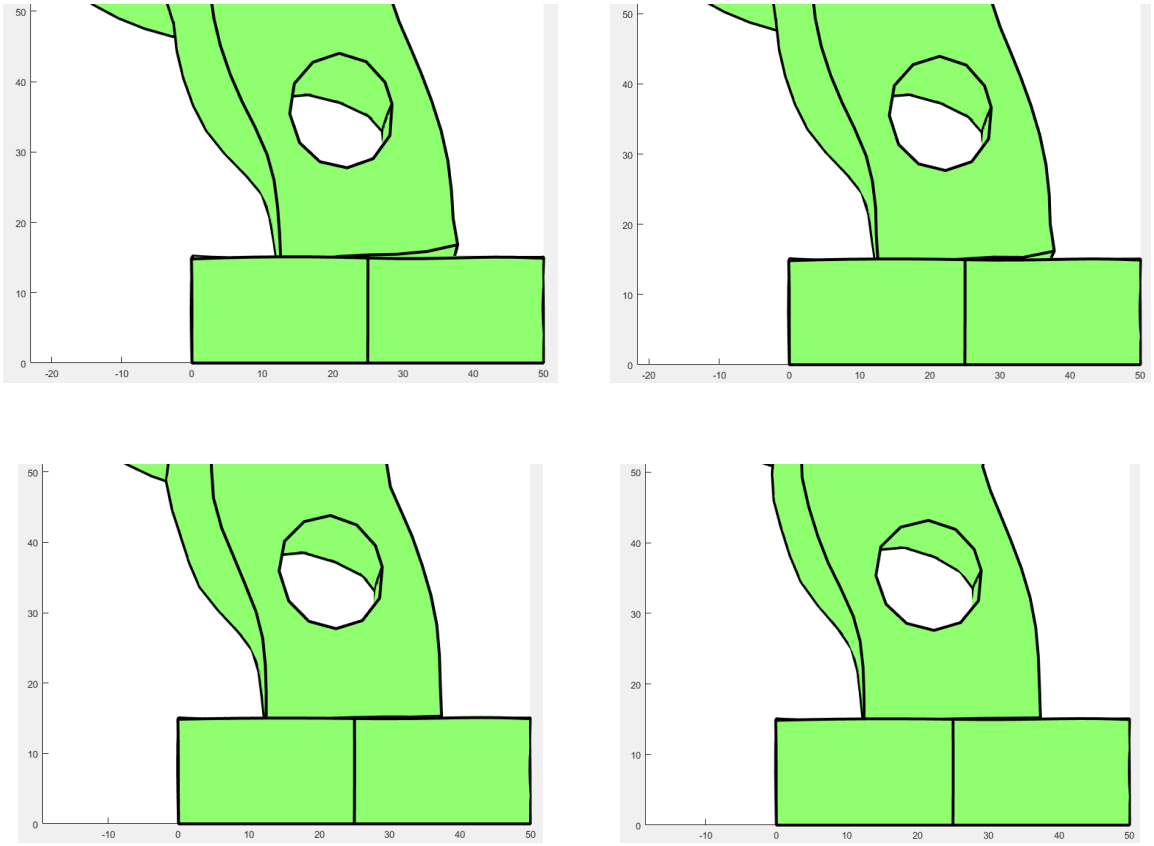


Fig. 8.43 Detail of coupling between patches 1 and 2, for coupling surface with triangle side equals to 16, 8, 4 and 2.

To highlight this effect, **Fig. 8.44** plots the difference between both patches in the deformed shape along the nine sample points of the path C. That difference is drastically reduced when the size of the triangles passes from $R = 16$ to $R = 2$.

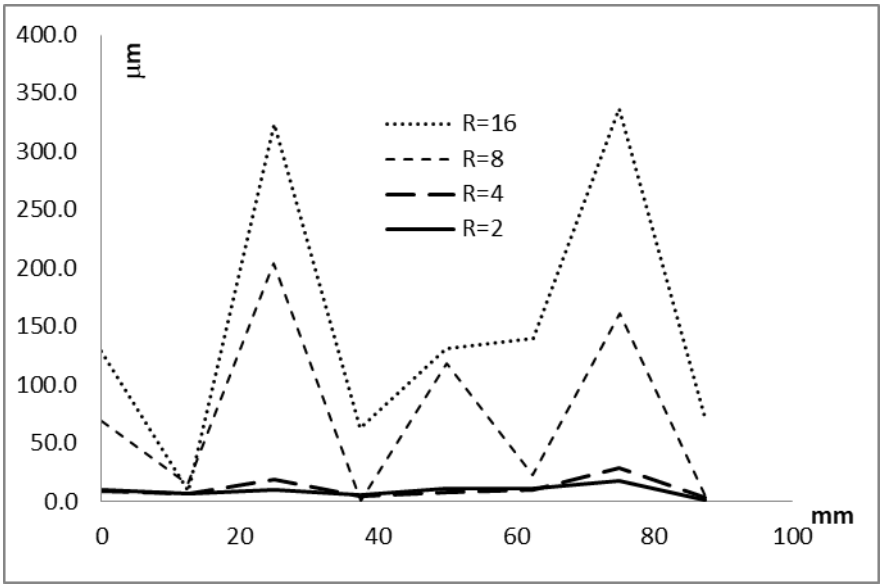


Fig. 8.44 Difference in the deformed shape between patches 1 and 2 at points of path C.

9. **Conclusions**

To conclude this work, this chapter revisits the objectives in section 9.1, where outlines how those objectives have been attained. The contributions and their location in the thesis are enumerated in section 9.2. Section 9.3 details the limitations of this work and the potential improvements and extensions. Finally, section 9.4 closes the thesis with a reflection about IGA from trimmed solids.

9.1. Revisiting the objectives

The implementation of a new algorithm for analysing solids imported from CAD in with IGA has been achieved in this thesis. The objectives have been covered along this work. First, the objectives proposed in Chapter 1 are repeated below for convenience. Then, explanation of how each one is attained is given afterwards.

Objectives:

> ***a***) Codification of an algorithm that encompasses all the processes: from IGES files interpretation to results visualization.
>
> ***b***) Creation of a translator from IGES files to numerical data readable by the code.
>
> ***c***) Parametrization of the volumes contained in the IGES files.
>
> ***d***) Discretization of the solids with adaptability to capture any trimmed shape.
>
> ***e***) Application of constraints to interfaces non-conformal with the solid domain.
>
> ***f***) Improvement of point projection technique to increase its robustness.
>
> ***g***) Representation of the surfaces efficiently and independently of their parametrization.

The objective ***a*** is accomplished as the code has been created in this thesis. The code (provided in the attached file) and has been developed by the author from scratch in Matlab®. Functions and classes are grouped by folders, numbered to facilitate their location. Chapter 3 briefs the whole algorithm, highlighting the main stages. Appendix 3A gives schemes of the code itself and the main variables.

The first part of the algorithm has a routine that transforms the information of IGES files into numerical arrays, which are readable by the code. In addition, that information is interpreted to prepare the solid parametrization and the discretization of boundary surfaces, which are done afterwards. The transformation of IGES to numbers and their interpretation is the objective ***b***. The solid parametrization is the objective ***c***. Both objectives are explained in Chapter 4.

The objective ***d*** is covered in Chapters 5 and 6. The trimmed solid is discretized in tetrahedrons that provide adaptability to any shape. Mixed-degree tetrahedrons are located at the trimming surfaces to enhance the accuracy. This tetrahedralization process is explained in Chapter 6. Previously, in Chapter 5, the trimming surfaces triangulation into quadratic triangles with error under control is explained.

Chapter 7 covers the objective ***e***. The constraints and couplings are weakly imposed using Lagrange multipliers. This method is well settled and extended in IGA and FEA, hence the novelty is not the method itself, but the adjacent procedures derived from coupling on trimming surfaces: approximation of the surface and tetrahedral discretization.

Since point projection is ubiquitous in the whole thesis, developing a robust technique was relevant (objective ***f***). Appendix 10A explains the novel point procedure approach developed here that is called marching point projection (*MPP*).

The objective ***g***, representation of the surfaces of the solid, is covered in Appendix 10B. These surfaces are triangulated with a new procedure called quasi-isotropic initial triangulation (*QIT*). The *QIT* algorithm delivers triangulation with equilateral triangles and uniformity in their sizes. These two features are desirable in the visualization of the analysis results. Equilateral triangles involve Jacobians close to regular, which provides accuracy when interpolating the stresses at the solid surface. Uniformity (nodes are evenly spread throughout the whole surface) allows detecting stress peaks at any location regardless of the parametrization of the surface.

9.2. Contributions and location in the thesis

Apart from the whole algorithm itself, which allows reading CAD information to be used directly for IGA, there are five key contributions in this thesis:

- A new approach for solid parametrization based on lattice analogy is proposed in section 4.2.3. This approach delivers quality parametrization with low computational cost. In brief, the control net of the solid is transformed into a three-dimensional truss. Prescribed displacements are applied to the nodes of such truss to achieve the desired arrangement by the direct stiffness method.
- A new procedure to approximate trimming surfaces by quadratic triangulations in the patch parameter space is proposed in sections 5.3, 5.4 and 5.5. To control the error, the derivatives of the surface in the patch parameter space are estimated by finite divided differences.
- Solid spatial discretization with mixed-degree tetrahedrons for trimmed solids is proposed in section 6.4. These tetrahedrons introduce flexibility to the capture the domain shape and the computational effort is reduced compared to purely linear or purely quadratic

meshes. The number and distribution of tetrahedrons is related to the untrimmed version of the solid.

- Point projection technique is improved in terms of robustness by the marching point projection (*MPP*), presented in Appendix 10A. *MPP* is based on estimation of lengths and directions to achieve the target point from one trial location. It uses the marching method, but it is expanded for curves and solids. The robustness of *MPP*, i.e. extension of domain where the trial point converges to the target, is greater than the traditional Newton-Raphson iterations.
- For surfaces triangulation a new algorithm called *QIT* is introduced in Appendix 10B. *QIT* delivers high quality triangulations because the majority of the triangles are equilateral regardless of the surface parametrization or shape. The process does not require remeshing, i.e. the triangles are generated at once.

9.3. Research limitations and future work

The algorithm developed in this thesis has limitations, which may be seen as potential improvements for the future. The most relevant in the author's opinion are explained in this section.

9.3.1. Restrictions in CAD geometries readable by the algorithm

CAD geometries that are readable in this algorithm can have any shape in case of *trimmed solids*, except a closed shape, i.e. one face of the solid coincides with another face of the same solid. To account for solids with closed shapes the *translator* from IGES files (Chapter 4) must be amended.

In case of untrimmed solids, that form the so-called *gross patches*, are limited to six faces. Extension to more faces may be possible by extending the options in the code (Chapter 4).

9.3.2. The parametrization with lattice analogy is rigid and not fully optimized

Parametrization of the solids by the lattice analogy uses initially six planes to approximate to the solid control net, one plane per face. Such technique reduces the admissible solid geometries to small curvatures. Certainly, if the solid has large curvatures the initial approximation by planes will introduce large errors that will be reproduced in the final lattice shape or even lead to intersection of opposite planes, which is not acceptable. To provide more flexibility, the initial truss shape may be approximated using different forms than plane, e.g. cylindrical surfaces.

In addition, the ratio of stiffness rods/diagonals is studied within the variation of one order of magnitude. However, extending such range may yield higher quality parametrization. These implementations should be done in the lattice fitting algorithm (section 4.2.3).

9.3.3. Inaccuracies in the error estimation when approximating to trimming surfaces

The approximation to trimming surfaces in the patch parameter space is achieved by fitting quadratic triangulations to such surfaces. The spacing between the nodes of the triangulations is controlled by the error estimation. The error depends on the third derivative (Isaacson and Keller, 2012) of the surface in the patch parameter space that is unknown. Hence the third derivative is estimated by finite divided differences (FDD), using a set of sample points.

The computation of those sample points involves point projection. The error committed in the point projection will reflect in the third derivatives estimation, i.e. the error in the FDD is linked to the error in the point projection of the sample points. This relationship, which has not been addressed in this work, requires further study to guarantee accuracy in the trimming surface approximation.

Apart from those two coupled errors, the distribution of sample points within the trimming surface is uniform. However, an improved approach would need denser sample points where the variation of the curvature is greater. These improvements should apply to section 5.3 of the thesis.

9.3.4. Tetrahedralization is not optimal and lacks robustness

Discretization of the trimmed solid is achieved by tetrahedral mesh subjected to trimming surface constraints. Although the results are acceptable, two issues appear in the current meshing algorithm.

The first item regards the tetrahedrons quality. The final mesh quality is not controlled and consequently distorted tetrahedrons may appear. The meshing algorithm is susceptible of improvement by nodal relocation or change mesh connections (Ruppert, 1995; Si, 2008). Secondly the constrained tetrahedralization process demands insertion of new nodes at the trimming surface (Shewchuk, 2009). This insertion for complex geometries (with large curvatures) may yield self-intersecting tetrahedrons that invalidate the mesh. This drawback happens because the process of new nodes insertion needs to be more robust, i.e. generate correct mesh whatever the geometry is. These two improvements apply to section 6.4.

9.3.5. Optimal number of Gauss points on the boundary surfaces

Constraints (coupling and Dirichlet boundary conditions) are achieved by integrating the domains basis functions across boundary surfaces (Apostolatos et al., 2014). If the number of used Gauss points is too low, the constraints are not fulfilled (see example of section 8.3.7). On the other hand, if the number of Gauss points is too high, many of them may be redundant increasing the computational cost unnecessarily. The problem of finding the optimal number of Gauss points in the boundary surfaces is not solved in this thesis. The answer to this question would be located in section 7.1.

9.3.6. Ill conditioned stiffness matrix

The imposition of constraints by Lagrange multipliers bring stiffness matrix ill-conditioned, i.e. the components at some of the matrix rows are several orders of magnitude lower than the rest. The system is solvable for the examples shown in this thesis, but for larger stiffness matrices the error might be unacceptable or the system can even become singular. One alternative is the Penalty method (Babuška, 1973), but it introduces an error in the displacements (the penalty factor). Other alternative more suitable is the augmented Lagrange multipliers approach (Hestenes, 1969; Bertsekas, 2014). The implementation of this improvement would affect to Chapter 7.

9.3.7. Non-linear plastic analysis

This algorithm is devised for linear elastic analysis with small displacements. To account for geometrical non-linearities, the same algorithm might be used but the time needs to be discretized, i.e. the forces are applied in steps. The updated Lagrange approach is the most used for these non-linearities where, in each step, the reference is updated to the deformed shape (Kojić and Bathe, 1987). In each step the domain and the trimming surfaces must be updated to account for the displacements.

To account for material non-linearities, i.e. plastic behaviour and damage, the return mapping is the most common method (Simo and Taylor, 1986; Krajcinovic and Lemaitre, 1987; Lee and Fenves, 2001; Neto et al., 2008). It requires also time discretization and in each step the equilibrium is recovered accounting for the material yielding. The equilibrium conditions alongside with the material internal variables that depend on the history loading are the inputs of a non-linear system of equations that is solved by Newton-Raphson iterations.

The structure of the code is prepared to include both improvements: non-linear geometry and plasticity, but they still need to be implemented.

9.3.8. Local refinement

As mentioned in section 2.1.3, local refinement is not allowed in NURBS. To allow local refinement the basis functions must be T-splines (Bazilevs *et al.*, 2010) or the other options mentioned in section 2.1.3. The implementation of T-splines in this work would impact on all stages of the algorithm, since all the routines that use or return NURBS need amendment for T-splines.

9.4. Final reflection about IGA for trimmed solids

In the traditional design process, the geometry is created in the CAD modelling step, with its own mesh (given by NURBS parametrization), and the analysis requires to create another mesh to integrate the domain and solve the PDE. The primitive driving force for IGA was to avoid this duplication of the mesh. The idea was to use the NURBS parametrization generated during the CAD modelling to integrate the PDE. In addition other advantages came up in IGA such us high inter-elemental continuity and fully accurate geometry. We have seen in Chapters 1 and 2 that these ideas fully operate for un-trimmed curves and surfaces.

In general, for un-trimmed solids, the main idea of using NURBS parametrization as integration mesh can be applied, and the other two advantages (high continuity and geometrical accuracy) remain. However, in the case of solids, CAD does not produce the solid itself but the enveloping surfaces. Therefore the solid needs to be parametrized, penalizing the efficiency of IGA in solids.

For trimmed solids we need to consider two additional items: the NURBS parametrization cannot be used to integrate the domain since it does not coincide with the solid to analyse; and the geometrical accuracy is not fulfilled on the trimming surfaces.

The lack of geometrical accuracy can be controlled by adjusting triangulated surface to the trimming surface, however the computational cost is increased. The integration mesh, that is different from the NURBS parametrization, can be carried out and the results are acceptable. However the mesh is duplicated: one mesh was done in the CAD modelling stage, and the second mesh is performed in the analysis to adapt to the trimmed domains.

In view of the previous comments, it seems that IGA for trimmed solids still needs to be improved, especially compared against untrimmed-surfaces and curves, where IGA fully complies with all the advantages.

APPENDIXES

Appendix 2A: Miscellanea of mathematics

2A.1 General operations

Table 2A.1 General operations

Item	Symbol	Examples / comments
Scalar (0 order tensor)	Lower case letters	a, b
Vector (1 order tensor)	Lower case bold letters	$\boldsymbol{u}, \boldsymbol{v}$
Matrices (2 order tensor)	Upper case letters	$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$
Higher order tensors	Upper case script letters	$\mathbb{S}, \mathbb{T}$
del operator	∇	$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)$
Laplace operator	Δ	$\Delta = \nabla \cdot \nabla$
Gradient of scalar	$\nabla a = a_{,i}$	$\nabla a = \begin{Bmatrix} a_{,x} \\ a_{,y} \\ a_{,z} \end{Bmatrix}$
Gradient of vector	$\nabla \boldsymbol{u} = u_{i,j}$	$\nabla \boldsymbol{u} = \begin{bmatrix} u_{1,x} & u_{1,y} & u_{1,z} \\ u_{2,x} & u_{2,y} & u_{2,z} \\ u_{3,x} & u_{3,y} & u_{3,z} \end{bmatrix}$
Divergence of vector	$\nabla \cdot \boldsymbol{u} = \nabla_i \cdot u_i = u_i \nabla_i$	$\nabla \cdot \boldsymbol{u} = u_{1,x} + u_{2,y} + u_{3,z}$
Divergence of matrix	$\nabla \cdot \boldsymbol{A} = \nabla_j \cdot A_{ij}$	$\nabla \cdot \boldsymbol{A} = \begin{Bmatrix} A_{i1}\nabla_1 \\ A_{i2}\nabla_2 \\ A_{i2}\nabla_2 \end{Bmatrix}$
Laplacian of scalar	$\Delta a = \nabla \cdot \nabla a = a_{,i}\, \nabla_i = a_{,ii}$	$\Delta a = a_{,xx} + a_{,yy} + a_{,zz}$
Laplacian of vector	$\Delta \boldsymbol{u} = \nabla \cdot \nabla \boldsymbol{u} = u_{i,j}\, \nabla_j = u_{i,jj}$	$\Delta \boldsymbol{u} = \begin{Bmatrix} u_{1,11} + u_{1,22} + u_{1,33} \\ u_{2,11} + u_{2,22} + u_{2,33} \\ u_{3,11} + u_{3,22} + u_{3,33} \end{Bmatrix}$
Dot product between 2 vectors	$\boldsymbol{u} \cdot \boldsymbol{v} = u_i v_i$	$\boldsymbol{u} \cdot \boldsymbol{v} = a$
Dot product between matrix and vector	$\boldsymbol{A} \cdot \boldsymbol{u} = A_{ij} u_j$	$\boldsymbol{A} \cdot \boldsymbol{u} = v_i$
Dot product between two matrices	$\boldsymbol{A} \cdot \boldsymbol{B} = A_{ik} B_{kj}$	$\boldsymbol{A} \cdot \boldsymbol{B} = C_{ij}$
Double dot product between two matrices	$\boldsymbol{A} : \boldsymbol{B} = A_{ij} B_{ij}$	$\boldsymbol{A} : \boldsymbol{B} = a$
Double dot product between tensor 4th order and matrix	$\mathbb{S} : \boldsymbol{A} = \mathbb{S}_{ijkl} A_{kl}$	$\mathbb{S} : \boldsymbol{A} = \boldsymbol{B}$
Inner product of two functions within domain Λ (or bilinear form)	$\langle f, g \rangle$	$\langle f, g \rangle = \int_\Lambda f g \; d\Lambda$
Infinitesimal strain tensor	$\boldsymbol{\varepsilon} = \varepsilon_{ij}$	$\varepsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right)$
Hooke's law	$\sigma_{ij} = c_{ijkl}\, \varepsilon_{kl}$	$\boldsymbol{\sigma} = \mathbb{C} : \boldsymbol{\varepsilon}$

2A.2 Green´s theorem for d-dimensional case

The Green's theorem, also known as divergence theorem, can be seen as an integration by parts. Given two functions f and g such that $f, g: \mathbb{R}^d \rightarrow \mathbb{R}^1$ the Green's theorem is expressed as:

$$\int_{\Omega} f\, g_{,j}\, d\Omega = -\int_{\Omega} f_{,j}\, g\, d\Omega + \oint_{\Gamma} f\, g\, n_j\, d\Gamma \tag{2A.1}$$

Where the domain Ω and its boundary Γ is represented in **Fig. 2A.1** for the case $d = 3$ (three-dimensional domain).

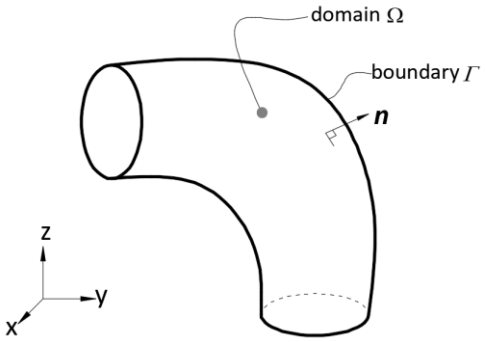


Fig. 2A.1 Domain Ω enclosed by the boundary Γ with normal vector $\boldsymbol{n}$.

Application of Green´s theorem to left-hand side of equilibrium equation

Let $\mathcal{L}$ be the left-hand side of the equilibrium equation:

$$\mathcal{L} = \int_{\Omega} \boldsymbol{w}^T\, (\nabla \cdot \boldsymbol{\sigma})\, d\Omega = \int_{\Omega} w_i\, \sigma_{ij,j}\, d\Omega$$

Identification of components:

$$f = w_i$$

$$g_{,j} = \sigma_{ij,j}$$

Then, applying Green´s theorem to $\mathcal{L}$:

$$\mathcal{L} = -\int_{\Omega} w_{i,j}\, \sigma_{ij}\, d\Omega + \int_{\Gamma} w_i\, \sigma_{ij}\, n_j\, d\Gamma$$

Expressed in tensor form:

$$\mathcal{L} = -\int_{\Omega} \partial \boldsymbol{w} : \boldsymbol{\sigma}\, d\Omega + \int_{\Gamma} \boldsymbol{w} \cdot \boldsymbol{\sigma} \cdot \hat{\boldsymbol{n}}\, d\Gamma$$

Using Voigt notation the matrix form is:

$$\mathcal{L} = -\int_{\Omega} \partial \boldsymbol{w}^T \, \boldsymbol{\sigma} \, d\Omega + \int_{\Gamma} \boldsymbol{w}^T \, \boldsymbol{\sigma} \, \hat{\boldsymbol{n}} \, d\Gamma$$

Recall that $\boldsymbol{\sigma} \, \boldsymbol{n} = \bar{\boldsymbol{t}}$ and $\boldsymbol{\sigma}$ is a function of displacement, the expression yields to:

$$\mathcal{L} = -\int_{\Omega} \partial \boldsymbol{w}^T (\boldsymbol{D} \, \partial \boldsymbol{u}) \, d\Omega + \int_{\Gamma} \boldsymbol{w}^T \, \bar{\boldsymbol{t}} \, d\Gamma$$

2A.3 Sobolev spaces

Let $H^1(\Omega)$ be the Sobolev space composed of all functions f defined as:

$$H^1(\Omega) = \{f | D^\alpha f \in L^2(\Omega), \alpha \leq 1\} \tag{2A.2}$$

Where $D^\alpha f$ is the derivative of order α of the function f, and $L^2(\Omega)$ is the space defined by the set of all functions g such that:

$$\int_\Omega g^2 d\Omega < +\infty \tag{2A.3}$$

Equation (2A.3) indicates that the square of g is integrable in the domain Ω.

2A.4 Lebesgue spaces

Lebesgue integral of f, for $1 \leq p \leq \infty$, in domain Ω:

$$\|f\|_{L^p,\Omega} := \left(\int_\Omega |f(x)|^p \, dx\right)^{1/p} \tag{2A.4}$$

Lebesgue spaces:

$$L^p, \Omega := \{f : \|f\|_{L^p,\Omega} < \infty\} \tag{2A.5}$$

2A.5 Closest location of one line to another line

Given two lines r and s, the closest location from r to s (and vice versa) may be obtained in an iterative process as follows.

We define the initial points on r and s, called $\boldsymbol{x}^r$ and $\boldsymbol{x}^s$ respectively. The closest location from $\boldsymbol{x}^r$ to s, called $\boldsymbol{x}^{crs}$, is computed with equation (2A.6), where $\boldsymbol{u}$ is a vector with the direction of the line s.

$$\boldsymbol{x}^{crs} = \boldsymbol{x}^s + \frac{\boldsymbol{u} \cdot (\boldsymbol{x}^r - \boldsymbol{x}^s)}{\|\boldsymbol{u}\|^2} \boldsymbol{u} \tag{2A.6}$$

Then the closest location from $\boldsymbol{x}^{crs}$ (that lies on line s) to line r is computed with the same procedure, obtaining $\boldsymbol{x}^{csr}$. The process is repeated, computing the location alternatively on line r and line s, until two consecutive values in one line differ less than a pre-established tolerance.

2A.6 Floor and ceiling functions

Floor

$$floor(x) = max\{m \in \mathbb{Z} \,|m \leq x\} \tag{2A.7}$$

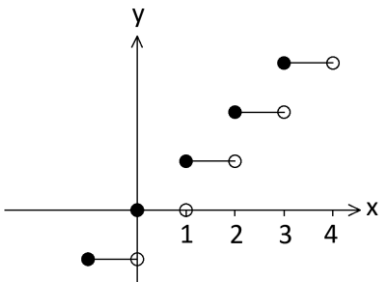


Fig. 2A.2 Floor function.

Ceiling

$$ceil(x) = min\{n \in \mathbb{Z} \,|n \geq x\} \tag{2A.8}$$

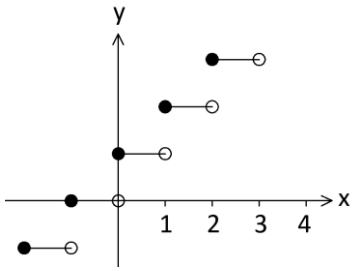


Fig. 2A.3 Ceiling function.

2A.7 Errors in Lagrange polynomial interpolations for curves

Mapping $\mathbb{R}^1 \rightarrow \mathbb{R}^1$

Given a function $x = f(u)$, such that $f: \mathbb{R}^1 \rightarrow \mathbb{R}^1$ defined in the domain Ω and a set of $n+1$ locations $u_0, \ldots, u_n$ equally spaced within Ω, called nodes. This function may be approximated by $g(u)$, defined as (2A.9), in the domain Ω. The order of the approximation is n.

$$g(u) = \sum_{i=0}^{n} L_i(u)\, f(u_i) \tag{2A.9}$$

where $L_i(u)$ are the Lagrangian polynomials, whose value is equal to $f(u_i)$ at u_i and zero at the rest of nodes locations. The Lagrangian polynomials are defined as follows:

$$L_i(u) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{u - u_j}{u_i - u_j} \tag{2A.10}$$

The difference between f and g a one location u_a, *i.e.* the error, when g is of order n is calculated as equation (2A.11).

$$E_n \leq \frac{f^{(n+1)}(u_\beta)}{(n+1)!} \prod_{j=0}^{n} (u_a - u_j) \tag{2A.11}$$

Where x_β is an unknown location within Ω that maximizes E_n. The error for $n = 2$ (quadratic approximation) is given in (2A.12). **Fig. 2A.4** illustrates one example.

$$E_2 \leq \frac{f'''(u_\beta)}{3!} (u_a - u_0)(u_a - u_1)(u_a - u_2) \tag{2A.12}$$

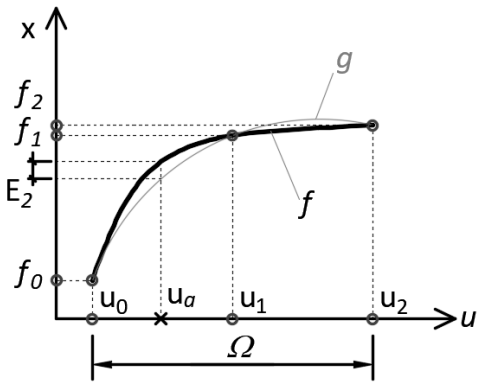


Fig. 2A.4 Quadratic approximation to f and error at *a*-location.

Mapping $\mathbb{R}^1 \rightarrow \mathbb{R}^d$

Given a function $\boldsymbol{x} = f(u)$, such that $f: \mathbb{R}^1 \rightarrow \mathbb{R}^d$ defined in the domain Ω and a set of $n+1$ locations $u_0, \ldots, u_n$ equally spaced within Ω, called nodes. This function may be approximated by $g(u)$, defined as (2A.13), in the domain Ω. The order of the approximation is n.

$$g(u) = \sum_{i=0}^{n} L_i(u)\, f(u_i) \tag{2A.13}$$

The Lagrange polynomials and the error of the approximation are computed in the same manner as equation (2A.11). In this case, the error is a vector with d components. Equation (2A.14) shows the error of the jth component for quadratic approximation. It might be convenient for some application to use the norm of the error vector as shown in equation (2A.15). **Fig. 2A.5** illustrates one example where f maps into $\mathbb{R}^3$.

$$E_{2\,j} \leq \frac{f'''(u_\alpha)_j}{3!}\,(u_a - u_0)(u_a - u_1)(u_a - u_2) \tag{2A.14}$$

$$E_2 = \left\| E_{2\,j} \right\| \tag{2A.15}$$

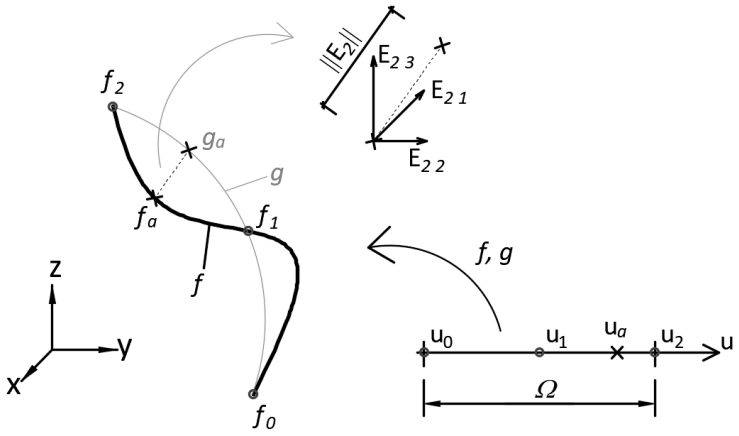


Fig. 2A.5 Quadratic approximation to f and error at *a*-location in 3D-mapping.

Maximization of intervals product in linear and quadratic interpolation error

Let us refer as p_n to the product of $n+1$ intervals for the error estimation. For linear and quadratic interpolations, this product is:

$$p_1(u) = (u_a - u_0)(u_a - u_1)\text{ , for } n = 1$$

$$p_2(u) = (u_a - u_0)(u_a - u_1)(u_a - u_2)\text{ , for } n = 2 \text{ (see equation 2A.13).}$$

We assume that Ω spans from -1 to +1. Therefore the coordinates of the nodes for linear and quadratic approximations are:

$$\boldsymbol{u} = (-1, 1)\text{ , for } n = 1$$

$$\boldsymbol{u} = (-1, 0, 1)\text{ , for } n = 2$$

The location where p is maximum (or minimum) is given by equating the derivative p' to zero. That occurs at locations presented in **Table 2A.2.**

Table 2A.2 Locations of maximums and minimums of increment products.

n	u for $\Omega \in [-1, +1]$	u for $\Omega \in [-L/2, +L/2]$	p for $\Omega \in [-1, +1]$
1	0	L	-1.000
2	-0.578	0.211 L	+0.385
	0.578	0.788 L	-0.385

Fig. 2A.6 plots both products, linear and quadratic, where the maximums indicated in **Table 2A.2** can be seen.

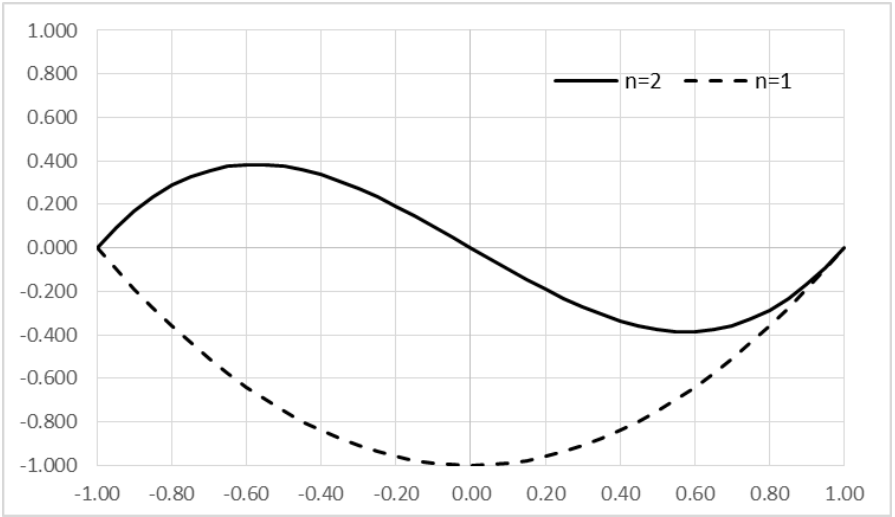


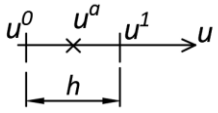
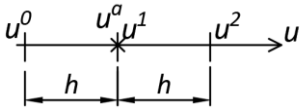
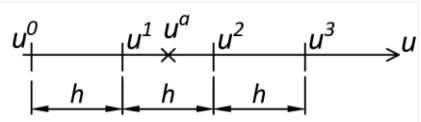
Fig. 2A.6 Plot of interval product function for linear and quadratic approximation and Ω span (-1,+1).

2A.8 Numerical differentiation: Finite divided differences

Mapping $\mathbb{R}^1 \to \mathbb{R}^d$

Given a function $\boldsymbol{x} = f(u)$, such that $f: \mathbb{R}^1 \to \mathbb{R}^d$, its derivatives up to 3rd order at one location u^a may be approximated as shown in **Table 2A.3**. The sample locations u^i, with $i = 0, \ldots, 3$, are equally spaced at step h. Note the derivatives in **Table 2A.3** are computed for each *d*-component. The location of f at u^i is indicated as f_i.

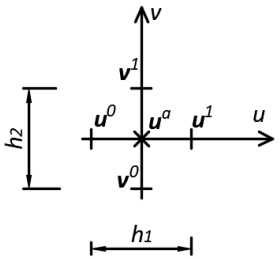
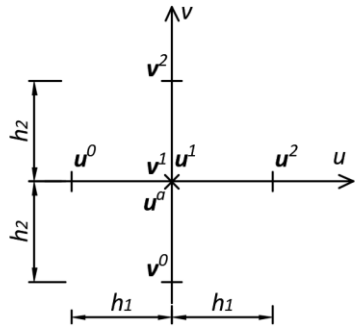
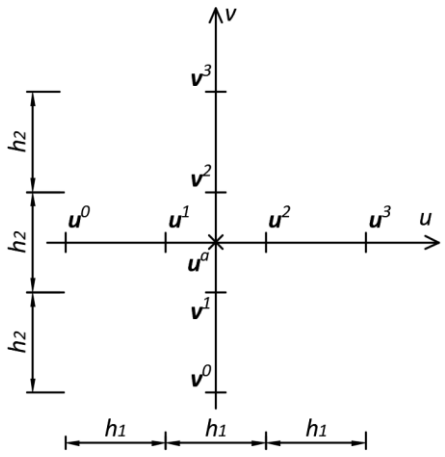
Table 2A.3 Approximation of derivatives centred at u^a location.

Order	Equation	Scheme
1	$f'(u^a) \cong \frac{f_1 - f_0}{h}$	
2	$f''(u^a) \cong \frac{f_2 - 2\,f_1 + f_0}{h^2}$	
3	$f'''(u^a) \cong \frac{f_3 - 3\,f_2 + 3\,f_1 - f_0}{h^3}$	

Mapping $\mathbb{R}^2 \rightarrow \mathbb{R}^d$

Given a function $\boldsymbol{x} = f(\boldsymbol{u})$, such that $f: \mathbb{R}^2 \rightarrow \mathbb{R}^d$, its derivatives up to 3rd order may be approximated as shown in **Table 2A.4**. The components of the $\boldsymbol{u}$ vector are $(u, v)^T$.The sample locations $\boldsymbol{u}^i$ are equally spaced at step $\boldsymbol{h}$. Note the derivatives in **Table 2A.4** are computed for each *d*-component. The location of f at u^i and v^i are indicated as f_{ui} and f_{vi} respectively.

Table 2A.4 Approximation of derivatives centred at $\boldsymbol{u}^a$ location.

Order	Equation	Scheme
1	$f_{,u}(\boldsymbol{u}^a) \cong \frac{f_{u1} - f_{u0}}{h_1}$ $f_{,v}(\boldsymbol{u}^a) \cong \frac{f_{v1} - f_{v0}}{h_2}$	
2	$f_{,uu}(\boldsymbol{u}^a) \cong \frac{f_{u2} - f_{u1} - f_{u0}}{{h_1}^2}$ $f_{,vv}(\boldsymbol{u}^a) \cong \frac{f_{v2} - f_{v1} - f_{v0}}{{h_2}^2}$	
3	$f_{,uuu}(\boldsymbol{u}^a) \cong \frac{f_{u3} - f_{u2} - f_{u1} - f_{u0}}{{h_1}^3}$ $f_{,vvv}(\boldsymbol{u}^a) \cong \frac{f_{v3} - f_{v2} - f_{v1} - f_{v0}}{{h_2}^3}$	

2A.9 Derivatives in NURBS

The formulation given in this appendix is based on (Piegl and Tiller, 1996).

Different expressions for NURBS entities

Next expressions are given for three dimensional parameter space, the reduction to 2D and 1D domains is trivial. The domain is defined as:

$$\boldsymbol{\Omega}(\xi,\eta,\chi) = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} R_{i,j,k}(\xi,\eta,\chi)\boldsymbol{P}_{i,j,k} \tag{2A.16}$$

Where $\boldsymbol{P}_{i,j,k}$ are the coordinates of the $(i,j,k)_{th}$ control point. The basis functions $R_{i,j,k}$ are calculated as:

$$R_{i,j,k}^{p,q,r}(\xi,\eta,\chi) = \frac{N_{i,p}(\xi)\, M_{j,q}(\eta)\, L_{k,r}(\chi)\, w_{i,j,k}}{\sum_{\hat{\imath}=1}^{n}\sum_{\hat{\jmath}=1}^{m}\sum_{\hat{k}=1}^{l} N_{\hat{\imath},p}(\xi)\, M_{\hat{\jmath},q}(\eta)\, L_{\hat{k},r}(\chi)\, w_{\hat{\imath},\hat{\jmath},\hat{k}}} = \frac{N_{i,p}(\xi)\, M_{j,q}(\eta)\, L_{k,r}(\chi)\, w_{i,j,k}}{W(\xi,\eta,\chi)} \tag{2A.17}$$

The domain might be expressed as:

$$\boldsymbol{\Omega}(\xi,\eta,\chi) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} N_{i,p}(\xi)\, M_{j,q}(\eta)\, L_{k,r}(\chi)\, w_{i,j,k}\, \boldsymbol{P}_{i,j,k}}{W(\xi,\eta,\chi)} \tag{2A.18}$$

Indexes i,j,k may be aggregated in I, and number of control points in $\aleph$. If dependencies are dropped off, then (2A.16) may be written as:

$$\boldsymbol{\Omega} = \sum_{I=1}^{\aleph} R_I\, \boldsymbol{P}_I \tag{2A.19}$$

With basis functions expressed as:

$$R_I = \frac{NML\, w_I}{\sum_{\hat{I}=1}^{\aleph} NML\, w_{\hat{I}}} = \frac{NML\, w_I}{W} \tag{2A.20}$$

An alternative expression for the domain is:

$$\boldsymbol{\Omega} = \frac{A}{W} \tag{2A.21}$$

With $\boldsymbol{A}$ and W being:

$$\boldsymbol{A} = \sum_{I=1}^{\aleph} NML\, w_I\, \boldsymbol{P}_I \tag{2A.22}$$

$$W = \sum_{\hat{l}=1}^{\aleph} NML\, w_{\hat{l}} \tag{2A.23}$$

Derivatives of NURBS entities

Next expressions are given for three dimensional parameter space, the reduction to 2D and 1D domains is trivial. We can compute the derivative of $\boldsymbol{\Omega}$ via derivative of $\boldsymbol{A}$ as shown below. Super-indexes k, l, m indicates derivatives in each parameter direction.

$$\boldsymbol{A}^{(k,l,m)} = [[[W\,\boldsymbol{\Omega}]^k]^l]^m \tag{2A.24}$$

That may be developed as shown below arriving to (2A.25).

$$\boldsymbol{A}^{(k,l,m)} = [[[W\,\boldsymbol{\Omega}]^k]^l]^m = \left[\left[\sum_{i=0}^{k}\binom{k}{i} W^{(i,0,0)}\,\boldsymbol{\Omega}^{(k-i,0,0)}\right]^l\right]^m$$

$$\boldsymbol{A}^{(k,l,m)} = \left[\sum_{i=0}^{k}\binom{k}{i}\sum_{j=0}^{l}\binom{l}{j} W^{(i,j,0)}\,\boldsymbol{\Omega}^{(k-i,l-j,0)}\right]^m$$

$$\boldsymbol{A}^{(k,l,m)} = \sum_{i=0}^{k}\binom{k}{i}\sum_{j=0}^{l}\binom{l}{j}\sum_{r=0}^{m}\binom{m}{r} W^{(i,j,r)}\,\boldsymbol{\Omega}^{(k-i,l-j,m-r)} \tag{2A.25}$$

If equation (2A.25) is developed to write derivatives of $\boldsymbol{\Omega}$ we obtain:

$$\begin{aligned}\boldsymbol{A}^{(k,l,m)} = W^{(0,0,0)}\,\boldsymbol{\Omega}^{(k,l,m)} &+ \sum_{i=1}^{k}\binom{k}{i} W^{(i,0,0)}\,\boldsymbol{\Omega}^{(k-i,l,m)} + \sum_{j=1}^{l}\binom{l}{j} W^{(0,j,0)}\,\boldsymbol{\Omega}^{(k,l-j,m)} \\ &+ \sum_{r=1}^{m}\binom{m}{r} W^{(0,0,r)}\,\boldsymbol{\Omega}^{(k,l,m-r)} \\ &+ \sum_{i=1}^{k}\binom{k}{i}\sum_{j=1}^{l}\binom{l}{j}\sum_{r=1}^{m}\binom{m}{r} W^{(i,j,r)}\,\boldsymbol{\Omega}^{(k-i,l-j,m-r)}\end{aligned} \tag{2A.26}$$

In equation (2A.26) we can isolate the target derivative $\boldsymbol{\Omega}^{(\mathrm{k,l,m})}$ as a function of previous derivatives of $\boldsymbol{\Omega}$ and derivatives of $\boldsymbol{A}$ and W, as expressed in equation (2A.26). Note that $W^{(0,0,0)}$ is the weight itself:

$$\boldsymbol{\Omega}^{(k,l,m)} = \frac{1}{W^{(0,0,0)}} \left[\boldsymbol{A}^{(k,l,m)} - \sum_{i=1}^{k} \binom{k}{i} W^{(i,0,0)}\, \boldsymbol{\Omega}^{(k-i,l,m)} - \sum_{j=1}^{l} \binom{l}{j} W^{(0,j,0)}\, \boldsymbol{\Omega}^{(k,l-j,m)} - \sum_{r=1}^{m} \binom{m}{r} W^{(0,0,r)}\, \boldsymbol{\Omega}^{(k,l,m-r)} - \sum_{i=1}^{k} \binom{k}{i} \sum_{j=1}^{l} \binom{l}{j} \sum_{r=1}^{m} \binom{m}{r} W^{(i,j,r)}\, \boldsymbol{\Omega}^{(k-i,l-j,m-r)} \right] \tag{2A.27}$$

$\boldsymbol{A}$ and W are equivalent to B-splines, recall equation (2A.17) and (2A.22), therefore their derivatives are computed as B-splines derivatives, that are a function of derivatives of their basis functions as shown below:

$$\boldsymbol{A}^{(k,l,m)} = \left[\sum_{I=1}^{\aleph} NML\, w_I\, \boldsymbol{P}_I \right]^{(k,l,m)} = \sum_{I=1}^{\aleph} N^{(k)} M^{(l)} L^{(m)}\, w_I\, \boldsymbol{P}_I = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{l} N_i^{(k)}\, M_j^{(l)}\, L_k^{(m)}\, w_{ijk} \boldsymbol{P}_{ijk} \tag{2A.28}$$

$$W^{(k,l,m)} = \left[\sum_{I=1}^{\aleph} NML\, w_I \right]^{(k,l,m)} = \sum_{I=1}^{\aleph} N^k M^l L^m\, w_I = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{l} N_i^{(k)}\, M_j^{(l)}\, L_k^{(m)}\, w_{ijk} \tag{2A.29}$$

Derivative of unidimensional B-spline basis function

The kth derivative is computed recursively as shown in (2A.23):

$$\left({}^{p}N_i \right)^{(k)} = \frac{p!}{(p-k)!} \sum_{j=0}^{k} \alpha_{k,j}\ {}^{p-k}N_{i+j} \tag{2A.30}$$

With:

$$\alpha_{0,0} = 1 \tag{2A.31a}$$

$$\alpha_{k,0} = \frac{\alpha_{k-1,0}}{\xi_{i+p-k+1} - \xi_i} \tag{2A.31b}$$

$$\alpha_{k,j} = \frac{\alpha_{k-1,j} - \alpha_{k-1,j-1}}{\xi_{i+p+j-k+1} - \xi_{i+j}} \quad with\ j = 1, \dots, k-1 \tag{2A.31c}$$

$$\alpha_{k,k} = \frac{-\alpha_{k-1,k-1}}{\xi_{i+p+1} - \xi_{i+k}} \tag{2A.31d}$$

First derivative of unidimensional NURBS basis function

The first derivative of a NURBS basis function may be computed with the general expression (2A.30) or as follows:

$$R_{i,\xi} = \frac{N_{i,\xi}\ w_i W - N_i\ w_i\ W_{,\xi}}{W^2} \tag{2A.32}$$

With:

$$W = \sum_{\hat{\imath}=1}^{n} N_{\hat{\imath},p}\ w_{\hat{\imath}} \tag{2A.33}$$

$$W_{,\xi} = \sum_{i=1}^{n} N_{i,\xi}\ w_i \tag{2A.34}$$

$$\left({}^{p}N_i\right)_{,\xi} = \frac{p}{\xi_{i+p} - \xi_i}\left({}^{p-1}N_i\right) - \frac{p}{\xi_{i+p+1} - \xi_{i+1}}\left({}^{p-1}N_{i+1}\right) \tag{2A.35}$$

Derivatives of physical coordinates w.r.t. parameter coordinates

These derivatives are calculated as follows:

$$\boldsymbol{x}_{,\xi} = \sum_{I=1}^{\aleph} R_{I,\xi}\ \boldsymbol{P}_I \tag{2A.36}$$

Where $\boldsymbol{P}_I$ are the coordinates of control points and $R_{I,\xi}$ the derivatives of basis functions.

Appendix 2B: Discretization of the equilibrium equation in Continuum Mechanics

This appendix briefs the fundamentals of the theory of elasticity and the numerical approximation used for the solution of the partial differential equation (PDE) that characterizes the linear elastic problem. In the discretization, the constraints are included using the approach from Apostolatos *et al.* (2014).

2B.1 Constitutive relations

The elastic constitutive tensor $\boldsymbol{D}$ contains the elastic properties of the material and relates the Cauchy stress tensor to the strain tensor as follows.

$$\boldsymbol{\sigma} = \boldsymbol{D}\boldsymbol{\varepsilon} \tag{2B.1}$$

$\boldsymbol{D}$ is a fourth order tensor, but in case of isotropic material and using Voigt notation, it can be expressed in matrix form:

$$\boldsymbol{D} = \begin{bmatrix} \lambda+2\mu & \lambda & \lambda & & & \\ \lambda & \lambda+2\mu & \lambda & & & \\ \lambda & \lambda & \lambda+2\mu & & & \\ & & & \mu & & \\ & & & & \mu & \\ & & & & & \mu \end{bmatrix} \tag{2B.2}$$

where:

$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$ is the Lamé first parameter,

$\mu = \frac{E}{2(1+\nu)}$ is the shear modulus

are the Lamé parameters, $\boldsymbol{\varepsilon}$ is the linear strain tensor obtained as in (2B.3), with $\boldsymbol{u}$ representing the displacement field. In this work, we refer to the operation (2B.3) as the *strain operator* ∂. Therefore, the expression (2B.3) is equivalent to (2B.4).

$$\boldsymbol{\varepsilon} = \varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{2B.3}$$

$$\boldsymbol{\varepsilon} = \partial\boldsymbol{u} \tag{2B.4}$$

The strain operator in three-dimensional case is given by:

$$\partial = \begin{bmatrix} \frac{\partial}{\partial x} & & \\ & \frac{\partial}{\partial y} & \\ & & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & \\ & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & & \frac{\partial}{\partial x} \end{bmatrix} \tag{2B.5}$$

The strain tensor (2B.6) can be expressed as vector (2B.7) using Voigt notation.

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_{\boldsymbol{x}} & \gamma_{\boldsymbol{xy}} & \gamma_{\boldsymbol{xz}} \\ \gamma_{\boldsymbol{yx}} & \varepsilon_{\boldsymbol{y}} & \gamma_{\boldsymbol{yz}} \\ \gamma_{\boldsymbol{zx}} & \gamma_{\boldsymbol{zy}} & \varepsilon_{\boldsymbol{z}} \end{bmatrix} \tag{2B.6}$$

$$\boldsymbol{\varepsilon} = \{\varepsilon_x \quad \varepsilon_y \quad \varepsilon_z \quad \gamma_{xy} \quad \gamma_{yx} \quad \gamma_{xz}\}^T \tag{2B.7}$$

$\boldsymbol{\sigma}$ is the Cauchy stress tensor[59] which is expressed as (2B.8) and its vector form is (2B.9) according to Voigt notation.

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_z \end{bmatrix} \tag{2B.8}$$

$$\boldsymbol{\sigma} = \{\sigma_x \quad \sigma_y \quad \sigma_z \quad \tau_{xy} \quad \tau_{yx} \quad \tau_{xz}\}^T \tag{2B.9}$$

2B.2 The strong formulation of the equilibrium

Let Ω be a domain whose boundary is denoted by Γ and its normal unit vector at each location by $\boldsymbol{n}$ (see **Fig. 2B.1**). The domain may be subjected to body forces $\boldsymbol{b}$. The boundary conditions are applied on portions of the boundary Γ_D and Γ_N. The former is where prescribed displacements $\overline{\boldsymbol{u}}$ are imposed - Dirichlet boundary conditions, while the latter is where the surface traction $\bar{\boldsymbol{t}}$ is applied - Neumann boundary conditions. Both portions must hold the next two conditions:

$$\Gamma_{\setminus \Gamma_D \cup \Gamma_N} \cup \Gamma_D \cup \Gamma_N = \Gamma \tag{2B.10}$$

$$\Gamma_D \cap \Gamma_N = \emptyset \tag{2B.11}$$

[59] There are four stress measures: Cauchy, first Piola, second Piola and Kirchhoff stresses according to the reference taken, deformed or undeformed domain. As this work assumes small displacements, both references are indistinguishable, and we use namely the Cauchy stress, that is referred to the deformed configuration.

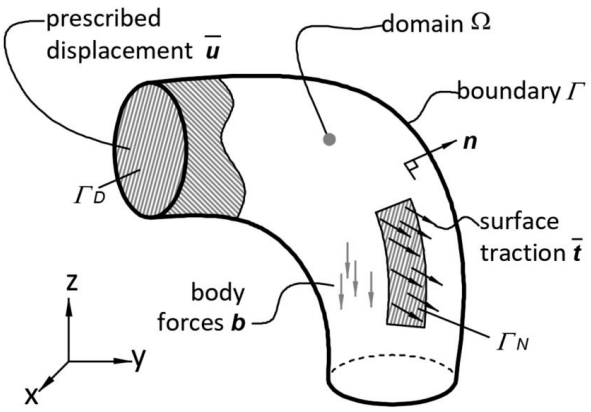


Fig. 2B.1 Domain and boundaries with applied conditions.

The Initial Boundary Value Problem (IBVP) is defined by the equilibrium equations (2B.12a) and its associated boundary conditions (2B.12b) and (2B.12c).

$$\nabla \cdot \boldsymbol{\sigma} + \boldsymbol{b} = \mathbf{0} \;\; in\, \Omega \tag{2B.12a}$$

$$\boldsymbol{u} = \bar{\boldsymbol{u}} \quad on\, \Gamma_D \tag{2B.12b}$$

$$\boldsymbol{\sigma}\, \boldsymbol{n} = \bar{\boldsymbol{t}} \quad on\, \Gamma_N \tag{2B.12c}$$

The problem to solve is defined as follows: find $\boldsymbol{u}: \Omega \rightarrow \mathbb{R}^d$ that satisfies the equations (2B.12), where d is the number of spatial dimensions. In this work d is equal to 3 unless noted otherwise. This formulation is called the differential or strong form of the equilibrium equations.

2B.3 The weak formulation of the equilibrium

Let us define the space of solutions $\boldsymbol{\mathcal{S}}$ as all the functions that match the Dirichlet boundary conditions (2B.13) and the space of weighting functions $\boldsymbol{\mathcal{V}}$ as all the functions that are zero at the Dirichlet boundary conditions (2B.14).

$$\boldsymbol{\mathcal{S}} = \left\{ \boldsymbol{u} \middle|\, \boldsymbol{u}|_{\Gamma_D} = \bar{\boldsymbol{u}} \right\} \tag{2B.13}$$

$$\boldsymbol{\mathcal{V}} = \left\{ \boldsymbol{w} \middle|\, \boldsymbol{w}|_{\Gamma_D} = \mathbf{0} \right\} \tag{2B.14}$$

To obtain the weak formulation, equation (2B.12a) is multiplied by a weighting function and integrated over the whole domain as shown in (2B.15).

$$\int_{\Omega} \boldsymbol{w}^T \; (\nabla \cdot \boldsymbol{\sigma} + \boldsymbol{b})\, d\Omega = 0 \tag{2B.15}$$

Where $\boldsymbol{w}$ is any weighting function that belongs to the space $\boldsymbol{\mathcal{V}}$. If the stress tensor is expressed in terms of displacements, recall (2B.1), and the integral summands are separated we arrive to (2B.16).

$$\int_{\Omega} \boldsymbol{w}^T \left(\nabla \cdot (\boldsymbol{D}\, \partial \boldsymbol{u})\right) d\Omega = -\int_{\Omega} \boldsymbol{w}^T\, \boldsymbol{b}\; d\Omega \tag{2B.16}$$

Applying the Green´s theorem[60] to the left-hand side of (2B.16), we arrive to:

$$\int_{\Omega} \partial \boldsymbol{w}^T (\boldsymbol{D}\, \partial \boldsymbol{u})\, d\Omega = \int_{\Omega} \boldsymbol{w}^T\, \boldsymbol{b}\; d\Omega + \int_{\Gamma_N} \boldsymbol{w}^T\, \bar{\boldsymbol{t}}\, d\Gamma_N \tag{2B.17}$$

Equation (2B.17) may be expressed in general form with differential operators as in (2B.18), being a and L the left and right-hand side differential operators respectively.

$$a(\boldsymbol{w}, \boldsymbol{u}) = L(\boldsymbol{w}) \tag{2B.18}$$

Equation (2B.18) is the weak formulation of the equilibrium problem, which can be defined as: find $\boldsymbol{u}$: $\Omega \rightarrow \mathbb{R}^d$ such that $\boldsymbol{u} \in \boldsymbol{\mathcal{S}}$ satisfies equation (2B.18) given any weighting function $\boldsymbol{w} \in \boldsymbol{\mathcal{V}}$.

The weak formulation is also called integral or variational formulation. The term *weak* refers to the requirements of the solution that in the strong form requires C^1 continuity meanwhile in the weak form requires only C^0 continuity. Weak and strong forms are equivalent, i.e. the solution for the weak form is valid for the strong form and vice-versa.

If we see $\boldsymbol{w}$ as any virtual displacement that satisfies the Dirichlet boundary conditions (recall that the space $\boldsymbol{\mathcal{V}}$ is zero at Γ_D (2B.14)), then equation (2B.18) is equivalent to the Principle of Virtual Work (PVW). The PVW states that if one domain Ω is in equilibrium the internal work equates the external work for any virtual displacements compatible with Dirichlet constraints. The external work is produced by virtual displacements exerted by external forces: body forces $\boldsymbol{b}$ and/or tractions $\bar{\boldsymbol{t}}$ on boundaries, i.e. the right-hand side of (2B.18). The internal work is given by the left-hand side of (2B.18). Let us remark that PVW must hold for any arbitrary virtual displacement, as long as it is compatible with Dirichlet conditions. **Fig. 2B.2** shows two possible configurations with virtual displacements (dashed lines), where the Dirichlet boundary conditions are kept.

[60] See Appendix 2A.

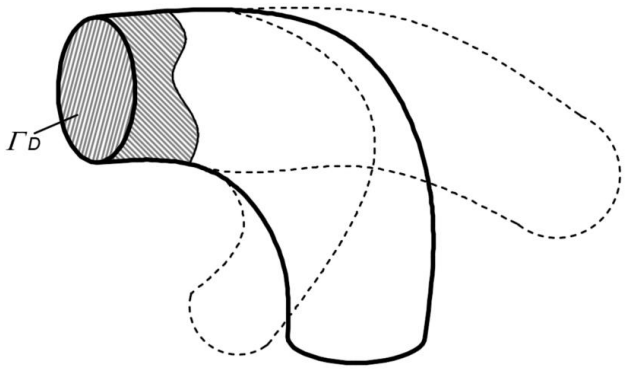


Fig. 2B.2 Allowed virtual displacements.

So far, no approximations have been applied to the problem, i.e. the solution $\boldsymbol{u}$ for the weak form (2B.18) would be exact. However, finding that solution is not possible in general being necessary to find an approximation. The way to achieve such approximations is presented in the next two sections.

2B.4 Weighted Residuals and the Galerkin´s method

The solution $\boldsymbol{u}$ and the weighting functions $\boldsymbol{w}$ for the weak form have infinite dimensions, i.e. $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{V}}$ are infinite-dimensional function spaces. That leads to the impossibility of finding analytical solution and therefore an approximation is required. To achieve such approximation, we restrict $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{V}}$ to a finite dimension.

Fig. 2B.3 depicts a scheme for the numerical approximations available for the elastic linear PDE. Two main approaches exist: approximating the equation in its original strong form or approximating the weak form. In this work Galerkin's method, which approximates the weak form, is used.

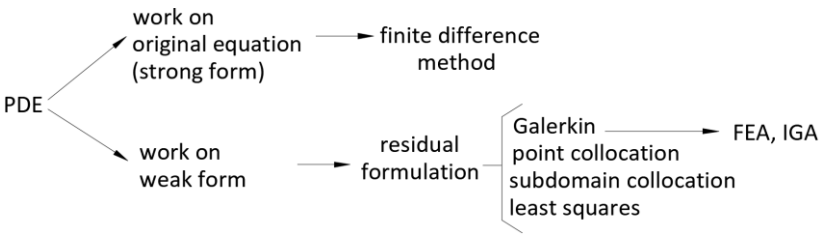


Fig. 2B.3 General view of methods for solving PDE.

Let us define the space of trial solutions $\boldsymbol{\mathcal{S}}^h \subset \boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{V}}^h \subset \boldsymbol{\mathcal{V}}$ as in (2B.19) and (2B.20) respectively.

$$\mathcal{S}^h = \left\{ \boldsymbol{u}^h \in H^1(\Omega),\ \boldsymbol{u}^h\big|_{\Gamma_D} = \overline{\boldsymbol{u}} \right\} \tag{2B.19}$$

$$\mathcal{V}^h = \left\{ \boldsymbol{w}^h \in H^1(\Omega),\ \boldsymbol{w}^h\big|_{\Gamma_D} = \boldsymbol{0} \right\} \tag{2B.20}$$

The Sobolev space $H^1(\Omega)$ forces the functions to have their derivatives square-integrable[61]. If the approximations are introduced in the weak form (2B.18), both sides of the equation are not equal but differ by a residual value called r, as shown in (2B.21).

$$\int_\Omega \partial \boldsymbol{w}^{h^T} \left(\boldsymbol{D}\, \partial \boldsymbol{u}^h \right) d\Omega - \int_\Omega \boldsymbol{w}^{h^T}\, \boldsymbol{b}\, d\Omega - \int_{\Gamma_N} \boldsymbol{w}^{h^T}\, \bar{\boldsymbol{t}}\, d\Gamma_N = r \tag{2B.21}$$

According the manner the residual r is minimized there are different methods. In the Galerkin´s method, the approximation (via discretization) for the weighting functions and the solution is the same.

2B.5 Spatial discretization and approximated solution

The strategy to find the finite-dimensional solution $\boldsymbol{u}^h$ consists of dividing the continua Ω into subdomains or elements Ω^e. This division is called spatial discretization. **Fig. 2B.4** provides one example with eight elements. Control points are represented by small circles. At the right-hand side of **Fig. 2B.4** element 1 is extracted with its influential control points.

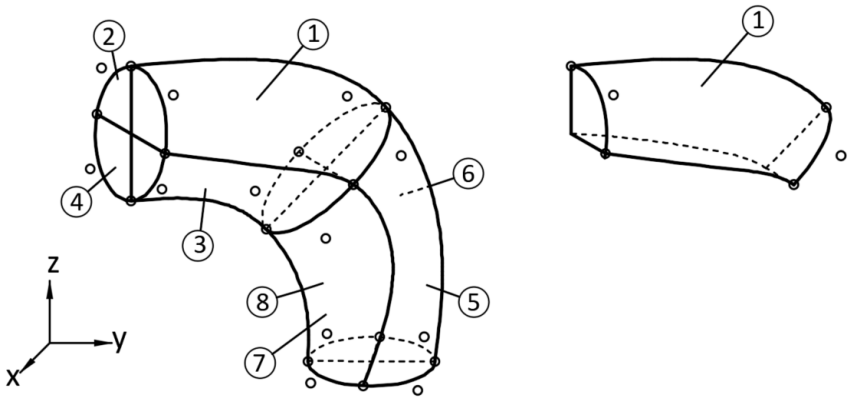


Fig. 2B.4 Spatial discretization of the domain.

The displacement field inside the *eth* element ($\boldsymbol{u}^e$) is approximated by equation (2B.22a) as well as the weighting functions (2B.22b). For further detail on spatial discretization refer to Appendix 2B.

[61] See Appendix 2A.

$$\boldsymbol{u}^{he} = \boldsymbol{R}^{e}\,\widehat{\boldsymbol{u}}^{e} \tag{2B.22a}$$

$$\boldsymbol{w}^{\boldsymbol{he}} = \boldsymbol{R}^{\boldsymbol{e}}\,\widehat{\boldsymbol{w}}^{\boldsymbol{e}} \tag{2B.22b}$$

where:

$\boldsymbol{R}^{e}$ is the matrix of basis functions with d rows and $n^{e}d$ columns.

$\widehat{\boldsymbol{u}}^{e}$ is the vector of displacements of the n^{e} control points with $n^{e}d$ rows.

The strain needs also to be approximated since it appears in the weak form (2B.18). The procedure consists of differentiating the basis functions as shown in (2B.23). The resultant $\boldsymbol{B}^{e}$ matrix is usually called the strain-displacement matrix for the element.

$$\boldsymbol{\varepsilon}^{e} \cong \boldsymbol{\varepsilon}^{he} = \partial\,\boldsymbol{R}^{e}\widehat{\boldsymbol{u}}^{\boldsymbol{e}} = \boldsymbol{B}^{e}\widehat{\boldsymbol{u}}^{\boldsymbol{e}} \tag{2B.23}$$

The strain-displacement matrix $\boldsymbol{B}^{e}$ is assembled as (2B.24).

$$\boldsymbol{B}^{e} = [\boldsymbol{B}_{1}^{e} \quad \ldots \quad \boldsymbol{B}_{n^{e}}^{e}] \tag{2B.24}$$

Where $\boldsymbol{B}_{i}^{e}$ is the submatrix for the i*th* control point which has influence on the eth element. For the three-dimensional case, $\boldsymbol{B}_{i}^{e}$ has the following form (2B.25):

$$\boldsymbol{B}_{i}^{\boldsymbol{e}} = \begin{bmatrix} R_{i,x} & & \\ & R_{i,y} & \\ & & R_{i,z} \\ R_{i,y} & R_{i,x} & \\ & R_{i,z} & R_{i,y} \\ R_{i,z} & & R_{i,x} \end{bmatrix} \tag{2B.25}$$

If the finite approximations $\boldsymbol{u}^{he}$ and $\boldsymbol{w}^{he}$ for each element are inserted on the weak form we get:

$$\int_{\Omega} \widehat{\boldsymbol{w}}^{\boldsymbol{e}T}\boldsymbol{B}^{\boldsymbol{e}T}\boldsymbol{D}\,\boldsymbol{B}\,\widehat{\boldsymbol{u}}^{\boldsymbol{e}}\,d\Omega - \int_{\Omega} \widehat{\boldsymbol{w}}^{\boldsymbol{e}T}\boldsymbol{R}^{\boldsymbol{e}T}\,\boldsymbol{b}\;d\Omega - \int_{\Gamma_N} \widehat{\boldsymbol{w}}^{\boldsymbol{e}T}\boldsymbol{R}^{\boldsymbol{e}T}\,\bar{\boldsymbol{t}}\,d\Gamma_N = 0 \tag{2B.26}$$

Since the weak form holds for any weighting function, the term $\widehat{\boldsymbol{w}}^{\boldsymbol{e}T}$vanishes and the following result is obtained:

$$\int_{\Omega} \boldsymbol{B}^{eT} \boldsymbol{D} \boldsymbol{B}^{e}\, \widehat{\boldsymbol{u}}^{\boldsymbol{e}}\, d\Omega = \int_{\Omega} \boldsymbol{R}^{eT}\, \boldsymbol{b}\, d\Omega - \int_{\Gamma_N} \boldsymbol{R}^{eT}\, \bar{\boldsymbol{t}}\, d\Gamma_N \tag{2B.27}$$

Or:

$$\boldsymbol{K}^{e}\, \widehat{\boldsymbol{u}}^{\boldsymbol{e}} = \hat{\boldsymbol{f}}^{e} \tag{2B.28}$$

where:

$$\boldsymbol{K}^{e} = \int_{\Omega} \boldsymbol{B}^{eT} \boldsymbol{D} \boldsymbol{B}^{e}\, \mathrm{d}\Omega \tag{2B.29}$$

$$\hat{\boldsymbol{f}}^{e} = \int_{\Omega} \boldsymbol{R}^{eT}\, \boldsymbol{b}\, d\Omega + \int_{\Gamma_N} \boldsymbol{R}^{eT}\, \bar{\boldsymbol{t}}\, d\Gamma_N \tag{2B.30}$$

Are the stiffness matrix and the force vector respectively. For each element, the stiffness matrix $\boldsymbol{K}^{e}$ depends on the material and geometry, and the force vector $\hat{\boldsymbol{f}}^{e}$ depends on boundary traction, body forces and geometry. Since all this information is known, $\boldsymbol{K}^{e}$ and $\hat{\boldsymbol{f}}^{e}$ may be computed for each element and then assembled into a global system of equations:

$$\boldsymbol{K}\, \widehat{\boldsymbol{u}} \;=\; \hat{\boldsymbol{f}} \tag{2B.31}$$

Expression (2B.31) is a system of linear equations whose solution is the vector of control points displacements $\widehat{\boldsymbol{u}}$. With these displacements at hand, the displacement field within each element is obtained as in (2B.22a).

2B.6 Discretization of other fields

In this thesis the Lagrange multipliers $\boldsymbol{\lambda}$, and the dots products $\langle \boldsymbol{\lambda}, \boldsymbol{w} \rangle$ and $\langle \boldsymbol{w}, \boldsymbol{u} \rangle$ are also to be discretized (see sections 2B.1.8 and 2B.1.9). The discretization for $\boldsymbol{\lambda}$ is going to be the same as for the $\boldsymbol{u}$ field:

$$\boldsymbol{\lambda}^{he} = \boldsymbol{R}^{e}\, \hat{\boldsymbol{\lambda}}^{e} \tag{2B.32}$$

The discretization for the dot products is shown in equations (2B.33) and (2B.34). See Appendix 2B for further explanations.

$$\langle \boldsymbol{\lambda}, \boldsymbol{w} \rangle \approx \int_{\Gamma} \widehat{\boldsymbol{w}}^T \boldsymbol{R}^T \boldsymbol{R} \, \widehat{\boldsymbol{\lambda}} \; d\Gamma \tag{2B.33}$$

$$\langle \boldsymbol{w}, \boldsymbol{u} \rangle \approx \int_{\Gamma_C} \widehat{\boldsymbol{w}}^T \boldsymbol{R}^T \boldsymbol{R} \, \widehat{\boldsymbol{u}} \; d\Gamma_C \tag{2B.34}$$

2B.7 Coupling constraints

To allow coupling of different domains we use the domain decomposition technique with Lagrange multipliers as proposed by Apostolatos *et al.* (2014), which enforces coupling in a weak sense. In section 2B.1.9, the same approach is presented for Dirichlet boundary conditions for any surface, regardless of the domains parametrization.

We need to re-define the strong form of the equilibrium equation to account for two subdomains. Although the explanation is for two subdomains, it is valid for *n* subdomains.

Let Ω be the domain composed of two subdomains Ω^A and Ω^B such that (2B.35) and (2B.36) apply.

$$\bar{\Omega} = \bigcup_{k=1}^{2} \bar{\Omega}^k \tag{2B.35}$$

$$\bigcap_{k=1}^{2} \Omega^k = 0 \tag{2B.36}$$

The closed domain $\bar{\Omega}$ is the union of Ω and its boundary Γ. The coupling boundary is defined as:

$$\Gamma_C = \bigcap_{k=1}^{2} \bar{\Omega}^k \tag{2B.37}$$

The reformulation of the equilibrium equation is as follows (see **Fig. 2B.5**).

$$\nabla \cdot \boldsymbol{\sigma} + \boldsymbol{b} = \boldsymbol{0} \;\; in \; \Omega \tag{2B.38a}$$

$$\boldsymbol{u} = \overline{\boldsymbol{u}} \quad on \; \Gamma_D \tag{2B.38b}$$

$$\boldsymbol{\sigma}\,\boldsymbol{n} = \bar{\boldsymbol{t}} \quad on\ \Gamma_N \tag{2B.38c}$$

$$\boldsymbol{u}^A - \boldsymbol{u}^B = \boldsymbol{0} \quad on\ \Gamma_C \tag{2B.38d}$$

$$\boldsymbol{q}^A + \boldsymbol{q}^B = \boldsymbol{0} \quad on\ \Gamma_C \tag{2B.38e}$$

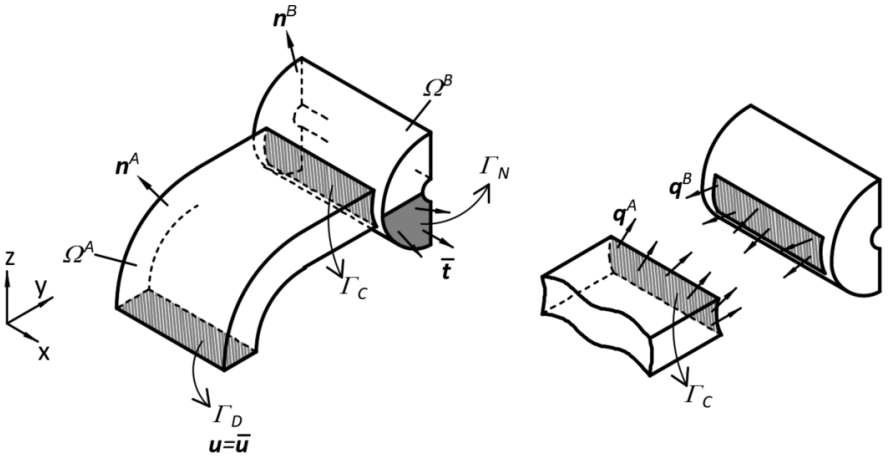


Fig. 2B.5 Decomposition of domain into subdomains Ω^A and Ω^B.

The subdomain A is called master while subdomain B slave. The normal to the surface Γ_C points outwards the master subdomain. The Dirichlet constraint (2B.38b) is addressed in section 2B.1.9.

The weak formulation (recall equation (2B.18)) for both domains is derived in equation (2B.39), which describes the equilibrium of both subdomains. However, the coupling constraint is still not considered.

$$a(\boldsymbol{w}^A, \boldsymbol{u}^A) + a(\boldsymbol{w}^B, \boldsymbol{u}^B) = L(\boldsymbol{w}^A) + L(\boldsymbol{w}^B) \tag{2B.39}$$

To include the coupling, conditions (2B.38d) and (2B.38e) are considered. The latter is introduced in (2B.39) as the internal work performed by tractions $\boldsymbol{q}^k$ at the boundary Γ_C. We call those tractions $\boldsymbol{\lambda}_C$ and the work they produce within each subdomain is $\int_{\Gamma_C} \boldsymbol{\lambda}_C{}^T\, \boldsymbol{w}^k\, d\Gamma_C$. Therefore, the expression (2B.39) is extended to:

$$a(\boldsymbol{w}^A, \boldsymbol{u}^A) + a(\boldsymbol{w}^B, \boldsymbol{u}^B) + \int_{\Gamma_C} \boldsymbol{\lambda}_C{}^T\, (\boldsymbol{w}^A + \boldsymbol{w}^B)\, d\Gamma_C = L(\boldsymbol{w}^A) + L(\boldsymbol{w}^B) \tag{2B.40}$$

The notation $\boldsymbol{\lambda}_C$ is used because these tractions have the role of the Lagrange multipliers in the minimization problem of the potential energy of the system.

In addition, we include the condition (2B.38d) in its weak form:

$$\int_{\Gamma_C} \boldsymbol{w}^T (\boldsymbol{u}^A - \boldsymbol{u}^B)\, d\Gamma_C = 0 \tag{2B.41}$$

Separating both subdomains in (2B.40) and considering (2B.41) we arrive to the system of equations (2B.42), that is the weak form of (2B.38).

$$a^A(\boldsymbol{w}, \boldsymbol{u}) + \int_{\Gamma_C} \boldsymbol{\lambda}_C{}^T \boldsymbol{w}^A\, d\Gamma_C = L^A(\boldsymbol{w}) \tag{2B.42a}$$

$$a^B(\boldsymbol{w}, \boldsymbol{u}) + \int_{\Gamma_C} \boldsymbol{\lambda}_C{}^T \boldsymbol{w}^B\, d\Gamma_C = L^B(\boldsymbol{w}) \tag{2B.42b}$$

$$\int_{\Gamma_C} \boldsymbol{w}^T \boldsymbol{u}^A\, d\Gamma_C - \int_{\Gamma_C} \boldsymbol{w}^T \boldsymbol{u}^B\, d\Gamma_C = 0 \tag{2B.42c}$$

Applying discretization to $a(\boldsymbol{w}^k, \boldsymbol{u}^k)$, $L(\boldsymbol{w}^k)$, and to the dot products $\langle \boldsymbol{\lambda}, \boldsymbol{w} \rangle$ and $\langle \boldsymbol{w}, \boldsymbol{u} \rangle$ (recall sections 2B.1.5 and 2B.1.6) we arrive to the discretized form of the weak formulation:

$$\boldsymbol{K}^A\, \hat{\boldsymbol{u}}^A + \boldsymbol{H}_C^A\, \hat{\boldsymbol{\lambda}}_C = \hat{\boldsymbol{f}}^A \tag{2B.43a}$$

$$\boldsymbol{K}^B\, \hat{\boldsymbol{u}}^B + \boldsymbol{H}_C^B\, \hat{\boldsymbol{\lambda}}_C = \hat{\boldsymbol{f}}^B \tag{2B.43b}$$

$$\boldsymbol{H}_C^{A^T}\, \hat{\boldsymbol{u}}^A + \boldsymbol{H}_C^{B^T}\, \hat{\boldsymbol{u}}^B = \boldsymbol{0} \tag{2B.43c}$$

That may be expressed in matrix form as:

$$\begin{bmatrix} \boldsymbol{K}^A & \boldsymbol{0} & \boldsymbol{H}_C^A \\ \boldsymbol{0} & \boldsymbol{K}^B & \boldsymbol{H}_C^B \\ \boldsymbol{H}_C^{A^T} & \boldsymbol{H}_C^{B^T} & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \hat{\boldsymbol{u}}^A \\ \hat{\boldsymbol{u}}^B \\ \hat{\boldsymbol{\lambda}}_C \end{Bmatrix} = \begin{Bmatrix} \hat{\boldsymbol{f}}^A \\ \hat{\boldsymbol{f}}^B \\ \boldsymbol{0} \end{Bmatrix} \tag{2B.44}$$

Where the $\boldsymbol{H}_C$ matrices are computed as (2B.45), being k equal to A or B.

$$\boldsymbol{H}_C^k = \int_{\Gamma_C} \boldsymbol{R}^{k^T} \boldsymbol{R}^A\, d\Gamma_C \tag{2B.45}$$

2B.8 Dirichlet constraints

The Lagrange multipliers approach will also be followed to impose Dirichlet boundary conditions. We consider the strong formulation (2B.46) for a domain (**Fig. 2B.6**) and its weak counterpart (2B.47) will be repeated here for convenience.

$$\nabla \cdot \boldsymbol{\sigma} + \boldsymbol{b} = \boldsymbol{0} \;\; in\, \Omega \tag{2B.46a}$$

$$\boldsymbol{u} = \overline{\boldsymbol{u}} \quad on\, \Gamma_D \tag{2B.46b}$$

$$\boldsymbol{\sigma}\,\boldsymbol{n} = \bar{\boldsymbol{t}} \quad on\ \Gamma_N \tag{2B.46c}$$

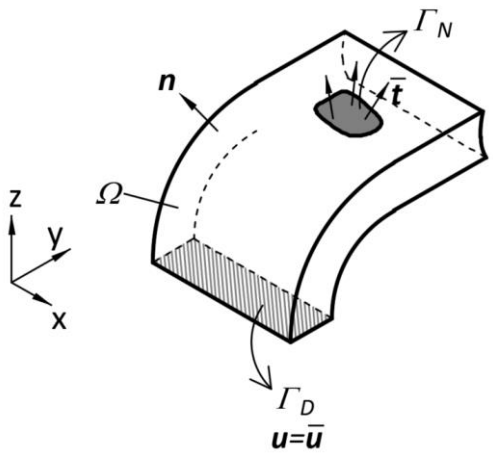


Fig. 2B.6 Dirichlet constraint on boundary Γ_C.

$$a(\boldsymbol{w},\boldsymbol{u}) = L(\boldsymbol{w}) \tag{2B.47}$$

Equation (2B.47) describes the equilibrium of the domain without Dirichlet constraint. The internal virtual work produced by the traction forces at the boundary Γ_D is $\int_{\Gamma_D} \boldsymbol{\lambda}_D{}^T\, \boldsymbol{w}^k\, d\Gamma_D$ and may be introduce at the left-hand side of (2B.47) as follows:

$$a(\boldsymbol{w},\boldsymbol{u}) + \int_{\Gamma_D} \boldsymbol{\lambda}_D{}^T\, \boldsymbol{w}^k\, d\Gamma_D = L(\boldsymbol{w}) \tag{2B.48}$$

In addition, the condition (2B.46b) is also included in its weak form:

$$\int_{\Gamma_D} \boldsymbol{w}^T(\boldsymbol{u} - \bar{\boldsymbol{u}})\, d\Gamma_D = 0 \tag{2B.49}$$

Aggregating both equations, (2B.48) and (2B.49), we arrive to the system of equations (2B.50), which is the weak form of (2B.46).

$$a(\boldsymbol{w},\boldsymbol{u}) + \int_{\Gamma_D} \boldsymbol{\lambda}_D{}^T\, \boldsymbol{w}\, d\Gamma_D = L(\boldsymbol{w}) \tag{2B.50a}$$

$$\int_{\Gamma_D} \boldsymbol{w}^T \boldsymbol{u}\, d\Gamma_D - \int_{\Gamma_D} \boldsymbol{w}^T \bar{\boldsymbol{u}}\, d\Gamma_D = 0 \tag{2B.50b}$$

Applying discretization to $a(\boldsymbol{w}^k, \boldsymbol{u}^k)$, $L(\boldsymbol{w}^k)$, and to the dot products $\langle \boldsymbol{\lambda}, \boldsymbol{w} \rangle$ and $\langle \boldsymbol{w}, \boldsymbol{u} \rangle$ (recall sections 2B.1.5 and 2B.1.6) we arrive to the following discretized form of the weak formulation:

$$\boldsymbol{K}\,\hat{\boldsymbol{u}} + \boldsymbol{H}_D\,\hat{\boldsymbol{\lambda}}_D = \hat{\boldsymbol{f}} \tag{2B.51a}$$

$$\boldsymbol{H_D}^T \, \hat{\boldsymbol{u}} = \hat{\overline{\boldsymbol{u}}} \tag{2B.51b}$$

Which can be expressed in matrix form as:

$$\begin{bmatrix} \boldsymbol{K} & \boldsymbol{H_D} \\ \boldsymbol{H_D}^T & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \hat{\boldsymbol{u}} \\ \hat{\boldsymbol{\lambda}}_D \end{Bmatrix} = \begin{Bmatrix} \hat{\boldsymbol{f}} \\ \hat{\overline{\boldsymbol{u}}} \end{Bmatrix} \tag{2B.52}$$

The $\boldsymbol{H_D}$ matrix is computed as:

$$\boldsymbol{H_D} = \int_{\Gamma_D} \boldsymbol{R}^T \boldsymbol{R} \, d\Gamma_D \tag{2B.53}$$

and vector $\hat{\overline{\boldsymbol{u}}}$ is composed of *n* $\overline{\boldsymbol{u}}$ vectors, being *n* the number of involved control points.

2B.9 Discretized weak formulation considering coupling and Dirichlet constraints

The coupling and Dirichlet constraints can be imposed simultaneously within the system by applying the techniques shown in sections 2B.1.8 and 2B.1.9. The resultant equation (2B.54) is called in this work the *aggregated system* of linear equations. **Fig. 2B.7** shows one example with Dirichlet constraints in two coupled subdomains.

$$\begin{bmatrix} \boldsymbol{K}^A & \boldsymbol{0} & \boldsymbol{H}_D^A & \boldsymbol{0} & \boldsymbol{H}_C^A \\ \boldsymbol{0} & \boldsymbol{K}^B & \boldsymbol{0} & \boldsymbol{H}_D^B & \boldsymbol{H}_C^B \\ \boldsymbol{H}_D^{A\,T} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{H}_D^{B\,T} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{H}_C^{A\,T} & \boldsymbol{H}_C^{B\,T} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \hat{\boldsymbol{u}}^A \\ \hat{\boldsymbol{u}}^B \\ \hat{\boldsymbol{\lambda}}_D^A \\ \hat{\boldsymbol{\lambda}}_D^B \\ \hat{\boldsymbol{\lambda}}_C \end{Bmatrix} = \begin{Bmatrix} \hat{\boldsymbol{f}}^A \\ \hat{\boldsymbol{f}}^B \\ \hat{\overline{\boldsymbol{u}}}^A \\ \hat{\overline{\boldsymbol{u}}}^B \\ \boldsymbol{0} \end{Bmatrix} \tag{2B.54}$$

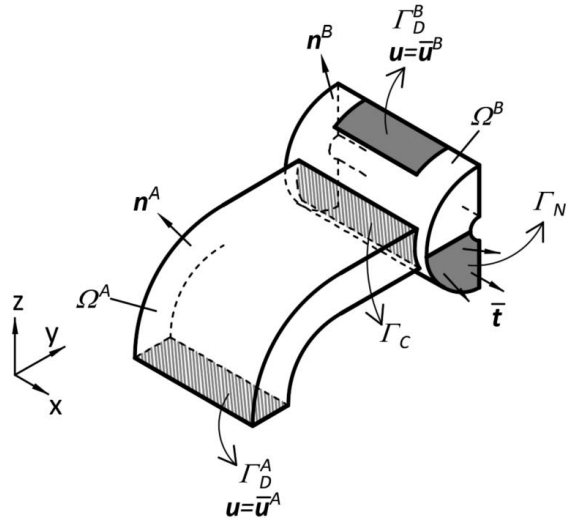


Fig. 2B.7 Decomposition of domain into subdomains A and B with Dirichlet constraints in both subdomains.

2B.10 A simplistic view of the discretization of the equilibrium equation

The equation (2B.12) applies to the domain, which is continuum in the reality. Since, in general, that equation is not solvable the domain is discretized into portions to find and approximation to the solution. The shape of each portion is governed by a set of point, that in Isogeometric Analysis are called control points. To find the approximated solution, the linear system of equations (2B.54) is constructed. This equation can be simplified as follows:

$$\boldsymbol{K}\,\hat{\boldsymbol{u}} = \hat{\boldsymbol{f}} \tag{2B.55}$$

where $\boldsymbol{K}$ represents the resistance of the control points to the displacement (stiffness); and $\hat{\boldsymbol{u}}$ are the displacements of the control points due the forces $\hat{\boldsymbol{f}}$ applied on the control points.

The transformation from the *real* continuum domain (a) to the discretized one (b) is shown in **Fig. 2B.8**. The *real* forces (body forces and surface tractions) are converted into forces at the control points. Similarly, the Dirichlet boundary conditions imposed in the reality are converted to restraints to the control points. The results of the analysis are the displacements at control points (c).

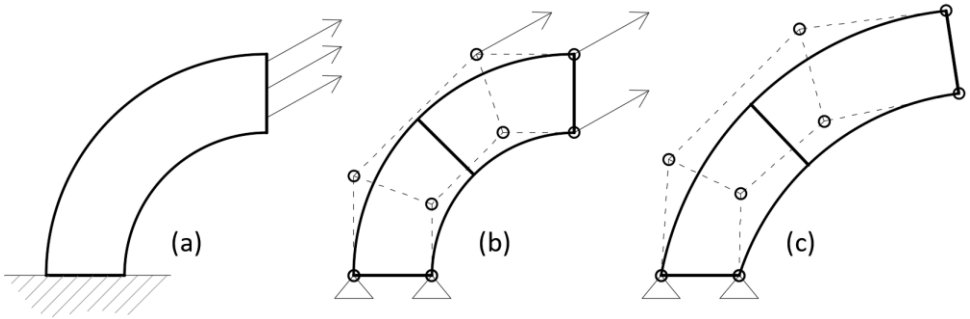


Fig. 2B.8 Discretization of the domain and results of the analysis.

Appendix 2C: Spatial discretization

Let us assume one domain Ω with two continuous vector fields $\boldsymbol{a}$ and $\boldsymbol{w}$ with d number of components, *i.e*. degrees of freedom. The domain may be divided into portions called elements. Let us assume a set of points having influence on each element, called control points. The fields are known at these points, and are called $\widehat{\boldsymbol{a}}_i$ and $\widehat{\boldsymbol{w}}_i$ for the ith control point. Each control point has one basis function[62] (R_i) whose value depends on the location considered within the element.

2C.1 Discretization of one field

The field $\boldsymbol{a}$ (or $\boldsymbol{w}$) may be estimated within each element as linear combination of vectors $\widehat{\boldsymbol{a}}_i$ that have influence on the element. The estimated field at one location $\boldsymbol{\xi}$ within the element is called $\boldsymbol{a}^{eh}$, whose calculation is as follows:

$$\boldsymbol{a}^e \cong \boldsymbol{a}^{he} = \sum_{i=1}^{n^e} R_i \, \widehat{\boldsymbol{a}}_i \tag{2C.1}$$

Where:

$\boldsymbol{a}^{he}$ is the approximation to the field within the *eth* element.

R_i is the basis function value of the i*th* control point at $\boldsymbol{\xi}$ location.

$\widehat{\boldsymbol{a}}_i$ is the field vector for the i*th* control point.

n^e is the number of control points that have influence on the eth element.

Equation (2C.1) can be expressed in matrix form as:

$$\boldsymbol{a}^{he} = \boldsymbol{R}^e \, \widehat{\boldsymbol{a}}^e \tag{2C.2}$$

where:

$\boldsymbol{R}^e$ is the matrix of basis functions with d rows and $n^e x\ d$ columns, as shown in (2C.3).

$\widehat{\boldsymbol{a}}^e$ is the vector of field values at the n^e control points with $n^e x\ d$ rows.

$$\boldsymbol{R}^e = [\boldsymbol{R}_1^e \quad \ldots \quad \boldsymbol{R}_{n^e}^e] \tag{2C.3}$$

In equation (2C.3) $\boldsymbol{R}_i^e$ is the $d\ x\ d$ diagonal submatrix of the ith control point, that is expressed in (2C.4) for $d = 3$.

$$\boldsymbol{R}_i^e = \begin{bmatrix} R_i & & \\ & R_i & \\ & & R_i \end{bmatrix} \tag{2C.4}$$

[62] Also called shape functions.

2C.2 Discretization of fields dot product

The dot product of fields $\boldsymbol{a}$ and $\boldsymbol{w}$ is:

$$\langle \boldsymbol{a}, \boldsymbol{w} \rangle = \int_{\Gamma} \boldsymbol{a}^T \, \boldsymbol{w} \; d\Gamma \tag{2C.5}$$

The discretization of the dot product is obtained as the discretization of each field:

$$\langle \boldsymbol{a}, \boldsymbol{w} \rangle \approx \int_{\Gamma} \widehat{\boldsymbol{a}}^T \, \boldsymbol{R}^T \, \boldsymbol{R} \, \widehat{\boldsymbol{w}} \; d\Gamma \tag{2C.6}$$

In case the field $\boldsymbol{w}$ is needed to be placed first, it can be moved as follows:

$$\langle \boldsymbol{w}, \boldsymbol{a} \rangle \approx \int_{\Gamma} \widehat{\boldsymbol{w}}^T \, \boldsymbol{R}^T \, \boldsymbol{R} \, \hat{\boldsymbol{\lambda}} \; d\Gamma \tag{2C.7}$$

Appendix 2D: Volumetric locking and B-bar method

One solid discretized into elements suffers of the so-called volumetric locking if the material is incompressible. This phenomenon increases the stiffness of the domain leading to non-valid analysis results. The cause of volumetric locking lies the rigidity of the interpolation of the displacement field within the elements. If the control points are not enough, the interpolated field is rigid, i.e. no able to capture displacements variations, and the displacement is under-estimated. There are four well-settled methods to eliminate the volume locking: strain projection (or B-bar method), mixed formulation, reduced integration and mesh refinement. The calculation of B-bar matrix is detailed in this appendix.

The B-bar method removes the volumetric locking by under-integrating the volumetric strain within the elements. As a result, the deformation matrix is modified and re-called B-bar matrix ($\overline{\boldsymbol{B}}$), that must be used in stiffness calculation. Then, the element stiffness matrix is computed as:

$$\boldsymbol{K}^e = \int_\Omega \overline{\boldsymbol{B}}^{e^T} \boldsymbol{D} \overline{\boldsymbol{B}}^e \, \mathrm{d}\Omega \tag{2D.1}$$

The B-bar method removes the volumetric locking by prescribing volumetric strain within the elements. The strain can be split into deviatoric and volumetric components (2D.2).

$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^d + \boldsymbol{\varepsilon}^v \cong \boldsymbol{B}^d \hat{\boldsymbol{u}} + \boldsymbol{B}^v \hat{\boldsymbol{u}} \tag{2D.2}$$

Then, the deformation matrix is split into volumetric and deviatoric parts (2D.3).

$$\boldsymbol{B} = \boldsymbol{B}^v + \boldsymbol{B}^d \tag{2D.3}$$

The volumetric deformation matrix is obtained as (2D.4) for solid domains[63] in the standard form (i.e. without B-bar method). The deviatoric part is obtained as difference (2D.5).

[63] Volumetric component of strains is obtained as the average of trace of strain tensor ($\varepsilon^v = \frac{1}{3}\varepsilon_{ii}$), and the dilatational strain vector has the same three components equal to this value ($\varepsilon_i^v = \varepsilon^v$).

$$\boldsymbol{B^v} = \frac{1}{3}\begin{bmatrix} \frac{\partial R_i}{\partial x} & \frac{\partial R_i}{\partial y} & \frac{\partial R_i}{\partial z} \\ \frac{\partial R_i}{\partial x} & \frac{\partial R_i}{\partial y} & \frac{\partial R_i}{\partial z} \\ \frac{\partial R_i}{\partial x} & \frac{\partial R_i}{\partial y} & \frac{\partial R_i}{\partial z} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2D.4}$$

$$\boldsymbol{B^d} = \boldsymbol{B} - \boldsymbol{B^v} \tag{2D.5}$$

The B-bar method computes the volumetric matrix differently: $\overline{\boldsymbol{B}}^{\boldsymbol{v}}$ is obtained by averaging the volumetric strain over the whole element. Therefore, each of its basis functions derivatives are obtained as (2D.6), which may be approximated by Gauss quadrature. Then the $\overline{\boldsymbol{B}}^{v}$ matrix is assembled as (2D.7).

$$\overline{\frac{\partial R_\iota}{\partial x}} = \frac{1}{\Omega^e}\int_{\Omega^e} \frac{\partial R_i}{\partial x}\, d\Omega^e \tag{2D.6}$$

$$\overline{\boldsymbol{B}}^{v} = \frac{1}{3}\begin{bmatrix} \overline{\frac{\partial R_\iota}{\partial x}} & \overline{\frac{\partial R_\iota}{\partial y}} & \overline{\frac{\partial R_\iota}{\partial z}} \\ \overline{\frac{\partial R_\iota}{\partial x}} & \overline{\frac{\partial R_\iota}{\partial y}} & \overline{\frac{\partial R_\iota}{\partial z}} \\ \overline{\frac{\partial R_\iota}{\partial x}} & \overline{\frac{\partial R_\iota}{\partial y}} & \overline{\frac{\partial R_\iota}{\partial z}} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2D.7}$$

With $\overline{\boldsymbol{B}}^{v}$ and $\boldsymbol{B}^{d}$ at hand, the latter computed as (2D.5), the modified deformation matrix $\overline{\boldsymbol{B}}$ matrix is obtained by the summation as shown in (2D.8).

$$\overline{\boldsymbol{B}} = \boldsymbol{B^d} - \overline{\boldsymbol{B}}^{\boldsymbol{v}} \tag{2D.8}$$

Appendix 2E: Parent elements

The parent space coordinates are represented differently for rectangular and triangular elements. For rectangular elements (line, square and hexahedron) the parent coordinates are given by the $\tilde{\boldsymbol{\xi}}$ vector, which is $\tilde{\xi}$, $\{\tilde{\xi}, \tilde{\eta}\}^T$ and $\{\tilde{\xi}, \tilde{\eta}, \tilde{\chi}\}^T$ for one, two and three-dimensional parent spaces respectively. For triangular elements (triangle and tetrahedron) the parent coordinates are given by the $\boldsymbol{r}$ vector, which is $\{r, s\}^T$ and $\{r, s, t\}^T$ for two and three dimensions respectively.

One parent space has attached a set of nodes and each node has attached a basis function ϕ. The projected space from the parent space is called $\mathbb{C}$ and lies in $\mathbb{R}^m$, with m typically from one to three. $\mathbb{C}$-coordinates are represented by the vector $\boldsymbol{c}$.

Nodes have coordinates in the parent and the projected space. The parent and projected coordinates at the i-location are called $\tilde{\boldsymbol{\xi}}_i$ and $\boldsymbol{c}_i$. The latter is mapped from $\tilde{\boldsymbol{\xi}}_i$ using equation (2E.1), where the dependency on the parent coordinates is omitted for clarity, e.g. we write ϕ instead $\phi(\tilde{\boldsymbol{\xi}})$.

$$\boldsymbol{c}_i = \sum_{j=1}^{n} \phi_j \, \boldsymbol{c}_j \tag{2E.1}$$

where:

n is the number of nodes of the parent element.

ϕ_j is the value of the *j*th basis function at $\tilde{\boldsymbol{\xi}}_i$.

$\boldsymbol{c}_j$ is the $\mathbb{C}$ –coordinate of the *j*th node.

Jacobians are computed as equations shown in **Table 2E.1**.

Table 2E.1 Calculation of Jacobians.

Dimensions of the parent space	Dimension of projected space	Jacobian
1	1, 2 or 3	$\mathcal{J} = \left\| \boldsymbol{c}_{,\tilde{\xi}} \right\|$
2	2 or 3	$\mathcal{J} = \left\| \boldsymbol{c}_{,\tilde{\xi}} \times \boldsymbol{c}_{,\tilde{\eta}} \right\|$
3	3	$\mathcal{J} = det\left[\boldsymbol{c}_{,\tilde{\xi}} \, \boldsymbol{c}_{,\tilde{\eta}} \, \boldsymbol{c}_{,\tilde{\chi}}\right]$

The derivatives $\boldsymbol{c}_{,\tilde{\xi}}$ are computed using the linear combination (2E.1), *i.e.* deriving the basis functions and summing as follows:

$$\frac{\partial \boldsymbol{c}_i}{\partial \tilde{\xi}} = \sum_{j=1}^{n} \frac{\partial \phi_j}{\partial \tilde{\xi}} \, \boldsymbol{c}_j \tag{2E.2}$$

In this appendix basis function of the next parent elements are presented.

- Linear line.
- Linear square.
- Linear hexahedron.
- Linear and quadratic triangle.
- Linear and quadratic tetrahedron.

In the figures below, the filled dots represent nodes and the hollow dot a generic i-location.

2E.1 Linear line

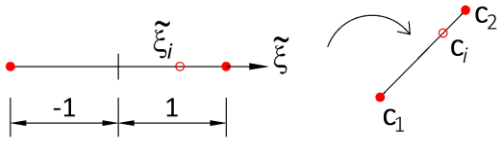


Fig. 2E.1 Parent space and mapping onto the $\mathbb{C}$-space of i-location.

Basis functions at i location are:

$$\phi_1 = \frac{1}{2}(1 - \tilde{\xi}_i) \tag{2E.3}$$

$$\phi_2 = \frac{1}{2}(1 + \tilde{\xi}_i) \tag{2E.4}$$

2E.2 Linear square

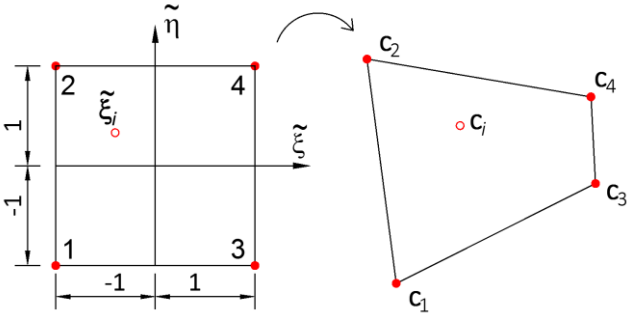


Fig. 2E.2 Parent space and mapping onto the $\mathbb{C}$-space of i-location.

Basis functions at i-location are:

$$\phi_1 = \frac{1}{4}(1 - \tilde{\xi}_i)(1 - \tilde{\eta}_i) \tag{2E.5}$$

$$\phi_2 = \frac{1}{4}(1 - \tilde{\xi}_i)(1 + \tilde{\eta}_i) \tag{2E.6}$$

$$\phi_3 = \frac{1}{4}(1 + \tilde{\xi}_i)(1 - \tilde{\eta}_i) \tag{2E.7}$$

$$\phi_4 = \frac{1}{4}(1+\tilde{\xi}_i)(1+\tilde{\eta}_i) \tag{2E.8}$$

2E.3 Linear hexahedron

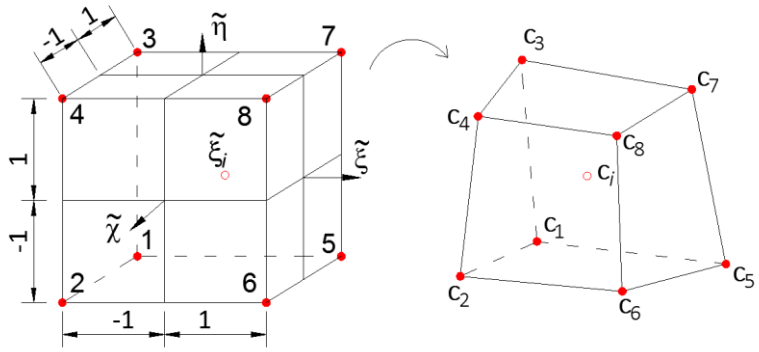


Fig. 2E.3 Parent space and mapping onto the $\mathbb{C}$-space of i-location.

Basis functions at i-location are:

$$\phi_1 = \frac{1}{8}(1-\tilde{\xi}_i)(1-\tilde{\eta}_i)(1-\tilde{\chi}_i) \tag{2E.9}$$

$$\phi_2 = \frac{1}{8}(1-\tilde{\xi}_i)(1-\tilde{\eta}_i)(1+\tilde{\chi}_i) \tag{2E.10}$$

$$\phi_3 = \frac{1}{8}(1-\tilde{\xi}_i)(1+\tilde{\eta}_i)(1-\tilde{\chi}_i) \tag{2E.11}$$

$$\phi_4 = \frac{1}{8}(1-\tilde{\xi}_i)(1+\tilde{\eta}_i)(1+\tilde{\chi}_i) \tag{2E.12}$$

$$\phi_5 = \frac{1}{8}(1+\tilde{\xi}_i)(1-\tilde{\eta}_i)(1-\tilde{\chi}_i) \tag{2E.13}$$

$$\phi_6 = \frac{1}{8}(1+\tilde{\xi}_i)(1-\tilde{\eta}_i)(1+\tilde{\chi}_i) \tag{2E.14}$$

$$\phi_7 = \frac{1}{8}(1+\tilde{\xi}_i)(1+\tilde{\eta}_i)(1-\tilde{\chi}_i) \tag{2E.15}$$

$$\phi_8 = \frac{1}{8}(1+\tilde{\xi}_i)(1+\tilde{\eta}_i)(1+\tilde{\chi}_i) \tag{2E.16}$$

2E.4 Linear triangle

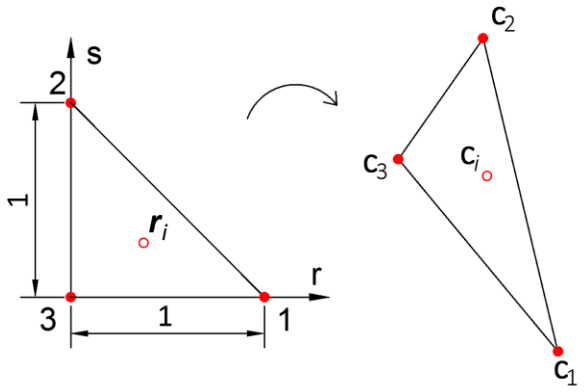


Fig. 2E.4 Parent space and mapping onto the $\mathbb{C}$-space of i-location.

Basis functions at i-location are:

$$\phi_1 = r_i \tag{2E.17}$$

$$\phi_2 = s_i \tag{2E.18}$$

$$\phi_3 = t_i \tag{2E.19}$$

Where $t = 1 - r - s$.

2E.5 Quadratic line

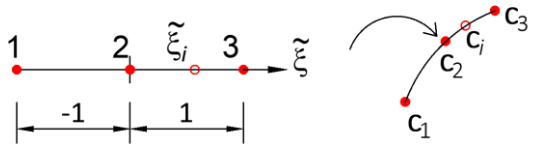


Fig. 2E.5 Parent space and mapping onto the $\mathbb{C}$-space of i-location.

Basis functions at i location are:

$$\phi_1 = \frac{1}{2}\left(\tilde{\xi}_i^{\,2} - \tilde{\xi}_i\right) \tag{2E.3}$$

$$\phi_2 = \left(1 - \tilde{\xi}_i^{\,2}\right) \tag{2E.4}$$

$$\phi_3 = \frac{1}{2}\left(\tilde{\xi}_i^{\,2} + \tilde{\xi}_i\right) \tag{2E.3}$$

2E.6 Quadratic triangle

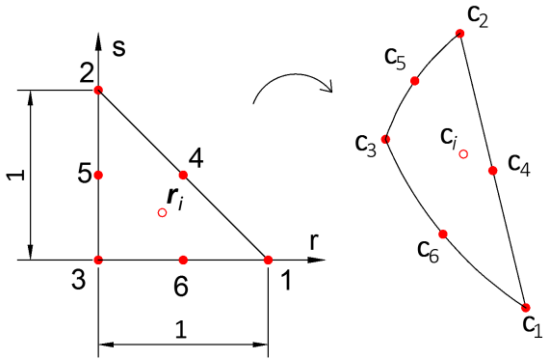


Fig. 2E.6 Parent space and mapping onto the $\mathbb{C}$-space of i-location.

Basis functions at i-location are:

$$\phi_1 = r_i\,(2r_i - 1) \tag{2E.20}$$

$$\phi_2 = s_i\,(2s_i - 1) \tag{2E.21}$$

$$\phi_3 = t_i\,(2t_i - 1) \tag{2E.22}$$

$$\phi_4 = 4r_i s_i \tag{2E.23}$$

$$\phi_5 = 4s_i t_i \tag{2E.24}$$

$$\phi_6 = 4t_i r_i \tag{2E.25}$$

Where $t = 1 - r - s$.

2E.7 Linear tetrahedron

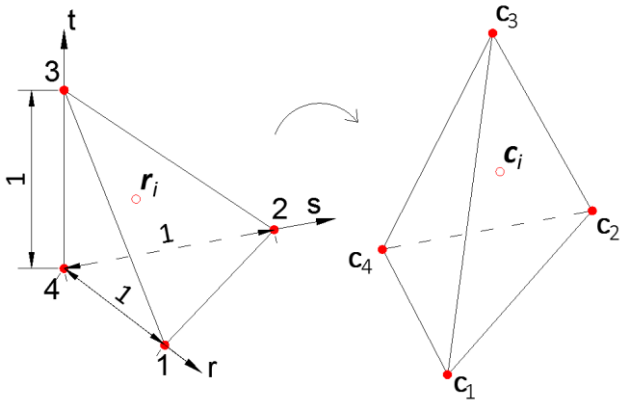


Fig. 2E.7 Parent space and mapping onto the $\mathbb{C}$-space of i-location.

Basis functions at i-location are:

$$\phi_1 = r_i \tag{2E.26}$$

$$\phi_2 = s_i \tag{2E.27}$$

$$\phi_3 = t_i \tag{2E.28}$$

$$\phi_4 = u_i \tag{2E.29}$$

Where $u = 1 - r - s - t$.

2E.8 Quadratic tetrahedron

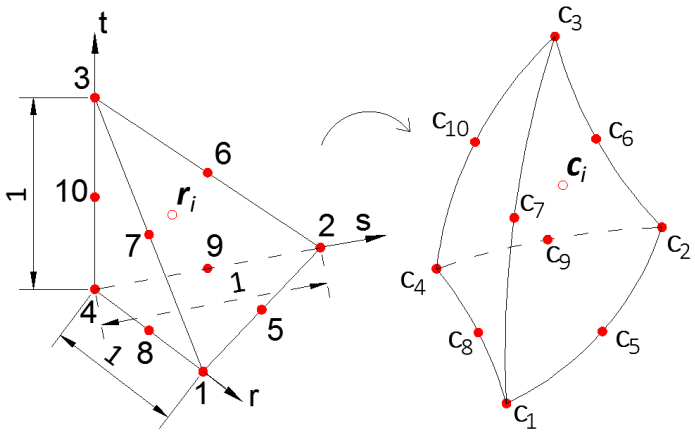


Fig. 2E.8 Parent space and mapping onto the $\mathbb{C}$-space of i-location.

Basis functions at i-location are:

$$\phi_1 = r_i - 2\, r_i\, (s_i + t_i + u_i) \tag{2E.30}$$

$$\phi_2 = s_i - 2\, s_i\, (r_i + t_i + u_i) \tag{2E.31}$$

$$\phi_3 = t_i - 2\, t_i\, (r_i + s_i + u_i) \tag{2E.32}$$

$$\phi_4 = u_i - 2\, u_i\, (r_i + s_i + t_i) \tag{2E.33}$$

$$\phi_5 = 4\, r_i s_i \tag{2E.34}$$

$$\phi_6 = 4\, s_i t_i \tag{2E.35}$$

$$\phi_7 = 4\, t_i r_i \tag{2E.36}$$

$$\phi_8 = 4\,u_i r_i \tag{2E.37}$$

$$\phi_9 = 4\,u_i s_i \tag{2E.38}$$

$$\phi_{10} = 4\,u_i t_i \tag{2E.39}$$

Where $u = 1 - r - s - t$.

Appendix 2F: Jacobians

Given the c −dimensional $\tilde{A}$ space whose coordinates are denoted by $\tilde{a}_1, \dots, \tilde{a}_c$, and the d −dimensional A space whose coordinates are denoted by $a_1, \dots, a_d$. Let us assume a set of d functions $f_1, \dots, f_d$ such that $f: \tilde{A} \to A$. The differentials of $\boldsymbol{f}$ may be written as:

$$\begin{Bmatrix} df_1 \\ \vdots \\ df_d \end{Bmatrix} = \begin{bmatrix} f_{1,\tilde{a}1} & & f_{1,\tilde{a}c} \\ & \ddots & \\ f_{d,\tilde{a}1} & & f_{d,\tilde{a}c} \end{bmatrix} \begin{Bmatrix} d\tilde{a}_1 \\ \vdots \\ d\tilde{a}_d \end{Bmatrix} \tag{2F.1}$$

Where the matrix is known as the Jacobian matrix which is composed of c columns. The ith column is the vector $\boldsymbol{f}_{,\tilde{a}i}$. The Jacobian matrix may be expressed in indicial notation as (2F.2). The norm of such matrix is used for transformation of boundaries in integration. **Table 2F.1** summarizes the most common cases for different values of c and d.

$$\boldsymbol{\mathcal{J}} = \frac{\partial f_i}{\partial \tilde{a}_j} \tag{2F.2}$$

Table 2F.1 Computation of Jacobian norm.

c	d	$\|\boldsymbol{\mathcal{J}}\|$
1	1	$\frac{df}{d\tilde{a}}$
	2 or 3	$\left\|\frac{\partial f_i}{\partial \tilde{a}}\right\| = \|\boldsymbol{f}_{,\tilde{a}1}\|$
2	2 or 3	$\|\boldsymbol{f}_{,\tilde{a}1} \times \boldsymbol{f}_{,\tilde{a}2}\|$
3	3	$det\, \boldsymbol{\mathcal{J}}$

One example is provided here. In the mapping from parameter ($\hat{\Omega}$) to physical space (Ω) in solids both spaces are three-dimensional, i.e. $c = 3$ and $d = 3$. Therefore the Jacobian is:

$$\mathcal{J}_1 = det\, \boldsymbol{\mathcal{J}}_1 = det \begin{bmatrix} x_{,\xi} & x_{,\eta} & x_{,\chi} \\ y_{,\xi} & y_{,\eta} & y_{,\chi} \\ z_{,\xi} & z_{,\eta} & z_{,\chi} \end{bmatrix} \tag{2F.3}$$

Note that the functions are the physical coordinates themselves.

2F.1 Frobenius norm of the Jacobian

Given the Jacobian matrix with n rows and m columns, its Frobenius norm is calculated as:

$$\|\boldsymbol{J}\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} |J_{ij}|^2} \tag{2F.4}$$

Appendix 2G: IGES files text structure

This appendix briefs the text structure of the IGES files, highlighting the most relevant features to this work. **Fig. 2G.1** shows one example. The file has six sections called *flag*, *start*, *global*, *directory entry*, *parameter* and *terminate*. In all sections the row number, or *address*, is indicated by the last number. For example, in **Fig. 2G.1** *directory entry section* is shown from row 1 to row 15, and *parameter data* from 1 to 20. These two sections are used in this research and are further detailed below. For clear identification of the IGES graphic objects, they are written in ***bold-italic***, and its type number indicated in brackets.

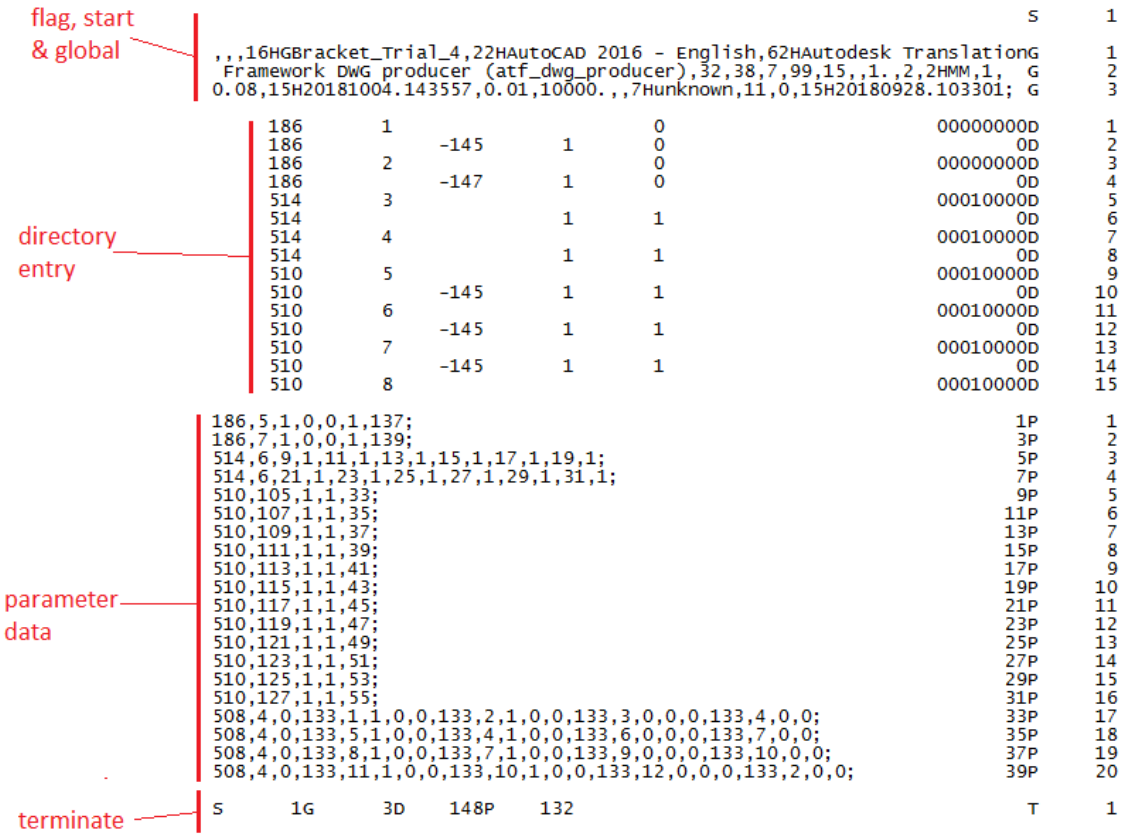


Fig. 2G.1 Aspect of IGES file and different sections within it. Fourth and fifth sections have been cut to fit the image in one page.

Directory entry section contains two lines per graphical object. In the first line, the first number is the type of object (e.g. 514 for ***shell***, 510 for ***face***, etc.) and the second number is the row address in the *parameter data* section. The second line first number is again the type and the second number (negative) is another address that we designate as the *colour address*. For some types c*olour address* code is 0 or not defined, e.g. ***shell***(514) does not have it.

For example, in **Fig. 2G.1** first four lines of directory entry section indicate that there are two ***manifold solid B-reps*** (186) and next four lines indicate two ***shells*** (514). The four objects are located at rows 1, 2, 3 and 4 in *parameter data* section. The *colour addresses* of ***manifold solid B-rep*** are 145 and 147. **Fig. 2G.2** provides one more example: three ***curves*** (126) whose addresses in *parameter data* section are 96, 105 and 107.

Fig. 2G.2 Address storage in *directory entry* and its location in *parameter data* section.

Parameter data section stores the information of objects. First number is the type (e.g. 514 for ***shell***, 510 for ***face***, etc.) and the meaning of next numbers vary with each type. They store geometrical information, including NURBS features, and addresses of dependant objects in *directory entry* section. The numbers are separated by comas. Number of lines per object varies. Below there are two examples for clarification.

Parameter data section. Example 1:

126,1,1,0,0,1,0,0.,0.,100.,100.,1.,1.,0.,99.9999999999999,0.,0., 57P 29
-5.6843418860808D-14,0.,0.,100.,0.,0.,0.;

It is a curve object (type 126). First two numbers indicate number of control points minus one and degree. Therefore it has two control points and degree is one. Next three numbers are flags (value 1 or 0) indicating if it is closed or open curve, polynomial or rational and periodic or not. In this case it is an open rational non-periodic curve. Next numbers indicate knot vector, control points weights and coordinates. Hence this curve knot vector is {0,0,100,100}, control points weights are {1,1}, control point coordinates are {0,99.99,0} and {0,0,0}. The latest 4 numbers of curves are not used in this work.

Parameter data section. Example 2:

510,177,1,1,69; 45P 23

It is a face object (type 510). First number indicates *parameter data section* address of underlying surface, second number indicates the number of loops and the third is an outer loop flag (0 or 1 for inner and outer). In this case there is one outer loop. Last number is the *parameter data section* address to the unique loop. In case there were more loops their addresses would be written behind.

At the rear of *parameter data section* they are stored the colour definition entities (type number 314). They are referred in the *directory entry* section by the *colour address* (mentioned above). This address number is followed by a "P" and it is located before the last number. **Fig. 2G.3** illustrates one example. The first number of the line is the type, the next three numbers correspond to the RBG colour code given in percentage (100 % corresponds to 255). For example, first colour code in **Fig 2G.3** has next three colour numbers: {201, 100, 0} that are the percentages of 255 shown in the IGES file. Next number is the *colour address* (145P and 147P) and the last number is the address in parameter data section.

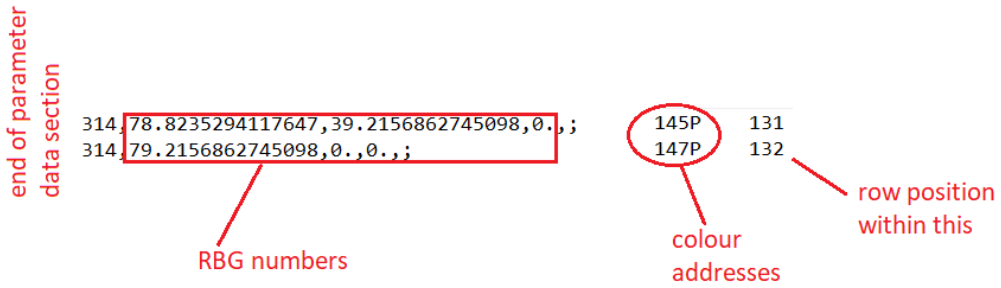


Fig. 2G.3 Colour entities in *parameter data section*.

Appendix 3A: Code notes

This appendix describes the most important variables used in the algorithm, and summarizes the algorithm itself in blocks diagrams. The two principal variables are the patch class and the boundary surface class. The former hosts the patches parametrization and discretization. The later contains the boundary surfaces parametrization, discretization and the boundary conditions values (prescribed displacements and surface tractions). They are referred in the code as `Spatches` and `Bentities`. Previous to the description of those two classes, the auxiliary structures of data used in the algorithm are presented in four sections as follows:

- Auxiliary structures 1: transference from IGES files. They store the information from the IGES files keeping the same arrangement as per those files.
- Auxiliary structures 2: IGES information arranged. They store the information from the IGES files but arranged to facilitate their use in this algorithm.
- Auxiliary structures 3: prepared for skins, patches and boundary surfaces generation. These structures are devised to compute the skins, patches and surface generation easily.
- Auxiliary structures 4: tetrahedralization. The structures used in the tetrahedralization of the solids are described here.

3A.1 Auxiliary structures 1: transference from IGES files

The IGES objects name, reference number in the IGES files and the array of the code where it is stored are shown in **Table 3A.1**.

Table 3A.1 IGES objects storage.

Type number in IGES system	Graphic object	Structure
186	***manifold solid B-rep***	`MSBO`
514	***shell***	`shells`
510	***face***	`faces`
128	***rational B-spline surface*** (***surface***)	`surf`
508	***loop***	`loops`
504	***edges***	`edges`
126	***curve***	`curves`
502	***vertex***	`vertexes`
143	***bounded surface***	`bsurf`
141	***boundary***	`bound`

Next tables show the components of each structure that contain IGES file information. Each structure is an array of *sub-structures*. For example, `Sshells` is an array of *n sub-structures*, where each `Sshells` has the components `R`, `Nof`, `faR` and `faO`, being *n* is the number of shells. In the code schemes (section 3A.7), the gross and bounded patches are stored in the so-called *sG arrays* and *sB arrays* respectively.

Smsbo

Component	Description
`shR`	Pointer to correspondent `Sshell`

Sshells

Component	Description
`R`	Reference in the IGES file
`Nof`	Number of faces attached
`faR`	Pointer to attached `Sfaces` (`Nof` faces per shell)
`faO`	Orientation of face: 1 / 0 if points outside / inside the shell

Sfaces

Component	Description
`R`	Reference in the IGES file
`suR`	Pointer to attached `Ssurf` (one surface per face)
`Nol`	Number of loops attached
`IOL`	Inner/outer loop flag, value is 1/0
`loR`	Pointer to attached `Sloops` (`Nol` loops per face); first loop is the outer

Sloops

Component	Description
`R`	Reference in the IGES file
`Noe`	Number of edges attached
`edR`	Pointer to attached collection of `Sedges`
`eLI`	Pointer to attached `Sedges` within the collection (`Noe` edges per loop)
`edO`	Orientation of edge: 1 / 0 leaves computable surface at left / right hand side

Sedges

Component	Description
R	Reference in the IGES file
Noc	Number of curves attached
cuR	Pointer to attached Scurves (Noc curves per edge)
v1R and v2R	Pointer to attached collection of Svrtx, for initial and final vertexes
v1LR and v2LR	Pointer to attached Svrtx within the collection (two vertexes per edge)

Svrtx

Component	Description
R	Reference in the IGES file
Nop	Number of points in the collection
P	Coordinates of the vertexes of the collection (Nop sets of coordinates)

Sbsurf

Component	Description
R	Reference in the IGES file
Nob	Number of boundaries attached
boR	Pointer to attached Sbound (Nob boundaries per bounded surface)
suR	Pointer to attached Ssurf (one surface per bounded surface)

Sbound

Component	Description
R	Reference in the IGES file
Noc	Number of curves attached
cuR	Pointer to attached Scurves (Noc curves per boundary)
cuO	Orientation of curve: 1 / 0 leaves computable surface at left / right hand side

Ssurf

Component	Description
R	Reference in the IGES file
NURBS	Ncp, Degree, Usize, KnotV, w, P
OC	Open / closed surface flag, whose value is 1 / 0; for two parameter directions
PL	Plane surface flag, whose value is 1 / 0 for plane / curved surface

Scurves

Component	Description
`R`	Reference in the IGES file
`NURBS`	`Ncp`, `Degree`, `Usize`, `KnotV`, `w`, `P`
`OC`	Open / closed curve flag, whose value is 1 / 0
`PL`	Plane curve flag, whose value is 1 / 0 for plane / curved line

The relationship between the arrays is depicted in Fig. 3A.1. The scheme is valid for both *sG arrays* and *sB arrays*.

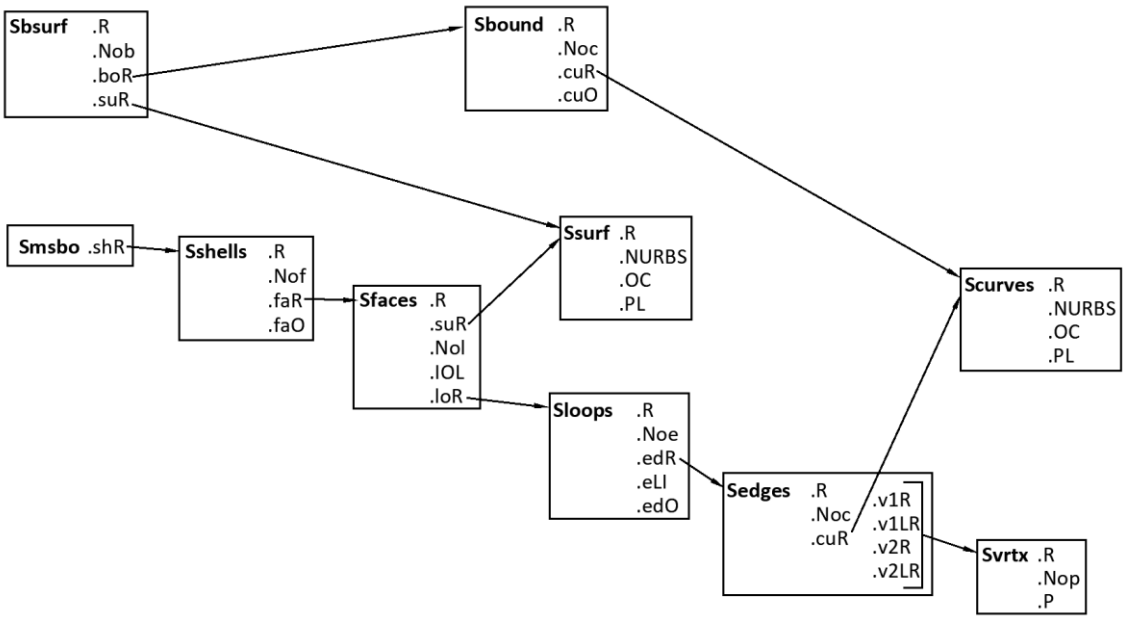


Fig. 3A.1 Relation bewteen arrays that contain IGES file information.

3A.2 Auxiliary structures 2: IGES information arranged

The tables of this section provide the components of the structures that store information of the patches and boundary surfaces prepared specifically for solid parametrization and identification of surfaces.

gShell

Component	Description
`vP(8,xdim)`	Vertex coordinates (arranged hierarchically)
`face(6,1)`	Faces features (see below)
`eCurves(12,1)`	Edge curves features

face

Component	Description
`eCurveR(2,2)`	Edge curves refences
`fSurface`	Features of surface at the background

Fig. 3A.2 illustrates the relationship between the structures contained within `gShell`. **Fig. 3A.3** shows one example of `bShell`, where `face` 2 is detailed. `eCurves` are arranged as follows:

- Curves references (1,1) and (1,2) parallel to ξ direction, bottom and top.
- Curves references (2,1) and (2,2) parallel to η direction, left and right.

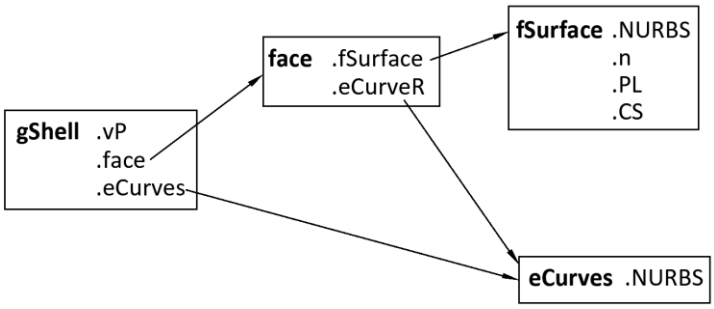


Fig. 3A.2 Structures referred by `gShell`.

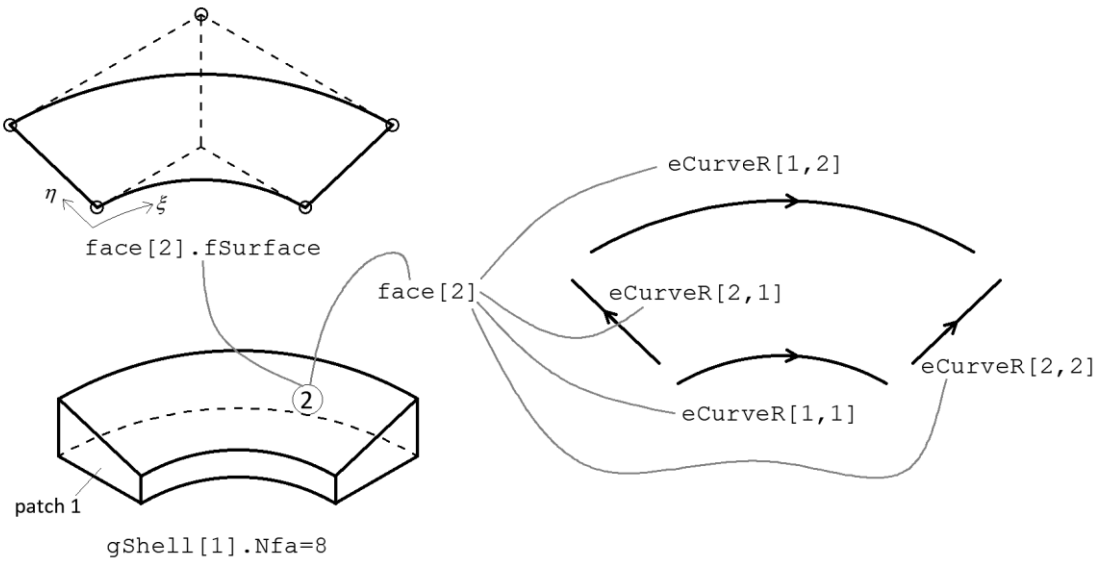


Fig. 3A.3 Example of `gShell` structure.

gbShell and bShell

Component	Description
`Nfa`	Number of faces. In case of `gbShell`, `Nfa` = 6 always
`face(Nfa,1)`	Faces features (see below)

face

Component	Description
`fSurface`	Features of surface at the background
`Nol`	Number of loops of the face
`fLoop(Nol,1)`	Features of curves of loop (see below)

fSurface

Component	Description
`NURBS features`	
`n`	Normal versor (for plane surfaces only)
`PL`	Plane surface flag, 1/0 for plane / curved surfaces
`CS(udim,1)`	Closed surface flag, 1/0 for closed / open surfaces

fLoop

Component	Description
`Noc`	Number of curves of the loop
`LCurves(Noc,1)`	NURBS features

The structures contained within `gbShell` or `bShell` are represented in **Fig. 3A.4**. One example of `bShell` is given in **Fig. 3A.5**, where face 2 is detailed. `LCurves` are arranged such that the computable surface lies at left-hand side of the advancing parameter direction of the curve.

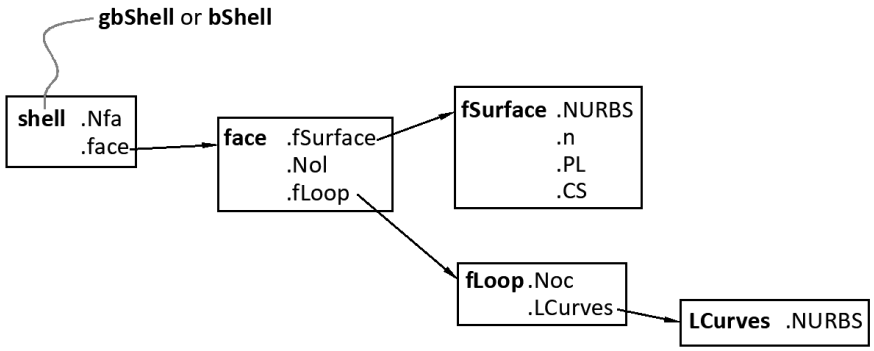


Fig. 3A.4 Structures referred by `gbShell` or `bShell`.

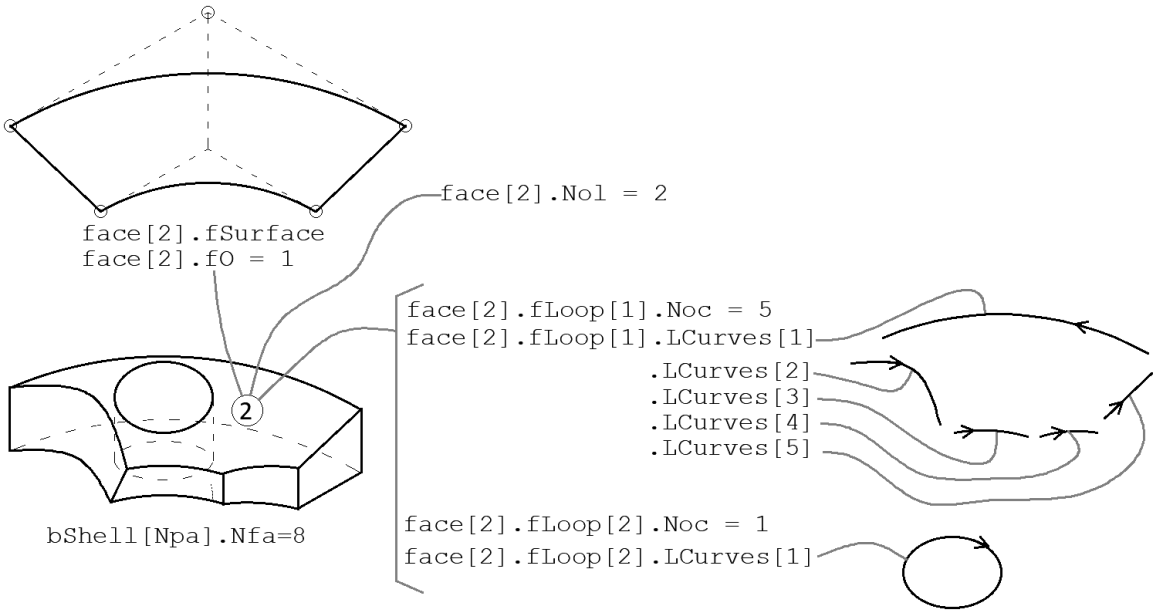


Fig. 3A.5 Example of `bShell` structure.

3A.3 Auxiliary structures 3: for generation of skins, patches and boundary surfaces

The tables of this section provide the structures that store information to generate:

- The NURBS features and `skins` of the patches (`Spatches`), both gross and trimmed. This information is stored in `gFace`, `tFace`, `tCurv` and `FgPatch`.
- The `skins` of the boundary surfaces (`Bentities`). This information is stored in `bEnti` and `bCurv`.
- Additionally coupling, trimming and reverse flags for surfaces are stored in `bCou`, `bTr` and `revs` respectively.

gFace and tFace

Component	Description
`Nfa`	Number of faces, for gross faces it is always six
`mSur(Nfa,1)`	Surfaces features

There is one `tFace` per patch.

mSur

Component	Description
`NURBS features`	
`PL`	Plane surface flag, 1/0 for plane / curved surfaces
`CS(udim,1)`	Closed surface flag, 1/0 for closed / open surfaces
`Nol`	Number of loops of the face
`Noc(Nol,1)`	Number of curves per loop of the face
`Rcu(Nol,Noc)`	Pointer to curves attached to each loop
`Tr`	Trimming flag, `Tr`=1 if trims the gross patch, otherwise `Tr`=0

Fig. 3A.6 shows the information contained in `gFace`, which is applicable also to `tFace`. There is an important difference between `gFace` and `tFace`. The former faces are coincident with the gross patches, i.e. form a regular shell enveloping the gross patch with each face contoured by four curves that are the intersection with the adjacent faces. By contrast, the `tFace` stores the background surfaces, whose do not form a regular shell. These surfaces may be trimmed by one or more loops, therefore may extend further than the contours, i.e. the computable surface does not coincide with the background surface.

The number of patches, is stored in `Npa`. There is one `gFace` and one `tFace` per patch. `gFace` wraps the gross patch with a regular shell of faces, and `tFace` wraps the trimmed patch with non-regular shell of faces, which are trimmed by contour curves. **Fig 3A.7** illustrates one example of

gFace and **Fig. 3A.8** one example with tFace. In **Fig. 3A.8** the bEnti structures are also represented.

Each surface of tFace and bEnti is defined by a background NURBS surface and one or more bound loops of curves. These curves may trim or not the NURBS surfaces. The number of curves contouring tFace and bEnti are stored in Ntc and Nbc respectively.

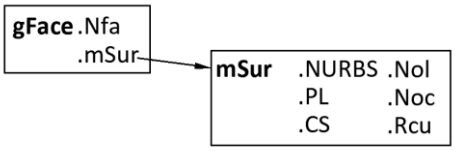


Fig. 3A.6 Structures contained by gFace and tFace.

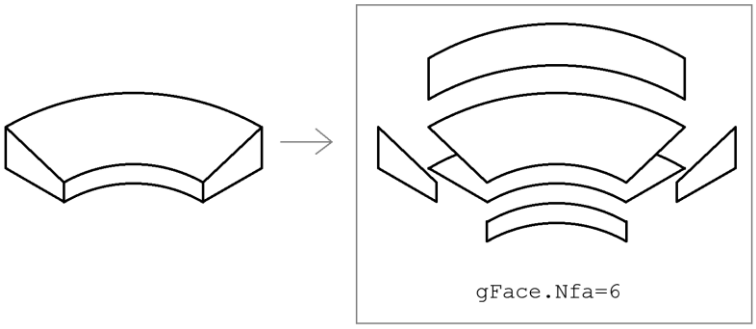


Fig. 3A.7 Example of gFace.

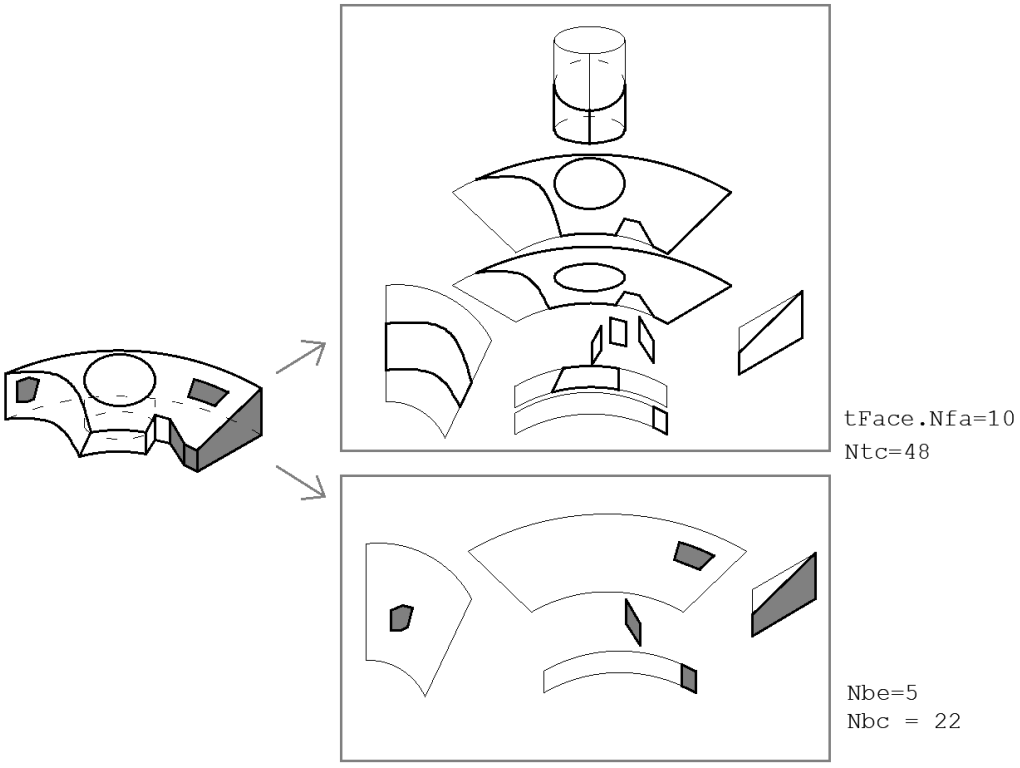


Fig. 3A.8 Example of tFace and bEnti.

FgPatch

Component	Description
NURBS features	
nbe	Number of attached boundary entities
Rbe	Point to attached boundary entities

The FgPatch structure contains all the information of the parametrized solid. One example is given in **Fig. 3A.9**.

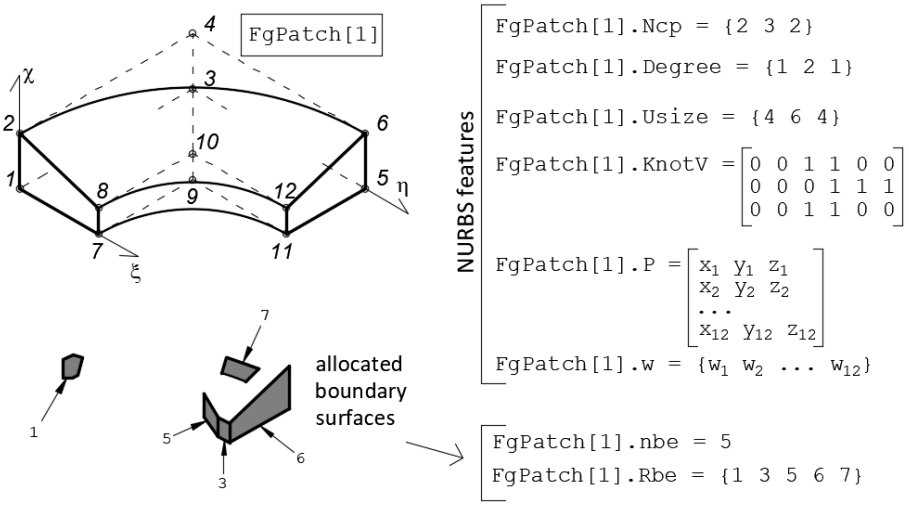
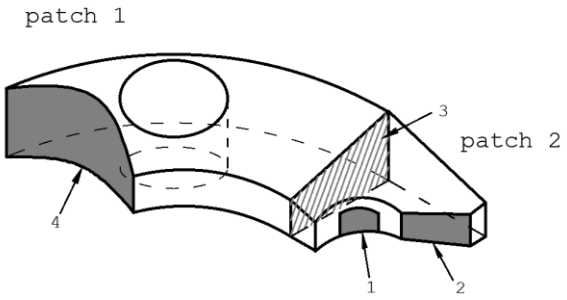


Fig. 3A.9 Example of FgPatch structure.

bEnti

Component	Description
NURBS features	
PL	Plane surface flag, 1/0 for plane / curved surfaces
CS(udim,1)	Closed surface flag, 1/0 for closed / open surfaces
Nol	Number of loops of the face
Noc(Nol,1)	Number of curves per loop of the face
Rcu(Nol,Noc)	Pointer to curves attached to each loop
Tr(2,1)	Trimming flag, 0 / 0.5 / 1 for external / internal / trimming
Cou	Coupling reference in case it is a coupling surface, otherwise it is zero
Pa(2,1)	Patches reference, the first is the master in case the entity is coupling

The boundary surfaces are all stored in bEnti regardless the patch where they belong. See one example in **Fig. 3A.10**. The patch where the each boundary surface applies is stored in Pa array. In case of coupling surfaces, Pa stores both patches references, otherwise the first component stores the involved patch and the second is zero.




```
bEnti[1]              bEnti[3]
    .Tr = [0.5 0]         .Tr = [0 0]
    .Cou= 0               .Cou= 1
    .Pa = [2 0]           .Pa = [1 2]

bEnti[2]              bEnti[4]
    .Tr = [1 0]           .Tr = [1 0]
    .Cou= 0               .Cou= 0
    .Pa = [2 0]           .Pa = [1 0]
```

Fig. 3A.10 `bEnti` structures.

tCurv and bCurv

Component	Description
`NURBS features`	
`CD`	Closed line flag, 1/0 for closed / open
`conF`	Contour flag, 1/0 for curves at edge of surface / internal

The curves that contour `tFace` and `bEnti` are stored in `tCurv` and `bCurv` respectively. Next explanation, given for `bEnti`, is also valid for `tFace`.

`bEnti` and `bCurv` are transferred from *sB arrays*, i.e. `sBbsurf` and `sBcurves`. In `bEnti` each surface has its own contour curves, not shared by others, and leaving the computable surface at the left-hand side. The references of the surfaces in the IGES file and the user references can be different. The relationship between both references is stored in `beoR`. **Fig.3A.11** provides one example of three surfaces, where each surface in `bEnti` has its own contour curves, meanwhile in `bBsurf` the curves are shared.

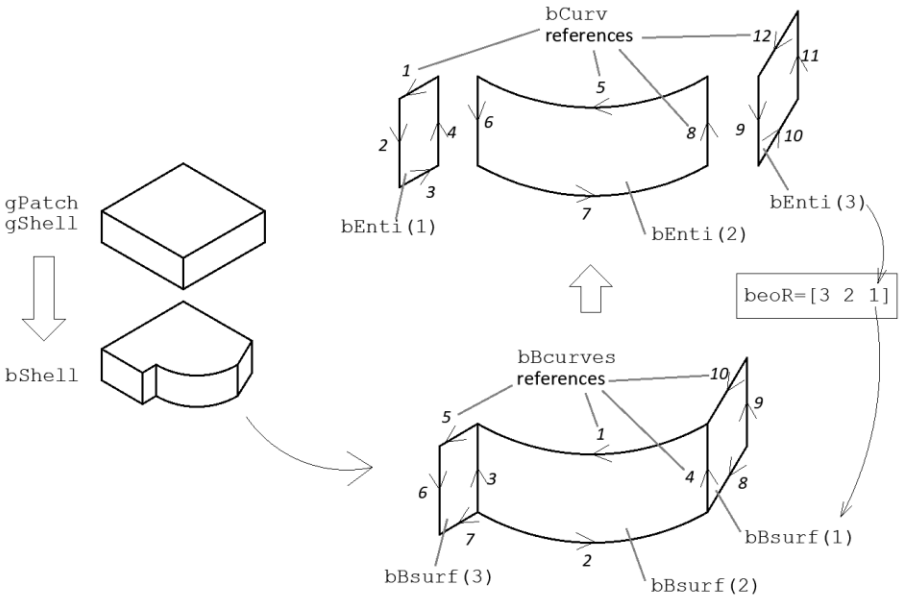


Fig. 3A.11 Relation between `sBsurf/sBcurves` and `bEnti/bCurv`.

Coupling, trimming and reversal flags of boundary surfaces

Boundary surfaces need to be defined in terms of coupling, trimming and reversal. This definition is stored in arrays of flags, that are called `bCou`, `bTr` and `revs` respectively. One example is illustrated in **Fig. 3A.12**. In addition, the coupled patches references are stored in `pCou` array. For example, in **Fig. 3A.12** there is a single coupling between patches 1 and 2.

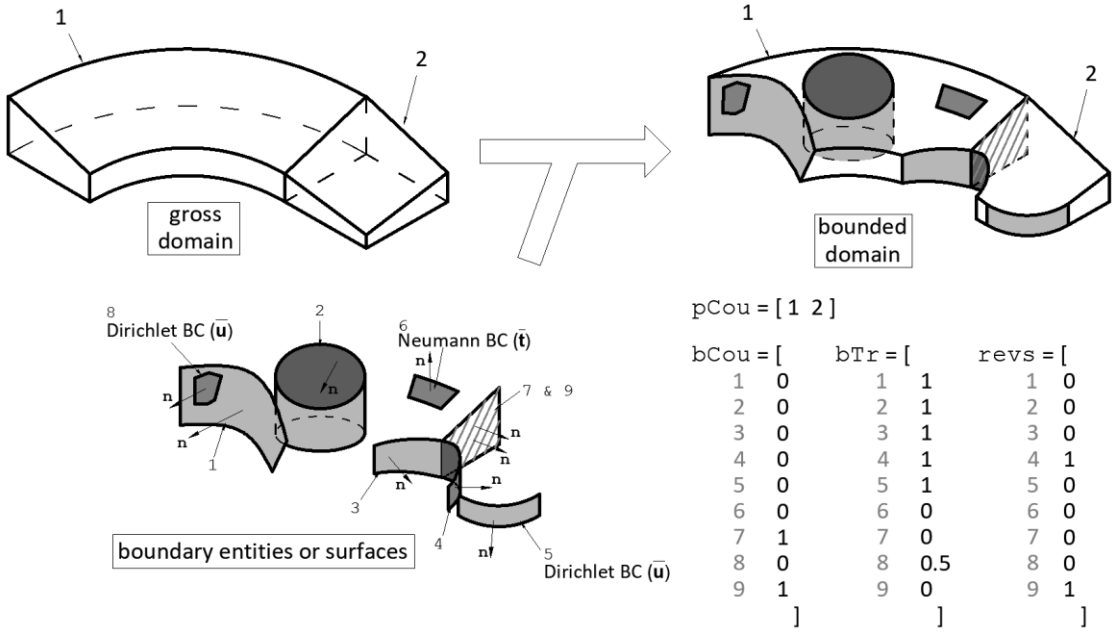


Fig. 3A.12 Examples of `bCou`, `bTr`, `revs` and `pCou`.

Contour curves of trimmed patches (ccTp)

The array `ccTp` stores the information of contour of curves of trimmed patches that are aggregated, i.e. non-repeated. There is one `ccTp` per patch. Its components are:

Component		**Description**
`Nf, Nc`		Number of faces and contour curves of the patch
`nlo(Nf,1)`		Number of loops per face
`ncu(Nf,nlo)`		Number of curves per loop of each face
`agCurv(Nc,1))`		Curves features (each one is a `tCurv` structure)
`SCcon(Nf,nlo,ncu)`		Surface VS curve connectivity matrix, passing through the loop reference
`CScon(Nc,*)`		Curve VS surface connectivity matrix. In case only one surface is attached to a curve, the rest of the row components are zero.
`NFC(Nc,1)`	`n`	Number of vertexes of the curve
	`Nsr`	Number of surfaces where the curve lies in
	`sr(Nsr,1)`	Surfaces references
	`Un(Nsr,n,2)`	Coordinates in surfaces parameter spaces
	`Bn(n,1)`	Coordinates in curve parameter space
	`An(n,3)`	Coordinates in patch parameter space
	`Xn(n,3)`	Coordinates in physical space

3A.4 Auxiliary structures 4: tetrahedralization

GPT

Acronym of *gross patch trimmed*. Contains the features of the tetrahedralization of the gross-trimmed patch.

Component	Description
`NE`	Number of nodes (only end-nodes, nod mid-edge nodes)
`Nn`	Number of elements (tetrahedrons)
`Xv(Nn,xdim)`	Nodes coordinates in physical space
`Av(Nn,pudim)`	Nodes coordinates in parameter space
`EVcon(NE,4)`	Connectivity matrix : Tetrahedron VS vertex
`DT`	Delaunay object (from Matlab®)
`Lax(NE,2)`	Average of edges lengths of tetrahedrons in parameter and physical spaces
`Lv(Nn,2)`	Average of lengths of all edges attached to each node in parameter and physical spaces

THC

Contains the coordinates of the nodes of the tetrahedral mesh. The number of nodes is designated by `Nn`.

Component	Description
`Xv(Nn,xdim)`	Nodes coordinates in the physical space
`Av(Nn,pudim)`	Nodes coordinates in the parameter space of the patch
`Uv(Nn,sudim)`	Nodes coordinates in the parameter space of the surface (if applies)
`Bv(Nn,cudim)`	Nodes coordinates in the parameter space of the curve (if applies)
`NCS(Nn,2)`	Curve and surface references where to node lies on (if applies)
`rm(Nn,1)`	Reference of the node in the MLT (if applies)
`ANv(Nn,pudim)`	Norma vectors to nodes that belong to the MLT (if applies) in the patch parameter space.

MLT

Acronym of *merged linear triangulations*.

Component	Description
`tNE`	Number of elements (triangles)
`tNV`	Number of nodes (only end-nodes, nod mid-nodes from `rTskin`)
`tNL`	Number of lines
`tTVcon(tNE,3)`	Connectivity matrix: Element vs Vertexes
`tLVcon(tNL,3)`	Connectivity matrix: Lines vs Vertexes
`tVLcon(tNV,*)`	Connectivity matrix: Vertex vs Edge Lines
`tTLcon(tNE,3)`	Connectivity matrix: Element vs Lines
`tVTcon(tNV,*)`	Connectivity matrix: Vertex vs Element
`tLTcon(tNE,2)`	Connectivity matrix: Edge line vs Element
`tUv(tNV,2)`	Parameter coordinates of surface where the node lies in
`tBv(tNV,1)`	Parameter coordinates of curve where the node lies in
`tAv(tNV,3)`	Coordinates in patch parameter space of vertexes
`tXv(tNV,3)`	Coordinates in physical space of vertexes
`tNCS(tNV,2)`	Curve and surface attached to each node (0 if it does not apply)
`tANv(tNV,3)` `tXNv(tNV,3)`	Normal versor in patch parameter space and physical space to each node (averaged adjacent triangles normal)
`tANe(tNE,3)` `tXNe(tNE,3)`	Normal versor in patch parameter space and physical space to each triangle
`tAtetIN(tNE,3)` `tAtetOU(tNE,3)`	Coordinates of node to from optimal tetrahedron for each triangle. The node projection lies at the triangle centroid. In patch parameter space.
`tXtetIN(tNE,3)`	Coordinates of node to from optimal tetrahedron for each triangle in physical space and only for node the lie within the patch domain
`tDtet(tNE,1)`	Distance to triangle from optimal tetrahedron node

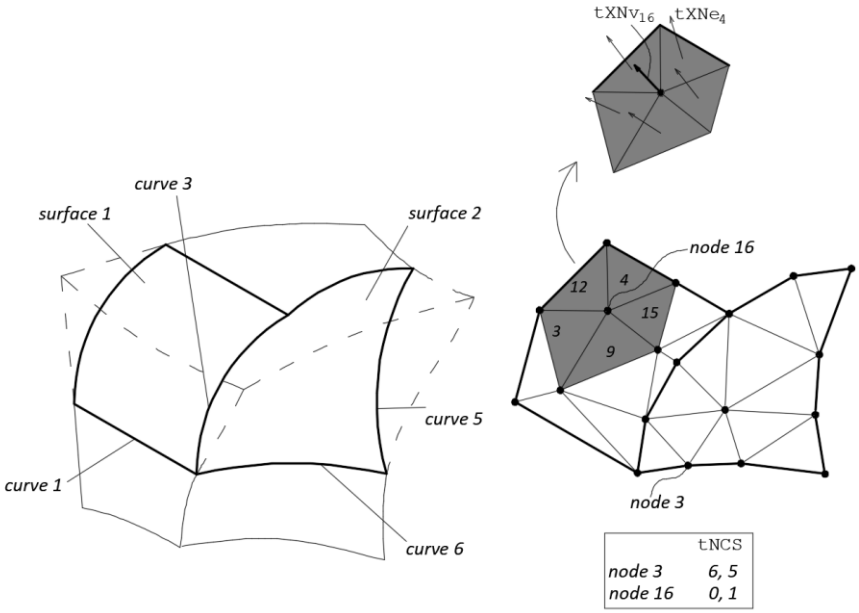
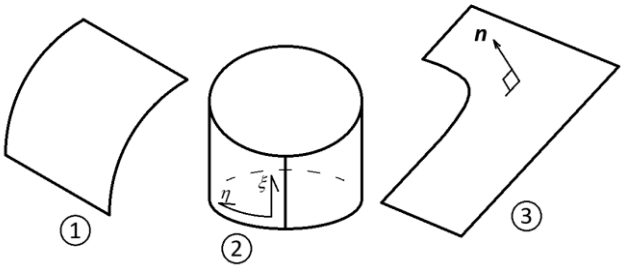


Fig. 3A.13 `MLT` structure.

rTskin (cTskin)

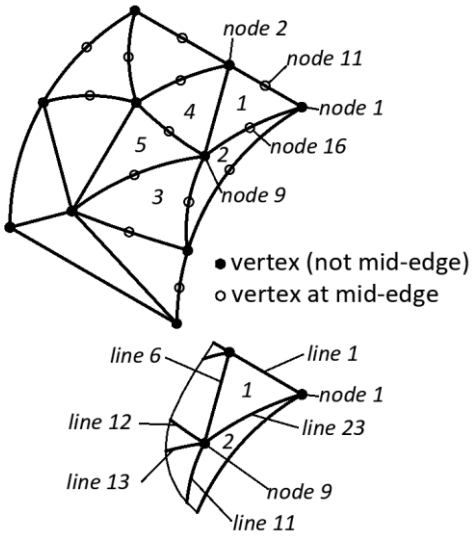
These triangles are used as tetrahedron facets in contact with the trimming surfaces.

Properties	Description
NURBS features	
mCS(udim,1)	Closed surface flag, equals to 1 in case the surface is closed
mPN	Plane surface flag
mNE	Number of triangles
mNV	Number of vertexes, with no mid-edge vertexes
mNVm	Number of vertexes including mid-edges, for quadratic surfaces
mBv(mNVm,1)	Parameter coordinates in curve parameter space
mAv(mNVm,3)	Patch parameter space coordinates of vertexes (master patch)
mUv(mNVm,2)	Surface parameter space coordinates of vertexes
mXv(mNVm,xdim)	Physical space coordinates of vertexes
mTVcon(mNE,*)	Connectivity matrix: Element vs Vertexes
mVTcon(mNV,*)	Connectivity matrix: Vertex vs Elements
mLVcon(*,*)	Connectivity matrix: Line vs Vertexes. In case of mid-vertex it appears at the second position
mVLcon(mNV,*)	Connectivity matrix: Vertex vs Lines
mTLcon(mNE,3)	Connectivity matrix: Element vs Lines
mArea(mNE,1)	Area or each triangle (assumed linear) in physical space
mANv(mNV,3)	Normal to each vertex in patch parameter space
mANe(mNE,3)	Normal to each triangle (assumed linear) in patch parameter space
mXNv(mNV,3)	Normal to each vertex in physical space
mXNe(mNE,3)	Normal to each triangle (assumed linear) in physical space
mNlo	Number of contour loops of the NURBS surface
mNlv(mNlo,1)	Number of vertexes per contour loop. The final vertex, that coincides with the first vertex is not considered
mNC(mNVm,1)	Curve reference where the node lies in (equals 0 if not applies)



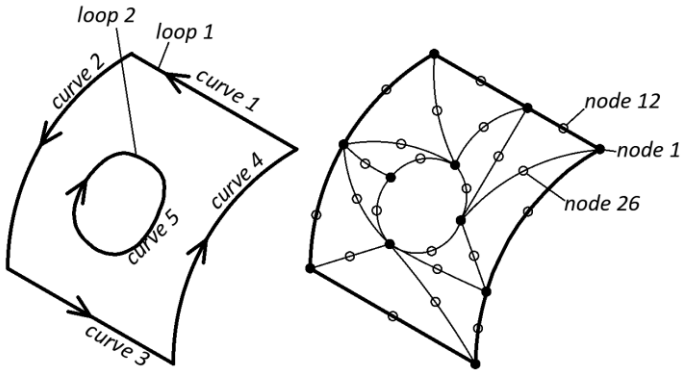
rTskin(1)	.mCS = (0 , 0)
	.mPN = 0
rTskin(2)	.mCS = (0 , 1)
	.mPN = 0
rTskin(3)	.mCS = (0 , 0)
	.mPN = 1

Fig. 3A.14 Closed and plane flags of rTskin.



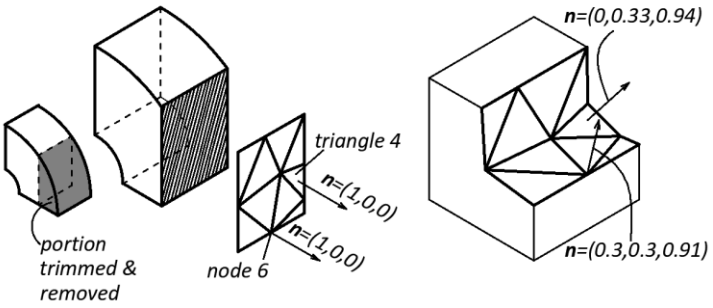
rTskin(1)	.mNE = 11
	.mNV = 10
	.mNVm = 24
	.mTVcon(1,…)=(1,2,9,16,11,0)
	.mVTcon(9,…)=(1,2,3,4,5)
	.mLVcon(6,…) =(1,16,9) .mLVcon(23,…)=(2,9,0)
	.mVLcon(9,…)=(6,12,13,11,23)
	.mTLcon(1,…)=(2,6,23)
	.iem(5,…,…)= 0 1 2 1 0 0 2 0 0

Fig. 3A.15 Connectivity matrices of `rTskin`.



rTskin(1)	.mNlo = 2
	.mNCL = (4,1)
	.mNlv = (7,4)
	.mNlcv(1,2) = 5

Fig. 3A.16 Contour curves loops of `rTskin`.



rTskin(1)	.mXNv(6,…) = (1,0,0)
	.mXNe(4,…) = (1,0,0)
	.mANv(6,…) = (0,0.33,0.94)
	.mANe(4,…) = (0.3,0.3,0.91)
	.mANvS(6,…) = (1,0,0)
	.mANeS(4,…) = (1,0,0)

Fig. 3A.17 Normal vectors of `rTskin`.

3A.5 Patch Class

In this section `Spatches`, that is class `cPatch`, is presented. **Fig. 3A.18** shows all the components of the class `cPatch`. When refer to one object**,** its name in the code is shown first, and afterwards in brackets the class name. For example `Spatches` (`cPatch`).

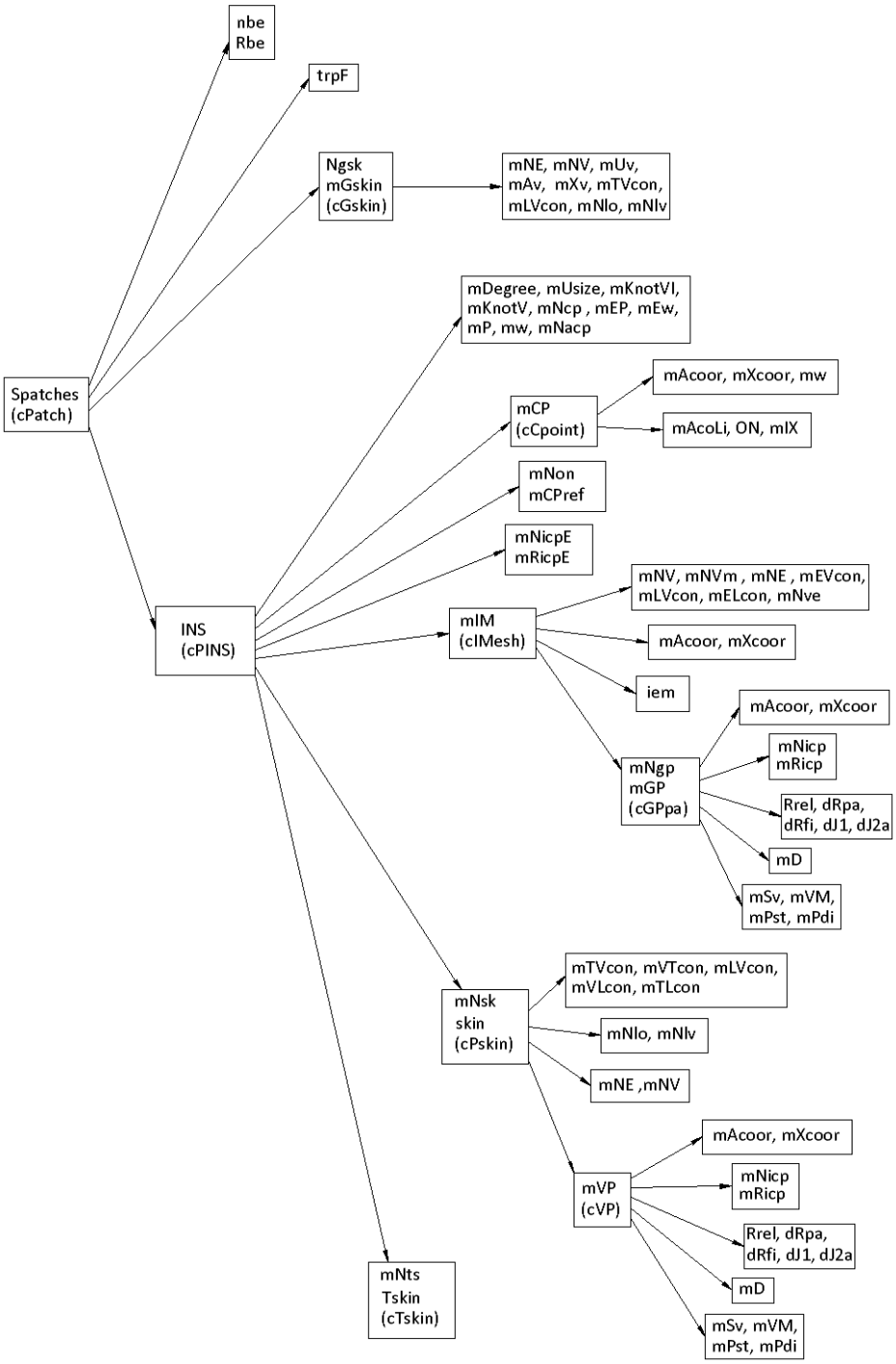
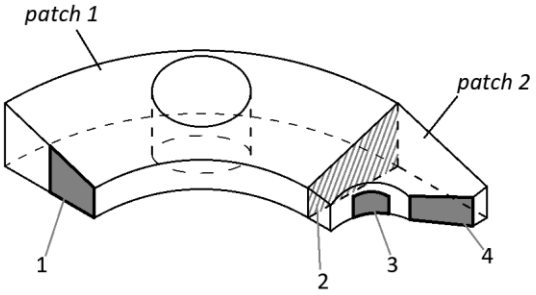


Fig. 3A.18 Components of `Spatches` object.

cPatch(Npa)

Properties	Description
`nbe`	Number of boundary entities attached to the patch
`Rbe(nbe,1)`	References of the attached boundary entities
`trpF`	Trimmed patch flag, equal to 1 if trimmed, 0 otherwise
`Ngsk`	Number of gross skins (restrained to six)
`mGskin(Ngsk,1)`	`cGskin` class
`INS(Nts,1)`	`cPINS` class




```
Spatches(1).nbe = 2
Spatches(1).Rbe = [1 2]
Spatches(1).trpF = 1

Spatches(2).nbe = 3
Spatches(2).Rbe = [2 3 4]
Spatches(2).trpF = 1
```

Fig. 3A.19 `Spatches` object.

mGskin (cGskin)

Properties	Description
`mNE`	Number of triangles
`mNV`	Number of vertexes
`mTVcon(mNE,*)`	Connectivity matrix: Element vs Vertexes
`mLVcon(*,2)`	Connectivity matrix: Line vs Vertexes.
`mUv(mNV,2)`, `mAv(mNV,3)`, `mXv(mNV,xdim)`	Vertexes coordinates in surface parameter space, patch parameter space and physical space
`mNlo`	Number of loops that contour the skin
`mNlv(mNlo,1)`	Number of vertexes per loop (the first vertex is not repeated)

INS (cPINS)

Properties	Description
`NURBS features`	
`mCP(mNacp,1)`	`cCpoint` class
`mNon`	Number of control points that are active
`mCPref(mNon,1)`	References of active control points, if control point is deactivated then its reference is 0
`mNicpE(nbe,1)`	Number of control points that are influential on each boundary entity
`mRicpE(nbe,mNicpE)`	References of influential control points in boundary entities
`mIM`	`cIMesh` class
`mNsk`	Number of skins
`skin(mNsk,1)`	`cPskin` class

Some components of instances (INS) are shown in **Fig. 3A.20**, the active control points are filled at the bottom. One example of influential control points is given in **Fig. 3A.21**, where influential control points on boundary surfaces 1 and 2 are shown at left and right-hand side respectively.

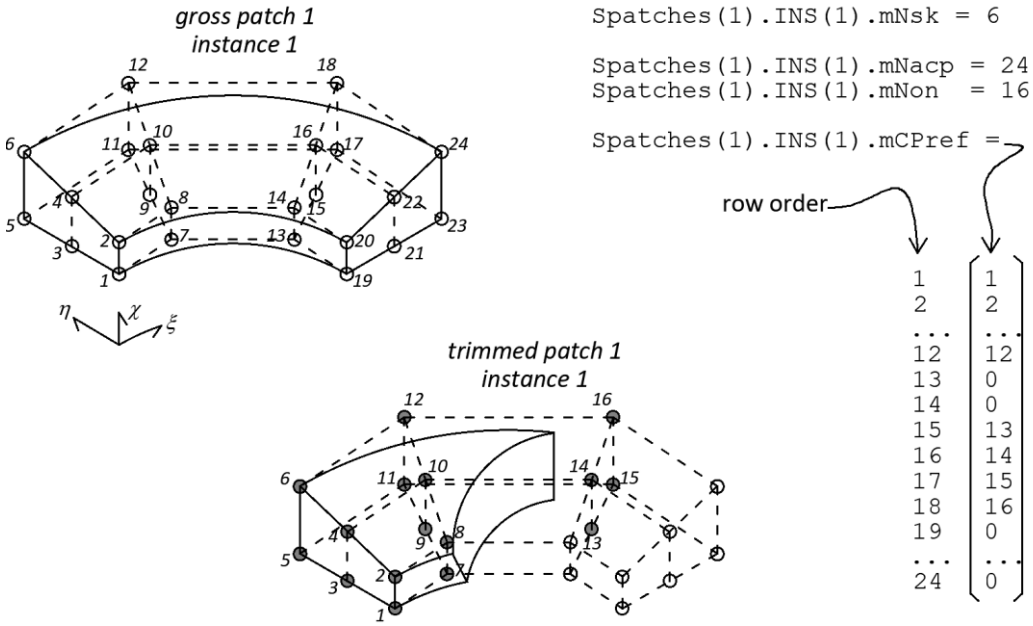


Fig. 3A.20 INS object.

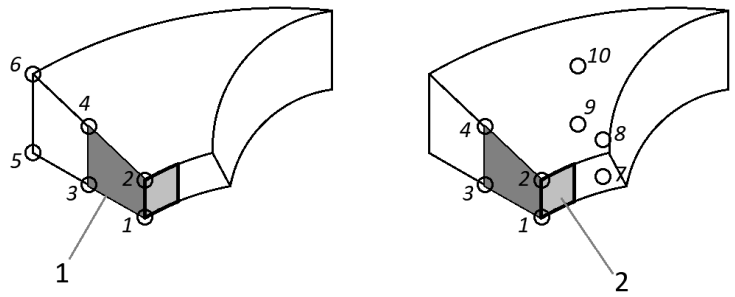
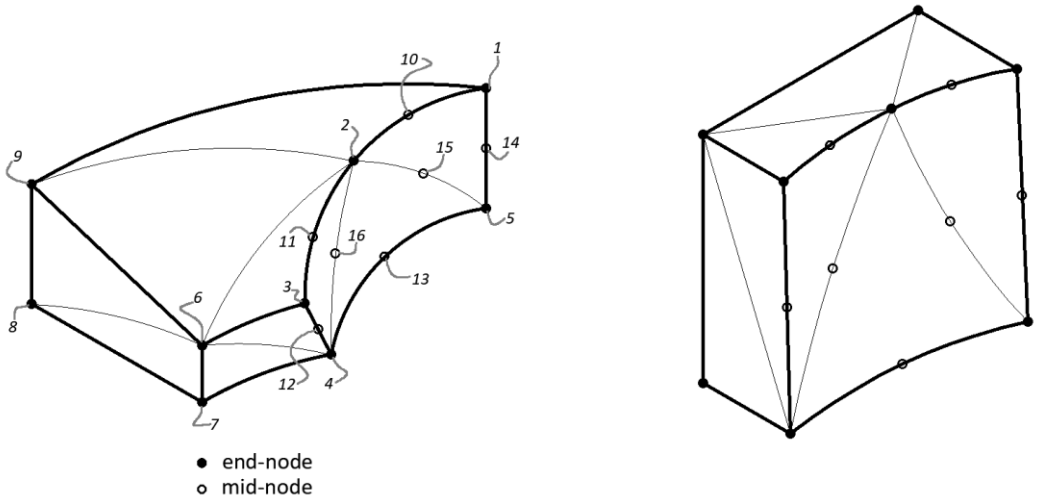


Fig. 3A.21 Influential control point on boundary surfaces.

mIM (cIMesh)

Properties	Description
`mNV`	Number of vertexes, without the mid-edge vertexes
`mNVm`	Number of vertexes in total, counting also the mid-edge vertexes
`mNE`	Number of tetrahedrons
`mEVcon(mNE,*)`	Connectivity matrix: Element vs Vertexes
`mLVcon(mNL,*)`	Connectivity matrix: Line vs Vertexes. Mid-vertex appears at 2nd position.
`mELcon(mNE,6)`	Connectivity matrix: Element vs Lines. Lines-vertexes: 1-2, 2-3, 3-1, 4-1, 4-2, 4-3
`mNve(mNE,1)`	Number of vertexes per element
`mAcoor(mNVm,3)`, `mXcoor(mNVm,3)`	coordinates of vertexes in patch parameter and physical spaces
`iem(mNE,4,4)`	Inserted edges matrix
`mNgp`	Number of Gauss points
`mGP(mNgp,1)`	`cGPpa` class

Fig. 3A.22 shows one tetrahedral mesh in physical and parameter spaces. The first nodes numbered are the end-nodes, followed by the mid-nodes. **Fig. 3A.23** shows the detail of one linear and one mixed-degree tetrahedrons. In case of quadratic edges (as line 25 of tetrahedron 3), the mid-node appears at the second position in the connectivity matrix `mLVcon`.




```
Spatches(1).INS(1).mIM.mNE =   8
Spatches(1).INS(1).mIM.mNV =   9
Spatches(1).INS(1).mIM.mNVm =  16
```

Fig. 3A.22 Integration mesh in physical and parameter spaces.

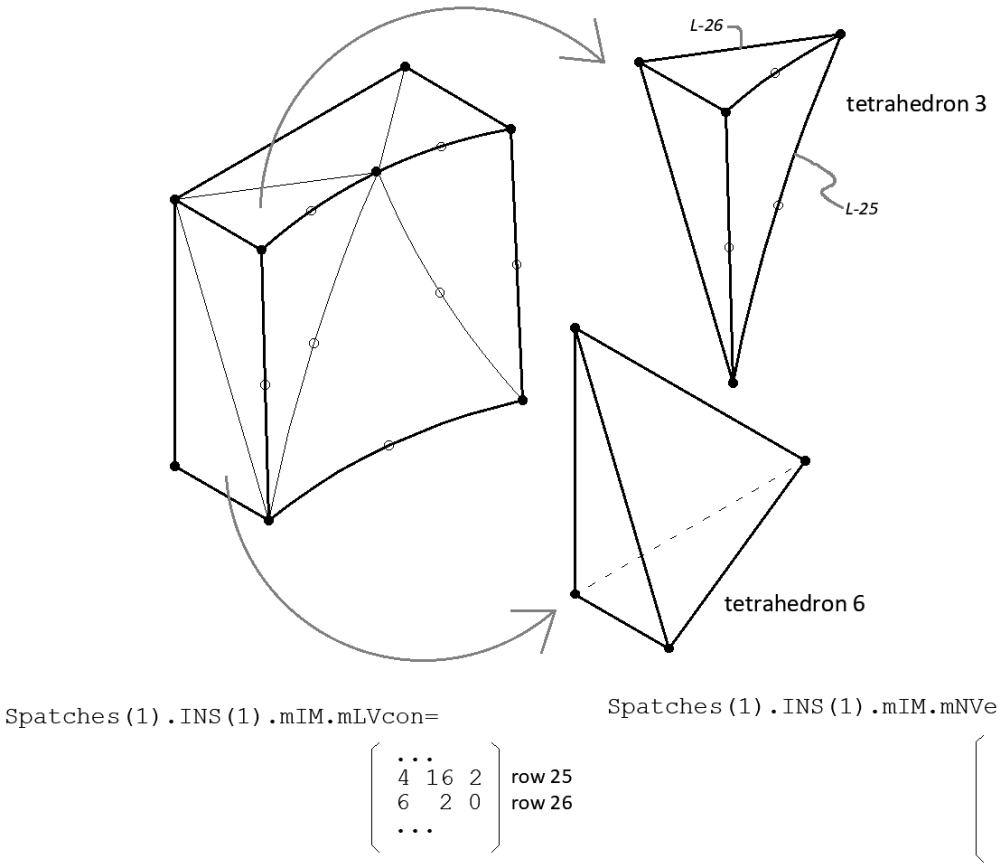


Fig. 3A.23 Linear and mixed-degree tetrahedrons.

mCP (cCpoint)

Properties	**Description**
`mAcoor(pudim,1)`, `mXcoor(xdim,1)`	Coordinates in patch parameter and physical spaces
`mw`	Weight
`mAcoLi(pudim,2)`	Limits of influential volume in patch parameter space
`ON`	Flag for indicating the control point is active (`ON` = 1) or not (`ON` = 0)
`mIX(udim,1)`	Indexes of the position of the control point within the control points net in each parameter direction

The influential volume of control points in parameter space, given by `mAcoLi` is given for one example in **Fig. 3A.24**. At left-hand side the trimmed parameter space is shown. The circles indicate control points, being the active control points filled. The patch Gauss points are represented by crosses. At centre and right, the influential volume is shown for control points 12 and 14. In the latter case there are two Gauss points within the volume.

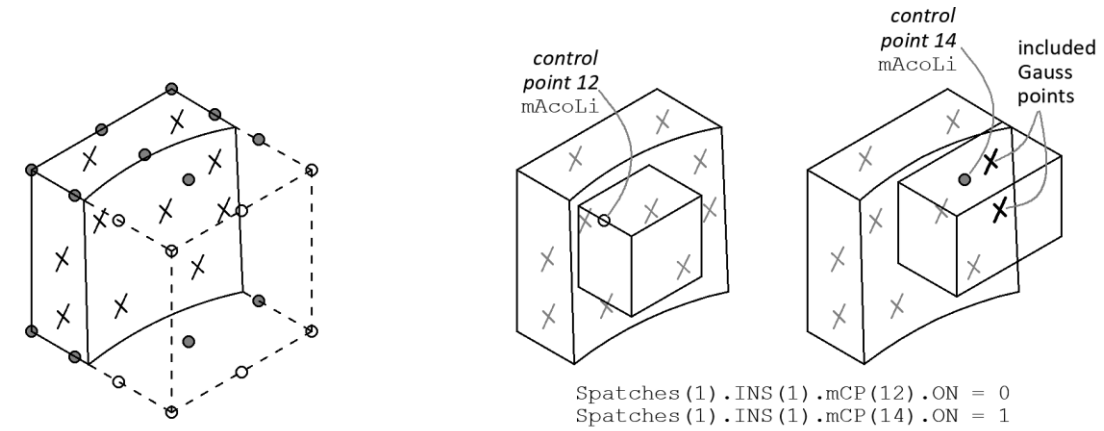
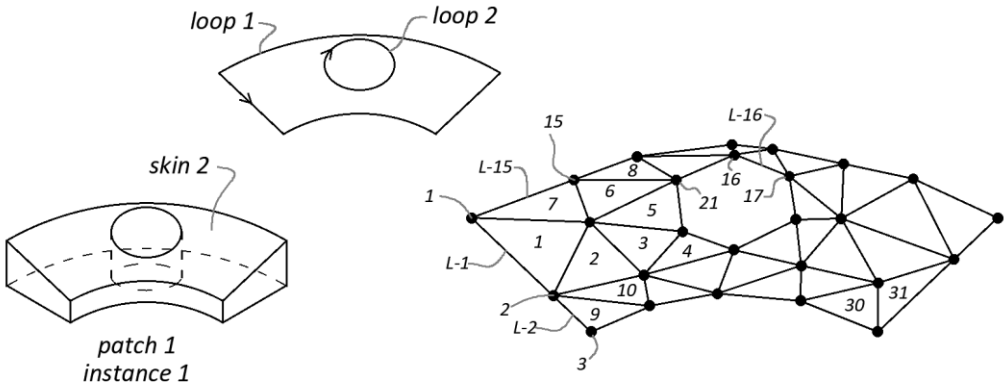


Fig. 3A.24 Influential volume of control points.

Skin (cPskin)

Properties	**Description**
`mNE`	Number of triangles
`mNV`	Number of vertexes
`mTVcon(mNE,*)`	Connectivity matrix: Element vs Vertexes
`mVTcon(mNV,*)`	Connectivity matrix: Vertex vs Elements
`mLVcon(*,*)`	Connectivity matrix: Line vs Vertexes. In case of mid-vertex it appears at the second position
`mVLcon(mNV,*)`	Connectivity matrix: Vertex vs Lines
`mTLcon(mNE,3)`	Connectivity matrix: Element vs Lines
`mVP(mNV,1)`	`cVP` class
`mNlo`	Number of loops that contour the skin
`mNlv(mNlo,1)`	Number of vertexes per loop (the first vertex is not repeated)

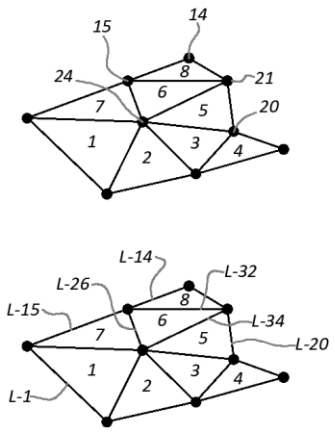
Fig. 3A.25 provides one example of `cPskin`, where the numbers of some of the nodes are given. The first numbered nodes corresponds to the contour loops. **Fig. 3A.26** shows one example with the connectivity matrices of one `cPskin`.




```
Spatches(1).INS(1).skin(2).mNE = 31
Spatches(1).INS(1).skin(2).mNV = 26

Spatches(1).INS(1).skin(2).mNlo = 2
Spatches(1).INS(1).skin(2).mNlv = [15 6]
```

Fig. 3A.25 Numbering of nodes in skins.



mTVcon=

...	
20 21 24	row 5
21 15 24	row 6
...	

mVLcon=

...	
14 15 26 32	row 15
...	

mVTcon=

...	
6 7 8	row 15
...	
1 2 3 5 6 7	row 24
...	

mTLcon=

...	
26 32 34	row 6
...	

mLVcon=

...	
20 21	row 20
...	
15 21	row 32
...	

Fig. 3A.26 Connectivity matrices in skins.

mGP (cGPpa) and mVP (cVP)

Properties	Description
`mAcoor(3,1)`, `mXcoor(3,1)`	Coordinates in patch parameter and physical spaces
`mNicp`, `mRicp(mNicp,1)`	Number of influential control points from the patch on the point, and their references
`Rrel(mNicp,1)`	Basis functions values of the influential control points at Gauss point location
`dRpa(mNicp,udim)`, `dRfi(mNicp,xdim)`	Derivatives of `Rrel` with respect to parameter directions and physical directions
`dJ1`	Determinant of first Jacobian at the Gauss point location
`dJ2a`	Aggregated value from all elements where the Gauss point is of determinant of second Jacobian times Gauss point weight
`mD(6,6)`	Constitutive matrix
`mSv(6,1),mVM`	Stresses vector and Von Mises stress
`mPst(3,1),mPdi(3,3)`	Principal stresses and principal directions (one per column)

3A.6 Boundary surface class

Fig. 3A.27 shows all the components of the class `cBentities`. One object is referred with its followed by the class name in brackets. For example `Bentities (cBentities)`.

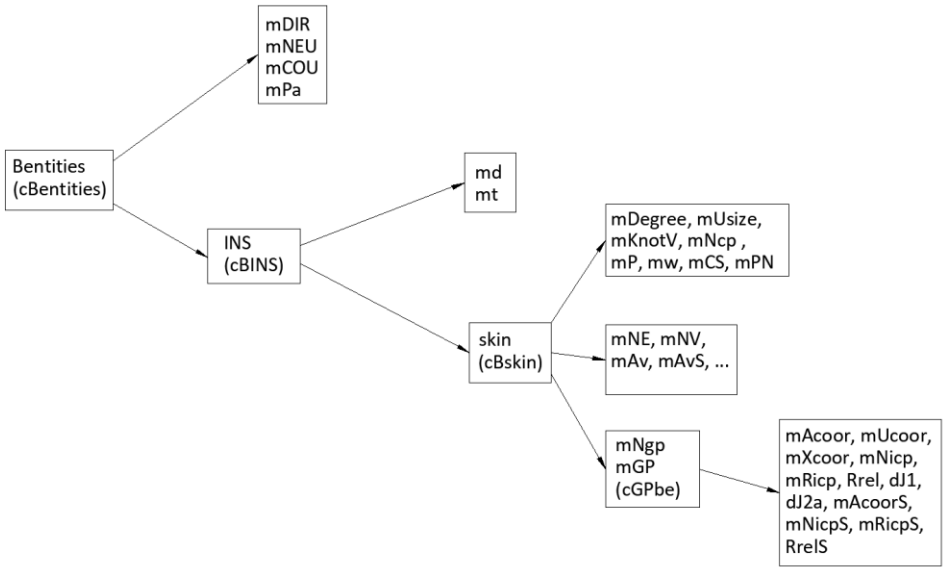


Fig. 3A.27 Components of `Bentities` object.

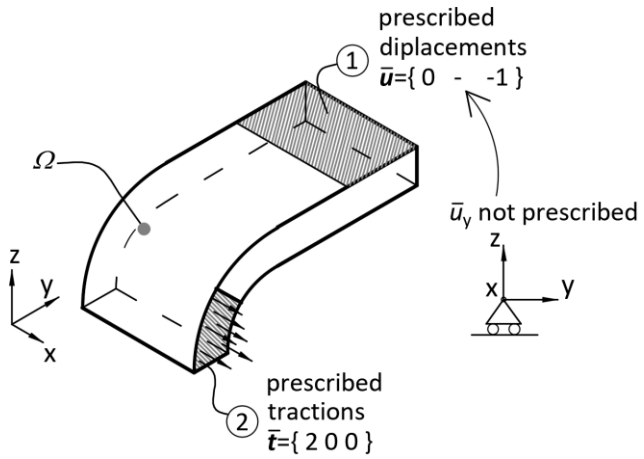
cBentities(Nbe)

Properties	Description
`mDIR`	Flags, equal to 1 if Dirichlet or Neumann condition is attached to the boundary entity.
`mNEU`	
`mCOU`	Coupling flag. It contains the coupling reference. Otherwise, if it is not coupling equals 0
`mPa(2,1)`	Patch where the surface belongs to. In case of coupling surface there are two patches stored
`INS(Nts,1)`	`cBINS` class

INS (cBINS)

Properties	Description
`md(xdim,1)`	Prescribed displacements (Dirichlet). In case of not prescribed displacement, the value is `nan`
`mt(xdim,1)`	Prescribed tractions (Dirichlet)
`skin`	`cBskin` class

One example of `cBentities` and its first instance is given in **Fig. 3A.28**. There are two boundary surfaces, one with Dirichlet and Neumann boundary conditions. In case there is not prescribed displacement in the Dirichlet condition, the component of the vector is to be `nan` (*not a number*).




```
cBentities(1)
       .mDIR = 1
       .INS.md=(0 nan -1)
       .INS.skin
```

```
cBentities(2)
       .mNEU = 1
       .INS.mt=(2 0 0)
       .INS.skin
```

Fig. 3A.28 Boundary conditions applied to boundary surfaces.

Skin (cBskin)

Properties	**Description**
`NURBS features`	
`mNE`	Number of triangles
`mNV`	Number of vertexes, with no mid-edge vertexes
`mAv(mNVm,3)`	Patch parameter space coordinates of vertexes (master patch)
`mAvS(mNVm,3)`	Patch parameter space coordinates of vertexes (slave patch)
`mUv(mNVm,2)`	Surface parameter space coordinates of vertexes
`mXv`	Physical space coordinates of vertexes
`mTVcon(mNE,*)`	Connectivity matrix: Element vs Vertexes
`mVTcon(mNV,*)`	Connectivity matrix: Vertex vs Elements
`mLVcon(*,*)`	Connectivity matrix: Line vs Vertexes. In case of mid-vertex it appears at the second position
`mVLcon(mNV,*)`	Connectivity matrix: Vertex vs Lines
`mTLcon(mNE,3)`	Connectivity matrix: Element vs Lines
`mNlo`	Number of contour loops of the NURBS surface
`mNCL(mNlo,1)`	Number of curves per contour loop of the NURBS surface
`mNlv(mNlo,1)`	Number of vertexes per contour loop. The final vertex, that coincides with the first vertex is not considered
`mNlcv(mNlo, mNCL)`	Number of vertexes per contour curve
`mNgp`	Number of Gauss points of the triangulated surface
`mGP(mNgp,1)`	`cGPbe` class
`mPa(2,1)`	Patch where the surface is attached to. Second one is slave patch in case of coupling
`mCou`	It is zero in case it is not a coupling surface.

mGP (cGPbe)

Properties	Description
`mAcoor(3,1)`	Patch parameter coordinates
`mUcoor(2,1)`	Surface parameter coordinates
`mXcoor(3,1)`	Physical parameter coordinates
`mNicp`	Number of influential control points from the patch on the Gauss point
`mRicp(mNicp,1)`	References of the influential control points
`Rrel(mNicp,1)`	Basis functions values of the influential control points at Gauss point location
`dJ1`	Determinant of first Jacobian at the Gauss point location
`dJ2a`	Aggregated value from all elements where the Gauss point is of determinant of second Jacobian times Gauss point weight
`mAcoorS(3,1)`	Same as `mAcoor` for slave patch
`mNicpS`	Same as `mNicp` for slave patch
`mRicpS(mNicpS,1)`	Same as `mRicp` for slave patch
`RrelS(mNicpS,1)`	Same as `Rrel` for slave patch

3A.7 Diagrams of blocks

This section presents the blocks diagrams for the main routines used in the algorithm. Not all the routines are presented but the most relevant, i.e. this is not an exhaustive list.

The routines are represented with a grey colour background meanwhile the variables are presented in a box with no fill (see examples in **Fig. 3A.29**).

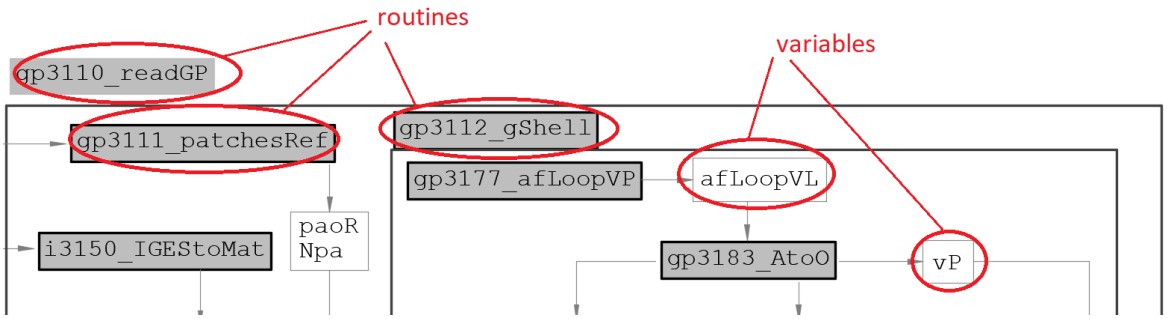


Fig. 3A.29 Example of routine and variable in the code schemes used in this section.

The first five diagrams show the main stages (A to E) of the algorithm. Some of the routines are marked with an asterisk, which indicates the routine is further detailed in blocks diagrams 6 to 12.

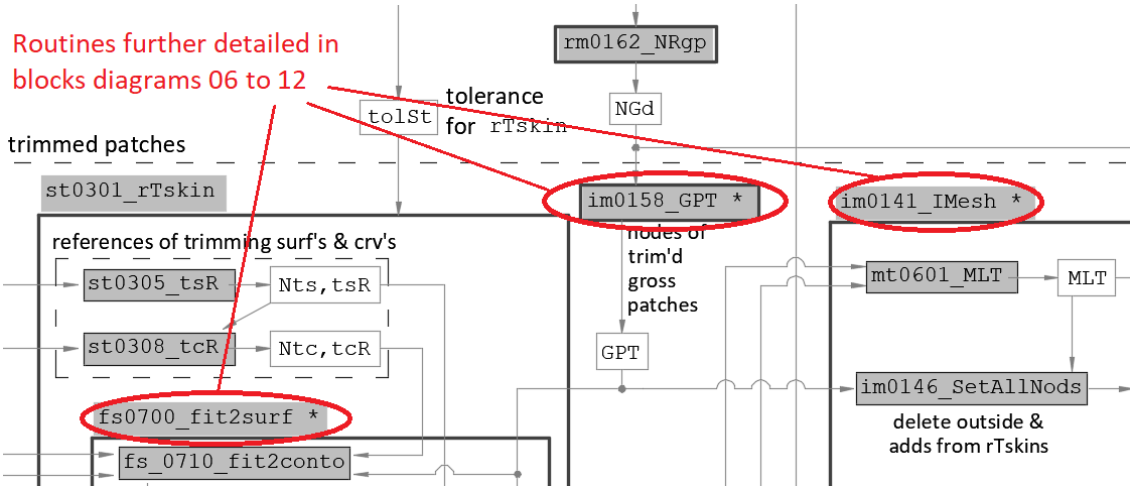
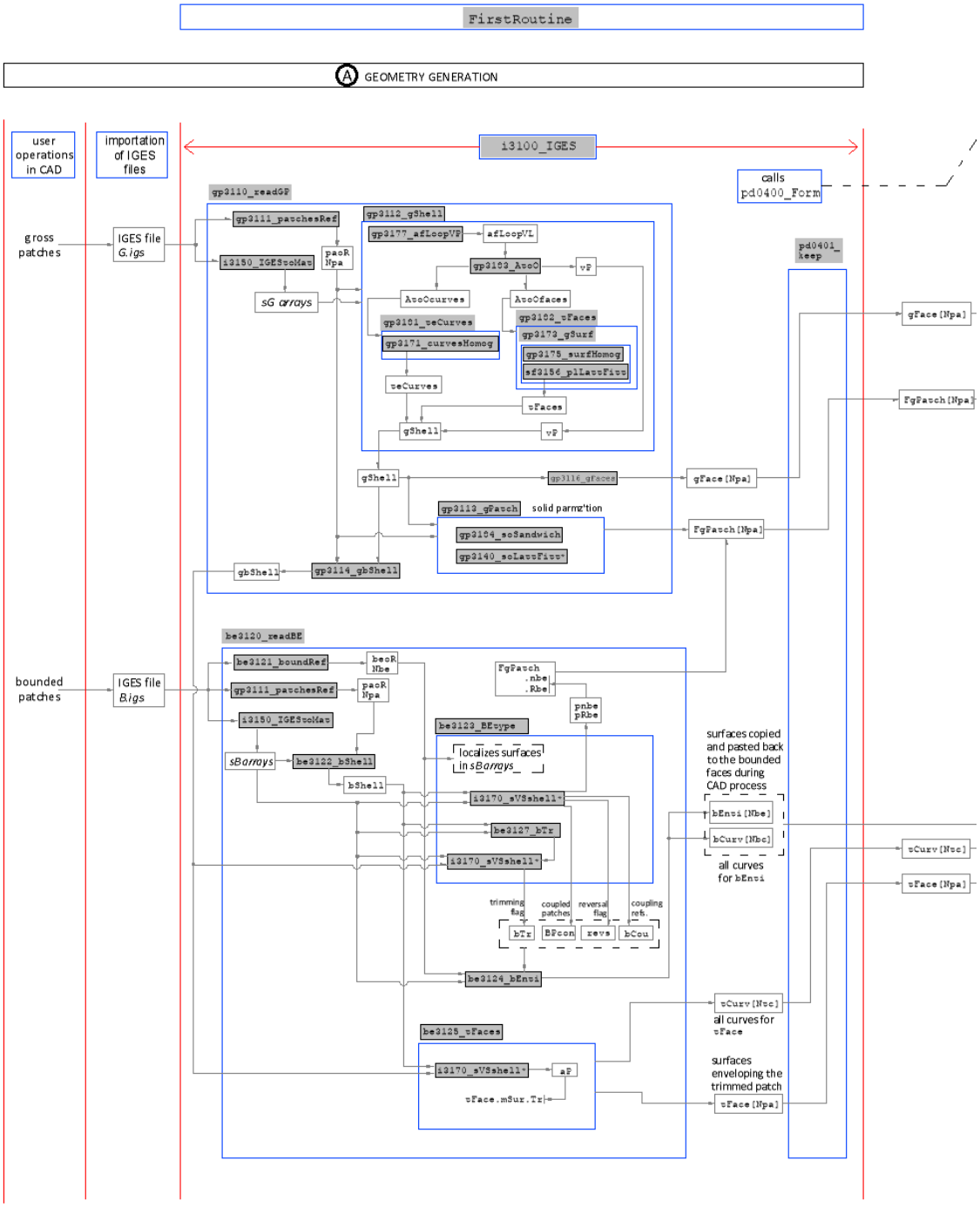


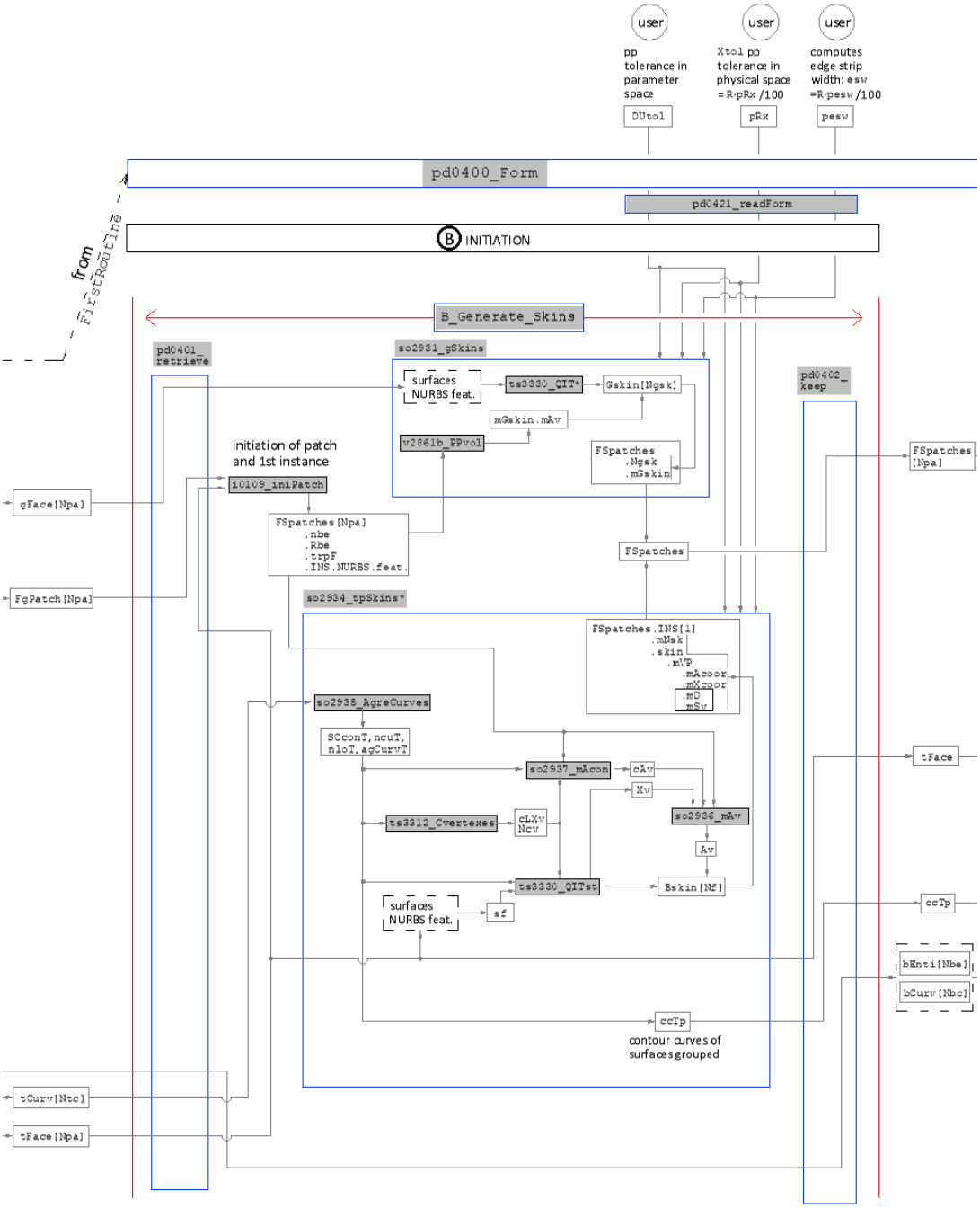
Fig. 3A.30 Asterisk indicates the routine is further detailed in another blocks diagram.

GIVEN THE POOR RESOLUTION OF THE BLOCKS DIAGRAMS, ADDITIONAL ANNEX IS SUBMITTED WITH THIS THESIS WHERE THIS DIAGRAMS ARE PRESENTED IN A MORE LEGILIBLE MANNER.

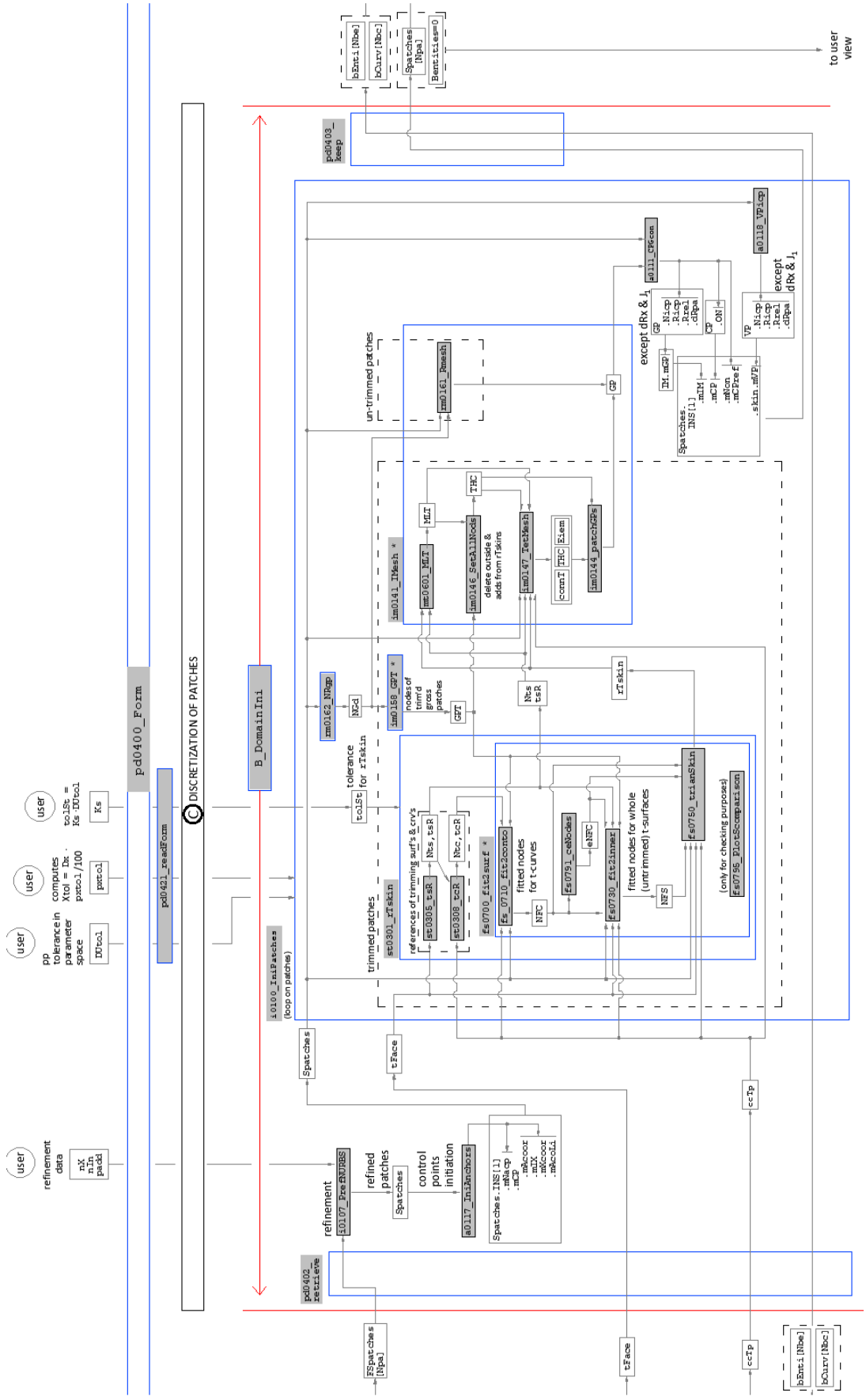
BLOCKS DIAGRAM 01



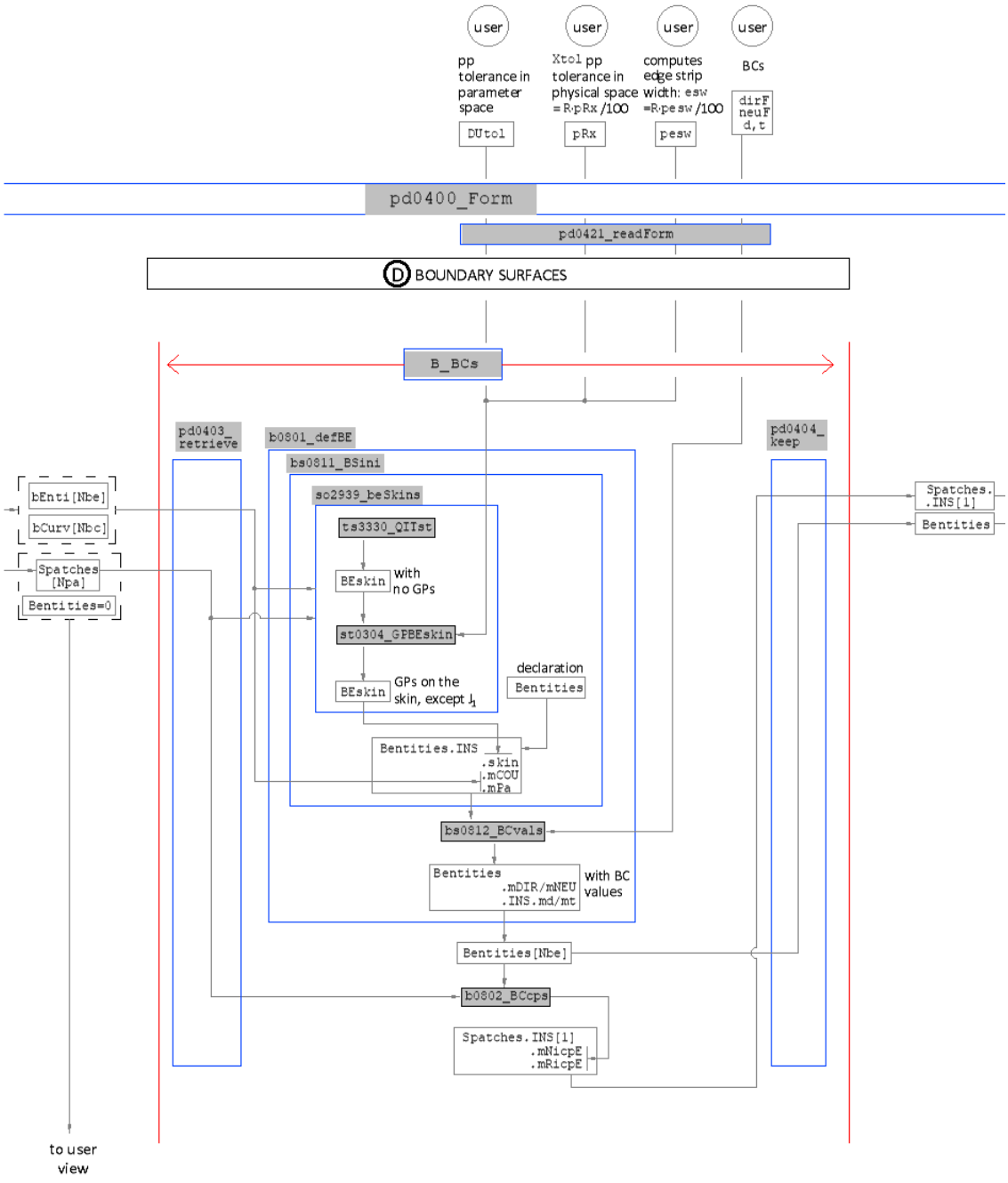
BLOCKS DIAGRAM 02



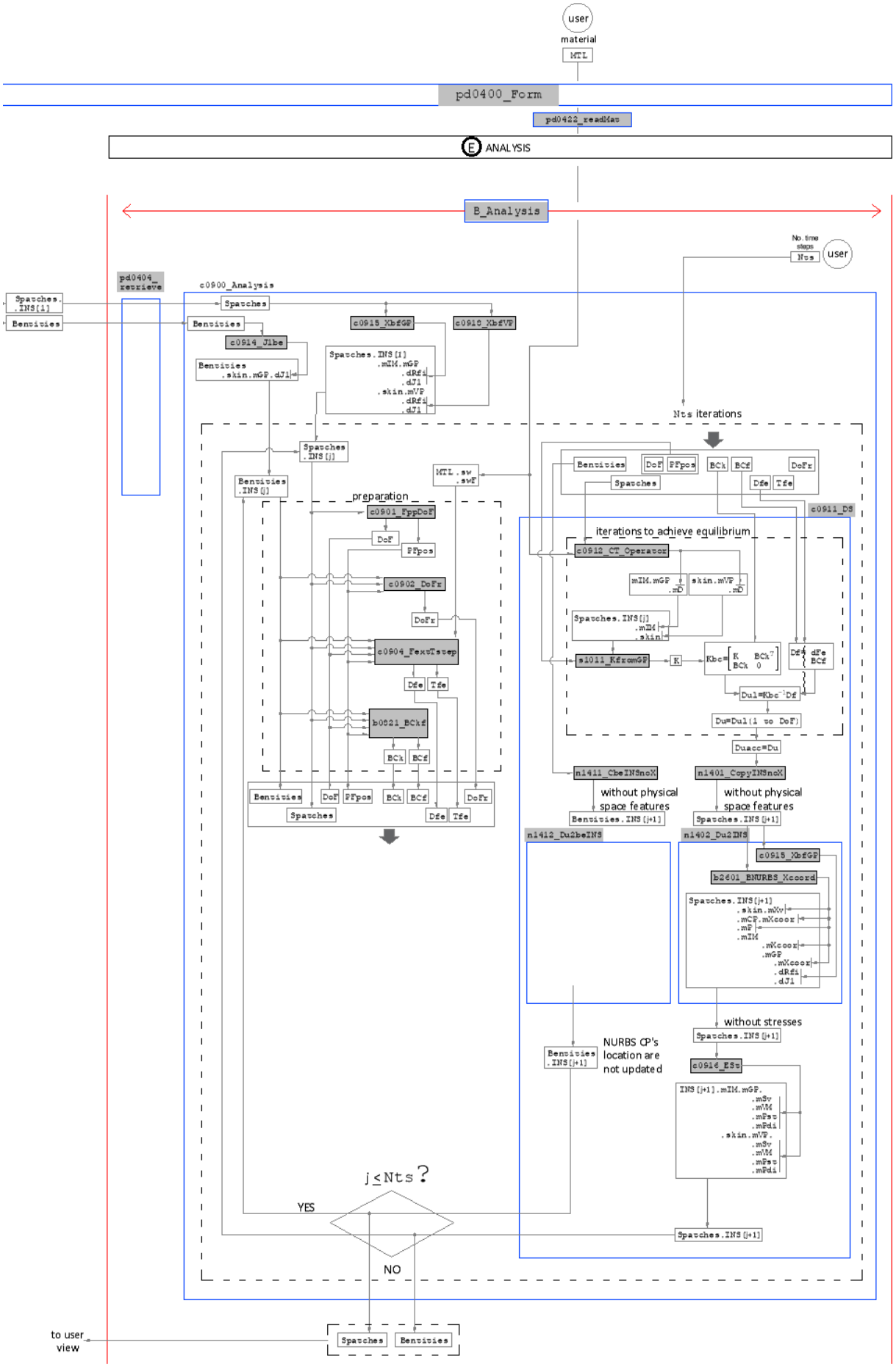
BLOCKS DIAGRAM 03



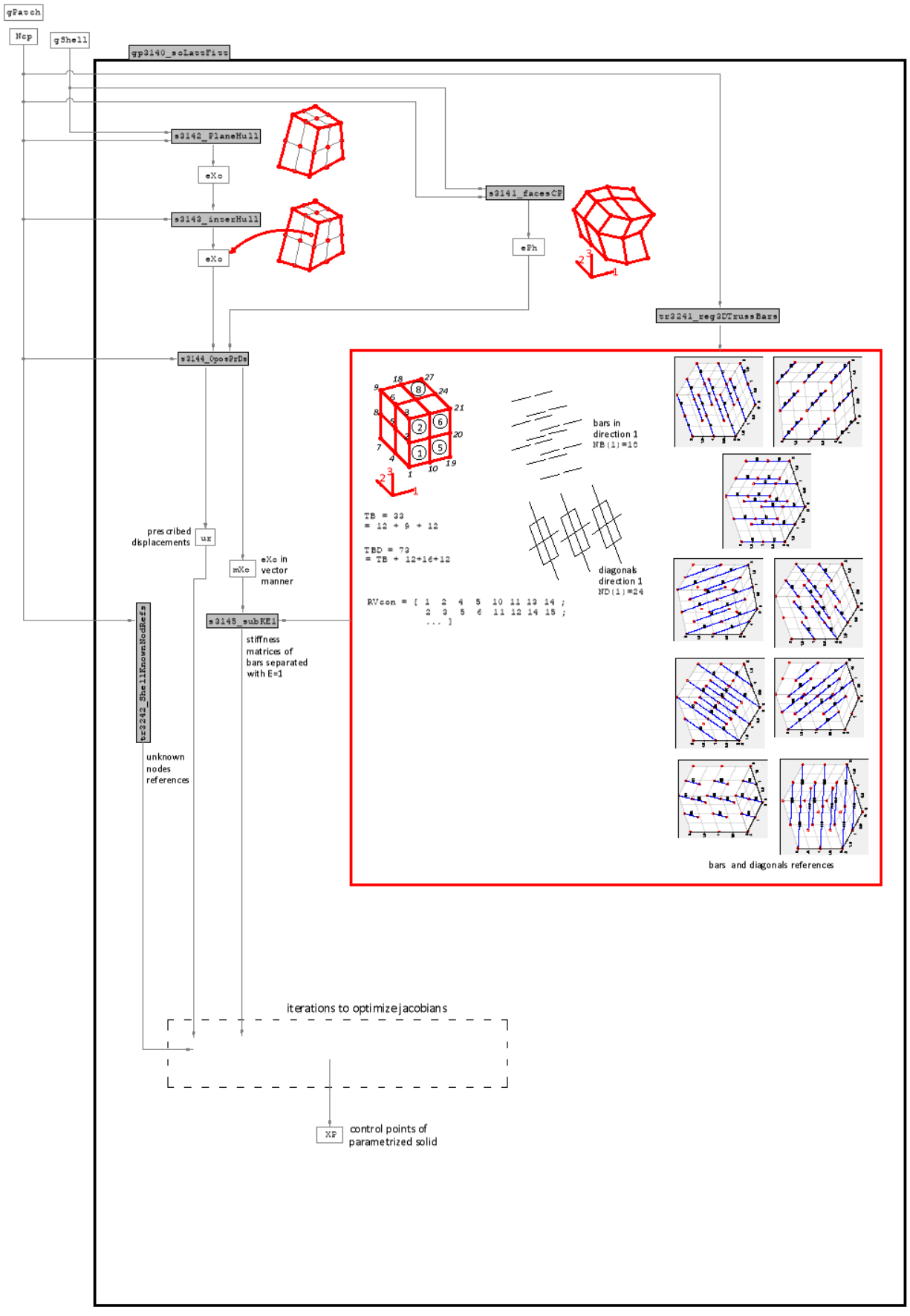
BLOCKS DIAGRAM 04



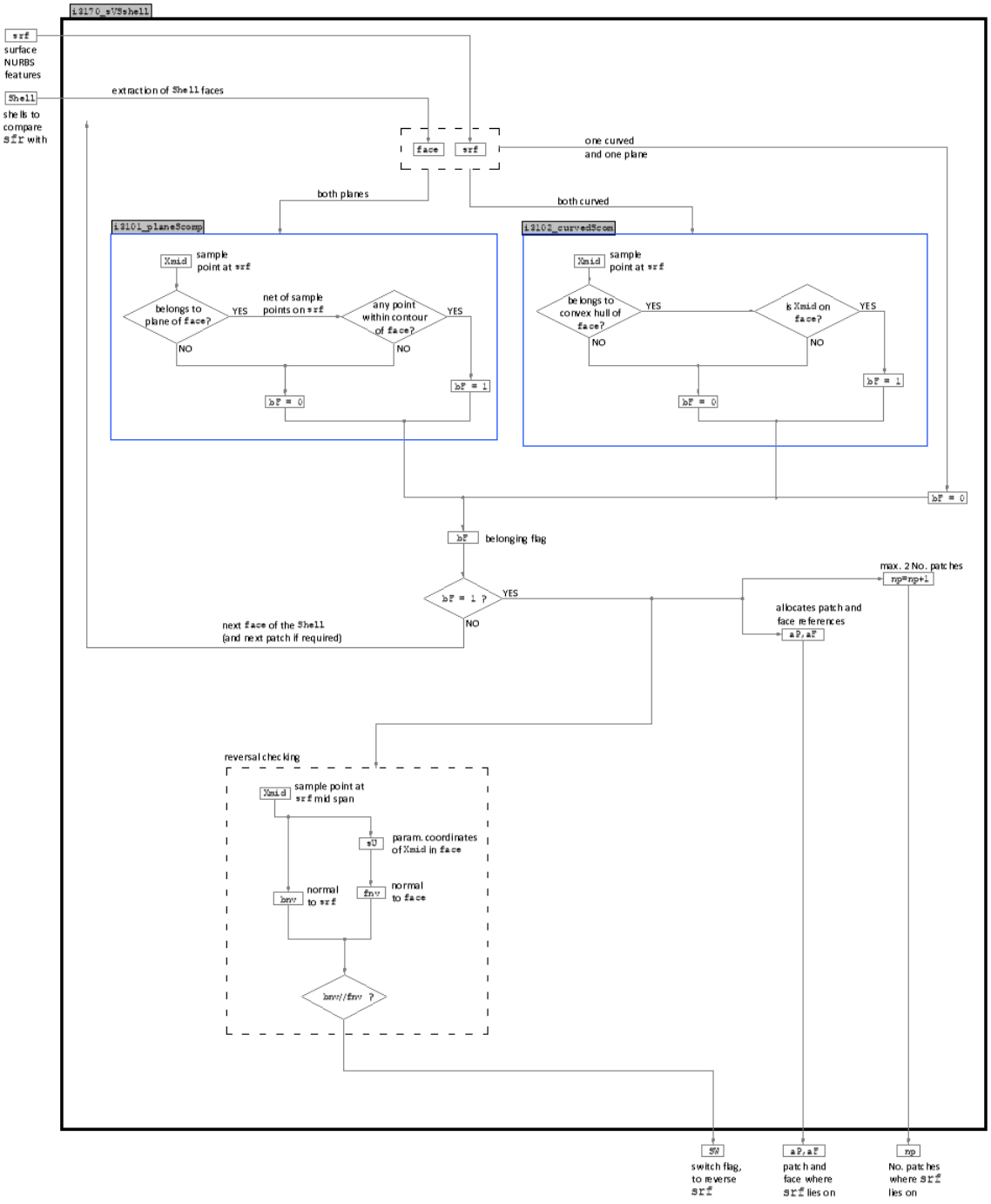
BLOCKS DIAGRAM 05



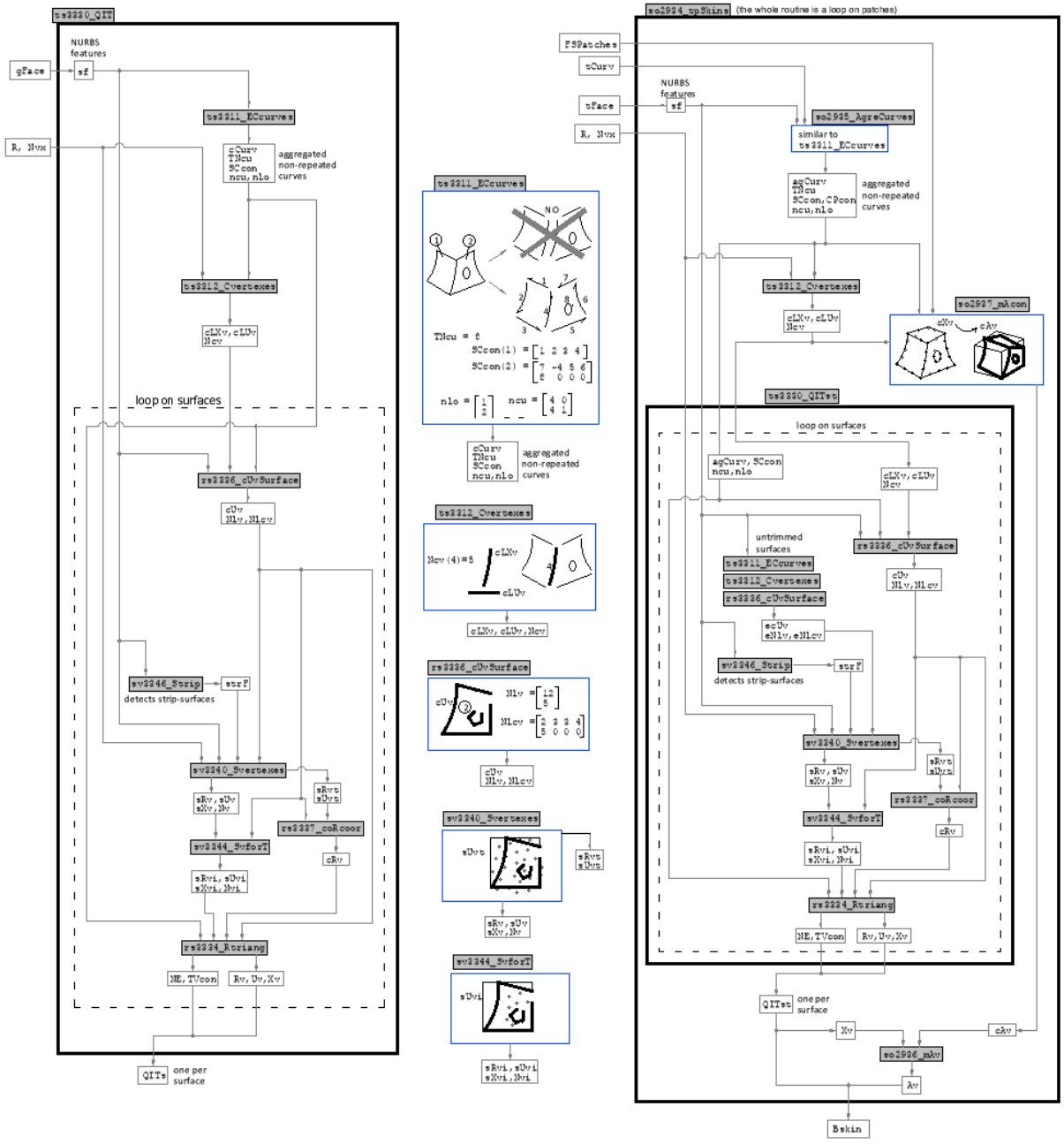
BLOCKS DIAGRAM 06



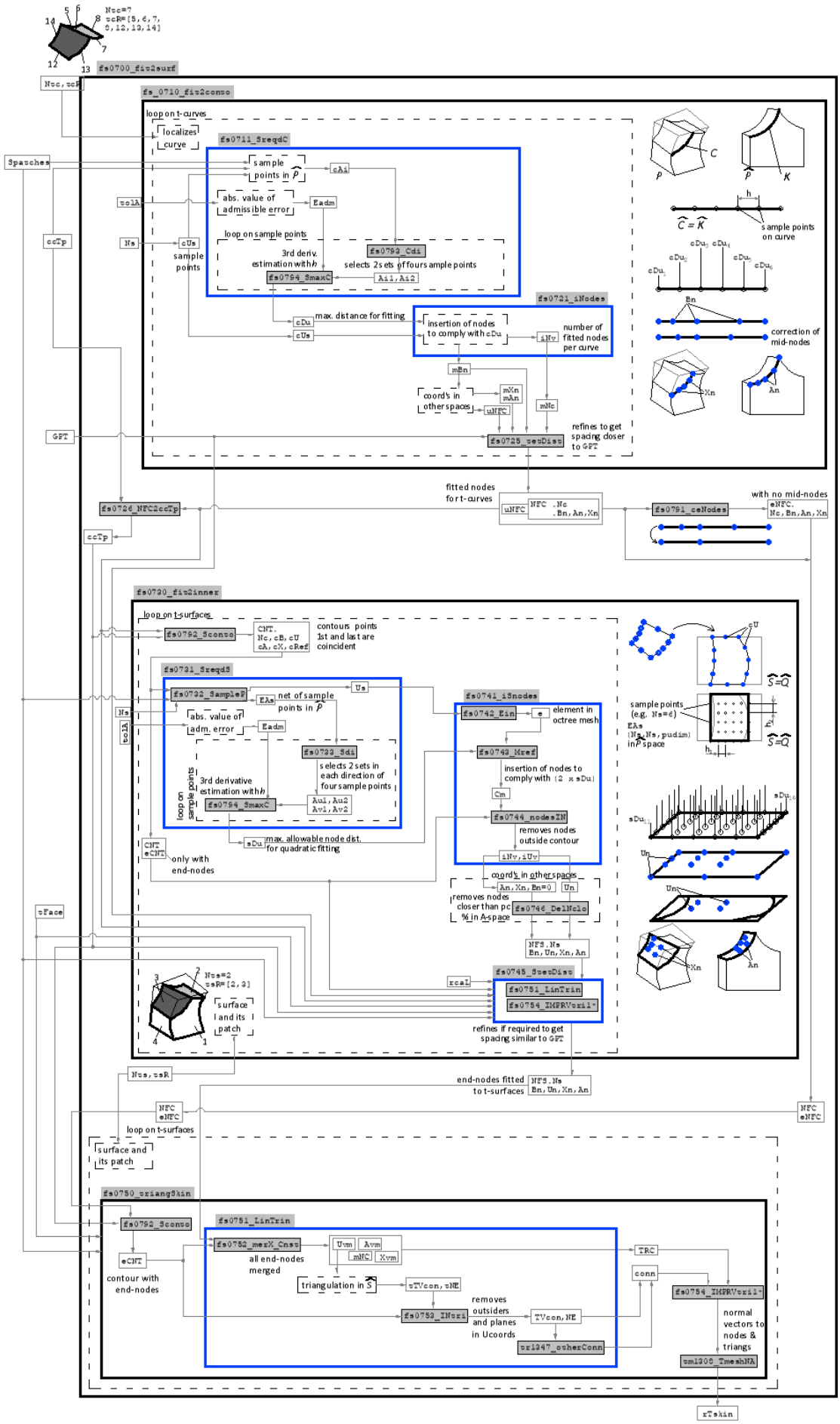
BLOCKS DIAGRAM 07



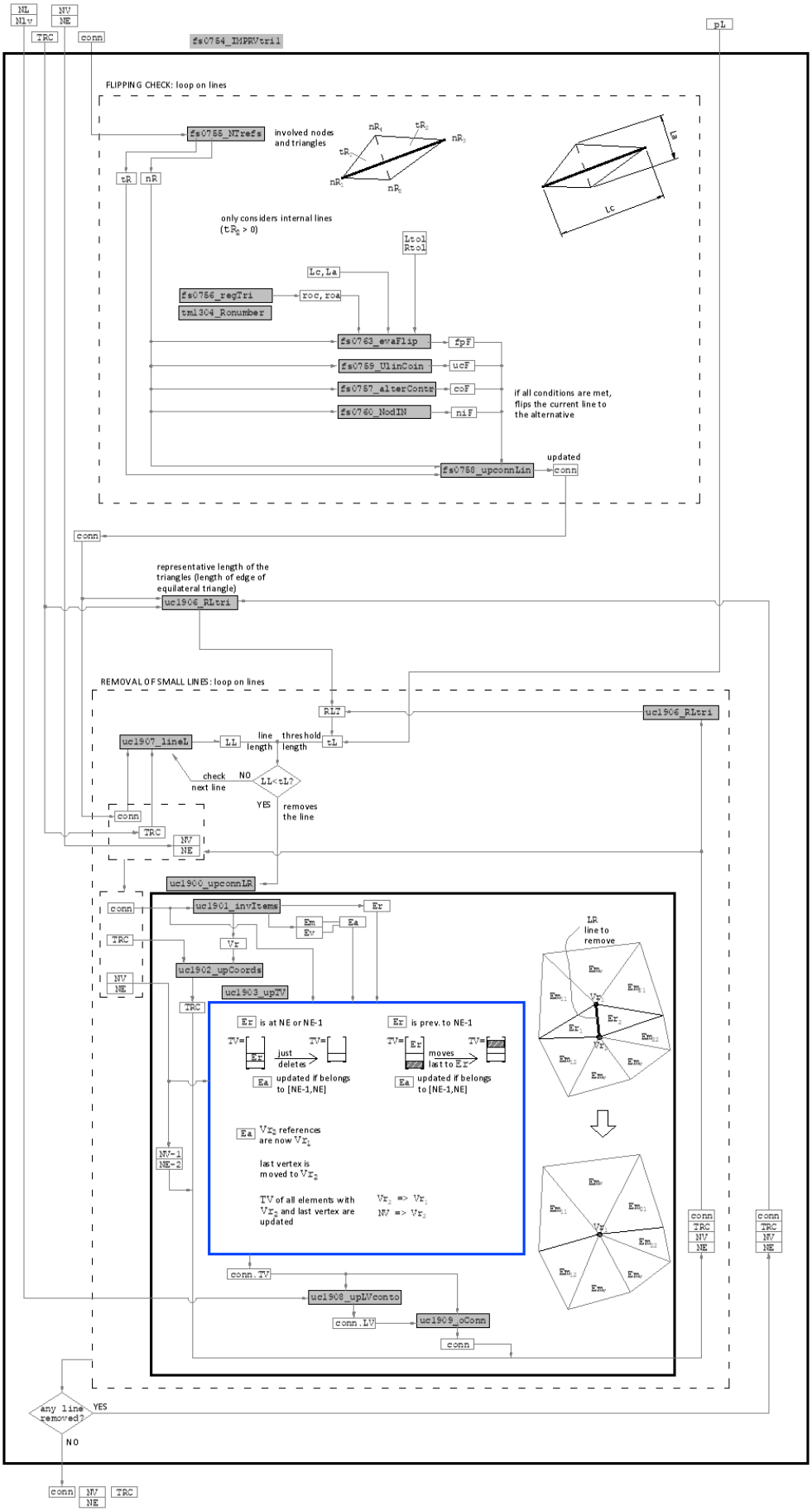
BLOCKS DIAGRAM 08



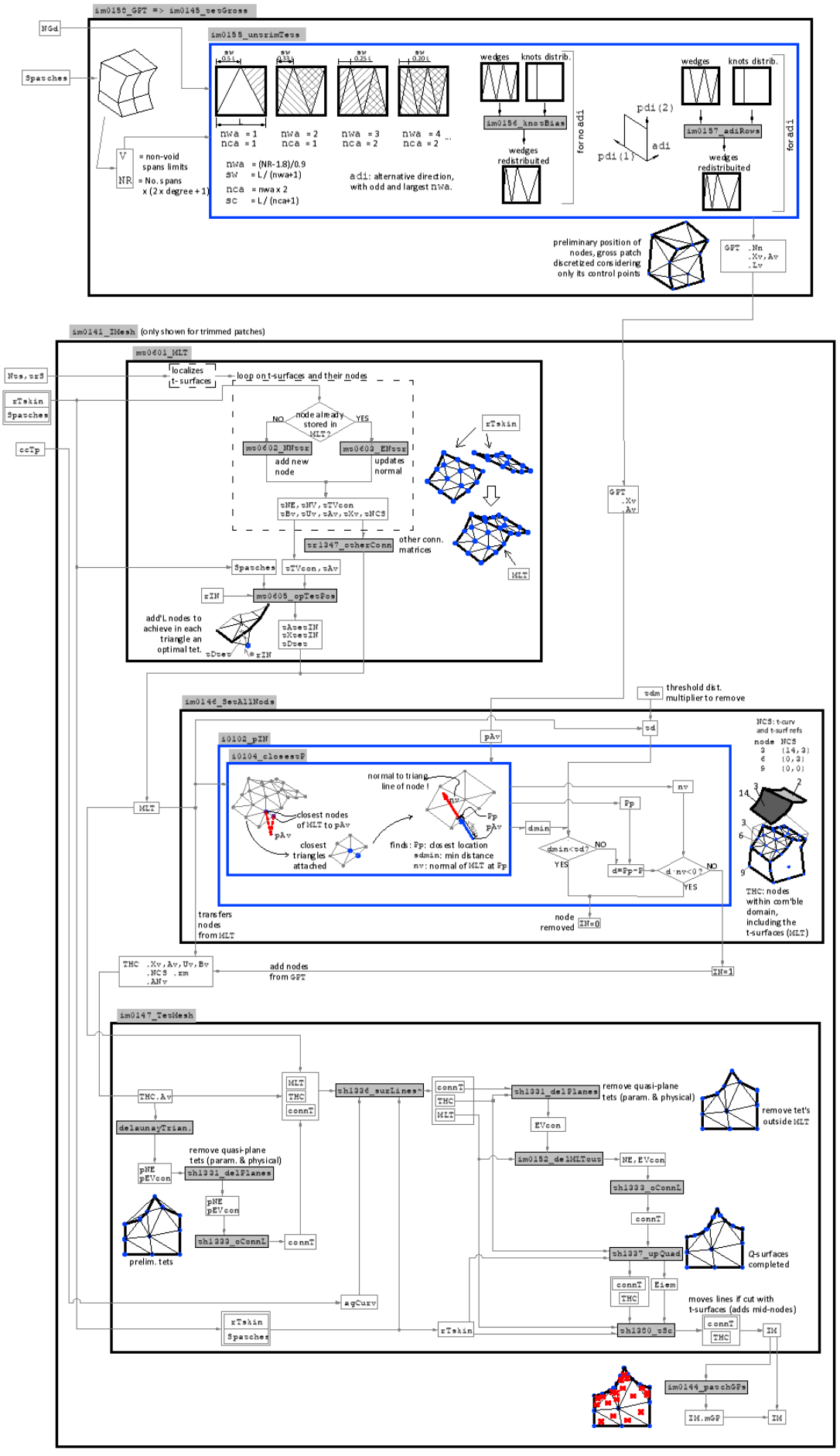
BLOCKS DIAGRAM 09



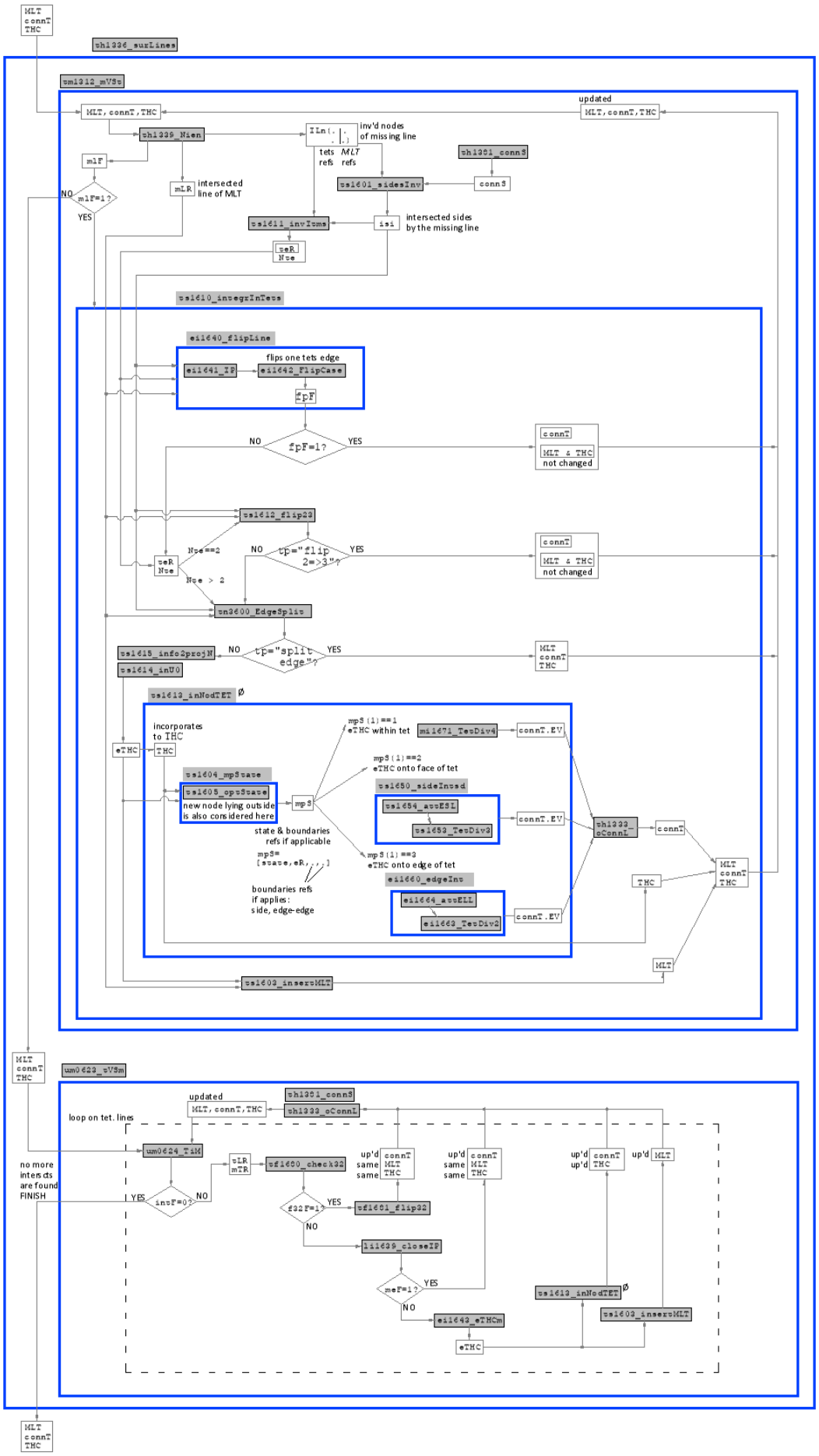
BLOCKS DIAGRAM 10



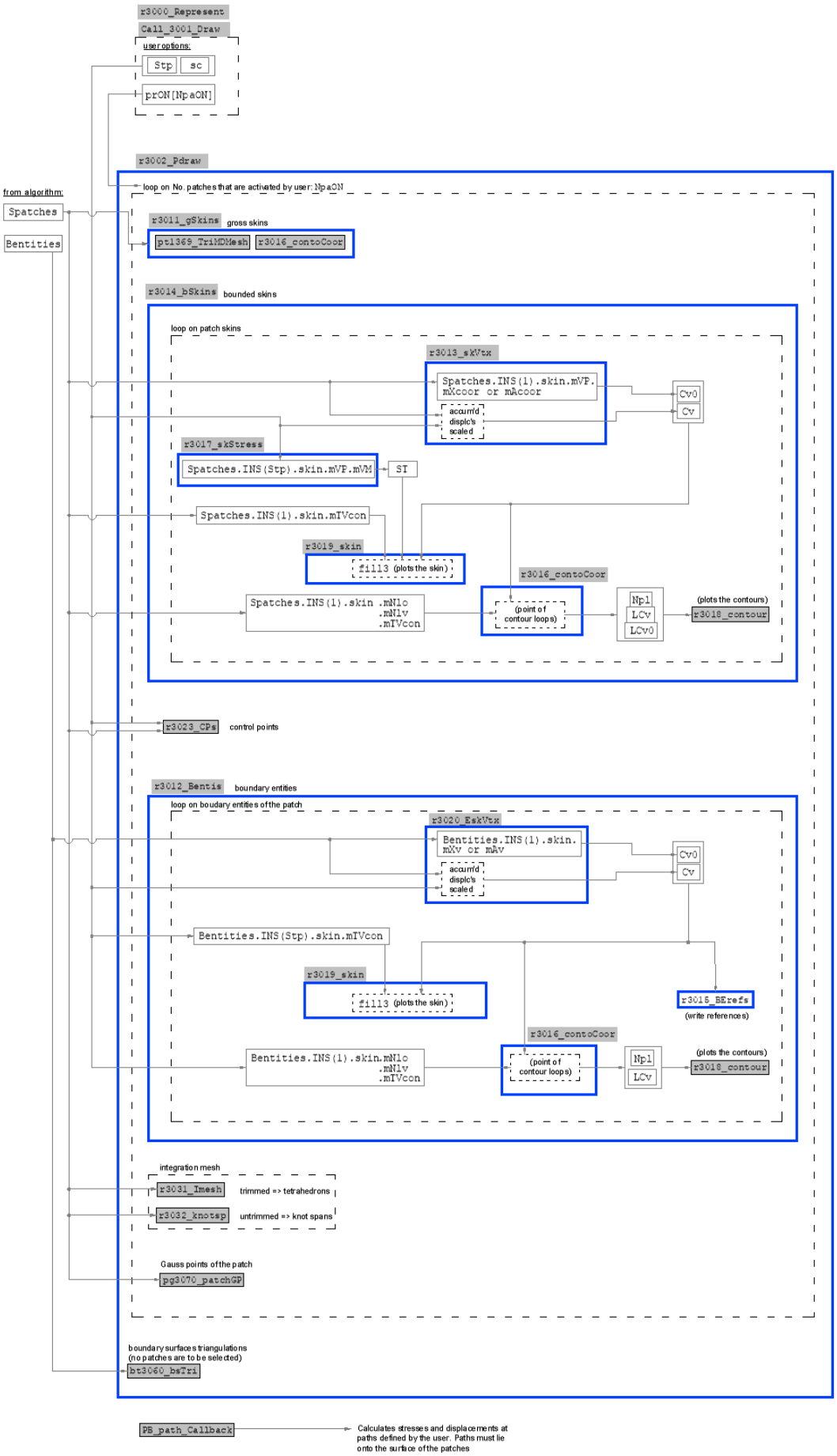
BLOCKS DIAGRAM 11



BLOCKS DIAGRAM 12



BLOCKS DIAGRAM 13



Appendix 3B: References to patches and boundary entities

3B.1 Allocation of references by the user

Patches and boundary surfaces must be numbered by the user in order to refer to them when applying boundary conditions. Both, patches and boundary entities, are numbered independently to each other with correlative natural numbers starting by 1. The user allocates this number via RBG colour code, during the CAD design of the geometry and before exporting the IGES files. That code is given in CAD by three numbers that span from 0 to 255, e.g. 201,155, 80. Only the first number is used to allocate the patch or boundary reference. The rule applied in this work is 200 + reference.

The only restraint to consider is that the patches references must coincide between gross and bounded patches. These reference numbers allocated by the user might differ from the order established by CAD when drawing the domain (IGES file order). However, the algorithm re-arranges patches and boundary entities to follow the user´s allocation.

One example with two patches and seven boundary surfaces is provided in **Fig. 3B.1**. The figure shows the trimmed patches with numbers 1 and 2. Let us remark that these numbering must match the gross patches. The first number of the RGB colour code is to be 201 and 202, as shown. The other two numbers of each code are not relevant for the algorithm. In **Fig. 3B.1** the boundary surfaces are separated from the trimmed patches for clarity, but let us recall that the bounded patches have the boundary surfaces ON THEM, to be suitable to export the IGES files and create the *B.igs*.

Same procedure is followed to number boundary surfaces. Hence their first number of the RGB code goes from 201 to 207 as illustrated in **Fig. 3B.1**.

3B.2 Retrieving references in the algorithm

Patches references (`paoR`) and their number (`Npa`) are returned by `gp3111_patchesRef` function. In a similar manner, boundary surfaces references (`beoR`) and their number (`Nbe`) are returned by `be3121_boundRef` function. Both functions work in the same manner, that are explained below for boundary surfaces case. For patches the explanation is also applicable.

In general, the order of boundary surfaces in IGES files is not the same as the order given by the user. The array `beoR` links both references, that in reality is a pointer: to know the IGES position of one boundary surface whose user reference is *i*, one enters in `beoR` at the *i*th row reads the reference in the IGES files. **Fig. 3B.2** illustrates one example graphically and **Fig. 3B.3** in IGES file.

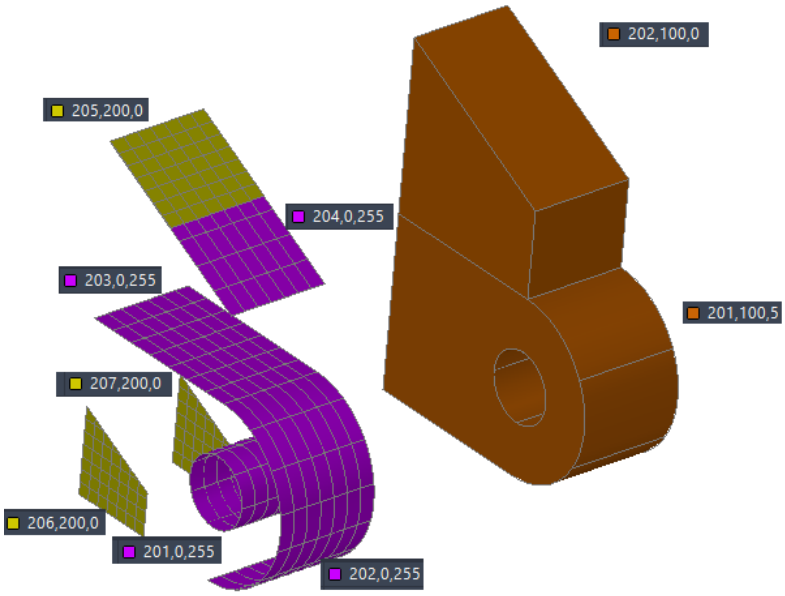
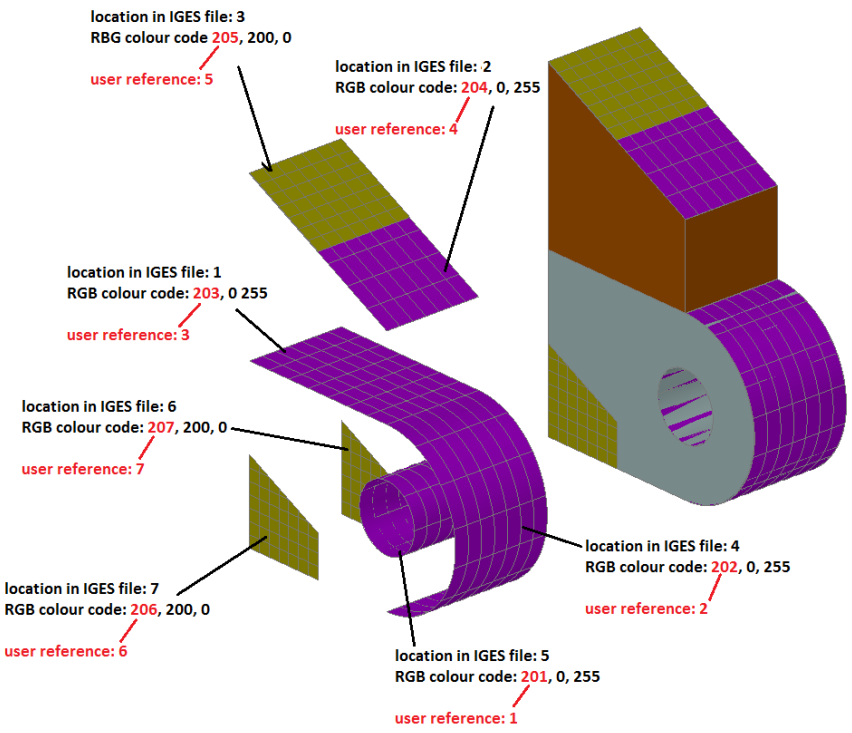


Fig. 3B.1 Allocation of references for boundary surfaces (left) and for trimmed patches (right).

In **Fig. 3B.3** one can observe seven boundary surfaces, therefore `Nbe=7`. Their *colour addresses* (recall section 3.1) in *parameter data section* are circled in red. In that section, the colour code first number indicates the user's reference. As stated in section 3.1, the colour code is the percentage of 255. Then, for example the first BE in IGES file corresponds to the third user's reference, as next computation:

$$\frac{79.60784}{100}255 \approx 203 \rightarrow user's\ reference = 203 - 200 = 3$$



$$\text{beoR} = \begin{Bmatrix} 5 \\ 4 \\ 1 \\ 2 \\ 3 \\ 7 \\ 6 \end{Bmatrix}$$

Fig. 3B.2 Example of user reference BS allocation and `beoR` array.

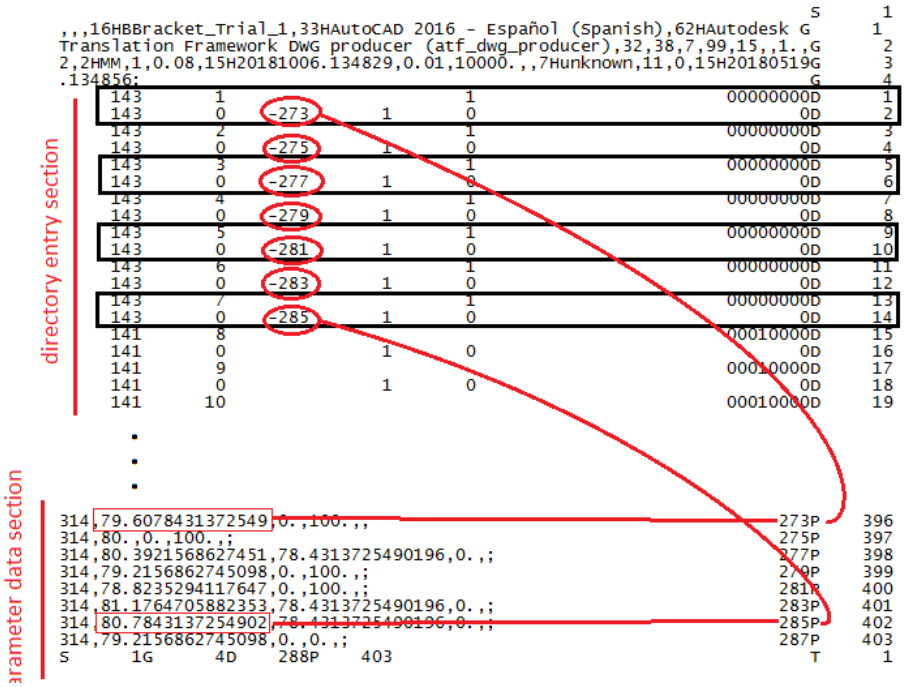


Fig. 3B.3 Identification of number of boundary surface, their colour addresses and the first colour number.

Appendix 4A: Direct stiffness method with prescribed displacements

Let Λ be a truss made of bars that only work under axial loads, tension or compression in $\mathbb{R}^d$. For plane structures $d = 2$ and for three dimensional $d = 3$. The connection between rods are designated as nodes. Let $\boldsymbol{K}$ be the stiffness matrix, $\boldsymbol{f}$ the nodal force vector and $\boldsymbol{u}$ de nodal displacement vector of Λ, that are related as per equation (4A.1).

$$\boldsymbol{f} = \boldsymbol{K}\,\boldsymbol{u} \tag{4A.1}$$

The ith node has d degrees of freedom, then its displacement may be represented as $\boldsymbol{u}^i = \left(u_1^i, \dots u_d^i\right)^T$. Each rod has a stiffness matrix in local axis $\boldsymbol{k}'$ as (4A.2) and (4A.3) for plane and three-dimensional trusses respectively.

$$\boldsymbol{k}' = \begin{bmatrix} +EA/_L & 0 & -EA/_L & 0 \\ 0 & 0 & 0 & 0 \\ -EA/_L & 0 & +EA/_L & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4A.2}$$

$$\boldsymbol{k}' = \begin{bmatrix} EA/L & 0 & 0 & -EA/L & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -EA/L & 0 & 0 & EA/L & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4A.3}$$

E, A and L are the elastic modulus, the cross-sectional area and the length of each rod. The rotation matrix of each rod is computed as (4A.4) and (4A.5) for plane and three-dimensional trusses respectively.

$$\boldsymbol{R}^T = \begin{bmatrix} cos(\alpha_{x'x}) & cos(\alpha_{y'x}) & & \\ cos(\alpha_{x'y}) & cos(\alpha_{y'y}) & & \\ & & cos(\alpha_{x'x}) & cos(\alpha_{y'x}) \\ & & cos(\alpha_{x'y}) & cos(\alpha_{y'y}) \end{bmatrix} \tag{4A.4}$$

$$\boldsymbol{R}^T = \begin{bmatrix} cos(\alpha_{x'x}) & cos(\alpha_{y'x}) & cos(\alpha_{z'x}) & & & \\ cos(\alpha_{x'y}) & cos(\alpha_{y'y}) & cos(\alpha_{z'y}) & & & \\ cos(\alpha_{x'z}) & cos(\alpha_{y'z}) & cos(\alpha_{z'z}) & & & \\ & & & cos(\alpha_{x'x}) & cos(\alpha_{y'x}) & cos(\alpha_{z'x}) \\ & & & cos(\alpha_{x'y}) & cos(\alpha_{y'y}) & cos(\alpha_{z'y}) \\ & & & cos(\alpha_{x'z}) & cos(\alpha_{y'z}) & cos(\alpha_{z'z}) \end{bmatrix} \quad (4A.5)$$

Where $cos(\alpha_{i'j})$ refers to the cosine from local axis i' to global axis j. $\boldsymbol{k}'$ is transformed to global axis $\boldsymbol{k}$ as per equation (4A.6). Matrix $\boldsymbol{K}$ is obtained by assembling matrices of rods in global axis. The location of $\boldsymbol{k}$ within $\boldsymbol{K}$ is given by the rod nodes references.

$$\boldsymbol{k} = \boldsymbol{R}^T \boldsymbol{k}' \boldsymbol{R} \quad (4A.6)$$

To prescribe displacements to some of the nodes one needs to operate with submatrices. The stiffness matrix and vectors can be subdivided as shown in equation (4A.7). Sub-indexes a and b refer to un-known and known displacements respectively. Since $\boldsymbol{u}_b$ are the displacements to be enforced and are known, the subsystem expressed in equation (4A.8) allows to compute the unknowns $\boldsymbol{u}_a$. In the absence of forces, the unknown displacement vector is computed as (4A.9).

$$\begin{Bmatrix} \boldsymbol{f}_a \\ \boldsymbol{f}_b \end{Bmatrix} = \begin{bmatrix} \boldsymbol{K}_{aa} & \boldsymbol{K}_{ab} \\ \boldsymbol{K}_{ba} & \boldsymbol{K}_{bb} \end{bmatrix} \begin{Bmatrix} \boldsymbol{u}_a \\ \boldsymbol{u}_b \end{Bmatrix} \quad (4A.7)$$

$$\boldsymbol{f}_a = \boldsymbol{K}_{aa}\boldsymbol{u}_a + \boldsymbol{K}_{ab}\boldsymbol{u}_b \quad (4A.8)$$

$$\boldsymbol{u}_a = \boldsymbol{K}_{aa}{}^{-1} - \boldsymbol{K}_{ab}\boldsymbol{u}_b \quad (4A.9)$$

The displacements of all nodes of Λ are then given by the vectors $\boldsymbol{u}_b$ (prescribed) and $\boldsymbol{u}_a$ (computed).

Appendix 4B: Least squares method

4B.1 Fitting a bi-dimensional line

Let us define set of m points and the line $\boldsymbol{l}$ that passes through the point $\boldsymbol{A} = (a, b)$, all in $\mathbb{R}^2$. The line is expressed parametrically as (4B.1), where $\boldsymbol{v}$ is the unit length vector that provides the direction and t the free parameter.

$$\boldsymbol{l}(t) = \boldsymbol{A} + \boldsymbol{v}\, t \tag{4B.1}$$

The error at each point is defined as the distance from the line to that point, measured orthogonally to the line. The direction $\boldsymbol{v}$ that minimizes the errors is given by the first eigenvector of the matrix $\boldsymbol{L}$:

$$\boldsymbol{L} = \delta\, \boldsymbol{I} - \begin{bmatrix} \sum_{i=1}^{m} (x_i - a)^2 & \sum_{i=1}^{m} (x_i - a)(y_i - b) \\ \sum_{i=1}^{m} (x_i - a)(y_i - b) & \sum_{i=1}^{m} (y_i - b)^2 \end{bmatrix} \tag{4B.2}$$

Where δ is:

$$\delta = \sum_{i=1}^{m} (x_i - a)^2 + \sum_{i=1}^{m} (y_i - b)^2 \tag{4B.3}$$

And (x_i, y_i) are the coordinates of the ith point.

4B.2 Fitting a three-dimensional plane

Let us define set of m points and the plane $\boldsymbol{\pi}$ that passes through the point $\boldsymbol{A} = (a, b, c)$, all in $\mathbb{R}^3$. The plane is expressed as (4B.4), where $\boldsymbol{n}$ is the unit length vector that provides the orientation of the plane and $\boldsymbol{a}$ is any point on the plane.

$$\boldsymbol{n} \cdot (\boldsymbol{A} - \boldsymbol{a}) = 0 \tag{4B.4}$$

The error at each point is defined as the distance from the plane to that point, measured orthogonally to the plane. The normal direction $\boldsymbol{n}$ that minimizes the errors is given by the first eigenvector of the matrix $\boldsymbol{P}$:

$$\boldsymbol{P} = \sum_{i=1}^{m} \begin{bmatrix} (x_i - a)^2 & (x_i - a)(y_i - b) & (x_i - a)(z_i - c) \\ (x_i - a)(y_i - b) & (y_i - b)^2 & (y_i - b)(z_i - c) \\ (x_i - a)(z_i - c) & (y_i - b)(z_i - c) & (z_i - c)^2 \end{bmatrix} \quad \text{(4B.5)}$$

Where (x_i, y_i, z_i) are the coordinates of the ith point.

Appendix 5A: Isotropic triangulation and measure of isotropy

One triangulation is isotropic if all its angles are 60 degrees and all nodes are attached to six triangles. **Fig. 5A.1** illustrates the isotropic triangulation.

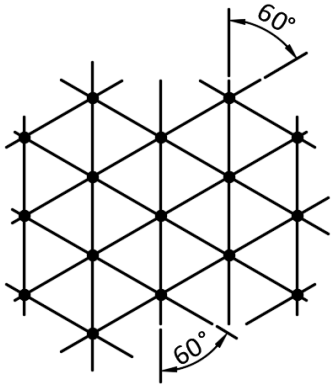


Fig. 5A.1 Isotropic triangulation.

The **angles** of the triangles and the number of triangles attached to each node, called **valence**, are used to measure the isotropy of the triangulation. The closer the angles to 60 degrees and the valences to 6, the more isotropic is the triangulation. Only infinite triangulation (with no borders) can be purely isotropic. When the isotropic triangulation is contoured angles different from 60 degrees and valences different from six appear as illustrated in **Fig. 5A.2**.

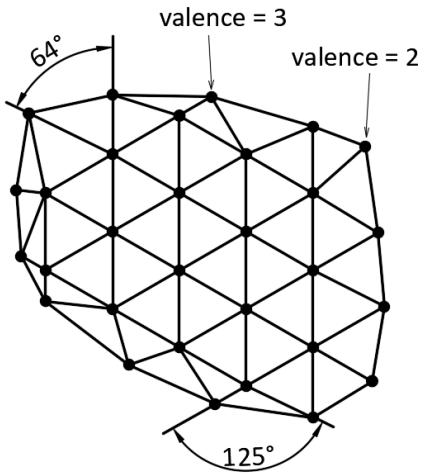


Fig. 5A.2 Quasi-isotropic triangulation.

When one contoured triangulation has all the angles equal to 60 degrees and the valences to 6 except at those triangles in the vicinity of the contours it is called quasi-isotropic triangulation (see **Fig. 5A.2**).

Appendix 5B: Measurement of triangles distortion

5B.1 Linear triangles

The distortion of one linear triangle may be measured by comparison with the equilateral triangle with same area. The number to compare is the ratio (5B.1).

$$r_A = \frac{A_C}{A} \tag{5B.1}$$

Where A_C is the area of the inscribed circle and A the area of the triangle itself. For one equilateral triangle, this ratio is called r_A^e and always approximately equal to 0.606. For other triangles r_A is lower: the more distorted the triangle the lower value of r_A. The relation given in (5B.2), between one triangle r_A and the equilateral, is used to characterize the triangle.

$$\rho = \frac{r_A}{0.606} \tag{5B.2}$$

If $\rho = 1$ the triangle is equilateral, *i.e.* it has the highest quality possible. Otherwise the value of ρ decreases as the distortion raises.

To compute the ratio r_A, one needs the area of the triangle and its inscribed circle. The former is half the norm of the cross product of two vectors corresponding to two edges. For the area of the inscribed circle one first calculates the bisectors of two angles and their intersection, *i.e.* the incentre. Then the distance from the incentre to any of the triangle edges is the radius of the inscribed circle, which allows obtaining its area. **Fig. 5B.1** shows three examples.

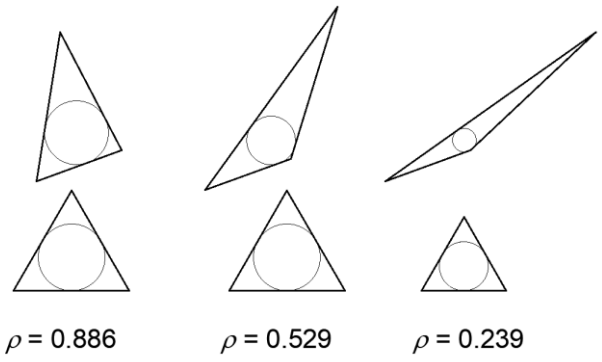


Fig. 5B.1 Example of ρ number for three different cases. Below each case it is represented the equilateral version (with same area than the triangle above).

5B.2 Quadratic triangles

Quadratic triangles have nodes at the vertexes (numbered as 1,2,3) and nodes at the edges mid locations (numbered as 4,5,6), see **Fig. 5B.2**. Each edge of the triangle is a quadratic curve of length L_{q_i}, where i varies between 1 and 3. This arrangement leads to a non-plane nodes distribution that impedes the comparison via inscribed circumference (as per linear triangles). One sphere should be used instead. However finding the sphere is a non-linear problem that results computationally expensive, therefore another approach is presented here.

The reference to compare with is the edge length of the equivalent equilateral triangle (L_{eq}). The equilateral triangle has the same area as the current triangle (A_{tv}) considering only nodes at vertexes (1,2,3), as shown in **Fig. 5B.2**.

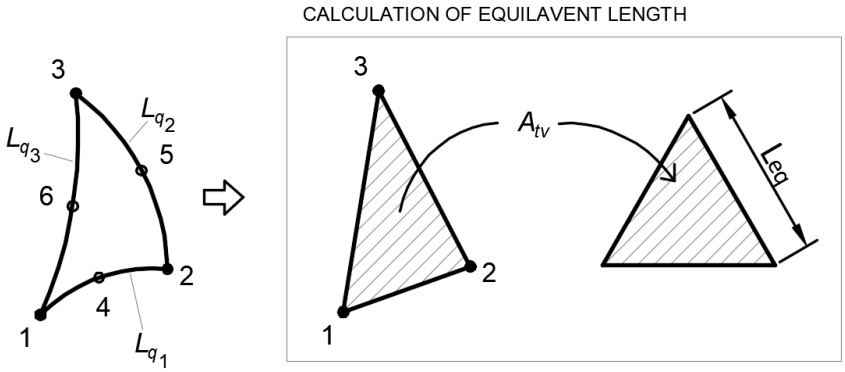


Fig. 5B.2 Numbering of nodes of quadratic triangle and construction of equivalent equilateral triangle.

The length L_{eq} is obtained as (5B.3), which comes from equating the area A_{tv} of the equilateral triangle to the area of the current triangle defined between nodes at vertexes.

$$L_{eq} = 1.52\ {A_{tv}}^{1/2} \tag{5B.3}$$

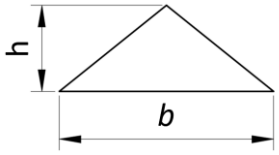
The variable that is measured is the coefficient of variation, in percentage, of the three quadratic edges lengths (L_{q_i}) with respect to the equivalent length (L_{eq}), as expressed in (5B.4).

$$CV = \frac{100}{L_{eq}} \left(\frac{\sum_{i=1}^{3} \left(L_{q_i} - L_{eq} \right)^2}{3} \right)^{1/2} \tag{5B.4}$$

For triangles with low distortion, *i.e. closer* to equilateral, CV is near to zero. The larger the distortion the larger CV. Two examples are provided further on, comparing the variation of CV varying one aspect parameter. In **Fig. 5B.3** the height is changed, from equilateral triangle to

smaller heights, getting more distorted triangles. In consequence CV increases with such variation. In **Fig. 5B.4** one edge is curved, increasing its mid-node eccentricity from zero (equilateral triangle) to certain value. With such eccentricity increment the CV becomes larger.

b	h	A_{tv}	CV
10	8.66	43.30	0
10	4.00	20.00	28
10	2.00	10.00	63
10	1.00	5.00	119
10	0.50	2.50	203
10	0.25	1.25	324



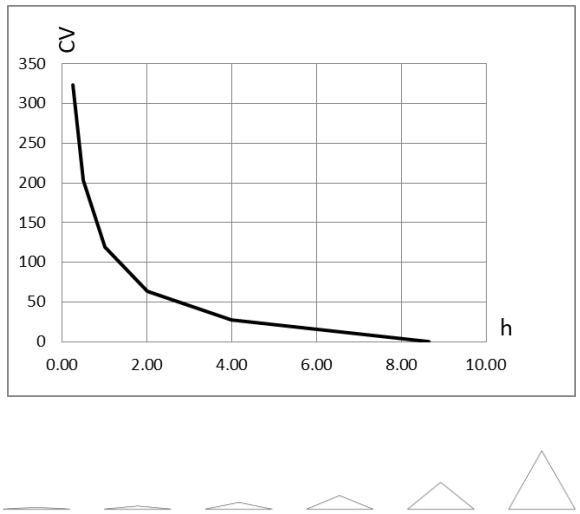
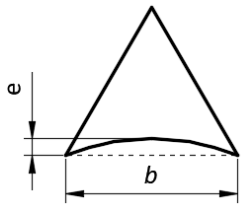


Fig. 5B.3 Variation of CV with triangle distortion (height variation).

b	e	L_{q1} (arc)	CV
10	0	10	0.0
10	0.25	10.02	0.2
10	0.5	10.07	0.7
10	1	10.27	2.7
10	2	11.03	10.3
10	4	13.83	38.3



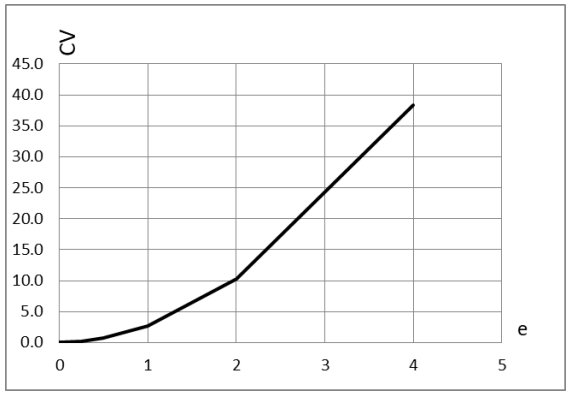


Fig. 5B.4 Variation of CV with edge curvature.

In view of **Fig. 5B.3** and **Fig. 5B.4** one can infer that the closer the triangle to the equilateral, *i.e.* the less distorted, the lower is the value of CV.

Appendix 5C: Comparison between linear and quadratic triangulation

This appendix shows the better performance of quadratic triangles versus the linear ones for approximating one NURBS surface S. Performance in this context refers to number of nodes required to define the approximated surface, called $\bar{S}$, such that the error is equal or less than admissible value E_{adm}.

To compare both triangulations, linear and quadratic, next assumptions are made (see **Fig. 5C.1**):

- the surface parameter domain is $[0,1]\otimes[0,1]$;
- the triangulation of $\bar{S}$ is isotropic, except at edges;
- the number of triangles in each direction is n and m, being the triangles arranged in n columns as illustrated in **Fig. 5C.1**;
- the side of each triangle is equal to 1.16 the width of the columns, *i.e.* $\Delta v \approx 1.16\Delta u$;
- the third derivative is one order of magnitude larger than the second, i.e. $\frac{S'''}{S''} = 10$;
- the error is measured only in one dimension of S (d=1).

Fig. 5C.1 illustrates one triangulation under these assumptions with 5 x 8 triangles. In vertical direction, the first and last triangles of each column are considered as one.

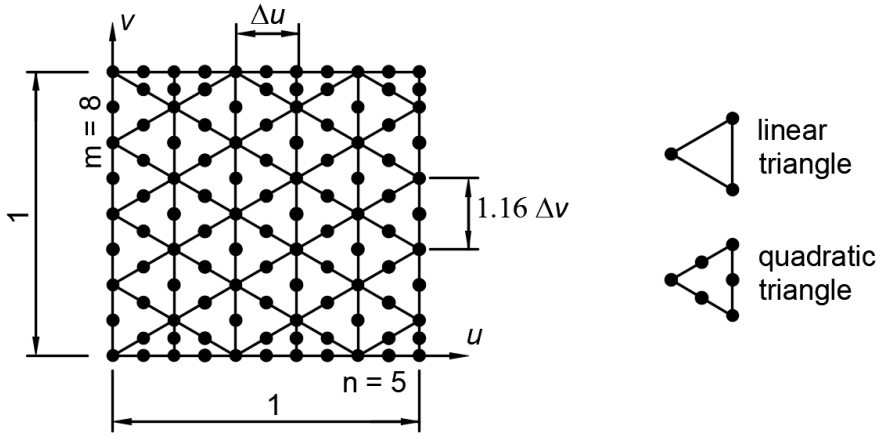


Fig. 5C.1 Triangulation with 5 x 8 triangles.

The number of nodes (N), given the number of triangles in each direction *n* and *m*, for linear and quadratic triangles are calculated as equations (5C.1) and (5C.2) respectively (one can deduct these expressions by looking at **Fig. 5C.1**).

$$N_1 = (\mathrm{m}-1)(\mathrm{n}+1) \tag{5C.1}$$

$$N_2 = (m-1)(n+1) + (m-2)(n+1) + (m+2)(n+1) \tag{5C.2}$$

The error in one parameter direction of S with respect to $\bar{S}$ is as per equations (5C.3) and (5C.4), for linear and quadratic approximations respectively. These equations are obtained from Appendix 2A, considering $S'' = 1$ (therefore $S''' = 10$).

$$E_1 \leq 0.125\, {h_1}^2 \tag{5C.3}$$

$$E_2 \leq 0.642\, {h_2}^3 \tag{5C.4}$$

Where h is the nodal spacing. Note that the line length is $L_1 = h_1$ and $L_2 = 2\, h_2$ for linear and quadratic lines respectively (see **Fig. 5C.2**).

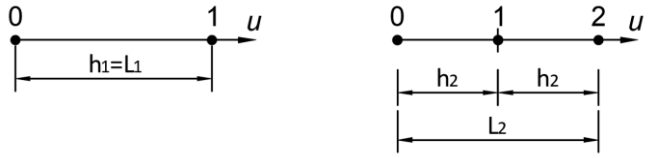


Fig. 5C.2 Line length and nodal spacing for linear and quadratic approximation.

The key of the quadratic efficiency lies in the exponent of h, that makes the nodal spacing to increase faster in quadratic interpolation when the error decreases. Let us fix the error to a value E_{adm}. The nodal distances to suit that error with linear and quadratic triangles may be calculated from equations (5C.3) and (5C.4), which yield expressions (5C.5) and (5C.6).

$$h_1 \approx 2.83\, {E_{adm}}^{1/2} \tag{5C.5}$$

$$h_2 \approx 1.16\, {E_{adm}}^{1/3} \tag{5C.6}$$

The number of triangles in each direction is equal to the span, whose length is one, divided by the increment (linear) or twice the increment (quadratic), as shown in equations (5C.7) to (5C.10).

$$n_1 \approx \frac{1}{\frac{2.83}{1.16}\, {E_{adm}}^{1/2}} \approx \frac{0.41}{{E_{adm}}^{1/2}} \tag{5C.7}$$

$$m_1 \approx \frac{1}{2.83\, {E_{adm}}^{1/2}} \approx \frac{0.35}{{E_{adm}}^{1/2}} \tag{5C.8}$$

$$n_2 \approx \frac{1}{2\,\frac{1.16}{1.16}\,{E_{adm}}^{1/3}} \approx \frac{0.50}{{E_{adm}}^{1/3}} \tag{5C.9}$$

$$m_2 \approx \frac{1}{2\ \ 1.16\,{E_{adm}}^{1/3}} \approx \frac{0.43}{{E_{adm}}^{1/3}} \tag{5C.10}$$

With the number of triangles in each direction at hand, the number of nodes for linear and quadratic cases may be computed as equations (5C.1) and (5C.2). **Fig. 6.C3** shows the number of required nodes for both cases varying the fixed error from $E_{adm} = 0.0001$ to 0.005 (that corresponds to percentages of $e = 0.01$ to 0.50 % of span). The number of nodes is less in the quadratic case, and the difference increases with the lesser value of E_{adm}.

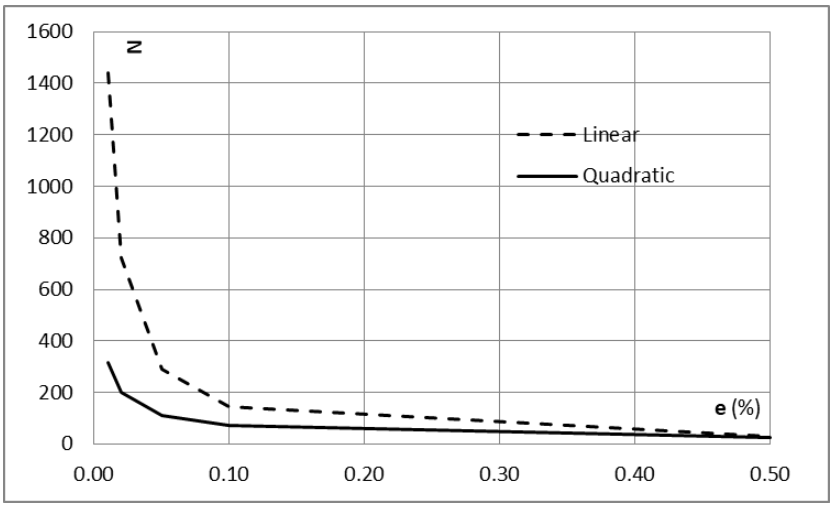


Fig. 5C.3 Number of nodes required for linear and quadratic triangles versus admissible error in percentage(**e**).

Linear and quadratic triangulations assuming error E_{adm}<0.0005 ($e = 0.05$ %) are depicted in **Fig. 5C.4**. By visual inspection one notice why the quadratic approximation is more efficient. Although this explanation made simplifications, it shows that quadratic triangulation tends to have less number of nodes than linear, for a fixed error.

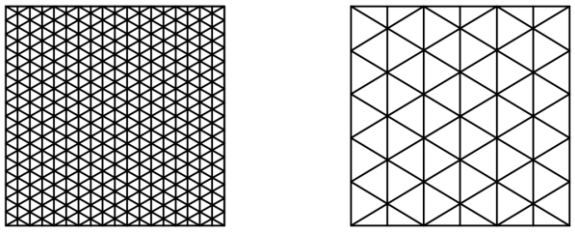


Fig. 5C.4 Triangulation for keeping the error $E_{adm} < 0.0005$, linear triangles (left) and quadratic (right).

Appendix 5D: Insertion of additional nodes

In the generation of trimming surfaces and the tetrahedral mesh, additional nodes to the initial set may be required, either to achieve quadratic degree (mid-nodes) or to refine/modify the tetrahedral mesh. The insertion of these nodes is not direct as explained in this appendix. Let us use the prefix Q and K for nodes and lines that lie on Q-surfaces and K-curves respectively, the rest of nodes / lines are may be preceded by the term *inner* (see **Fig. 5D.1**). Note that the K-nodes set is included in the Q-nodes set, *i.e.* K-nodes is a particular case of Q-nodes.

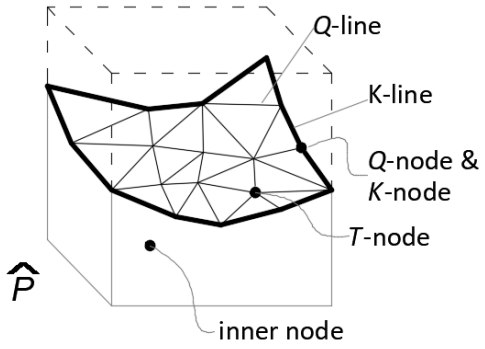


Fig 5D.1 Types of nodes and lines.

This appendix describes the computation of the coordinates of one additional inserted node in all the involved spaces: $\boldsymbol{x}_i$, $\boldsymbol{\xi}_i$, $\boldsymbol{u}_i$ and w_i. The node is preliminarily placed at the $\hat{P}$-space that is corrected to find its final coordinates. The process is different for inner, Q and K-nodes.

a. Inner nodes:

1. The position in the $\hat{P}$-space ($\boldsymbol{\xi}_i$) is given and coincides with the final position (there is no correction in this case).
2. The coordinates in the P-space are obtained by patch NURBS mapping ($\boldsymbol{x}_i$).

The coordinates in the $\hat{Q}$-space and the $\widehat{K}$-space ($\boldsymbol{u}_i$ and w_i) do not exist for inner nodes.

b. Q-nodes:

1. The position given in the $\hat{P}$-space is preliminary ($\boldsymbol{\xi}_p$).
2. Coordinates in the P-space are calculated by NURBS mapping ($\boldsymbol{x}_p$) from the $\hat{P}$-space.
3. $\boldsymbol{x}_p$ is point projected on the surface where the Q-node belongs, obtaining its parameter coordinate ($\boldsymbol{u}_i$).
4. The physical position is calculated by NURBS mapping of $\boldsymbol{u}_i$ from the $\hat{Q}$-space to the P-space ($\boldsymbol{x}_i$).
5. The position in the $\hat{P}$-space is obtained by point projection of $\boldsymbol{x}_i$ on the patch ($\boldsymbol{\xi}_i$).

Note that in general $\boldsymbol{x}_i \neq \boldsymbol{x}_p$ and $\boldsymbol{\xi}_i \neq \boldsymbol{\xi}_p$. The coordinates in the $\widehat{K}$-space (w_i) do not exist for Q-nodes. The process scheme is given in **Fig. 5D.2**.

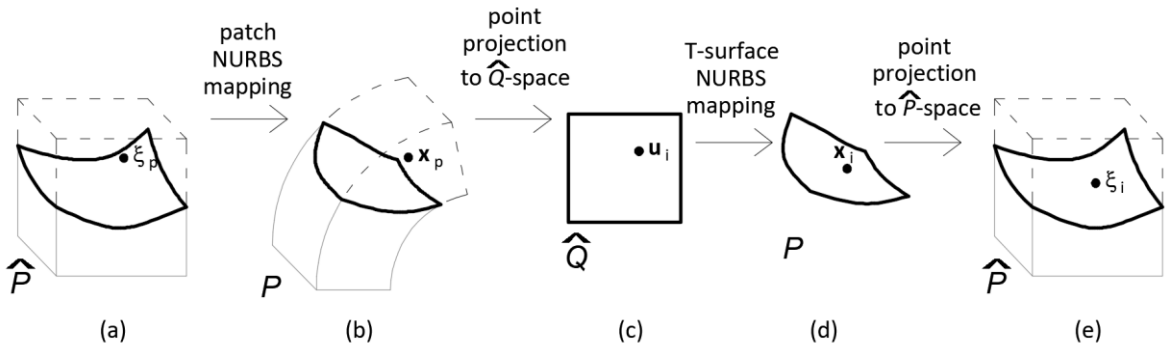


Fig 5D.2 Addition of Q-nodes.

c. K-nodes:

1. The position given in the $\hat{P}$-space is preliminary ($\boldsymbol{\xi}_p$).
2. Coordinates in the P-space are calculated by NURBS mapping ($\boldsymbol{x}_p$) from $\hat{P}$-space.
3. $\boldsymbol{x}_p$ is point projected on the curve where the K-node belongs, obtaining its parameter coordinate (w_i).
4. The physical position is calculated by NURBS mapping of w_i from the $\widehat{K}$-space to the P-space ($\boldsymbol{x}_i$).
5. The positions in the $\hat{P}$ and $\hat{Q}$-spaces are obtained by point projection of $\boldsymbol{x}_i$ on the patch and trimming surface ($\boldsymbol{\xi}_i$ and $\boldsymbol{u}_i$).

Note that in general $\boldsymbol{x}_i \neq \boldsymbol{x}_p$ and $\boldsymbol{\xi}_i \neq \boldsymbol{\xi}_p$. The process scheme is given in **Fig. 5D.3**.

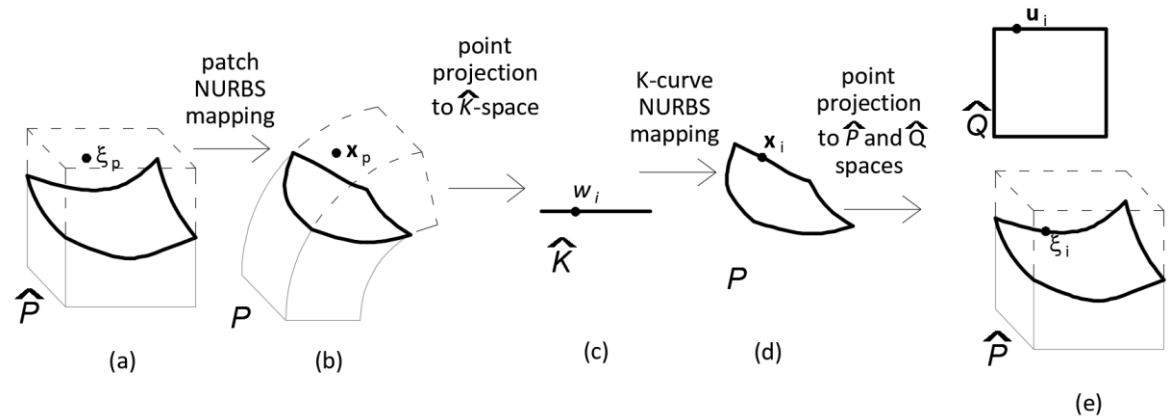


Fig 5D.3 Addition of K-nodes.

Appendix 5E: Example data

Table 5E.1 Gross patch NURBS features

Number of control points	4 4 5
Degree	2 2 2
Knot vector	000 0.5 111 000 0.5 111 000 0.5 0.8 111
Control points coordinates (x,y,z,w)	0.0000 50.0000 0.0000 1.0000 0.0000 50.0000 12.5070 0.9830 0.0000 55.1793 31.8363 0.9830 0.0000 60.2519 43.2120 0.9932 0.0000 62.7276 47.5000 1.0000 12.5000 50.0000 0.0000 1.0000 12.5000 50.0000 12.5070 0.9830 12.5000 55.1793 31.8363 0.9830 12.5000 60.2519 43.2120 0.9932 12.5000 62.7276 47.5000 1.0000 37.5000 50.0000 0.0000 1.0000 37.5000 50.0000 12.5070 0.9830 37.5000 55.1793 31.8363 0.9830 37.5000 60.2519 43.2120 0.9932 37.5000 62.7276 47.5000 1.0000 50.0000 50.0000 0.0000 1.0000 50.0000 50.0000 12.5070 0.9830 50.0000 55.1793 31.8363 0.9830 50.0000 60.2519 43.2120 0.9932 50.0000 62.7276 47.5000 1.0000 -5.0000 38.0000 0.0000 0.9615 -5.0000 38.0000 14.0868 0.9452 -5.0000 43.8335 35.8577 0.9452 -5.0000 49.5469 48.6704 0.9550 -5.0000 52.3353 53.5000 0.9615 8.4365 37.2561 0.0000 0.9808 8.1948 37.3922 14.2129 0.9683 8.1948 43.2404 36.0387 0.9683 8.3405 48.9170 48.9591 0.9758 8.4365 51.6910 53.8719 0.9808 30.5803 37.7519 0.0000 0.9426 30.2276 37.8912 14.1490 0.9308 30.2276 43.7115 35.8706 0.9308 30.4403 49.3599 48.7320 0.9379 30.5803 52.1204 53.6240 0.9426 41.0000 39.1200 0.0000 0.8853 41.0000 39.1200 13.9394 0.8702 41.0000 44.8924 35.4824 0.8702 41.0000 50.5460 48.1609 0.8792 41.0000 53.3052 52.9400 0.8853 -5.0000 12.0000 0.0000 0.9615 -5.0000 12.0000 17.5098 0.9452 -5.0000 19.2510 44.5708 0.9452 -5.0000 26.3527 60.4968 0.9550 -5.0000 29.8186 66.4999 0.9615 8.1052 9.5012 0.0000 0.9615 7.8602 9.7645 17.8511 0.9494 7.8602 17.1186 45.2968 0.9494

Table 5E.1 (cont.)

Control points coordinates (x,y,z,w)	8.0079 24.2020 61.5603 0.9567 8.1052 27.6546 67.7493 0.9615 30.6965 8.8611 0.0000 0.9234 30.3360 9.1383 17.9354 0.9119 30.3360 16.5257 45.5057 0.9119 30.5534 23.6353 61.8479 0.9188 30.6965 27.1002 68.0693 0.9234 41.0000 10.8800 0.0000 0.8853 41.0000 10.8800 17.6572 0.8702 41.0000 18.1920 44.9461 0.8702 41.0000 25.3535 61.0062 0.8792 41.0000 28.8486 67.0599 0.8853 0.0000 -0.0000 0.0000 1.0000 0.0000 -0.0000 19.0896 0.9830 0.0000 7.9052 48.5922 0.9830 0.0000 15.6476 65.9551 0.9932 0.0000 19.4263 72.4999 1.0000 12.0000 -5.0000 0.0000 0.9615 12.0000 -5.0000 19.7479 0.9452 12.0000 3.1778 50.2678 0.9452 12.0000 11.1872 68.2294 0.9550 12.0000 15.0961 74.9999 0.9615 38.0000 -5.0000 0.0000 0.9615 38.0000 -5.0000 19.7479 0.9452 38.0000 3.1778 50.2678 0.9452 38.0000 11.1872 68.2294 0.9550 38.0000 15.0961 74.9999 0.9615 50.0000 -0.0000 0.0000 1.0000 50.0000 -0.0000 19.0896 0.9830 50.0000 7.9052 48.5922 0.9830 50.0000 15.6476 65.9551 0.9932 50.0000 19.4263 72.4999 1.0000

Table 5E.2 Trimming surface NURBS features

Number of control points	2 8
Degree	1 3
Knot vector	00 11 0000 0.2 0.4 0.6 0.8 1111
Control points coordinates (x,y,z,w)	74.9484 -3.0000 55.0000 1.0000 74.9484 5.0000 45.0000 1.0000 74.9484 10.0000 33.0000 1.0000 74.9484 15.0000 35.0000 1.0000 74.9484 20.0000 30.0000 1.0000 74.9484 20.0000 20.0000 1.0000 74.9484 20.0000 10.0000 1.0000 74.9484 20.0000 -0.0000 1.0000 -25.0516 -3.0000 55.0000 1.0000 -25.0516 5.0000 45.0000 1.0000 -25.0516 10.0000 33.0000 1.0000 -25.0516 15.0000 35.0000 1.0000 -25.0516 20.0000 30.0000 1.0000 -25.0516 20.0000 20.0000 1.0000 -25.0516 20.0000 10.0000 1.0000 -25.0516 20.0000 -0.0000 1.0000

Appendix 6A: Arrangements of Gauss points

6A.1 Line

The parent space spans from -1 to 1. **Table 6A.1** provides the position in parent space and the weight of the control points, according to their number (NG).

Table 6A.1 Location and weights of Gauss points for lines.

NG	$\tilde{\xi}$	w
1	0	2
2	$\pm$0.57735	1
3	$\pm$0.77459 0	0.55555 0.88888
4	$\pm$0.86113 $\pm$0.33998	0.34785 0.65214
5	$\pm$0.90617 $\pm$0.53846 0	0.23692 0.47862 0.56888

6A.2 Square and hexahedron

For square and hexahedrons the Gauss points location and weights are obtained by tensor product of the involved directions. Each direction adopts the arrangement shown for lines.

6A.3 Triangle

The parent space is a rectangular triangle whose legs span from 0 to 1 in r and s axis. **Table 6A.2** provides the position in parent space and the weight of the control points, according to their number (NG).

Table 6A.2 Location and weights of Gauss points for triangles.

NG	$\boldsymbol{r}$	w
1	1/3, 1/3	0.5
3	0.5, 0.5 0, 0.5 0.5, 0	1/6 1/6 1/6
4	1/3, 1/3 0.6, 0.2 0.2, 0.6 0.2, 0.2	-27/96 25/96 25/96 25/96

6A.4 Tetrahedron

The parent space is a rectangular tetrahedron whose legs span from 0 to 1 in r, s and t axis. **Table 6A.3** provides the position in parent space and the weight of the control points, according to their number (NG).

Table 6A.3 Location and weights of Gauss points for tetrahedrons.

NG	$\boldsymbol{r}$	w
1	1/4, 1/4, 1/4	1/6
4	α, β, β β, α, β β, β, α β, β, β	1/24 1/24 1/24 1/24
5	1/4, 1/4, 1/4 1/3, 1/6, 1/6 1/6, 1/3, 1/6 1/6, 1/6, 1/3 1/6, 1/6, 1/6	-2/15 3/40 3/40 3/40 3/40

$\alpha = 0.58541020 \quad \beta = 0.13819660$

Appendix 6B: Tetrahedrons quality. Volume and regular tetrahedron

One tetrahedron volume is computed as equation (6B.1), where $\boldsymbol{v}_i^j$ is the vector from node i to node j.

$$V = \frac{1}{6}\, det[\boldsymbol{v}_4^1 \quad \boldsymbol{v}_4^2 \quad \boldsymbol{v}_4^3] \tag{6B.1}$$

One regular tetrahedron has all its edges with a length equal to b. In that particular case the volume may be calculated as follows:

$$V_r = \frac{\sqrt{2}}{12}\, b^3 \tag{6B.2}$$

Given one tetrahedron with edges length b_i, i varies from one to six. Let us define b_{ave} as the average of b_i. The regular version of that tetrahedron is another with all edges lengths equal to b_{ave} (see **Fig. 6B.1**).

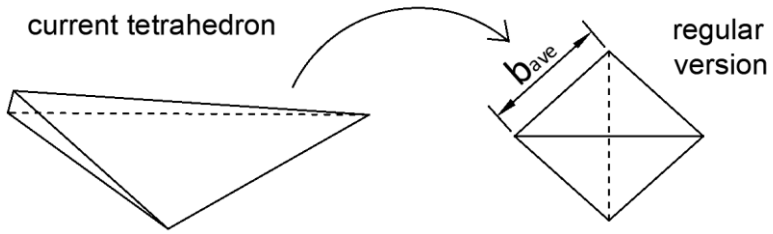


Fig. 6B.1 One tetrahedron and its regular version.

The regularity of the tetrahedron R is measured as the ratio (6B.3), which indicates proximity of one tetrahedron to its regular version. The upper limit is 1 that corresponds to a regular tetrahedron. The lower R the further is the current tetrahedron from its regular version and, hence, the less quality.

$$R = \frac{V}{V_r} \tag{6B.3}$$

Appendix 6C: Calculation of the optimal vertex location for one triangle

The volume of the tetrahedron may be calculated as follows:

$$V = \frac{1}{6}(\boldsymbol{u} \times \boldsymbol{v}) \cdot \boldsymbol{l} \tag{6C.1}$$

Where $\boldsymbol{u}$ and $\boldsymbol{v}$ vectors form one of the triangular bases and $\boldsymbol{l}$ is another vector that shares the initial point with $\boldsymbol{u}$ and $\boldsymbol{v}$ but does not belong to the triangular base. Let us call to such common node $\boldsymbol{a}_1$.

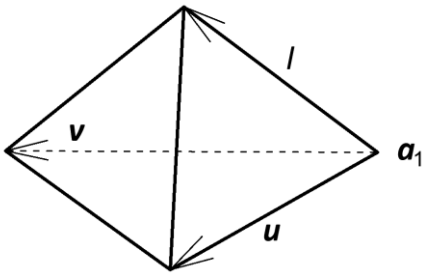


Fig. 6C.1 Three non-coplanar vectors can be used to compute the tetrahedron volume.

Given a triangle whose centre of gravity is $\boldsymbol{G}$, its vertexes are denoted by $\boldsymbol{a}_i$ and the sum of its edges is Lt. One tetrahedron can be formed by adding a fourth node ($\boldsymbol{a}_4$) on the line that passes through $\boldsymbol{G}$ perpendicularly to the triangle (see **Fig. 6C.2**). Let us call $\boldsymbol{n}$ to the normal versor to the triangle and t a free parameter, then the location of the fourth node is expressed as follows:

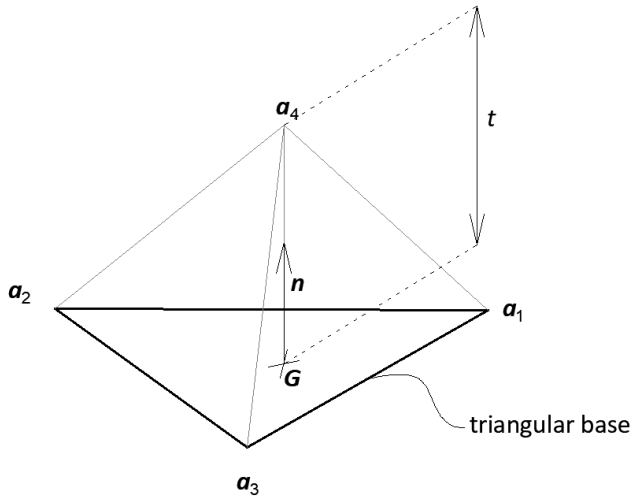


Fig. 6C.2 Tetrahedron generated from one triangular base.

$$\boldsymbol{a}_4 = \boldsymbol{G} + t\,\boldsymbol{n} \tag{6C.2}$$

The regularity (see Appendix 6B) of such tetrahedron can be calculated by equation (6C.3), where A and B are defined as (6C.4) and (6C.5) respectively.

$$R = 305.5\,\frac{A}{B} \tag{6C.3}$$

$$A = (\boldsymbol{u} \times \boldsymbol{v}) \cdot \boldsymbol{l} \tag{6C.4}$$

$$B = \left(Lt + \sum_{i=1}^{3} Lh_i \right)^3 \tag{6C.5}$$

where Lt is the summation of the edges of the triangular base and Lh_i is the length of the edges from each node of the triangular base ($\boldsymbol{a}_i$) to the fourth node ($\boldsymbol{a}_4$). Note that this second summand may be expressed as follows:

$$\sum_{i=1}^{3} Lh_i = \sum_{i=1}^{3} \|\boldsymbol{G} + t\,\boldsymbol{n} - \boldsymbol{a}_i\| \tag{6C.6}$$

There is a location for the fourth node, defined by the parameter t, where the tetrahedron regularity (R) is the maximum, *i.e.* the derivative is zero ($R_{,t} = 0$). That t value can be found by Newton-Raphson iterations as described here.

In A, the cross product $(\boldsymbol{u} \times \boldsymbol{v})$ ·depends on the selected triangular base and is constant. By contrast, the vector $\boldsymbol{l}$ can be expressed as function of t as follows:

$$\boldsymbol{l} = \boldsymbol{G} + t\,\boldsymbol{n} - \boldsymbol{a}_1 \tag{6C.7}$$

In B, the length of the edges of the triangular base (Lt) are constant, but the others are functions of the parameter t as follows:

$$Lh_i = \|\boldsymbol{G} + t\,\boldsymbol{n} - \boldsymbol{a}_i\| \tag{6C.8}$$

The first and second derivatives of A are:

$$A_{,t} = (\boldsymbol{u} \times \boldsymbol{v}) \cdot\ \boldsymbol{n} \tag{6C.9}$$

$$A_{,tt} = 0 \tag{6C.10}$$

The first and second derivatives of Lh_i are:

$$Lh_{i,t} = \frac{(\boldsymbol{G} + t\,\boldsymbol{n} - \boldsymbol{a}_i) \cdot\ \boldsymbol{n}}{Lh_i} \tag{6C.11}$$

$$Lh_{i,tt} = \frac{\boldsymbol{n} \cdot \boldsymbol{n} - Lh_{i,t}{}^2}{Lh_i} \tag{6C.12}$$

Therefore, the first and second derivatives of B are:

$$B_{,t} = 3\left(Lt + \sum_{i=1}^{3} Lh_i\right)^2 \sum_{i=1}^{3} Lh_{i,t} \tag{6C.13}$$

$$B_{,tt} = 6\left(Lt + \sum_{i=1}^{3} Lh_i\right)\left(\sum_{i=1}^{3} Lh_{i,t}\right)^2 + 3\left(Lt + \sum_{i=1}^{3} Lh_i\right)^2 \sum_{i=1}^{3} Lh_{i,tt} \tag{6C.14}$$

With all the derivatives at hand, the Newton-Raphson iterations may be applied as follows:

$$t^{k+1} = t^k - \frac{R_{,t}}{R_{,tt}} \tag{6C.15}$$

When the difference between t^{k+1} and t^k is less than a pre-established tolerance the iterations stop and the fourth of the tetrahedron can be found by inputting t^{k+1} in (6.C2).

Appendix 6D: Derivatives of Lagrangian quadratic curves

Let $\boldsymbol{C}$ be the parametric curve in $\mathbb{R}^d$ given as combination of Lagrange polynomials of order 2, *i.e.* the curve has three nodes. $\boldsymbol{C}$ is obtained by mapping $\mathbb{R}^1 \rightarrow \mathbb{R}^d$ from parent space to the curve space as:

$$\boldsymbol{C} = \sum_{i=1}^{3} \phi_i \, \boldsymbol{x}_i \tag{6D.1}$$

Where ϕ_i are the basis functions. The mapping is represented in **Fig. 6D.1**, where the coordinates in parent space are denoted by u and in $\boldsymbol{C}$ curve space by $\boldsymbol{x}$. The parent space is assumed to span from -1 to +1.

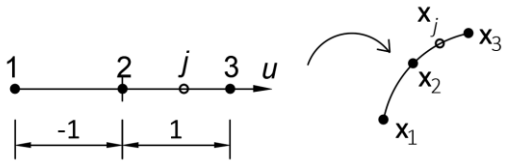


Fig. 6D.1 Mapping of Lagrangian quadratic parametric curve.

The basis functions at u location are:

$$\phi_1 = \frac{1}{2}(u^2 - u) \tag{6D.2}$$

$$\phi_2 = (1 - u^2) \tag{6D.3}$$

$$\phi_3 = \frac{1}{2}(u^2 + u) \tag{6D.4}$$

The derivative $\boldsymbol{C}_{,u}$ is calculated as:

$$\boldsymbol{C}_{,u} = \sum_{i=1}^{3} \frac{\partial \phi_i}{\partial u} \, \boldsymbol{x}_i \tag{6D.5}$$

Which may be extended as:

$$\boldsymbol{C}_{,u} = \frac{1}{2}(2u - 1)\boldsymbol{x}_1 + (2u)\boldsymbol{x}_2 + \frac{1}{2}(2u + 1)\boldsymbol{x}_3 \tag{6D.6}$$

At starting location ($u = -1$) the derivative is:

$$\boldsymbol{C}_{,u} = \frac{-3}{2} \, \boldsymbol{x}_1 + 2 \, \boldsymbol{x}_2 - \frac{1}{2}\boldsymbol{x}_3 \tag{6D.7}$$

Appendix 6E: Tetrahedrons of mixed degree

Tetrahedral meshes are well settled (Hughes 2000, Zienkiewicz, Taylor 2000) and extensively used due to its adaptability to any shape. Linear tetrahedrons have 4 nodes, meanwhile quadratic tetrahedrons have 10 nodes, with 4 at the tetrahedron vertex (called end-nodes) and one at the mid location of each edge[64] (called mid-nodes). We call mixed degree tetrahedrons in this work to those that have some of their lines with the mid-node. This appendix explains how to obtain the basis functions of mixed-degree tetrahedrons.

The number of nodes in mixed-degree tetrahedrons may vary from 4 to 10. Linear tetrahedrons have only end-nodes and are arranged as **Fig. 6E.1**.

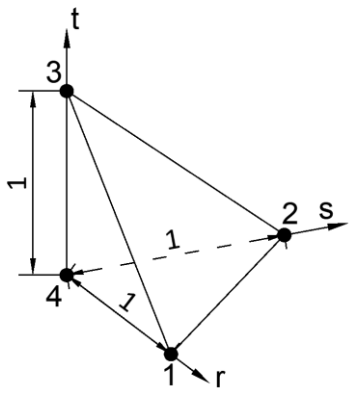


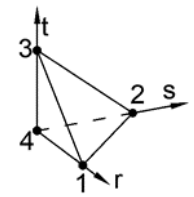
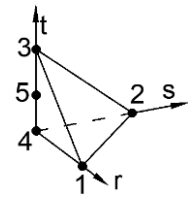
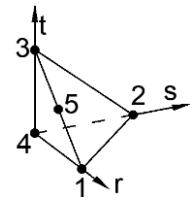
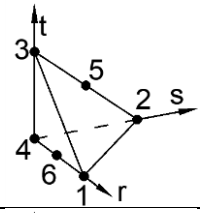
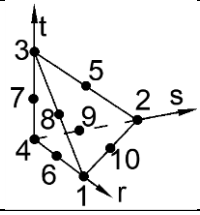
Fig. 6E.1 Nodes for linear tetrahedron in parent space.

Considering the linear tetrahedron as starting point, new nodes may be inserted at mid-location of its lines (mid-nodes). The insertion mid-nodes to all the lines leads to the quadratic tetrahedron.

One mid-node can be inserted at any edge. The matrix called *iem*[65] stores where the mid-nodes are inserted. The *iem* matrix has four rows and four columns, one per end-node. Let us focus on the ith row. Each of its four components (except the ith column) can contain a number which is the reference of the inserted mid-node, *i.e.* if jth column has k number, it means that the k mid-node is inserted between i and j end-nodes. If the component is zero there is no mid-node inserted. The *iem* matrix is symmetric and components at diagonal are all zero. **Table 6E.1** shows five examples of *iem* matrix.

[64] The *edges* are also called *lines* in this work.
[65] *iem* is the acronym of *inserted edge matrix*.

Table 6E.1 Examples of storage of information of inserted mid-nodes.

iem matrix	Tetrahedral parent space	Comments
$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$		Linear tetrahedral
$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 \\ 0 & 0 & 5 & 0 \end{bmatrix}$		One mid-node insertion between nodes 3-4
$\begin{bmatrix} 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$		One mid-node insertion between nodes 1-3
$\begin{bmatrix} 0 & 0 & 0 & 6 \\ 0 & 0 & 5 & 0 \\ 0 & 5 & 0 & 0 \\ 6 & 0 & 0 & 0 \end{bmatrix}$		Two mid-nodes insertion between nodes 1-4 and 2-3
$\begin{bmatrix} 0 & 10 & 8 & 6 \\ 10 & 0 & 5 & 9 \\ 8 & 5 & 0 & 7 \\ 6 & 9 & 7 & 0 \end{bmatrix}$		Quadratic tetrahedral

Given a location in parent space with coordinates $\boldsymbol{r} = (r\ s\ t)^T$, the basis functions at $\boldsymbol{r}$ might be obtained for each case as described in (Hughes 2000). Here we give a procedure that is valid for any mixed tetrahedral configuration.

First, the fourth coordinate is computed as equation (6E.1), then the four coordinates are stored in the vector $\hat{\boldsymbol{r}} = (r\ s\ t\ u)^T$.

$$u = 1 - r - s - t \tag{6E.1}$$

Second, the basis function of each *k*th mid-node inserted is calculated as equation (6E.2).

$$N_k = 4\,\hat{\boldsymbol{r}}(i)\,\hat{\boldsymbol{r}}(j) \tag{6E.2}$$

Where i and j are the row and column positions in *iem* matrix. Third, the basis functions of the affected end-nodes are calculated as (6E.3).

$$N_k = \hat{\boldsymbol{r}}(k) - \sum_{i=1}^{A} \frac{1}{2} N_i \tag{6E.3}$$

Where N_i are the basis function values of the *i*th mid-nodes, inserted at each edge attached to the *k*th corner-node. A is the number of attached edges with mid-insertion, which may vary from zero to three. We provide one example below for clarity. Mixed tetrahedral basis functions calculation is not new. The procedure explained here provides flexibility for any configuration and facilitates the implementation in the author's humble opinion.

Example of construction of basis function:

(Corresponding to the fourth case of **Table 6E.1**, with two mid-nodes insertions).

The location where we compute the basis functions is $\hat{\boldsymbol{r}} = (r\ s\ t\ u)^T$.

In view of the position of the 5th and 6th mid-nodes references in the *iem* matrix, their functions are calculated as follows:

$$N_5 = 4\,\hat{\boldsymbol{r}}(2)\,\hat{\boldsymbol{r}}(3) = 4st$$

$$N_6 = 4\,\hat{\boldsymbol{r}}(1)\,\hat{\boldsymbol{r}}(4) = 4ru$$

The corner nodes basis functions are:

$$N_1 = \hat{\boldsymbol{r}}(1) - \frac{1}{2} N_6 = r - 2ru$$

$$N_2 = \hat{\boldsymbol{r}}(2) - \frac{1}{2} N_5 = s - 2st$$

$$N_3 = \hat{\boldsymbol{r}}(3) - \frac{1}{2} N_5 = t - 2st$$

$$N_4 = \hat{\boldsymbol{r}}(4) - \frac{1}{2} N_6 = u - 2ru$$

Gauss points are inserted in the tetrahedrons according to Appendix 6A. Mixed-degree tetrahedrons are treated as quadratic, *i.e.* they need four Gauss points.

Appendix 7A: Calculation of displacements and stresses

The displacements and stresses at any location of the domain may be computed as long as the material and the displacements at the control points are known. In this appendix this computation is explained for any location within the domain, but it is focused on the domain boundaries (*skins*) because they are used for representation of results.

7A.1 Assumptions

- The displacements at control points are known.
- The domain has its boundaries discretized into linear triangles (called *skins*) for representation purposes.
- The material behaves linearly, *i.e.* the material properties do not change after the deformation.

7A.2 Calculation of displacements at any location within the domain

The displacement at the ath location within the domain is calculated as:

$$\boldsymbol{u}_a = \sum_{I=1}^{\aleph} \boldsymbol{R}(\boldsymbol{\xi}_a)_I \boldsymbol{u}_I \tag{7A.1}$$

where:

$\aleph$ is the number of control points of the domain.

$R(\boldsymbol{\xi}_a)_I$ is the matrix with the Ith basis function at $\boldsymbol{\xi}_a$.

$\boldsymbol{\xi}_a$ are the coordinates of a in patch parameter space.

$\boldsymbol{u}_I$ is the displacement of the Ith control point.

7A.3 Calculation of stresses at any location within the domain

The stress vector at the ath location within the domain is calculated as:

$$\boldsymbol{\sigma}_a = \boldsymbol{D} \sum_{I=1}^{\aleph} \boldsymbol{B}(\boldsymbol{\xi}_a)_I \boldsymbol{u}_I \tag{7A.2}$$

Where:

$\boldsymbol{D}$ is the constitutive matrix, that characterizes the material.

$\boldsymbol{B}(\boldsymbol{\xi}_a)_I$ is the strain-displacement matrix for the Ith basis function at $\boldsymbol{\xi}_a$.

7A.4 Displacements and stresses on the skins

The skins have nodes and triangles connecting them. The nodes are located at the domain boundaries, *i.e.* belong to the domain. The displacements and stresses may be computed at the nodes by equations (7A.1) and (7A.2). Then displacements and stresses at any location within one triangle may be interpolated from its nodes as follows:

$$\boldsymbol{s}_a = \sum_{j=1}^{3} \phi(\boldsymbol{r}_a)_j \, \boldsymbol{s}_j \tag{7A.3}$$

Where:

$\boldsymbol{s}_a$ is the displacement or stress at ath location, that lies on the triangle.

$\phi(\boldsymbol{r}_a)_j$ is the basis function value of the linear triangle at $\boldsymbol{r}_a$ location.

$\boldsymbol{r}_a$ are the coordinates in parent space of a.

The triangle parent space coordinates are $\boldsymbol{r} = (r, s)$. The parent space is a right triangle whose legs span from (0,0) to (1,0) and from (0,0) to (0,1). The basis functions are calculated as follows:

$$\phi_1 = r_a \tag{7A.4}$$

$$\phi_2 = s_a \tag{7A.5}$$

$$\phi_3 = 1 - r_a - s_a \tag{7A.6}$$

Appendix 8A: User guidance and generation of examples

This appendix provides the detailed steps to generate the domains of the examples in CAD and how to proceed with the user interphase (UI). Since the UI is not the main target of this research, the user windows are work in progress, and some of the options that appear in them are deactivated.

Section 1 of this appendix briefs the steps to follow to use the algorithm. Then sections 2 to 4 explain in detail the generation of the domains and the way to proceed with the UI for the examples provided in this thesis.

8A.1 Steps to use the algorithm

After extracting from CAD the IGES files (*G.igs* and *B.igs* for gross and bounded patches respectively) they are copied to the folder *201_TXTs* under to Matlab® directory. This directory may contain IGES files for other geometries (see **Fig. 8A.1**).

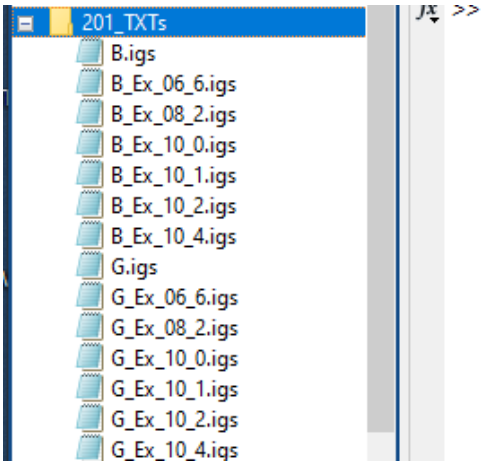


Fig. 8A.1 Folder containing the IGES files.

Then, in the routine `FirstRoutine` make sure the names called by the function `i3100_IGES` are the corresponding to *G.igs* and *B.igs*. (see **Fig. 8A.2**)

```
function [ ] = FirstRoutine ()


% 01.- EXTRACTS & DEFINES DOMAIN GEOMETRY FROM IGES FILES:

disp (  '01  -  Extraction from IGES file and solid parametrization'  );

%  example ref.

Gname = strcat('G.igs');
Bname = strcat('B.igs');


[Npa, Nbe, Nbc, Ntc, FgPatch, gFace, bEnti, tFace, bCurv, tCurv, pCou] = i3100_IGES (Gname, Bname);


pd0401_keep (Npa, Nbe, FgPatch, gFace, bEnti, tFace, bCurv, tCurv)


disp(  '                                                FINISHED'  );



% 02.- OPENS PROBLEM DEFINITION FORM:

pd0400_Form;


end
```

Fig.8A.2 Names called in `i3100_IGES` must coincide with names of IGES files.

To start the algorithm, in the Matlab® command window type `FirstRoutine` and enter. This routine reads the IGES files and generates the parametrization for the solids, trimming surfaces and the boundary surfaces. The inputs window, called `pd400_Form`, appears automatically, as shown in **Fig. 8A.3**.

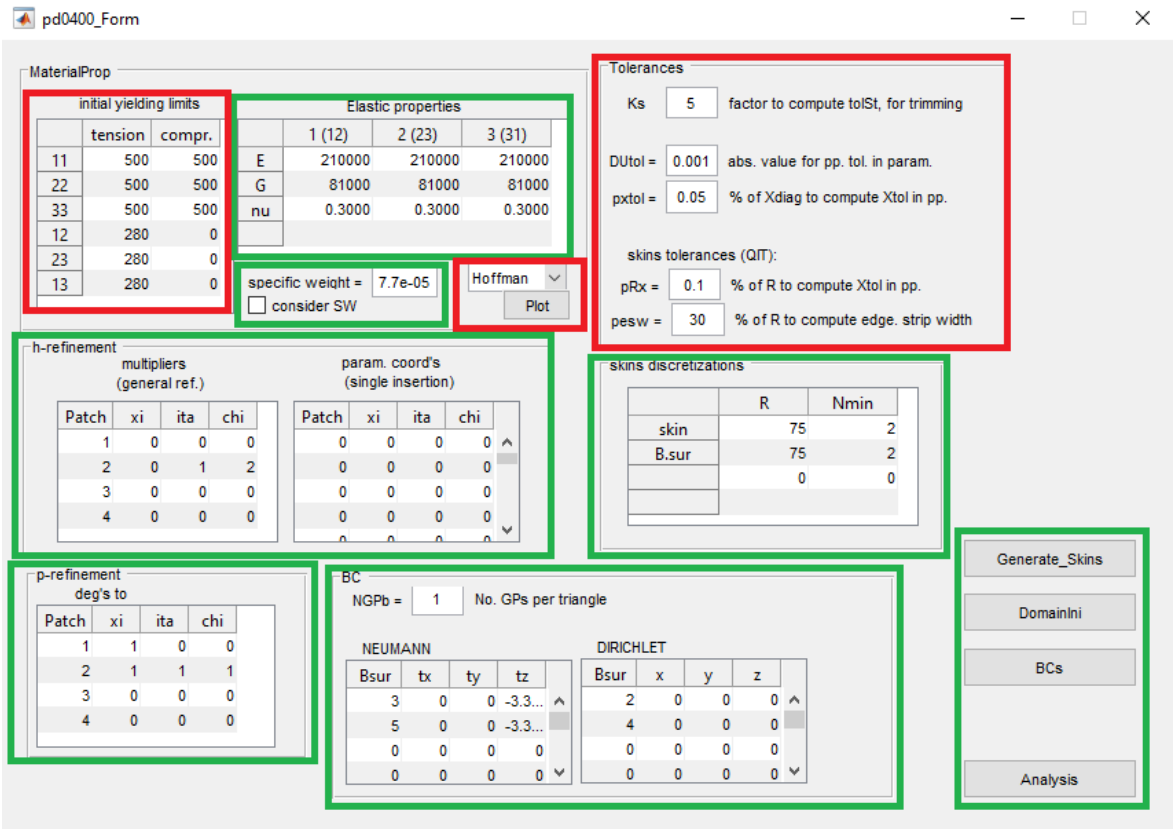


Fig. 8A.3 Inputs window.

There are some fields that are not used or are not advised to vary in the `pd400_Form`, as indicated in red contours in **Fig. 8A.3**. In the *Material properties* area, the initial yielding limits are not used. They are prepared for plastic analysis that is out of the scope of this research. The yielding model, set by default as *Hoffman* is not used neither. *Tolerances* area include the parameters used for discretization, which are advised to remain as the values given by default.

Fields that the user may vary, indicated on **Fig. 8A.3** with green contours, are listed below.

Elastic properties: the elastic constants are introduced here. The fields are prepared for orthotropic materials, however when this thesis was submitted the algorithm only worked with isotropic materials and only values at first column are considered.

Specific weight: may be disregarded if the check box below is not activated.

h-refinement: for each patch a *general* and *single* refinement may be applied. The first halves each non-void span and the second insert one knot at the specified location. For example, a *general h-refinement* of the knot vector 000 0.2 111 would yield 000 0.1 0.2 0.6 111. Then, one could insert an extra knot at 0.65 in the *single h-refinement* field.

p-refinement: for each patch the increment of degree may be set.

BC: this is de boundary conditions area. The number of Gauss points per triangle of the boundary surfaces can be set to 1, 3 or 4. Neumann and Dirichlet boundary conditions must be defined for the involved surfaces. They are introduced by Cartesian components.

The button area, at right-bottom of the window, allows the user to carry on with the algorithm steps as follows:

Button *Generate_Skins*: the skins of the patches, both gross and bounded, are generated by triangulation.

Button *Domaini*: the patches are discretized. After this process, the output window, called `r3000_represent`, is shown.

Button *BCs*: the boundary surfaces are triangulated and the boundary conditions applied to them.

Button *Analysis*: the displacements and stresses are computed.

These buttons must be used in the presented sequence. The output window (`r3000_represent`) allows the user to visualize the domain and the results (see **Fig. 8A.4**). After selecting the space and objects to visualize the button *Draw* must be pushed to refresh the view. At left-hand side of

that button the step may be changed to 2, which indicates results after analysis. This option is only available after analysis has finished (button *Analysis* pushed).

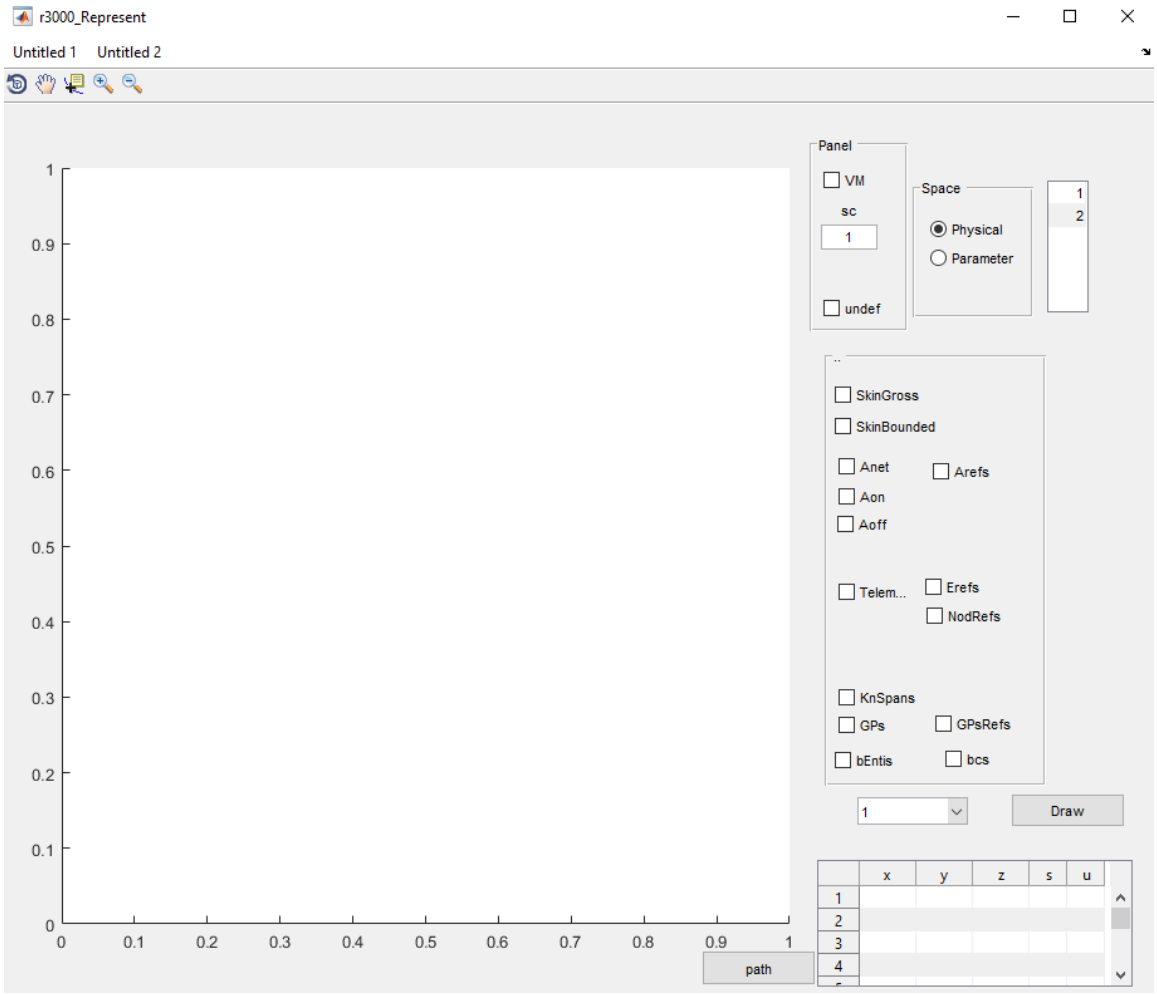


Fig. 8A.4 Output window.

8A.2 Example 8.1

Generation of domains in CAD

The gross patches are generated by extrusion of the contours as shown in **Fig. 8A.5** and **Fig. 8A.6**. We set patch 1 to the flange of the beam and patch 2 the web. The curve line of patch 1 has been created by spline fitting to a set of points.

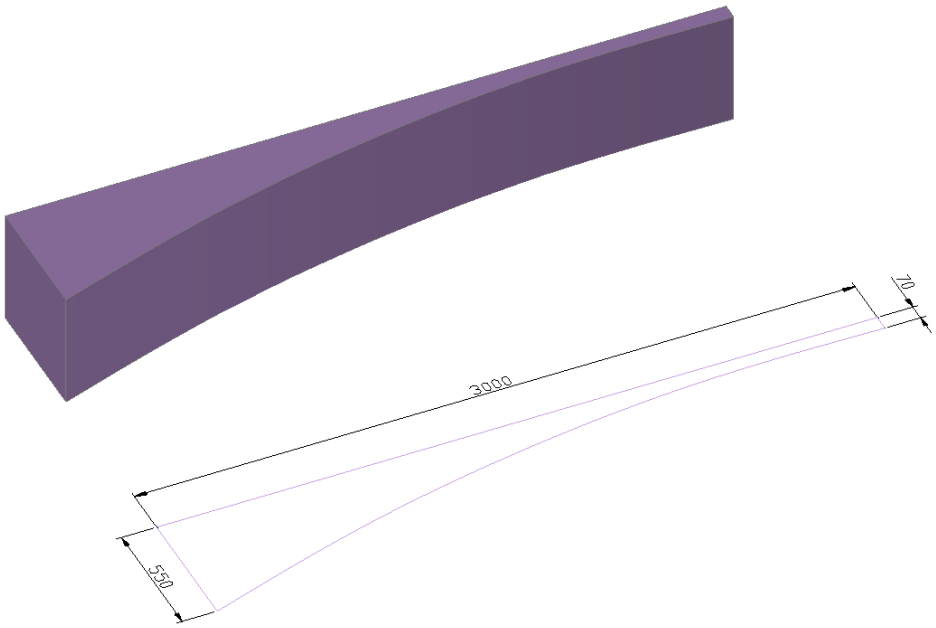


Fig. 8A.5 Generation of gross patch 1.

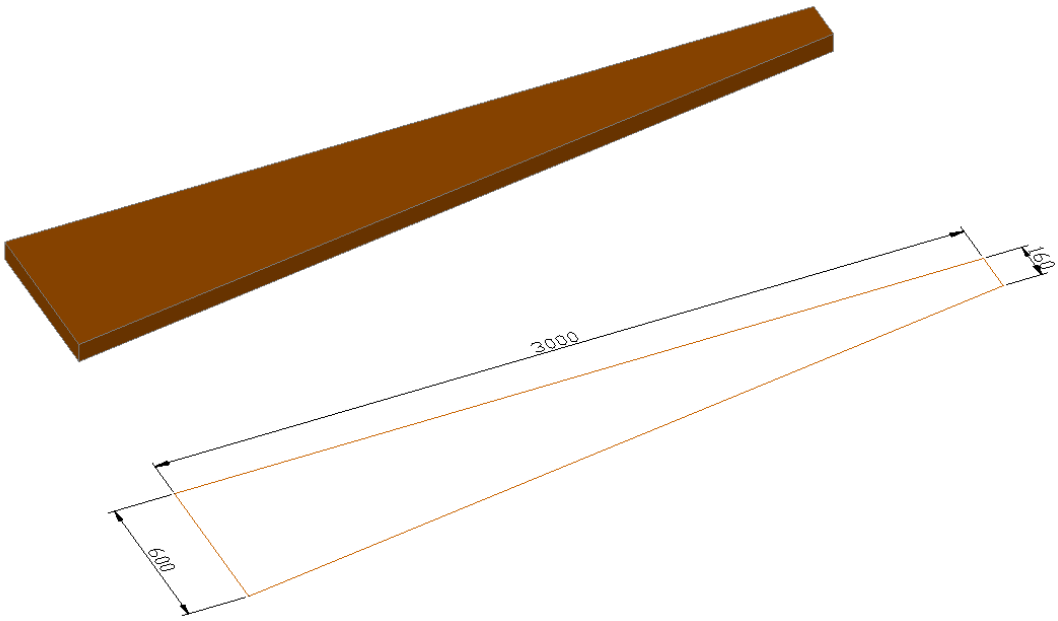


Fig. 8A.6 Generation of gross patch 2.

Both patches are assembled as illustrate din **Fig. 8A.7**. To allocate the patches references we use the first value of the RGB colour code (see Appendix 3B). In this case for patch 1 and patch 2 the

colour figures are 201-159-229 and 202-100-0 respectively. The IGES file if this configuration is exported as the gross patches version with the name *G_Ex_10_1.igs*.

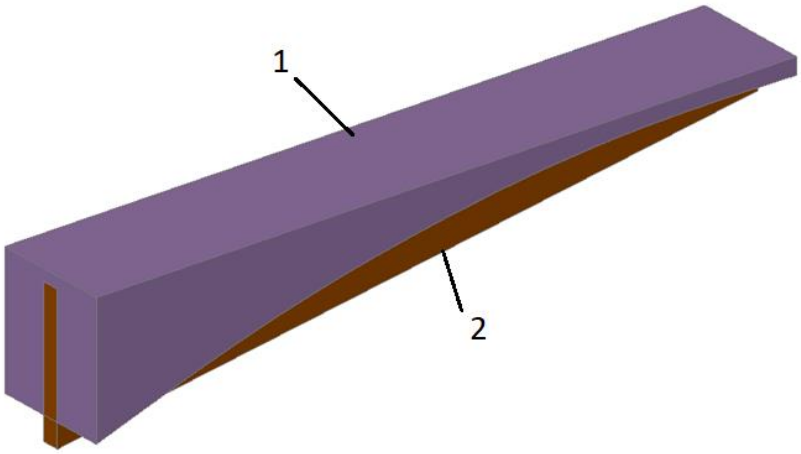


Fig. 8A.7 Assembling of gross patches.

To achieve the final shape, and avoid patches intersections, the gross patch 2 needs to be trimmed as shown in **Fig. 8A.8**. The trimmed domain must occupy the same position as the gross domain in the CAD space.

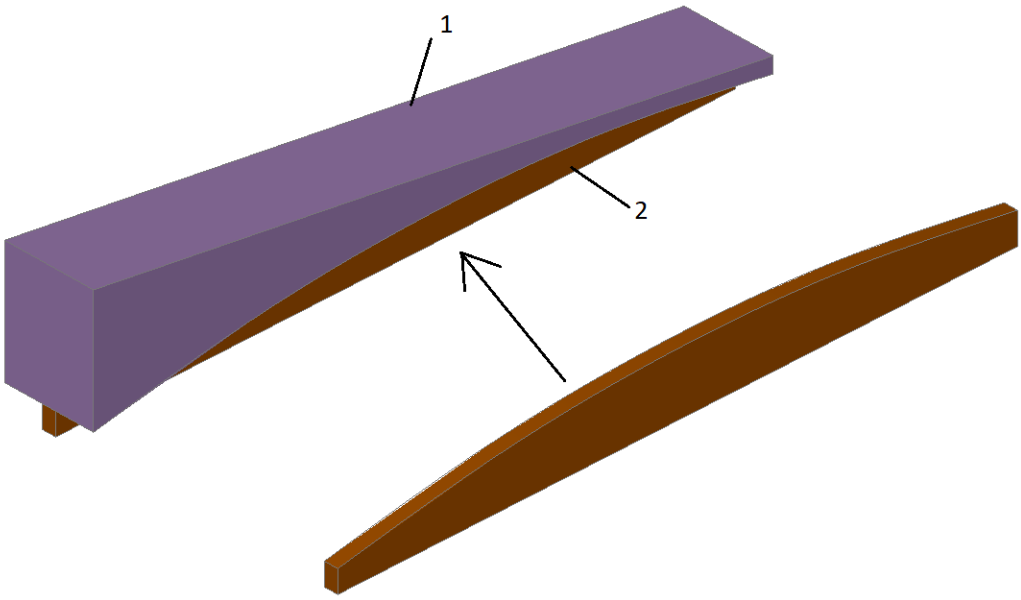


Fig. 8A.8 Trimming of patch 2 and assembling of the domain.

The boundary surfaces are obtained by extraction of faces of the trimmed patches (see **Fig. 8A.9**). Surfaces 2 and 4 will be used to impose zero displacement, surface 1 will couple both patches, and surfaces 3 and 5 will be used to impose tractions. Boundary surfaces references are given by their RGB colour code (see Appendix 3B). For surfaces 1 to 5 the RGB figures are listed below:

Surface 1:	201-202-201
Surface 2:	202-85-120
Surface 3:	203-85-120

Surface 4: 204-85-120
Surface 5: 205-85-120

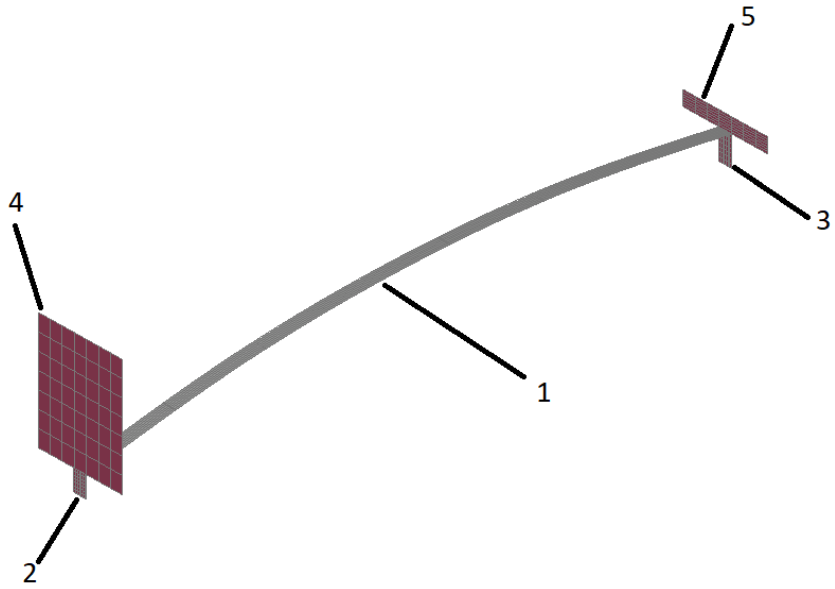


Fig. 8A.9 Boundary surfaces.

The boundary surfaces are moved onto the trimmed domain to achieve the bounded domain (see **Fig. 8A.10**). Recall that the bounded domain must hold the same position as the gross domain. The bounded domain is exported to the IGES file with the name *B_Ex_10_1.igs*.

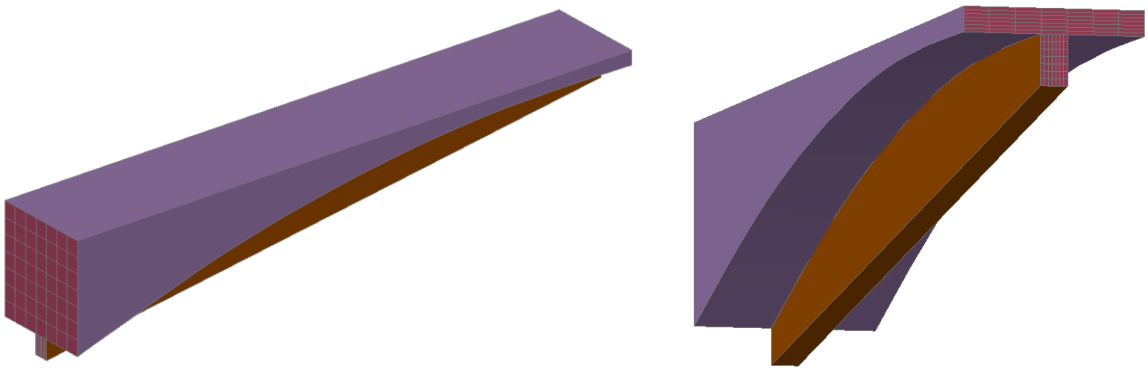


Fig. 8A.10 Bounded domain.

Preparation of patches for analysis

The `FirstRoutine` produces the parameterization of the gross patches as shown in **Fig. 8A.11** and their control points as **Fig. 8A.12**. In order to gain accuracy in the analysis both patches are refined as indicated in **Table 8A.1** and **Table 8A.2**. For path 1 degree elevation from 1 to 2 is done in ξ direction. For patch 2 the number of control points inserted in ξ, η and χ parameter directions are 1, 2 and 4 respectively. Degree elevation from 1 to 2 is done in the three parameter directions.

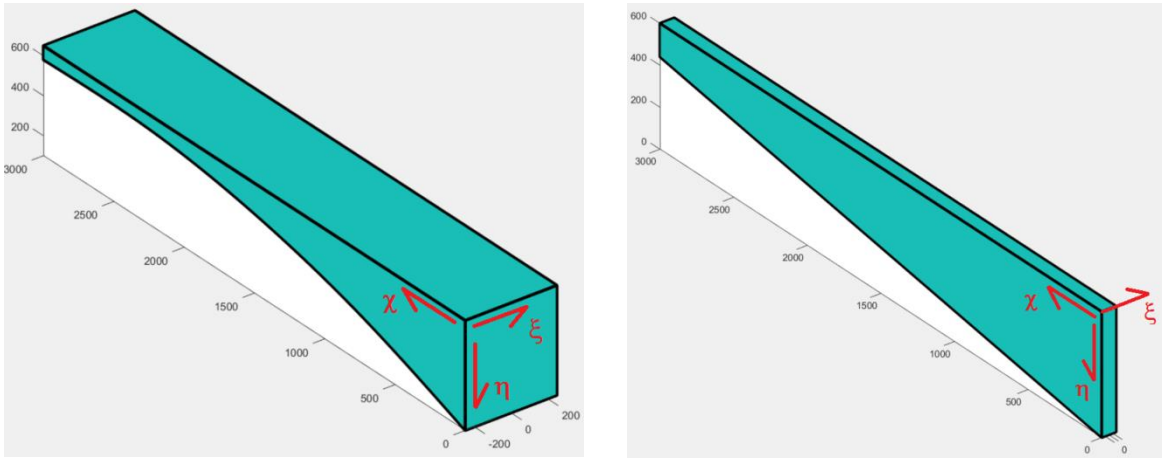


Fig. 8A.11 Parameter directions of the patches

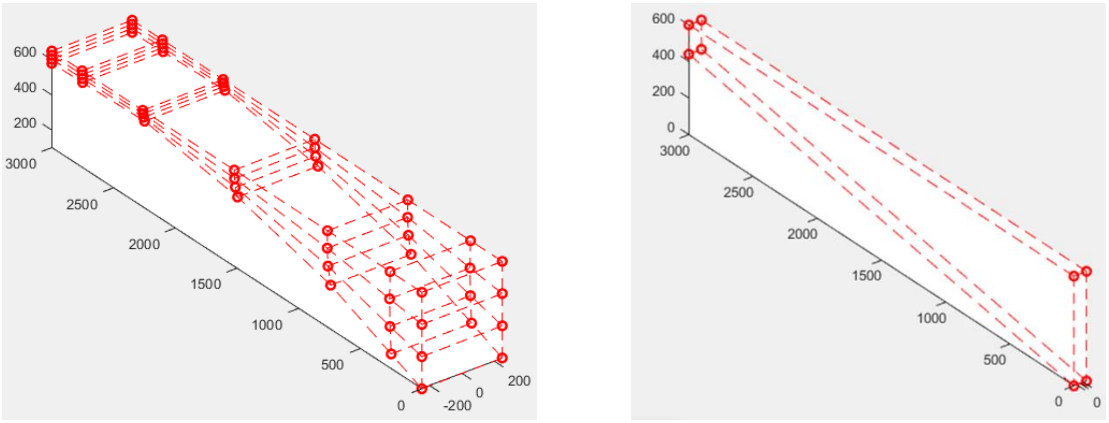


Fig. 8A.12 Initial control points arrangement.

Table 8A.1 Refinement in patch 1.

		Initial	Refined
Number of control points	n	2	3
	m	4	4
	l	7	7
Degree	p	1	2
	q	3	3
	r	3	3
Knot vectors	Ξ	00 11	000 111
	H	0000 1111	0000 1111
	X	0000 0.2573 0.5081 0.7547 1111	0000 0.2573 0.5081 0.7547 1111

Table 8A.2 Refinement in patch 2.

		Initial	Refined
Number of control points	n	2	3
	m	2	4
	l	2	6
Degree	p	1	2
	q	1	2
	r	1	2
Knot vectors	Ξ	00 11	000 111
	H	00 11	000 0.50 111
	X	00 11	000 0.25 0.50 0.75 111

The *h-refinement* and *p-refinement* areas in the input form need to be as shown in **Fig. 8A.13**. As result the refinement the control points arrangements are as per **Fig. 8A.14**.

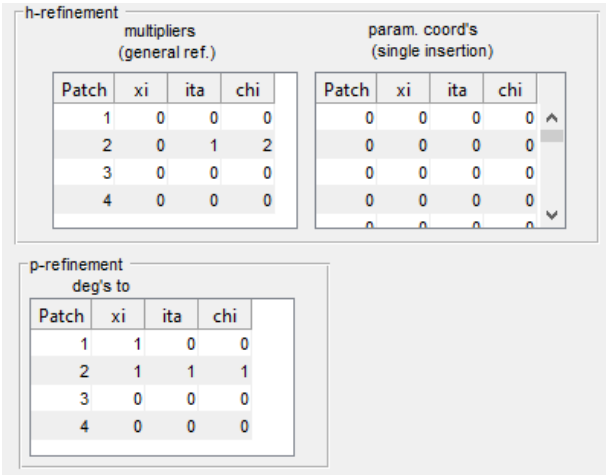


Fig. 8A.13 Refinement inputs required.

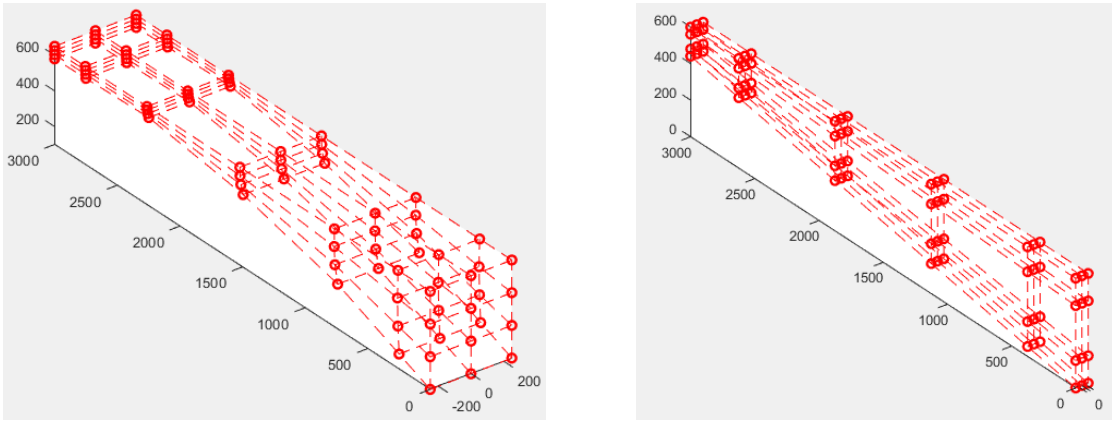


Fig. 8A.14 Refined control points arrangement.

8A.3 Example 8.2

Generation of domains in CAD

The gross patch is a cuboid generated by extrusion of 22 the contour shown in **Fig. 8A.15**.

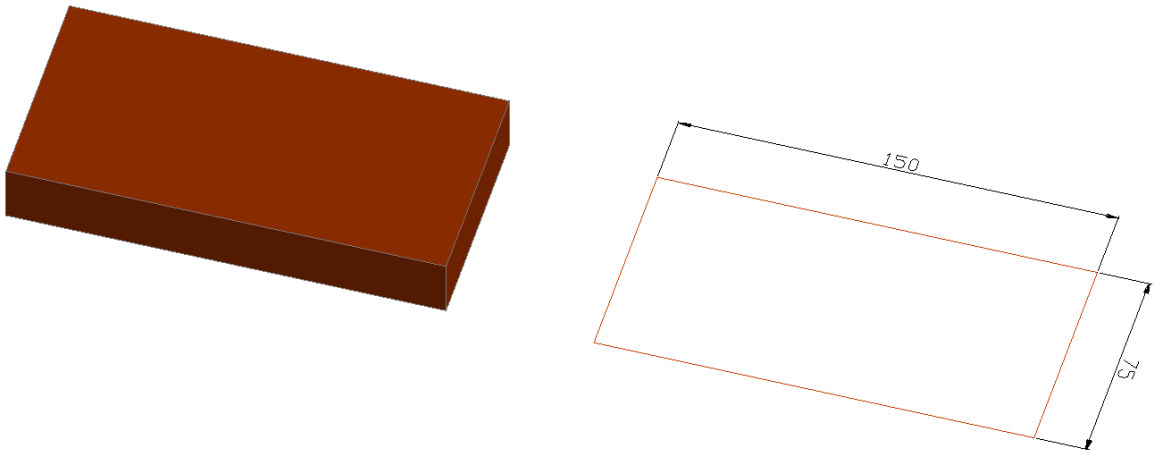


Fig. 8A.15 Generation of gross patch.

To allocate the patch reference we use the first value of the RGB colour code (see Appendix 4B), in this case the colour figures are 201-63-0. The IGES file if this configuration is exported as the gross patches version with the name *G_Ex_10_2.igs*.

The ground profile is generated by loft of the three curves shown in **Fig. 8A.16**. They are splines whose control points coordinates in the x-y plane are listed in **Table 8A.3**, the weights of all control points are one.

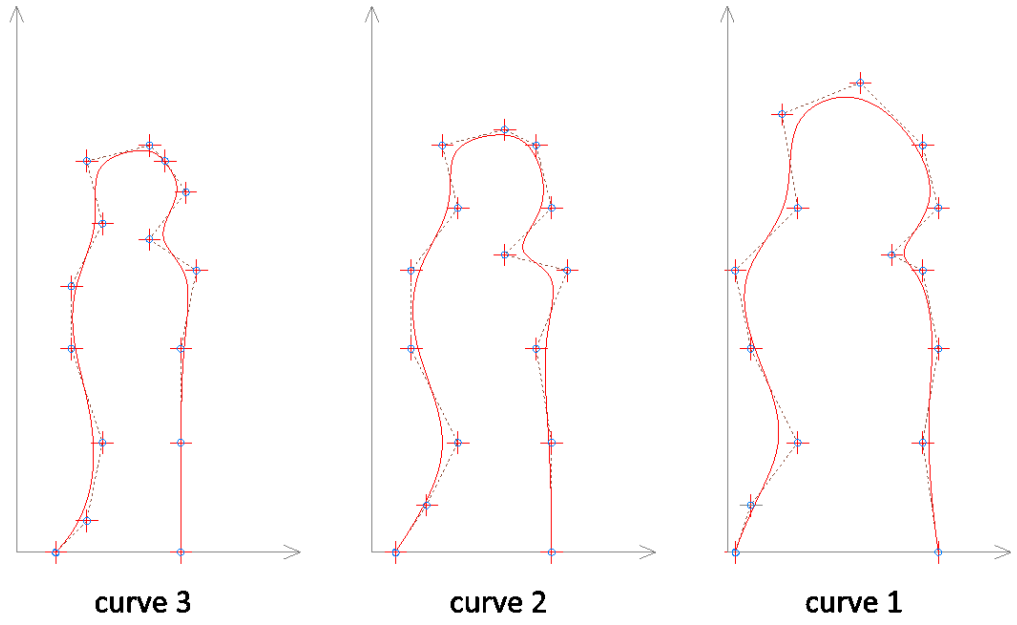


Fig. 8A.16 Curves used to generate the surface.

Table 8A.3 Coordinates of control points.

Control points	curve 1		curve 2		curve 3	
	x	y	x	y	x	y
1	2.5	0	7.5	0	12.5	0
2	7.5	15	17.5	15	22.5	10
3	22.5	35	27.5	35	27.5	35
4	7.5	65	12.5	65	17.5	65
5	2.5	90	12.5	90	17.5	85
6	22.5	110	27.5	110	27.5	105
7	17.5	140	22.5	130	22.5	125
8	42.5	150	42.5	135	42.5	130
9	62.5	130	52.5	130	47.5	125
10	67.5	110	57.5	110	54	115
11	52.5	95	42.5	95	42.5	100
12	62.5	90	62.5	90	57.5	90
13	67.5	65	52.5	65	52.5	65
14	62.5	35	57.5	35	52.5	35
15	67.5	0	57.5	0	52.5	0

The trimming surface is generated by loft across the three curves, that are set at heights 0, 10 and 24 as illustrated in **Fig. 8A.17**.

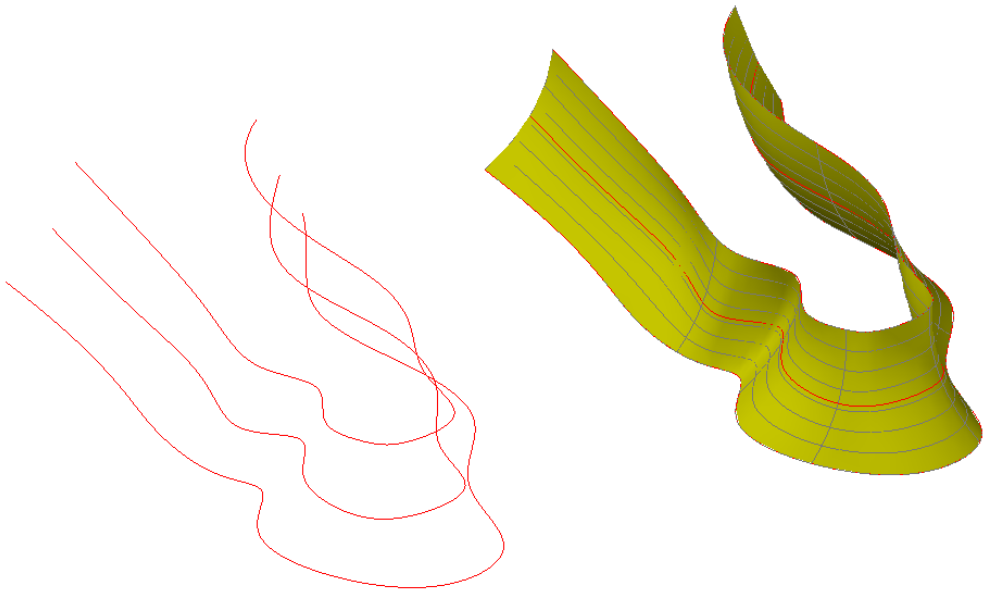


Fig. 8A.17 Trimming surface generated by loft.

To achieve the final shape the gross patch is trimmed by the trimming surface as indicated in **Fig. 8A.18**. The trimming surface flashes the bottom of the gross patch and protrude 2 over the top face. The setting out dimension is 7.5 from the corner as shown at right hand side of **Fig. 8A.18**. The resultant trimmed patch is depicted in **Fig. 8A.19**.

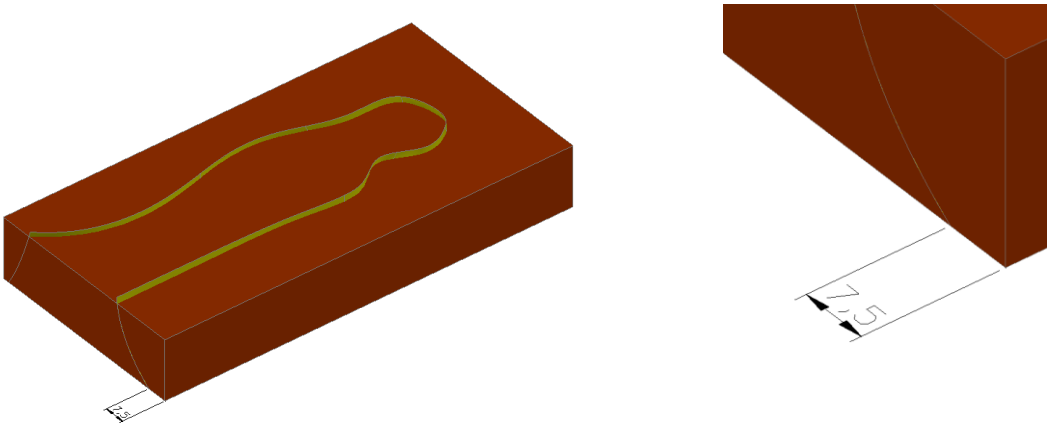


Fig. 8A.18 Relationship between the gross patch and the trimming surface.

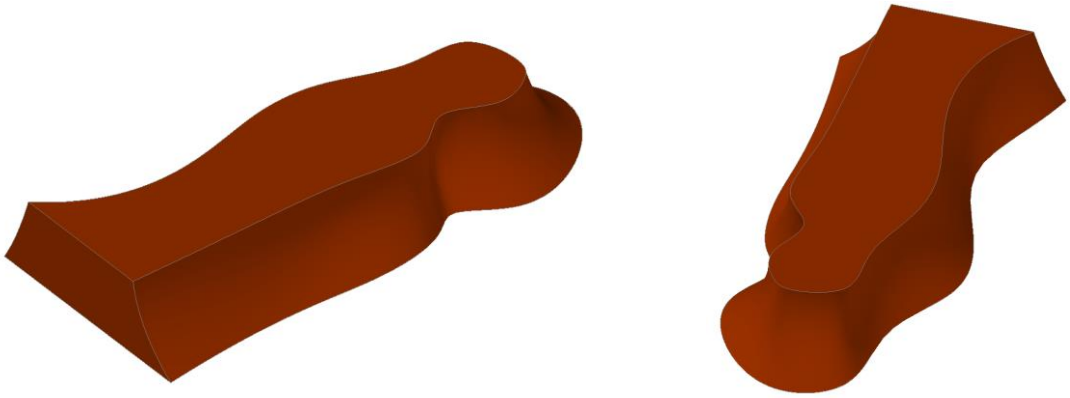


Fig. 8A.19 Trimmed patch.

The boundary surfaces are obtained by extraction of faces of the trimmed patch (see **Fig. 8A.20**). Surfaces 1 and 2 will be used to impose Dirichlet boundary conditions are attached to surfaces 1 and 2. Surface 1 has the displacements in all direction (x, y, and z) equal to zero, meanwhile surface 2 has only direction y equal to zero. Surface 3 will be used to impose tractions (400 in the negative direction of z). Boundary surfaces references are given by their RGB colour code (see Appendix 4B). For surfaces 1 to 3 the RGB figures are listed below:

Surface 1:	201-164-187
Surface 2:	202-164-187
Surface 3:	203-164-20

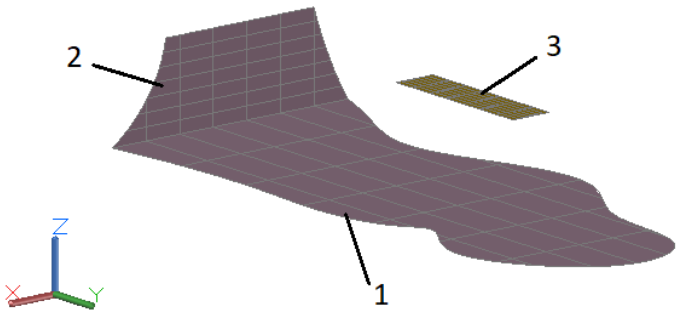


Fig. 8A.20 Boundary surfaces.

The boundary surfaces are placed onto the trimmed domain to achieve the bounded domain (see **Fig. 8A.21**). Recall that the bounded domain must hold the same position as the gross domain. The bounded domain is exported to the IGES file with the name *B_Ex_10_2.igs*.

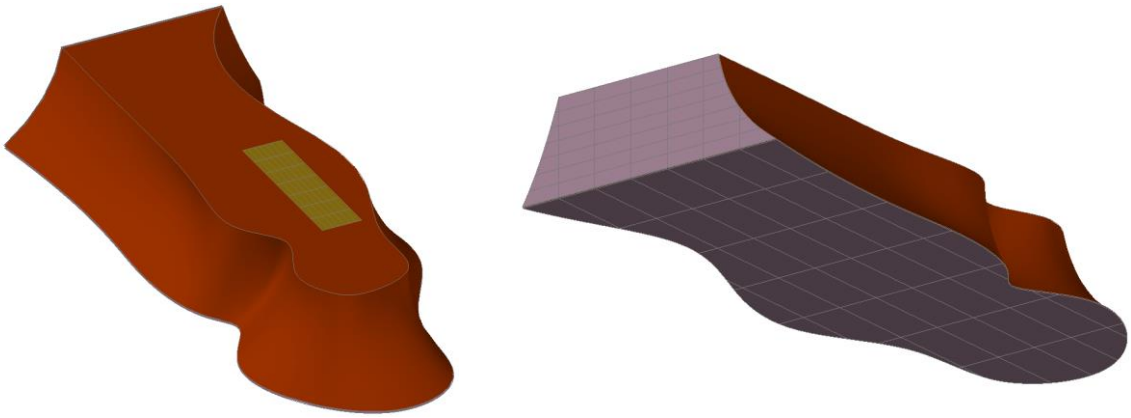


Fig. 8A.21 Bounded domain.

Preparation of patches for analysis

The `FirstRoutine` produces the parameterization of the gross patch as shown in **Fig. 8A.22** where the control points are also shown. To increase the degrees of freedom and gain accuracy in the analysis the refinement indicated in **Table 8A.4** is carried out. Degree elevation from 1 to 2 is done in ξ and η directions, meanwhile degree increment from 1 to 3 happens in χ direction. The number of control points inserted in ξ, η and χ parameter directions are 1, 1 and 3 respectively.

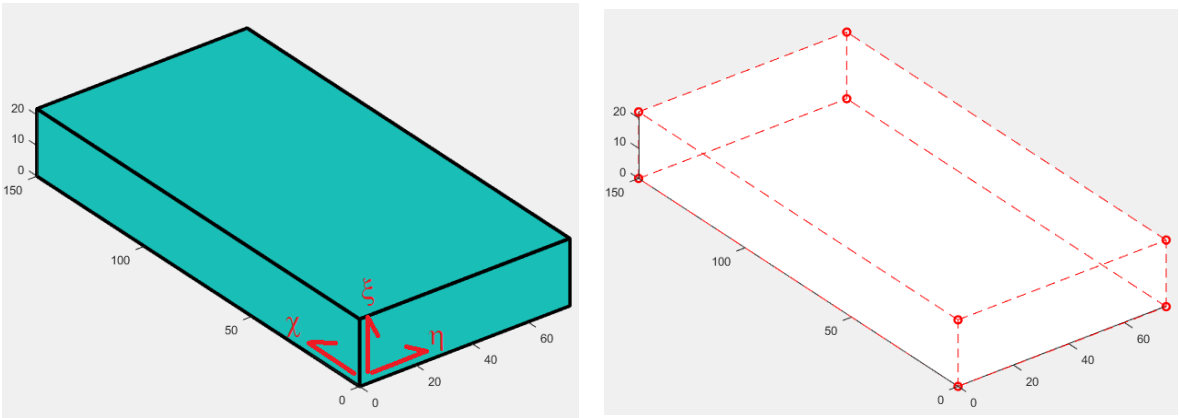


Fig. 8A.22 Parameter directions of the gross patch and initial control net.

Table 8A.4 Refinement.

		Initial	Refined
Number of control points	n	2	4
	m	2	4
	l	2	7
Degree	p	1	2
	q	1	2
	r	1	3
Knot vectors	Ξ	00 11	000 0.5 111
	H	00 11	000 0.5 111
	X	00 11	0000 0.25 0.50 0.75 1111

The *h-refinement* and *p-refinement* areas in the input form need to be as shown in **Fig. 8A.23**. As result the refinement the control points arrangements are as per **Fig. 8A.24**.

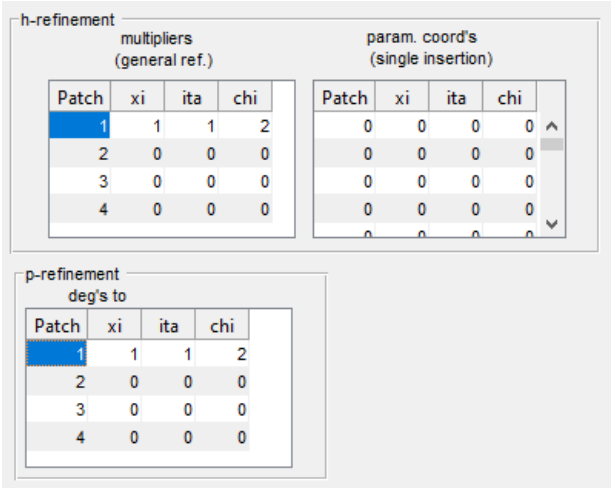


Fig. 8A.23 Refinement inputs required.

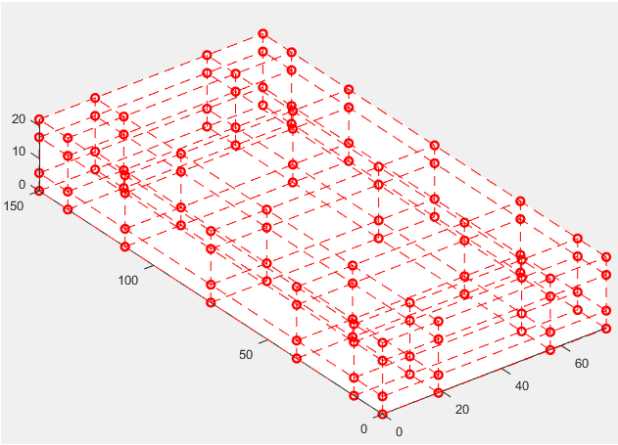


Fig. 8A.24 Refined control points arrangement.

8A.4 Example 8.3

Generation of domains in CAD

Gross patches are shown in **Fig. 8A.25**, with their allocated reference. For referencing, the RGB colour codes (see Appendix 4B) allocated to gross patches are 201-100-0 , 202-100-0 and 203-100-0.

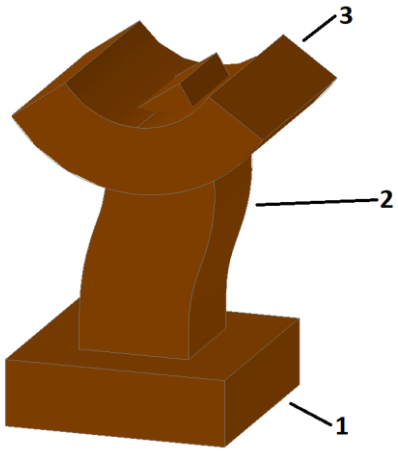


Fig. 8A.25 Gross patches.

Gross patch 1 is extruded from its contours as illustrated in **Fig. 8A.26** (left). Gross patch 3 is obtained by revolution of 110 degrees as shown in **Fig. 8A.26** (right).

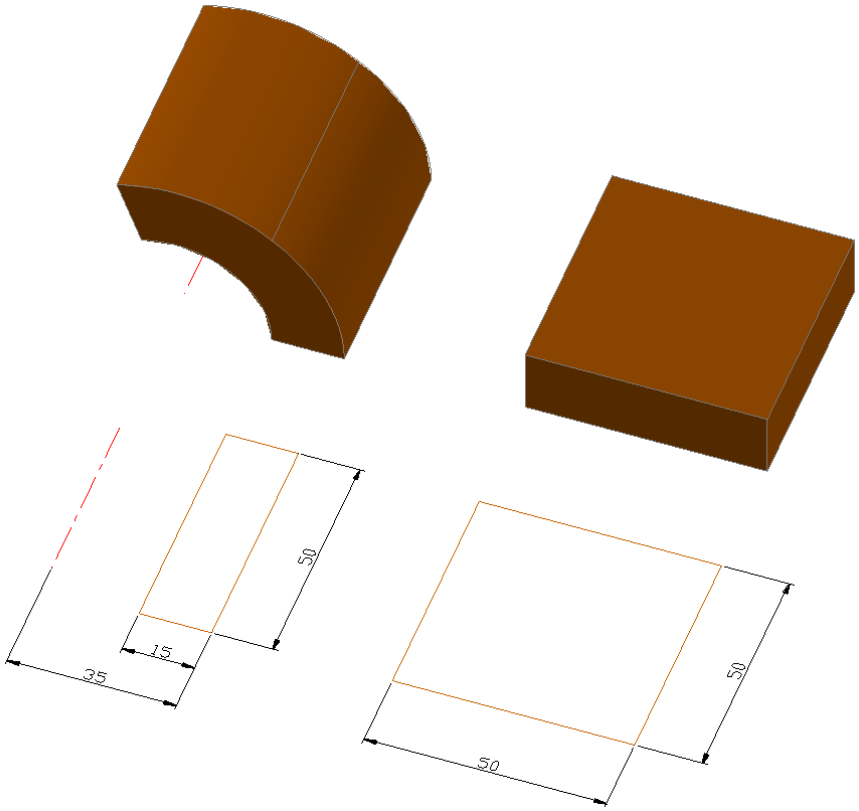


Fig. 8A.26 Generation of gross patches 1 and 3.

Gross patch 2 is constructed by sweep of the rectangular section along the axis line shown in **Fig. 8A.27**. The axis line is a plane cubic spline whose control points positions are given at left-hand side of **Fig. 8A.27**.

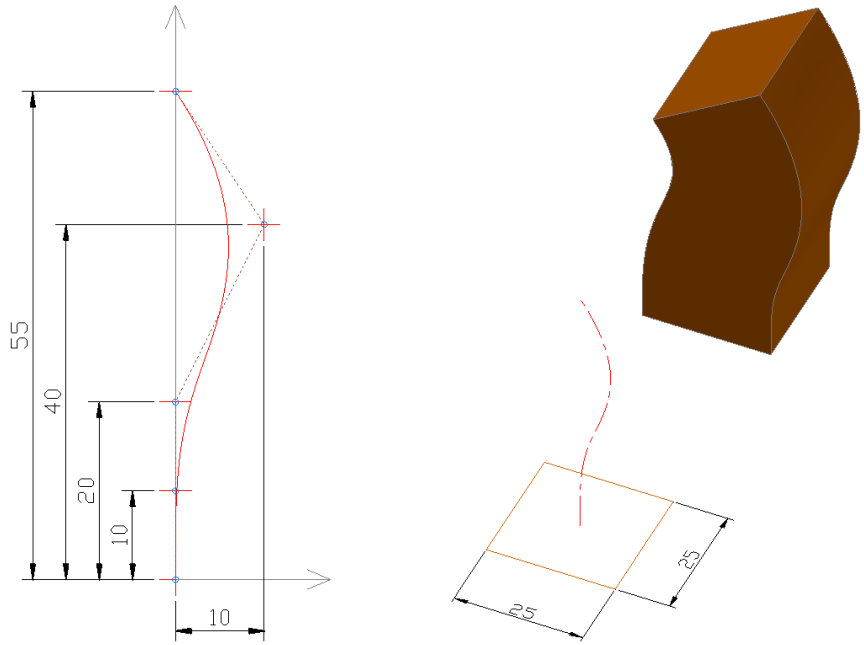


Fig. 8A.27 Construction of gross patches 2 by sweep.

Both patches are assembled as illustrated in **Fig. 8A.25**. The IGES file if this configuration is exported as the gross patches version with the name *G_Ex_10_3.igs*.

To achieve the final shape, and avoid patches intersections, the gross patches 1 and 2 need to be trimmed as shown in **Fig. 8A.28**. The trimmed domain, assembled as **Fig. 8A.29**, must occupy the same position as the gross domain in the CAD space.

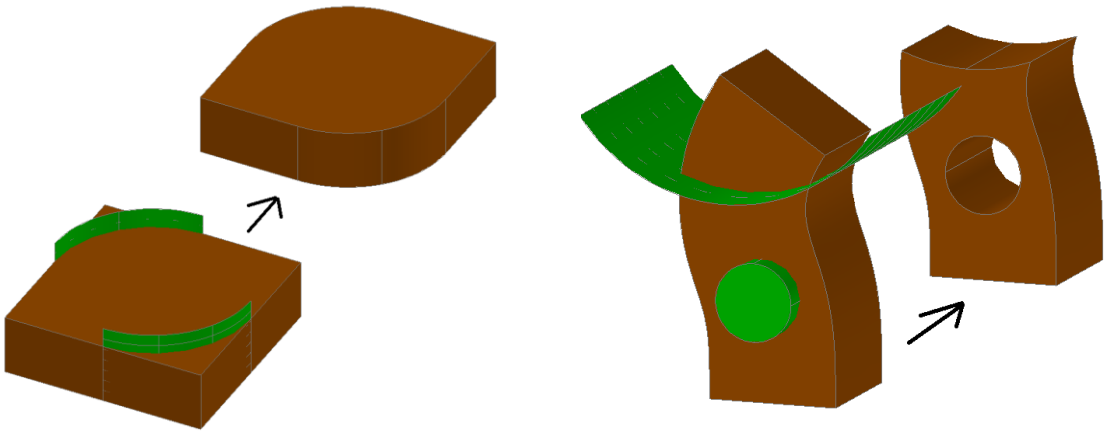


Fig. 8A.28 Trimming of patch 2 and assembling of the domain.

Fig. 8A.29 Trimmed domain.

The boundary surfaces are obtained by extraction of faces of the trimmed patches (see **Fig. 8A.30**). Surface 1 will be used to impose zero displacement, surfaces 2 and 3 will couple the patches, and surfaces 4 will be used to impose tractions. Boundary surfaces references are given by their RGB colour code (see Appendix 4B) as listed below:

Surface 1:	201-85-120
Surface 2:	202-85-120
Surface 3:	203-85-120
Surface 4:	204-85-120

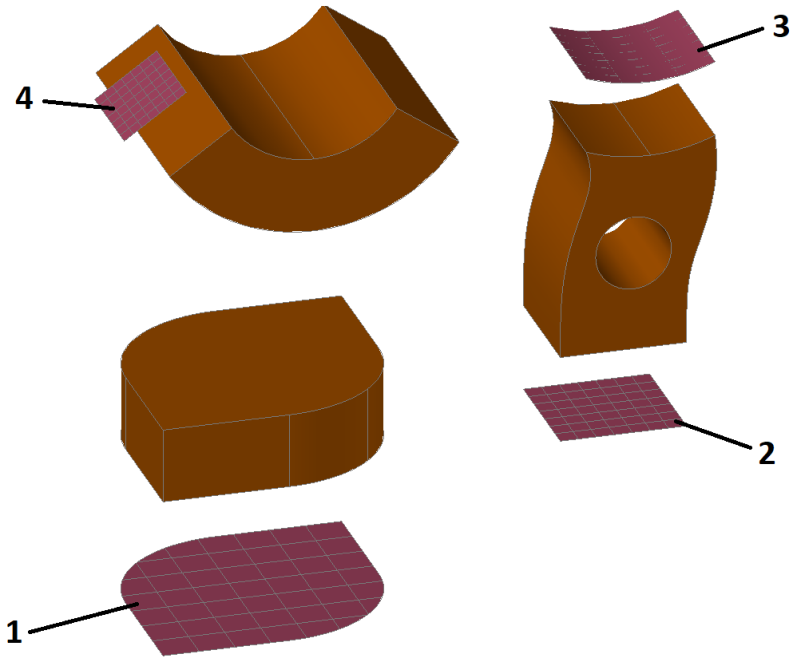


Fig. 8A.30 Boundary surfaces.

The boundary surfaces are moved onto the trimmed domain to achieve the bounded domain (see **Fig. 8A.31**). Recall that the bounded domain must hold the same position as the gross domain. The bounded domain is exported to the IGES file with the name *B_Ex_10_3.igs*.

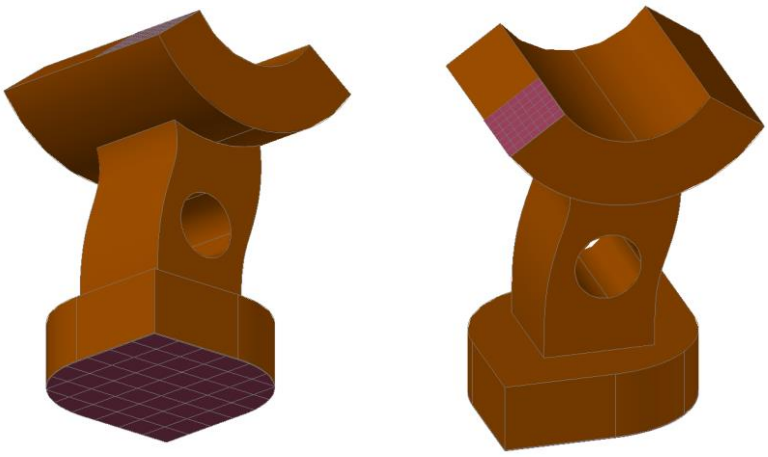


Fig. 8A.31 Bounded domain.

Preparation of patches for analysis

The `FirstRoutine` produces the parameterization of the gross patches as shown in **Fig. 8A.32** and their control points as **Fig. 8A.33**. In order to gain accuracy in the analysis the patches are refined as indicated in **Table 8A.5**, **Table 8A.6** and **Table 8A.7**. Let us remark two modifications that differ from the last two examples. For patch 2 the excessive number of control points in the third direction is reduced by knot removal. Since parameterization of patch 3 is not uniform in the second direction, one knot is inserted at 0.75 to achieve equally spaced knot spans.

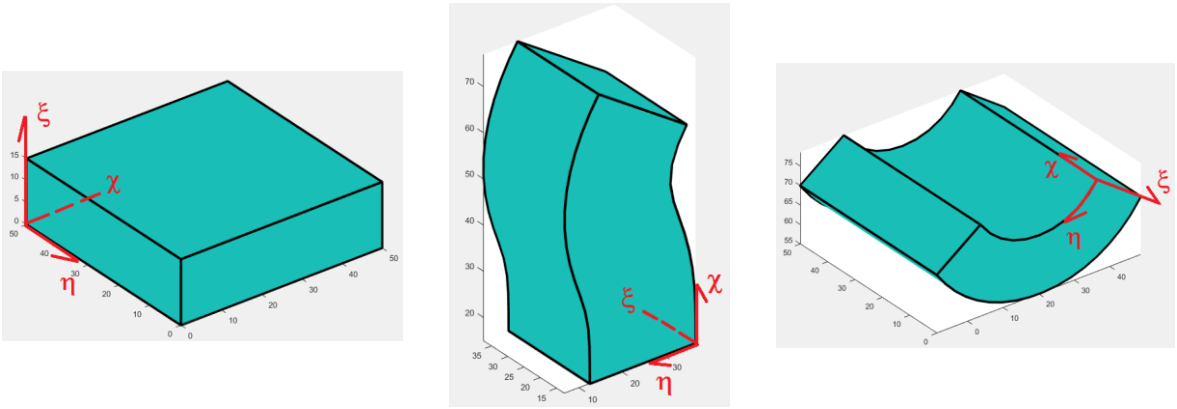


Fig. 8A.32 Parameter directions of the patches.

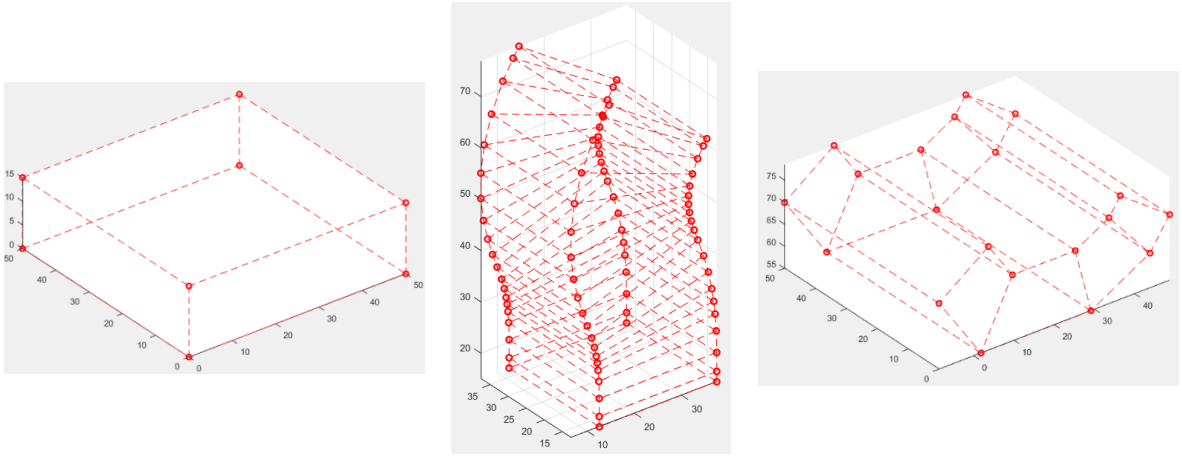


Fig. 8A.34 Initial control points arrangement.

Table 8A.5 Refinement in patch 1.

		Initial	Refined
Number of control points	n	2	4
	m	2	4
	l	2	4
Degree	p	1	2
	q	1	2
	r	1	2
Knot vectors	Ξ	00 11	000 0.5 111
	H	00 11	000 0.5 111
	X	00 11	000 0.5 111

Table 8A.6 Refinement in patch 2.

		Initial	Refined
Number of control points	n	2	4
	m	2	6
	l	20	13
Degree	p	1	2
	q	1	2
	r	3	3
Knot vectors	Ξ	00 11	000 0.5 111
	H	00 11	000 0.25 0.5 0.75 111
	X	0000 0.1 0.2 0.25 0.3 0.35 0.4 0.5 0.55 0.6 0.65 0.7 0.75 0.8 0.85 0.9 0.95 1111	0000 0.1 0.25 0.35 0.5 0.6 0.7 0.8 0.9 0.95 1111

Table 8A.7 Refinement in patch 3.

		Initial	Refined
Number of control points	n	2	4
	m	5	6
	l	2	6
Degree	p	1	2
	q	2	2
	r	1	2
Knot vectors	Ξ	00 11	000 0.5 111
	H	000 0.25 0.50 111	000 0.25 0.5 0.75 111
	X	00 11	000 0.25 0.5 0.75 111

The *h-refinement* and *p-refinement* areas in the input form need to be as shown in **Fig. 8A.35**. As result the refinement the control points arrangements are as per **Fig. 8A.36**.

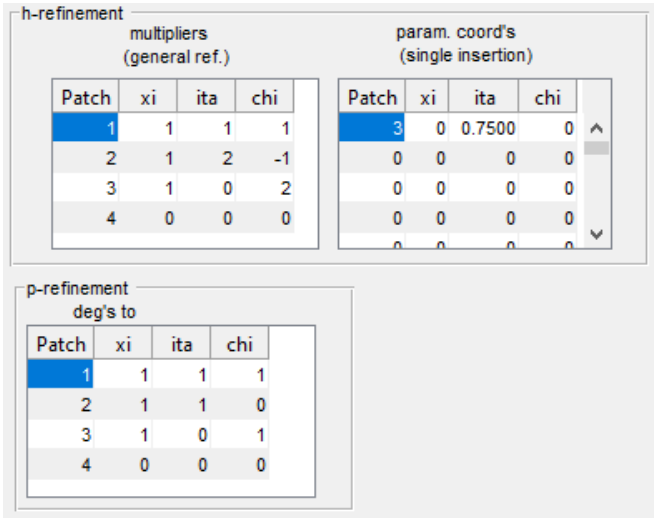


Fig. 8A.35 Refinement inputs required.

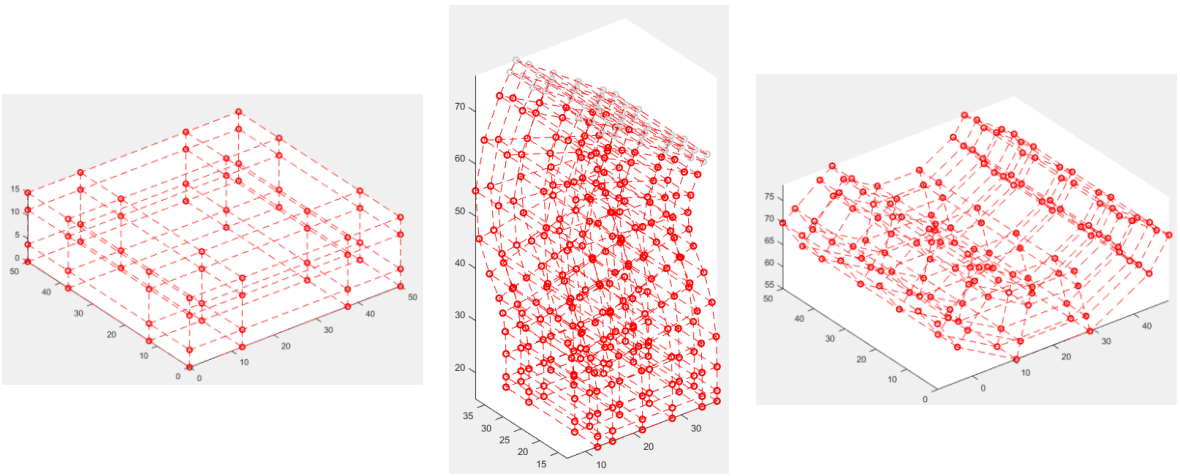


Fig. 8A.36 Final control points arrangement.

Appendix 10A: Marching Point Projection

The point projection (*PP*) technique finds the parameter coordinates corresponding to a position in the physical space (physical coordinates). *PP* applies to any NURBS domain: curve, surface or volume. We will refer to the NURBS domain as $\boldsymbol{H}$. *Main directions* refer to the orthogonal direction in the parameter space, *e.g.* for a surface the main directions are ξ and η, and for solids ξ, η and χ. The number of dimensions in parameter and physical spaces is denoted by c and d. The physical coordinates are denoted by $\boldsymbol{x}_t$, which are known. The target parameter coordinates are denoted by $\boldsymbol{\xi}_t$, which are unknown.

The *PP*, that is the reversal of the NURBS mapping, needs approximation since there is not an analytical solution for such process. This approximation is carried out by iterative procedures. The output of the *PP* is the estimated parameter location ${}^{e}\boldsymbol{\xi}_t$ whose physical coordinates are ${}^{e}\boldsymbol{x}_t$. The initial trial location is $\boldsymbol{\xi}_0$, that correspond to the $\boldsymbol{x}_0$ physical coordinates (see **Fig. 10A.1**). Then, the trial point is moved in each iteration closer to $\boldsymbol{\xi}_t$ by moving its location in the physical space an increment Δx. The process stops when one of the next two conditions is met:

- If $\boldsymbol{x}_t$ does not lie within the domain $\boldsymbol{H}$, the vector from ${}^{e}\boldsymbol{x}_t$ to $\boldsymbol{x}_t$, called $\boldsymbol{\Delta x}$, is *quasi-orthogonal* to $\boldsymbol{H}$ derivatives w.r.t. parameter directions. That implies the corresponding dot products $\boldsymbol{\Delta x} \cdot \boldsymbol{H}_{\xi i}$ are *close to zero*, in practise they are equal or less than a pre-established tolerance called *tol1* (see **Fig. 10A.1** top). This is called orthogonality condition.

- If $\boldsymbol{x}_t$ lies within the domain $\boldsymbol{H}$, the norm of the $\boldsymbol{\Delta x}$ vector is equal or less than a pre-established tolerance called *tol2*. In addition, the dot product referred above is *close to zero* (see **Fig. 10A.1** bottom). This is called coincidence condition.

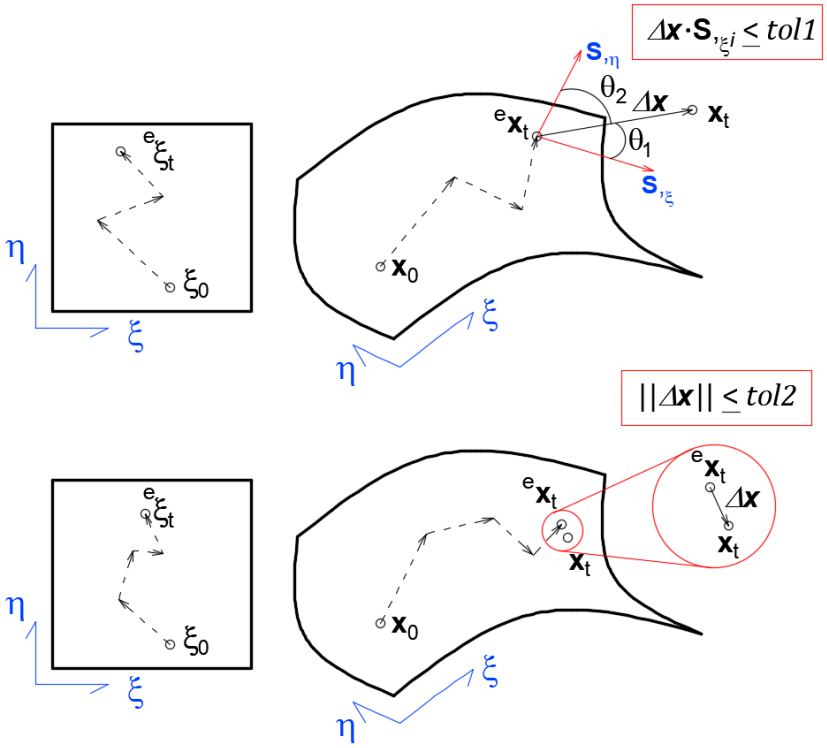


Fig. 10A.1 Point projection technique and conditions to stop the iterative location of $^e\boldsymbol{\xi}_t$. Above: orthogonality condition. Below: coincidence condition.

This appendix explains the Marching Point Projection (*MPP*) to find the NURBS parameter coordinates of one point with known physical coordinates. The *MPP* is based on the work done by Hoschek *et al.* (1993) that used it to find intersection of two surfaces. Here it is extended to solids and curves and it is compared to the traditional Newton-Raphson Iterations (*NRI*). The *MPP* overcomes the *NRI* in terms of robustness as demonstrated in the examples provided at the rear of this appendix.

10A.1 Preliminary concepts

Point projection on boundaries of the entity

When the target $\boldsymbol{x}_t$ lies outside domain and the number of parameter and physical dimensions is the same ($c = d$), *i.e.* plane surfaces and solids, the *PP* is applied on the boundary of the domain: boundary curves for plane surfaces, and boundary surfaces for solids, see **Fig. 10A.2**.

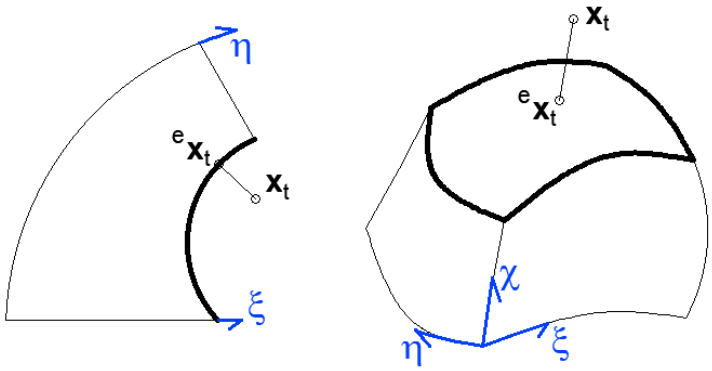


Fig. 10A.2 For points outside the domain, in case of plane surfaces or solids, the *PP* is done on the boundary of the domain instead the domain itself.

Directional derivatives in surfaces and solids

Let $\boldsymbol{\omega}$ be an orientation vector in the parameter space defined as equations (10A.1) and (10A.2) for surfaces and solids respectively, with the angles α, β and γ illustrated in **Fig. 10A.3**.

$$\boldsymbol{\omega} = \{cos\alpha \;\; cos\beta\}^T \tag{10A.1}$$

$$\boldsymbol{\omega} = \{cos\alpha \;\; cos\beta \;\; cos\gamma\}^T \tag{10A.2}$$

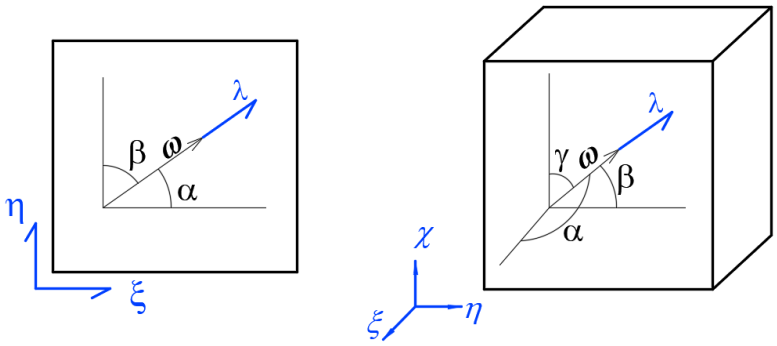


Fig. 10A.3 $\boldsymbol{\omega}$ vector in surface and solid parameter spaces, indicating orientation angles.

Let λ be the parameter direction with orientation $\boldsymbol{\omega}$. The directional derivative of $\boldsymbol{H}(\boldsymbol{\xi})$ w.r.t. λ, called $\boldsymbol{h}$, is given by equation (10A.3), where the dependency on $\boldsymbol{\xi}$ is removed for clarity. The partial derivatives $\boldsymbol{H}_{,\xi i}$ are along main directions, *e.g.* for a surface: $\boldsymbol{S}_{,\xi 2} = \boldsymbol{S}_{,\eta}$.

$$\boldsymbol{h} = \frac{d\boldsymbol{H}}{d\lambda} = \sum_{i=1}^{c} \boldsymbol{H}_{,\xi i} \;\; \omega_i \tag{10A.3}$$

The corresponding matrix form is given by equation (10A.4), where the $\boldsymbol{H}_{,\xi}$ matrix has d number of rows and c number of columns, *i.e.* it is composed of the main derivatives vectors as shown in equation (10A.5).

$$\boldsymbol{h} = \boldsymbol{H}_{,\xi}\ \boldsymbol{\omega} \tag{10A.4}$$

$$\boldsymbol{H}_{,\xi} = [\boldsymbol{H}_{,\xi 1} \quad \ldots \quad \boldsymbol{H}_{,\xi d}] \tag{10A.5}$$

Equations (10A.3) and (10A.4) show that the $\boldsymbol{h}$ vector is a linear combination of the main derivatives. **Fig. 10A.4** shows three examples.

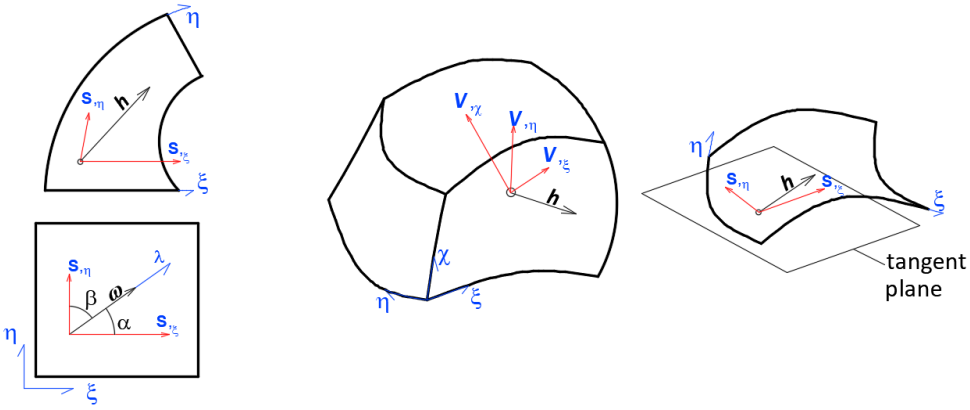


Fig. 10A.4 ***h*** vector for 2D surface, solid and 3D surface. In the last case ***h*** lies within the tangent plane. The parameter space is shown only for the first case for clarity.

Projected vector of increment in physical space

Let $\boldsymbol{\Delta x}$ be the vector defined in the physical space that goes from location $\boldsymbol{x}_1$ to $\boldsymbol{x}_2$. The projected vector $\boldsymbol{\Delta x}^p$ coincides with $\boldsymbol{\Delta x}$ if $c = d$, as illustrated in **Fig. 10A.5**. Otherwise, if $c < d$, $\boldsymbol{\Delta x}^p$ goes from $\boldsymbol{x}_1$ to ${\boldsymbol{x}_2}^p$, being the latter the projected location of $\boldsymbol{x}_2$ onto the tangent defined by the derivatives at $\boldsymbol{x}_1$. The tangent line in the curve is defined by the derivative at $\boldsymbol{x}_1$. The tangent plane in the surface is defined by the normal vector computed as cross product of main derivatives at $\boldsymbol{x}_1$ (see **Fig. 10A.6**).

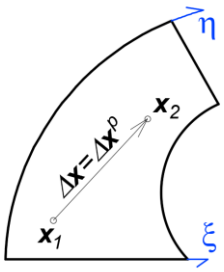


Fig. 10A.5 Projected vector $\boldsymbol{\Delta x}^p$ in a bi-dimensional surface ($c = d = 2$), in this case $\boldsymbol{\Delta x}^p$ coincides with $\boldsymbol{\Delta x}$.

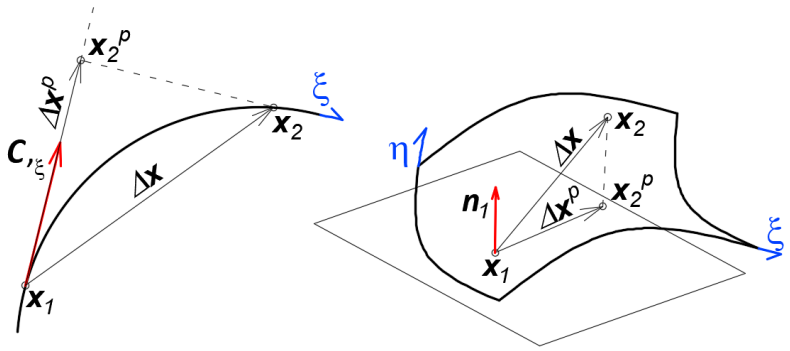


Fig. 10A.6 Projected vector $\Delta \boldsymbol{x}^p$ in a tri-dimensional curve ($c = 1, d = 3$) and in a tri-dimensional surface ($c = 2, d = 3$).

Orientation in the parameter space of a projected vector in physical space

The orientation vector $\boldsymbol{\omega}$ for $\boldsymbol{\Delta x}^p$ in the parameter space is computed as per equation (10A.6), whose unknowns are the components of $\boldsymbol{\omega}$. That expression comes from equation (10A.4), by substituting $\boldsymbol{h}$ with $\boldsymbol{\Delta x}^p$.

$$\boldsymbol{\omega} = \boldsymbol{H}_{,\xi}{}^{-1}\, \boldsymbol{\Delta x}^p \tag{10A.6}$$

Expression (10A.6) is a system of three equations for solids and two equations for plane surfaces. In case of three-dimensional surfaces ($c = 2$ and $d = 3$), expression (10A.6) has three equations with one of them dependent on the other two and, therefore, one of the three equations must be disregarded. In this work we remove arbitrarily the third one. Equation (10A.7) shows the particular case of tree-dimensional surfaces.

$$\boldsymbol{\omega} = \begin{bmatrix} S^x{}_{,\xi} & S^x{}_{,\eta} \\ S^y{}_{,\xi} & S^y{}_{,\eta} \\ S^z{}_{,\xi} & S^z{}_{,\eta} \end{bmatrix}^{-1} \begin{Bmatrix} \Delta x^{px} \\ \Delta x^{py} \\ \Delta x^{pz} \end{Bmatrix} \tag{10A.7}$$

As the third line is removed, both orientations are obtained as follows:

$$\boldsymbol{\omega} = \begin{bmatrix} S^x{}_{,\xi} & S^x{}_{,\eta} \\ S^y{}_{,\xi} & S^y{}_{,\eta} \end{bmatrix}^{-1} \begin{Bmatrix} \Delta x^{px} \\ \Delta x^{py} \end{Bmatrix} \tag{10A.8}$$

10A.2 The Marching Point Projection

In this section the marching point projection (*MPP*) formulation for curves, surfaces and solids is presented. The Newton-Raphson iterations (*NRI*) needs to be outlined, for completeness, because both methods are compared at the end if this appendix.

Point projection using Newton-Raphson method

Given a NURBS geometry $\boldsymbol{H}(\boldsymbol{\xi})$, the condition for one trial point $\boldsymbol{\xi}$ to be coincident with $\boldsymbol{\xi}_t$ is given by the set of equations (10A.9). This equation is the same expression given by (Piegl, Tiller 1996) but extended for any number of dimensions.

$$\boldsymbol{f}(\boldsymbol{\xi}) = \begin{cases} f_1(\boldsymbol{\xi}) = \boldsymbol{\Delta x} \cdot \boldsymbol{H}_{,\xi_1} = 0 \\ f_2(\boldsymbol{\xi}) = \boldsymbol{\Delta x} \cdot \boldsymbol{H}_{,\xi_2} = 0 \\ \vdots \\ f_n(\boldsymbol{\xi}) = \boldsymbol{\Delta x} \cdot \boldsymbol{H}_{,\xi_n} = 0 \end{cases} \tag{10A.9}$$

Where, $\boldsymbol{\Delta x} = \boldsymbol{x}_t - \boldsymbol{x}_0$ is the vector from $\boldsymbol{x}_0$ to $\boldsymbol{x}_t$ and $\boldsymbol{H}_{,\xi i}$ is the derivative of $\boldsymbol{H}$ w.r.t. *i*th parameter direction, e.g. for solids $\boldsymbol{H}_{,\xi 1} = \boldsymbol{V}_{,\xi}$ (see **Fig. 10A.7**).

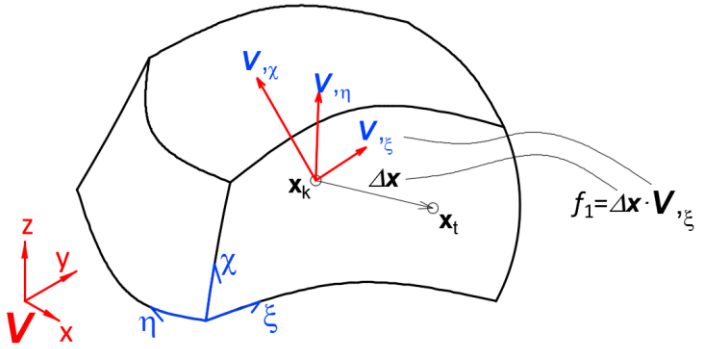


Fig. 10A.7 Vectors involved in equation (10A.9) for a target point within one volume $\boldsymbol{V}$ for the *k*th iteration.

The set of equations (10A.9) is a system of non-linear equations that may be solved by *NRI*, whose flowchart is shown in **Fig. 10A.8**. In each iteration the position is updated by adding the increment $\Delta\boldsymbol{\xi}$, as per equation (10A.10). Such increment is obtained as per equation (10A.11) and the Jacobian as per equation (10A.12). Dependencies on $\boldsymbol{\xi}_k$ are displayed in the last two equations to remark they are computed at the *k*th iteration. The iterative process stops when one of the two end conditions is met (recall section 10A.1).

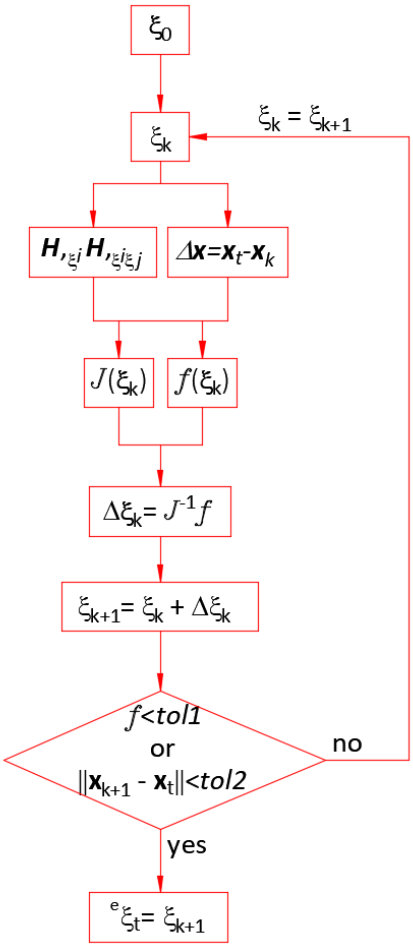


Fig. 10A.8 Flowchart for the Newton-Raphson iteration.

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k + \Delta\boldsymbol{\xi} \tag{10A.10}$$

$$\Delta\boldsymbol{\xi} = -\boldsymbol{\mathcal{J}}^{-1}(\boldsymbol{\xi}_k) \cdot \boldsymbol{f}(\boldsymbol{\xi}_k) \tag{10A.11}$$

$$\boldsymbol{\mathcal{J}}(\boldsymbol{\xi}_k) = \begin{bmatrix} \frac{\partial \mathrm{f}_1(\boldsymbol{\xi}_k)}{\partial \xi_1} & \frac{\partial \mathrm{f}_1(\boldsymbol{\xi}_k)}{\partial \xi_2} & \cdots & \frac{\partial \mathrm{f}_1(\boldsymbol{\xi}_k)}{\partial \xi_\mathrm{n}} \\ \frac{\partial \mathrm{f}_2(\boldsymbol{\xi}_k)}{\partial \xi_1} & \frac{\partial \mathrm{f}_2(\boldsymbol{\xi}_k)}{\partial \xi_2} & \cdots & \frac{\partial \mathrm{f}_2(\boldsymbol{\xi}_k)}{\partial \xi_\mathrm{n}} \\ \vdots & \vdots & & \vdots \\ \frac{\partial \mathrm{f}_n(\boldsymbol{\xi}_k)}{\partial \xi_1} & \frac{\partial \mathrm{f}_n(\boldsymbol{\xi}_k)}{\partial \xi_2} & \cdots & \frac{\partial \mathrm{f}_n(\boldsymbol{\xi}_k)}{\partial \xi_\mathrm{n}} \end{bmatrix} \tag{10A.12}$$

Each component for the matrix of equation (10A.12) is obtained as follows:

$$\mathcal{J}_{ij} = \boldsymbol{H}_{,\xi_i} \cdot \boldsymbol{H}_{,\xi_j} + \Delta\boldsymbol{x} \cdot \boldsymbol{H}_{,\xi_i\xi_j} \tag{10A.13}$$

The Marching Point Projection

The directional derivative of $\boldsymbol{H}(\boldsymbol{\xi})$ w.r.t. λ is given by equation (10A.3). Note that for curves, equation (10A.3) is reduced to equation (10A.14).

$$\boldsymbol{C}_{,\xi} = \frac{d\boldsymbol{C}}{d\xi} \tag{10A.14}$$

The length of one infinitesimal increment in the parameter space $(d\lambda)$ corresponding to one infinitesimal increment on the NURBS physical space $d\boldsymbol{H}$ is calculated as follows:

$$d\lambda = \frac{\|d\boldsymbol{H}\|}{\|\boldsymbol{h}\|} \tag{10A.15}$$

To estimate one increment in the parameter space $(\Delta\lambda)$ corresponding to one path length within the domain $\mathrm{L}_{\boldsymbol{H}}$, the approximation from equation (10A.16) might be used, see **Fig. 10A.9**. If the parametrization was constant the increment obtained would be exact, but it is not the case in general and hence the obtained value in (10A.16) is an approximation.

$$\Delta\lambda \cong \frac{\mathrm{L}_{\boldsymbol{H}}}{\|\boldsymbol{h}\|} \tag{10A.16}$$

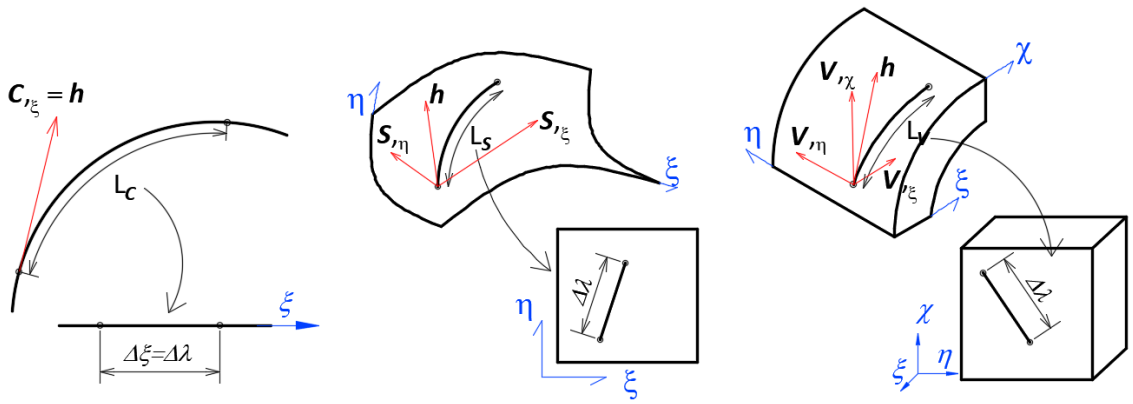


Fig. 10A.9 Estimation of the incremental parameter $\Delta\lambda$ corresponding to the length $\mathrm{L}_{\boldsymbol{H}}$ in the physical space, for curves, surfaces and solids.

Now, let us define one location in the physical space (within the domain or outside) as $\boldsymbol{x}_t$. To estimate its parameter coordinate $\boldsymbol{\xi}_t$ we use one initial trial point within the domain $\boldsymbol{x}_0$ with known parameter coordinates $\boldsymbol{\xi}_0$. Then the increment to *travel* from $\boldsymbol{\xi}_0$ to $\boldsymbol{\xi}_t$, whose exact value is $\Delta\lambda = \|\boldsymbol{\xi}_t - \boldsymbol{\xi}_0\|$, can be estimated as follows:

$$\Delta\lambda \cong \frac{\|\boldsymbol{\Delta x}^p\|}{\|\boldsymbol{h}\|} \tag{10A.17}$$

Where $\boldsymbol{\Delta x}^p$ is the increment vector from $\boldsymbol{x}_0$ to $\boldsymbol{x}_t$ as explained in section 10A.1. The increment in each main direction may be obtained from $\Delta\lambda$ as shown in equation (10A.18). The orientation $\boldsymbol{\omega}$ of the $\boldsymbol{\Delta x}^p$ vector is calculated by equations (10A.6).

$$\Delta\xi_i = \Delta\lambda\ \omega_i \tag{10A.18}$$

Equation (10A.17) introduces one additional error with respect to equation (10A.16) because the projected distance $\|\boldsymbol{\Delta x}^p\|$ does not coincide with $\mathrm{L}_{\boldsymbol{H}}$. However, the accuracy is enough to use this estimation in an iterative manner, by summing in each iteration the computed increment to the previous value, as shown in equation (10A.19) for the *k*th iteration. The increments in each parameter direction are obtained from (10A.18). The flow chart for the *MPP* is depicted in **Fig. 10A.10**. The iterative process stops when one of the two end conditions is met (section 10A.1).

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k + \boldsymbol{\Delta\xi} \tag{10A.19}$$

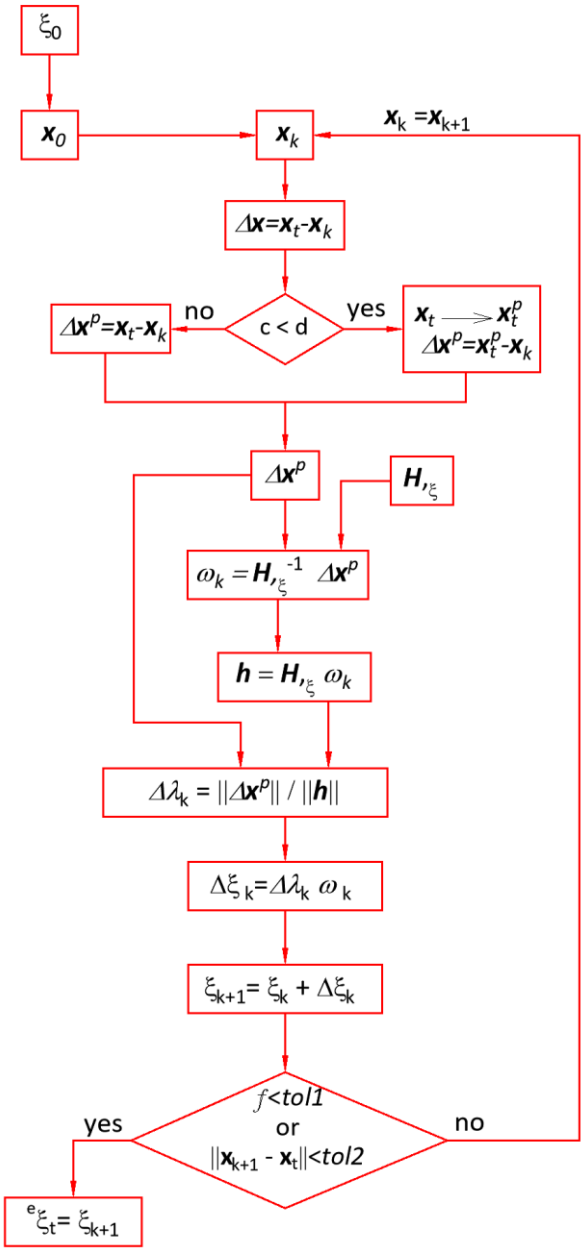


Fig. 10A.10 Flowchart for the marching point projection.

10A.3 Numerical examples

Both methods, *MPP* and *NRI* are compared in this section in terms of efficiency and robustness. These two terms are defined below and illustrated in **Fig. 10A.11** with a surface example.

- Efficiency: number of iterations to achieve the estimated value $\boldsymbol{\xi}_t$. The lower number of iterations the more efficiency.
- Robustness: portion of domain where the initial estimation converges to $\boldsymbol{\xi}_t$, expressed in percentage w.r.t. domain. This portion is called the *convergence portion*. The larger convergence portion the more robustness.

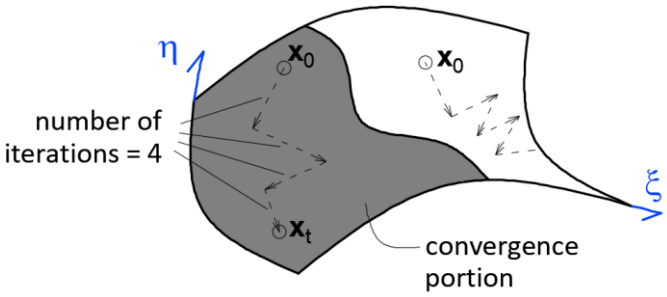


Fig. 10A.11 *PP* from two initial trial points ($\boldsymbol{x}_0$). If $\boldsymbol{x}_0$ lies within the convergence portion (the left-shaded portion) the approximation converges, otherwise not ($\boldsymbol{x}_0$ in the not shaded portion).

To carry out the comparison, one target location $\boldsymbol{x}_t$ is selected and a net of points $\boldsymbol{\xi}_{net}$ equally spaced over the whole parameter domain is defined (see **Fig. 10A.12**). The *PP* is done from each of these net points as initial trial location $\boldsymbol{\xi}_0$, with both methods: *NRI* and *MPP*, storing the number of iterations to achieve the target. It is considered that the iteration converges if the next two conditions are met:

- One of the convergence conditions is achieved (section 10A.1).
- The number of iterations is equal or less than a limit value (in this work it is 12 iterations).

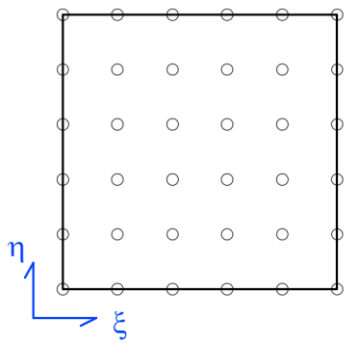


Fig. 10.12 Net of trial points for one surface.

To compare the results, the *iteration-convergence diagram* is used (**Fig. 10A.13** shows one surface example). This diagram plots in the parameter space vertical bars, whose bases lie on those $\boldsymbol{\xi}_{net}$ points. Only the vertical bars from those trial points that converge are drawn, *i.e.* the trial points that lie within the convergence portion. The height of the bars is proportional to the number of iterations at each location. Then, the diagram provides:

- Which initial trial locations converge. Their percentage w.r.t. total number of $\boldsymbol{\xi}_{net}$ points is the convergence portion. That is a measurement of the robustness.
- For each convergent point, the number of iterations required, that is a measurement of the efficiency.

For solids, the diagram shows the locations of the $\boldsymbol{\xi}_{net}$ that converge but not the number of iterations. However, these numbers are reflected in tables in the examples. There is one diagram for each iteration method: *NRI* and *MPP*.

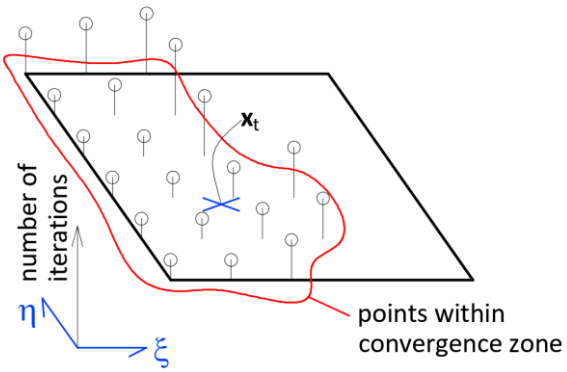


Fig. 10.13 Iteration-convergence diagram for one surface and target point $\boldsymbol{x}_t$.

Curves

The curve NURBS features are presented in **Table 10A.1**.

The target is $\boldsymbol{x}_t = (19.2124, 5.2983, 12.2554)$ and the net of sample points has 0.05 increment in the parameter space, *i.e.* there are 21 sample points.

The tolerance in the physical space is *tol2=0.01.* Tolerance of orthogonality (*tol1*) does not apply because the target point lies within the curve. The limit number of iterations is 12.

Table 10A.1 Curve NURBS features

Number of control points	10
Degree	3
Knot vector	0 0 0 0 0.1 0.3 0.55 0.7 0.8 0.9 1 1 1 1
Control points weights	1 1 0.5 1 0.5 1 0.5 1 0.5 1

Control points coordinates	0 0 0 ; 10 0 5; 20 0 10 ; 20 10 15 ; 20 20 20 ; 10 20 25 ; 0 20 30 ; 0 10 35 ; 0 0 40 ; 10 0 45

Fig. 10A.14 shows the curve in the physical space with the control net and the target location. **Fig. 10A.15** illustrates the iteration-convergence diagrams for *NRI* and *MPP*. Robustness and average number of iterations for each method are presented in **Table 10A.2**, which shows a better performance of *MPP*.

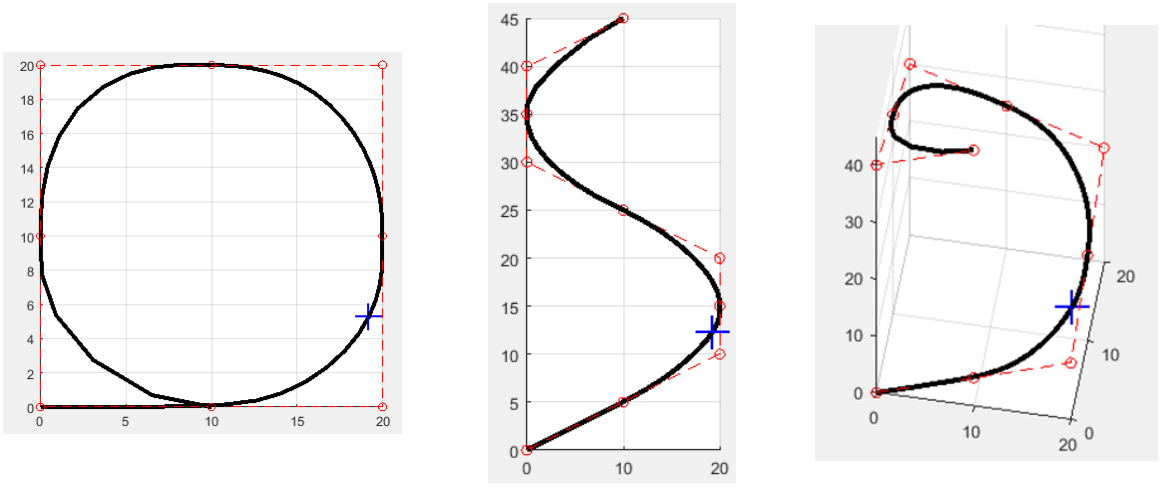


Fig. 10A.14 Views of the curve (plan, elevation and 3D) and the location of the target point $\boldsymbol{x}_t$ (blue cross).

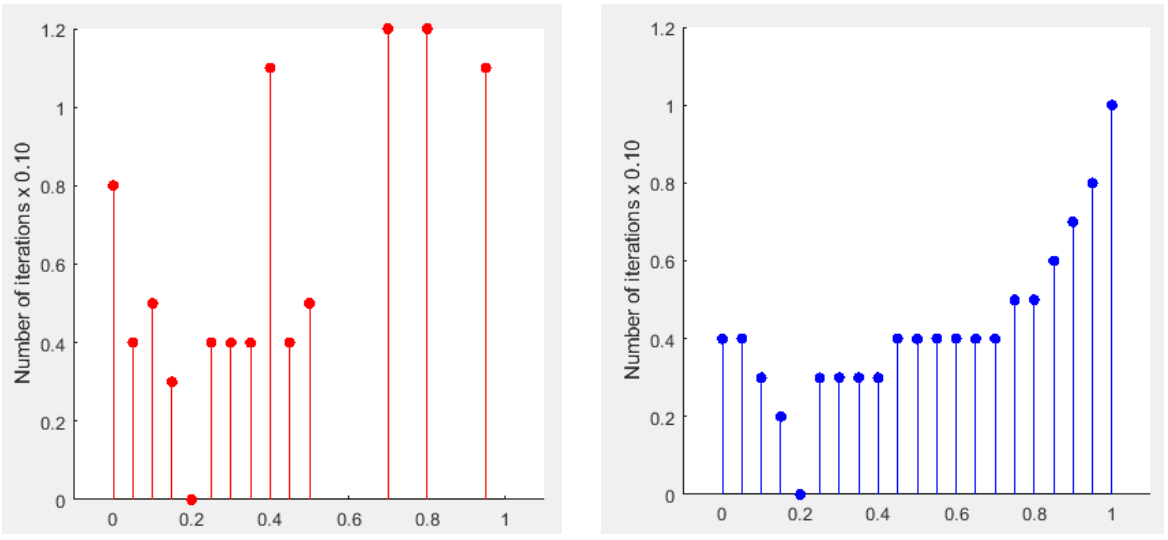


Fig. 10A.15 Iteration-convergence diagrams for *NRI* (left) and *MPP* (right). Number of iterations is scaled by 0.10.

Table 10A.2 Performance comparison for curve

	NRI	*MPP*
Robustness (%)	67	100
Number of iterations (average)	6.2	4.3

Surfaces

The NURBS features for the surface are presented in **Table 10A.3**.

The target is $\boldsymbol{x}_t = (11.6892, 12.5000, 13.0744)$ and the net of sample points has 0.125 increment in each parameter direction, i.e. there are 9x9=81 sample points.

The tolerance in the physical space is *tol2=0.01*. Tolerance of orthogonality (*tol1*) does not apply because the target point lies within the surface. The limit of number of iterations is 12.

Table 10A.3 Surface NURBS features

Number of control points	5 2
Degree	2 1
Knot vector	0 0 0 0.5 0.5 1 1 1 0 0 1 1
Control points weights	1 1 0.707 0.707 1 1 0.707 0.707 1 1
Control points coordinates	10 0 0 ; 0 50 0 ;10 0 20 ; 0 50 30 ;30 0 20 ; 30 50 30 ;50 0 20 ; 60 50 30 ; 50 0 0 ; 60 50 0

Fig. 10A.16 shows the surface in the physical space with the control net and the target location included. **Fig. 10A.17** illustrates the iteration-convergence diagrams for *NRI* and *MPP*. Robustness and average number of iterations for each method is presented in **Table 10A.4**, which shows greater robustness for *MPP*.

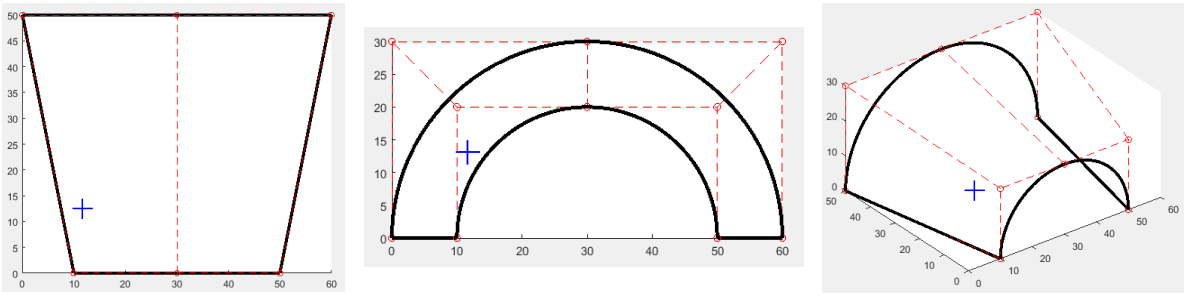


Fig. 10A.16 Views of the surface (plan, elevation and 3D) and the location of the target point $\boldsymbol{x}_t$ (blue cross).

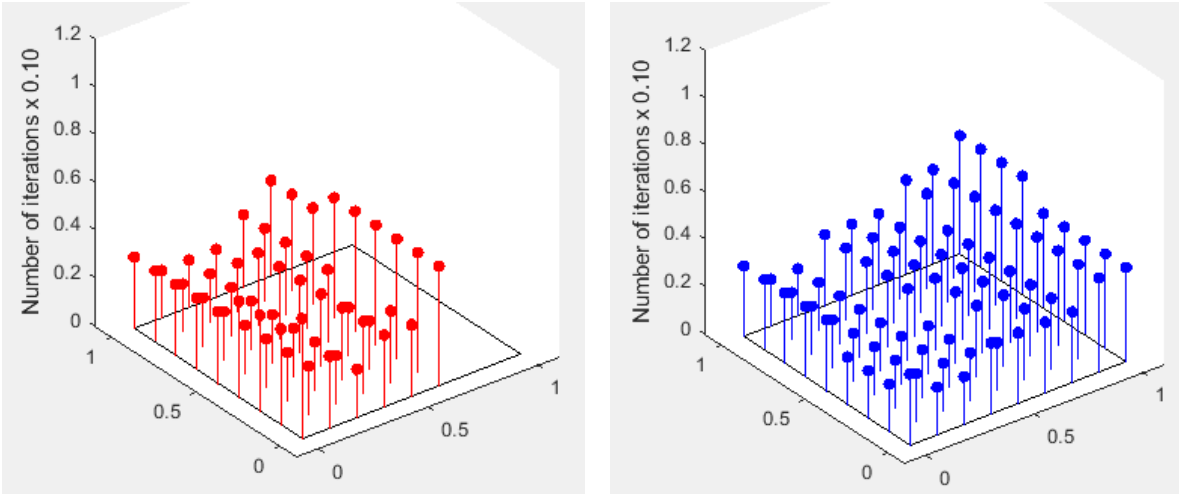


Fig. 10A.17 Iteration-convergence diagrams for *NRI* (left) and *MPP* (right). Number of iterations is scaled by 0.10.

Table 10A.4 Performance comparison for surface

	NRI	*MPP*
Robustness (%)	67	100
Number of iterations (average)	3.0	3.0

Solids

The NURBS features for the solid are presented in **Table 10A.5**. The target is $\boldsymbol{x}_t = (34.7893,\ 5.0000,\ 7.1331)$ and the net of sample points has 0.125 increment in each direction of the parameter space, i.e. there are 9x9x9=729 sample points.

The tolerance in the physical space is *tol2=0.01*. Tolerance of orthogonality (*tol1*) does not apply because the target point lies within the solid. The limit of number of iterations is 12.

Table 10A.5 Volume NURBS features

Number of control points	3 3 7
Degree	2 2 2
Knot vector	0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0.333 0.333 0.667 0.667 1 1 1
Control points weights	1 1 0.707 1 0.707 1 1 1 1 0.707 1 0.707 1 1 1 1 0.707 1 0.707 1 1 1 1 0.707 1 0.707 1 1 1 1 0.707 1 0.707 1 1 1 1 0.707 1 0.707 1 1 1 1 0.707 1 0.707 1 1 1 1 0.707 1 0.707 1 1 1 1 0.707 1 0.707 1 1
Control points coordinates	0 0 20; 0 0 40; 20 0 40; 40 0 40; 40 0 20; 40 0 0; 20 0 0; 0 10 20; 0 10 40; 20 10 40; 40 10 40; 40 10 20; 40 10 0; 20 10 0; 0 20 20; 0 20 40; 20 20 40; 40 20 40; 40 20 20; 40 20 0; 20 20 0; 5 0 20; 5 0 35; 20 0 35; 35 0 35; 35 0 20; 35 0 5; 20 0 5; 5 10 20; 5 10 35; 20 10 35; 35 10 35; 35 10 20; 35 10 5; 20 10 5; 5 20 20; 5 20 35; 20 20 35; 35 20 35; 35 20 20; 35 20 5; 20 20 5; 10 0 20; 10 0 30; 20 0 30; 30 0 30; 30 0 20; 30 0 10; 20 0 10; 10 10 20; 10 10 30; 20 10 30; 30 10 30; 30 10 20; 30 10 10; 20 10 10; 10 20 20; 10 20 30; 20 20 30; 30 20 30; 30 20 20; 30 20 10; 20 20 10

Fig. 10A.18 shows the volume in the physical space with the control net and the target location included. **Fig. 10A.19** illustrates the convergence diagrams for *NRI* and *MPP*. Robustness and

average number of iterations for each method is presented in **Table 10A.6**, which shows a greater robustness for the *MPP* method.

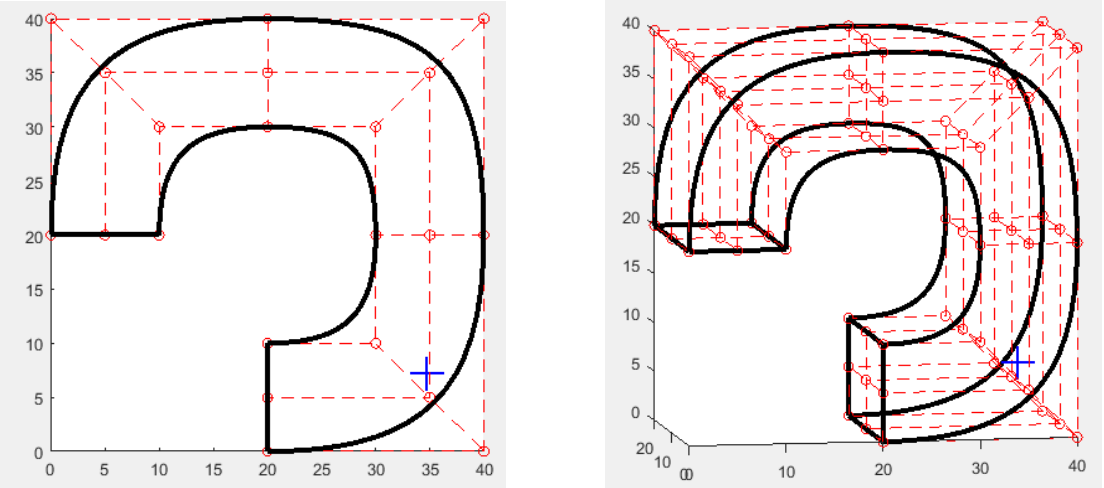


Fig. 10A.18 Views of the volume (elevation and 3D) and the location of the target point $\boldsymbol{x}_t$ (blue cross).

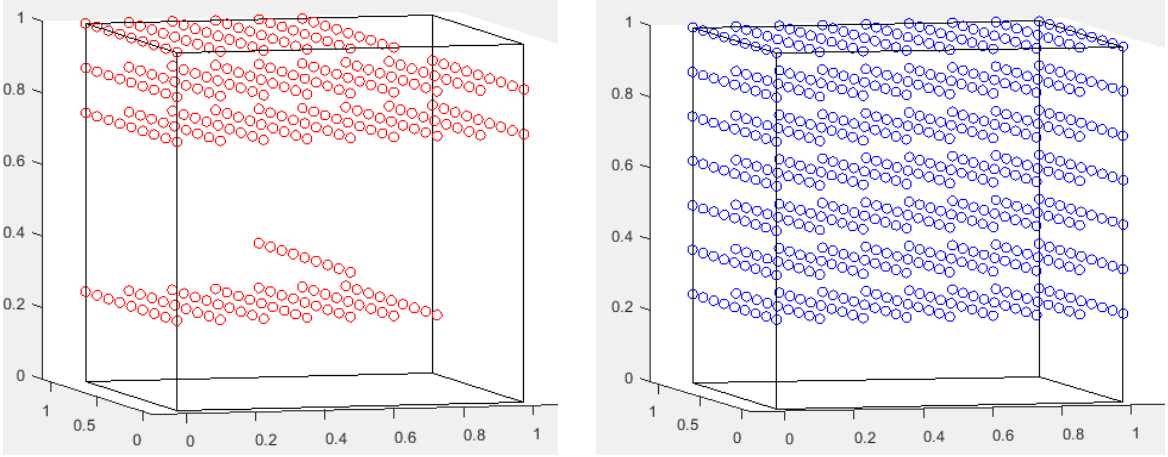


Fig. 10A.19 Diagrams showing sample points that converge, for *NRI* (left) and *MPP* (right).

Table 10A.6 Performance comparison for surface

	NRI	*MPP*
Robustness (%)	40	78
Number of iterations (average)	4.2	4.2

Appendix 10B: Quasi-isotropic Initial Triangulation

The triangulation of NURBS surfaces is required for their representation, either geometry or stress fields. Indeed to visualize a surface we need to define a set of points (called vertexes or nodes) lying on the surface and joint them by lines forming polygons, typically quadrilaterals or triangles. The use of triangles is easier to implement given the well-known Delaunay triangulation technique. To increase the quality of the representation, equilateral triangles are preferable which form what is called isotropic triangulation. In the isotropic triangulation all angles equal to 60 degrees and all vertexes valences equal to 6 (see Appendix 5A).

The Quasi-isotropic Initial Triangulation (*QIT*) discretizes the NURBS surfaces into equilateral-same size triangles, where they are not affected by the contours. The triangles distorted by the contours are adapted to mitigate such distortion. The resultant mesh is quasi-isotropic (refer to Appendix 5B) and it is obtained at once, without preliminary mesh.

In the *QIT* algorithm the vertexes are located in a wave propagation manner, starting form one initial node (chosen arbitrarily within the surface) and moving divergently to next outer contour. **Fig. 10B.1** illustrates one example. The length of the side of the triangles is called R, that is defined in the physical space.

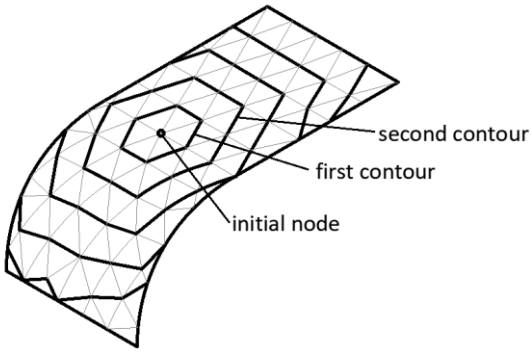


Fig. 10B.1 Positioning of vertexes in the QIT algorithm.

This appendix outlines the main ideas of the *QIT* algorithm that was presented by Adan and Cardoso (2020). For further details refer to such paper.

One may think that smaller triangles are only needed where curvature is pronounced, with triangles in plane zones larger. That is true if the triangulation is devised to represent only geometry. However, in this work the triangulation is used to plot the stress field as well. Therefore, the resolution of the triangulation must be able to capture the stress field profile regardless of the shape of the surface. This necessity is illustrated in **Fig. 10B.2**, where the zone of high stresses lies on a plane area of the surface. At the left-hand side the triangulation is uniform (as *QIT* triangulation) and the peak is captured in the representation. By contrast, at the right-

hand side the triangulation follows exclusively curvature criterion, having the plane zone only a few large triangles. These triangles are not able to represent the stress peak.

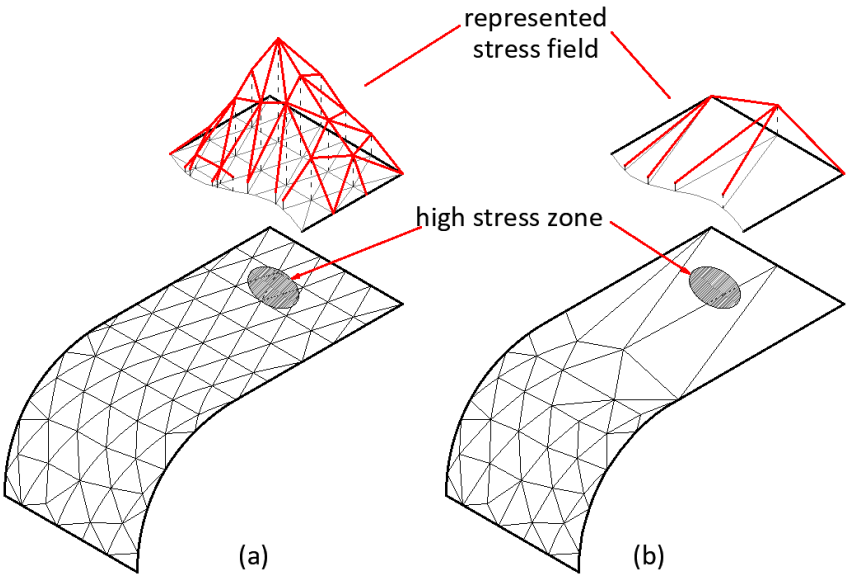


Fig. 10B.2 Stress peak represented by a uniform triangulation (a) and triangulation based on curvature of the surface (b).

The divergent propagation of vertexes is illustrated in **Fig. 10B.3**. Each vertex is computed by intersection of two arcs which are centred on vertexes that belong to the previous contour and the with arc length equal to R.

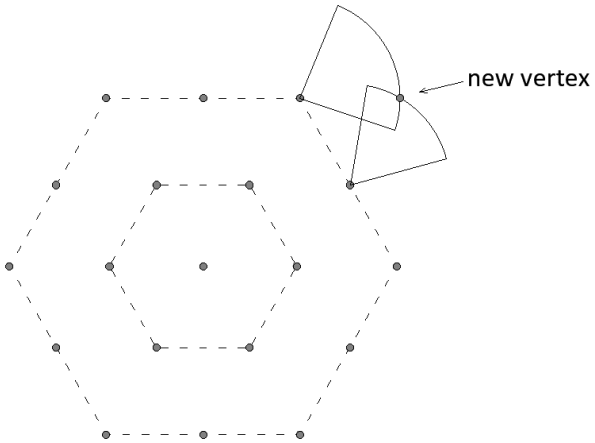


Fig. 10B.3 Calculation of vertexes of the triangulation by arcs intersections.

Since the contours are plotted in the physical space, regardless the shape of the surface, the theoretical arcs are not plane but lie onto such surface, being unknown. Therefore, it is necessary to discretize the arcs into segments.

The intersection between the theoretical arcs is then found by iterations. In each iteration the intersection between the segments is computed, the arc angle is reduced, the arc is discretized again and new segments intersection is computed. **Fig. 10B.4** illustrates this process. Note the

intersection can be found only in the parameter space of the surface since it is bi-dimensional. In the physical space (tri-dimensional) that would be impossible.

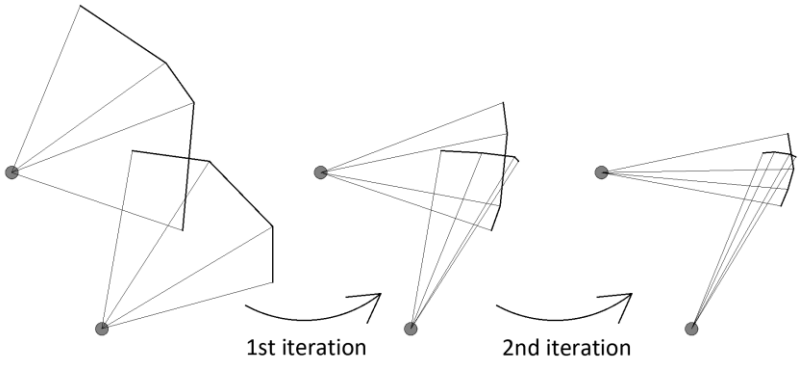


Fig. 10B.4 Estimation of arcs intersection by iterations using discretized arcs in the parameter space.

To compute the location of each point of the discretized arcs we integrate the length on the physical space using the trapezoidal rule to achieve R distance. The error of the trapezoidal rule depends on the length to integrate. To control such error the surface parameter space is divided into portions such that the trapezoidal rule within each one does not yield an error greater than a pre-established value. We also need the orientation to localize such points, which is inferred using a new space: the pattern space.

The pattern space is a bi-dimensional arrangement of concentric hexagons separated a distance equal to $R\ sin60$ (see **Fig. 10B.5**). The vertexes lie on these hexagons spaced at R. The pattern space leads naturally to an isotropic triangulation. The pattern space is the *guidance* followed by the *QIT* to compute the vertexes location.

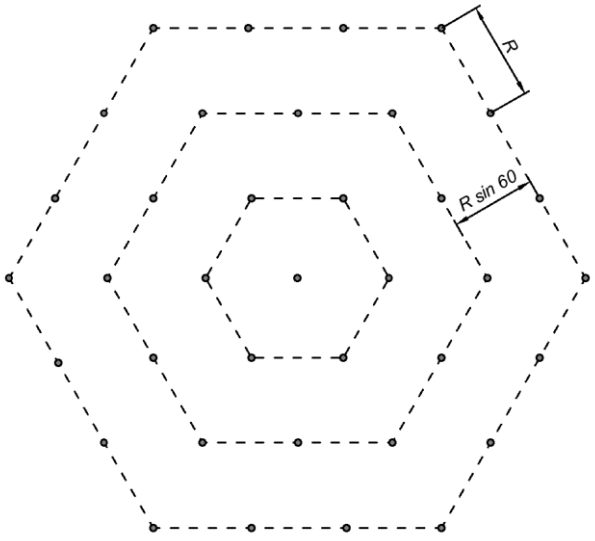


Fig. 10B.5 The pattern space.

The shape of the arcs in the parameter space depends on the parametrization of the surface and are unknown. The estimation of the correct orientation of the arcs is crucial to reduce the number of iterations, hence the importance of the pattern space.

The hexagonal contours are concentric and equally space in the pattern space. In the parameter space they are distorted according to the surface parametrization. In the physical space the resultant contours are the image of the pattern space. In other words, if we extract the hexagons from the surface on physical space and *unroll* to a plane, we recover the pattern space.

The *QIT* can be seen as a transfer of the pattern space to the physical space using the parameter space to localise the vertexes, as depicted in **Fig. 10B.6**.

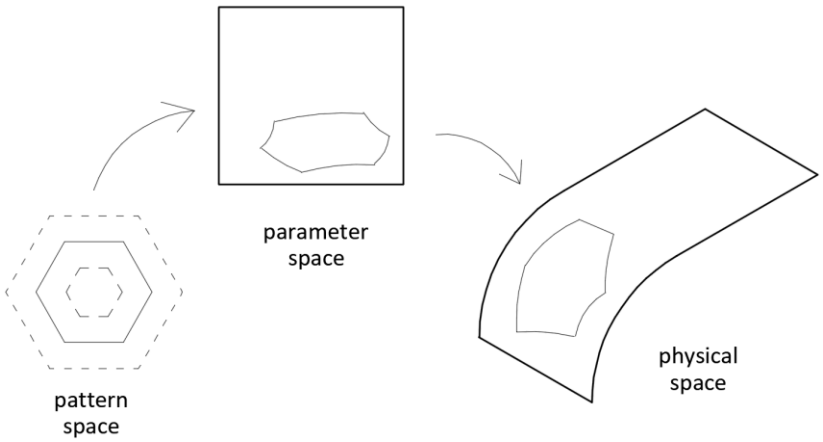


Fig. 10B.6 Transfer of the contours from the pattern space to the physical space using the parameter space.

With the vertexes computed the triangulation is carried out in the pattern space, which leads to an isotropic triangulation as mentioned. This isotropy will be reflected in the physical space. Only at the edge zones some distortion appears, *i.e*. it is a quasi-isotropic triangulation.

References

References:

Adan, D.H. and Cardoso, R.P. (2020) Quasi-isotropic initial triangulation of NURBS surfaces. *European Journal of Computational Mechanics* [online]. (Accepted) .

Antolin, P., Buffa, A. and Martinelli, M. (2019) Isogeometric Analysis on V-reps: first results. *arXiv Preprint arXiv:1903.03362* [online].

Apostolatos, A., Schmidt, R., Wüchner, R. and Bletzinger, K. (2014) A Nitsche-type formulation and comparison of the most common domain decomposition methods in isogeometric analysis. *International Journal for Numerical Methods in Engineering* [online]. 97 (7), pp.473-504.

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

Auricchio, F., Da Veiga, L.B., Hughes, T., Reali, A. and Sangalli, G. (2010) Isogeometric collocation methods. *Mathematical Models and Methods in Applied Sciences* [online]. 20 (11), pp.2075-2107.

Auricchio, F., Calabrò, F., Hughes, T.J.R., Reali, A. and Sangalli, G. (2012) A simple algorithm for obtaining nearly optimal quadrature rules for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*. 249-252 pp.15-27.

Babuška, I. (1973) The finite element method with penalty. *Mathematics of Computation* [online]. 27 (122), pp.221-228.

Baker, T.J. (2005) Mesh generation: Art or science? *Progress in Aerospace Sciences* [online]. 41 (1), pp.29-63.

Bathe, K. (2006) *Finite Element Procedures* [online]. Klaus-Jurgen Bathe.

Bauer, I., Catanese, F. and Pignatelli, R. (2011) Surfaces of general type with geometric genus zero: a survey. In: (2011) *Complex and Differential Geometry* [online]. Springer, pp.1-48.

Bazilevs, Y., Calo, V.M., Cottrell, J.A., Evans, J.A., Hughes, T.J.R., Lipton, S., Scott, M.A. and Sederberg, T.W. (2010) Isogeometric analysis using T-splines. *Computer Methods in Applied Mechanics and Engineering* [online]. 199 (5), pp.229-263.

Bazilevs, Y., Hsu, M. and Scott, M. (2012) Isogeometric fluid-structure interaction analysis with emphasis on non-matching discretizations, and with application to wind turbines. *Computer Methods in Applied Mechanics and Engineering* [online]. 249 pp.28-41.

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

Belgacem, F.B. and Maday, Y. (1997) The mortar element method for three dimensional finite elements. *ESAIM: Mathematical Modelling and Numerical Analysis* [online]. 31 (2), pp.289-302.

Belgacem, F.B. (1999) The mortar finite element method with Lagrange multipliers. *Numerische Mathematik* [online]. 84 (2), pp.173-197.

Benson, D., Bazilevs, Y., Hsu, M. and Hughes, T. (2010) Isogeometric shell analysis: the Reissner–Mindlin shell. *Computer Methods in Applied Mechanics and Engineering* [online]. 199 (5-8), pp.276-289.

Benson, D., Hartmann, S., Bazilevs, Y., Hsu, M. and Hughes, T. (2013) Blended isogeometric shells. *Computer Methods in Applied Mechanics and Engineering* [online]. 255 pp.133-146.

Bernardi, C., Debit, N. and Maday, Y. (1990) Coupling finite element and spectral methods: First results. *Mathematics of Computation* [online]. 54 (189), pp.21-39.

Bernšteın, S. (1912) Démonstration du théoreme de Weierstrass fondée sur le calcul des probabilities. *Comm.Soc.Math.Kharkov* [online]. 13 pp.1-2.

Bertsekas, D.P. (2014) *Constrained Optimization and Lagrange Multiplier Methods* [online]. Academic press.

Bézier, P. (1970) Numerical control: mathematics and applications. [online].

Borden, M.J., Scott, M.A., Evans, J.A. and Hughes, T.J. (2011) Isogeometric finite element data structures based on Bézier extraction of NURBS. *International Journal for Numerical Methods in Engineering* [online]. 87 (1-5), pp.15-47.

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

Bouclier, R., Elguedj, T. and Combescure, A. (2015) An isogeometric locking-free NURBS-based solid-shell element for geometrically nonlinear analysis. *International Journal for Numerical Methods in Engineering* [online]. 101 (10), pp.774-808.

Bouclier, R., Elguedj, T. and Combescure, A. (2013) On the development of NURBS-based isogeometric solid shell elements: 2D problems and preliminary extension to 3D. *Computational Mechanics* [online]. 52 (5), pp.1085-1112.

Breitenberger, M., Apostolatos, A., Philipp, B., Wuchner, R. and Bletzinger, K. (2015) Analysis in computer aided design: Nonlinear isogeometric B-Rep analysis of shell structures. *Computer Methods in Applied Mechanics and Engineering* [online]. 284 pp.401-457.

Brivadis, E., Buffa, A., Wohlmuth, B. and Wunderlich, L. (2015) Isogeometric mortar methods. *Computer Methods in Applied Mechanics and Engineering* [online]. 284 pp.292-319.

Budd, T. (2002) *An Introduction to Object-Oriented Programming*. 3rd ed. London; Boston: Addison-Wesley.

Cazzani, A., Malagù, M. and Turco, E. (2016) Isogeometric analysis of plane-curved beams. *Mathematics and Mechanics of Solids* [online]. 21 (5), pp.562-577.

Chan, C.L., Anitescu, C. and Rabczuk, T. (2017) Volumetric parametrization from a level set boundary representation with PHT-splines. *Computer-Aided Design*. 82 pp.29-41.

Chen, X., Yong, J., Wang, G., Paul, J. and Xu, G. (2008) Computing the minimum distance between a point and a NURBS curve. *Computer-Aided Design* [online]. 40 (10-11), pp.1051-1054.

Chen, L., Xu, G., Wang, S., Shi, Z. and Huang, J. (2019) Constructing volumetric parameterization based on directed graph simplification of ℓ1 polycube structure from complex shapes. *Computer Methods in Applied Mechanics and Engineering*. 351 pp.422-440.

Clough, R.W. (2001) Thoughts about the origin of the finite element method. *Computers and Structures* [online]. 79 (22), pp.2029-2030.

Cohen, E., Martin, T., Kirby, R.M., Lyche, T. and Riesenfeld, R.F. (2010) Analysis-aware modeling: Understanding quality considerations in modeling for isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*. 199 (5), pp.334-356.

Coox, L., Greco, F., Atak, O., Vandepitte, D. and Desmet, W. (2017) A robust patch coupling method for NURBS-based isogeometric analysis of non-conforming multipatch surfaces. *Computer Methods in Applied Mechanics and Engineering* [online]. 316 pp.235-260.

Cottrell, J.A., Hughes, T.J.R. and Bazilevs, Y. (2009) *Isogeometric Analysis: Towards Integration of CAD and FEA*. 1st ed. Wiley.

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

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

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

de Borst, R. and Chen, L. (2018) The role of Bézier extraction in adaptive isogeometric analysis: Local refinement and hierarchical refinement. *International Journal for Numerical Methods in Engineering* [online]. 113 (6), pp.999-1019.

Dede, L., Jaggli, C. and Quarteroni, A. (2015) Isogeometric numerical dispersion analysis for two-dimensional elastic wave propagation. *Computer Methods in Applied Mechanics and Engineering* [online]. 284 pp.320-348.

Dokken, T. (2010) Locally refined splines. [online].

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

Düster, A., Parvizian, J., Yang, Z. and Rank, E. (2008) The finite cell method for three-dimensional problems of solid mechanics. *Computer Methods in Applied Mechanics and Engineering* [online]. 197 (45-48), pp.3768-3782.

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

Escobar, J.M., Cascón, J.M., Rodríguez, E. and Montenegro, R. (2011) A new approach to solid modeling with trivariate T-splines based on mesh optimization. *Computer Methods in Applied Mechanics and Engineering*. 200 (45), pp.3210-3222.

Evans, E., Scott, M., Li, X. and Thomas, D. (2015) Hierarchical T-splines: Analysis-suitability, Bézier extraction, and application as an adaptive basis for isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering* [online]. 284 pp.1-20.

Farin, G. and Hansford, D. (1999) Discrete Coons patches. *Computer Aided Geometric Design*. 16 (7), pp.691-700.

Felippa, C.A. (2004) Introduction to finite element methods. *Course Notes, Department of Aerospace Engineeing Sciences, University of Colorado at Boulder, Available at Http://Www.Colorado.Edu/Engineering/Aerospace/CAS/Courses.D/IFEM.D* [online].

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

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

Geuzaine, C. and Remacle, J. (2009) Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering* [online]. 79 (11), pp.1309-1331.

Ghorashi, S.S., Valizadeh, N., Mohammadi, S. and Rabczuk, T. (2015) T-spline based XIGA for fracture analysis of orthotropic media. *Computers and Structures*. 147 pp.138-146.

Ghorashi, S.S., Valizadeh, N. and Mohammadi, S. (2012) Extended isogeometric analysis for simulation of stationary and propagating cracks. *International Journal for Numerical Methods in Engineering*. 89 (9), pp.1069-1101.

Goldman, R.N. and Filip, D.J. (1987) Conversion from Bézier rectangles to Bézier triangles. *Computer-Aided Design* [online]. 19 (1), pp.25-27.

Goldstein, B.L., Kemmerer, S.J. and Parks, C.H. (1998) A Brief History of Early Product Data Exchange Standards–NISTIR 6221 WERB-approved: 2 September 1998. *Technology* [online].

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

Hestenes, M.R. (1969) Multiplier and gradient methods. *Journal of Optimization Theory and Applications* [online]. 4 (5), pp.303-320.

Hoschek, J., Lasser, D. and Schumaker, L.L. (1993) *Fundamentals of Computer Aided Geometric Design* [online]. AK Peters, Ltd.

Hosseini, S., Remmers, J.J., Verhoosel, C.V. and De Borst, R. (2013) An isogeometric solid-like shell element for nonlinear analysis. *International Journal for Numerical Methods in Engineering* [online]. 95 (3), pp.238-256.

Hu, S. and Wallner, J. (2005) A second order algorithm for orthogonal projection onto curves and surfaces. *Computer Aided Geometric Design* [online]. 22 (3), pp.251-260.

Hughes, T.J.R., Cottrell, J.A. and Bazilevs, Y. (2005) Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering* [online]. 194 (39), pp.4135-4195.

Hughes, T.J.R., Reali, A. and Sangalli, G. (2010) Efficient quadrature for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering* [online]. 199 (5), pp.301-313.

Hughes, T.J.R. (2000) *The Finite Element Method : Linear Static and Dynamic Finite Element Analysis* [online]. New York: Dover Publications.

Isaacson, E. and Keller, H.B. (2012) *Analysis of Numerical Methods* [online]. Courier Corporation.

Jaxon, N. and Qian, X. (2014) Isogeometric analysis on triangulations. *Computer-Aided Design* [online]. 46 pp.45-57.

Johannessen, K.A., Kvamsdal, T. and Dokken, T. (2014) Isogeometric analysis using LR B-splines. *Computer Methods in Applied Mechanics and Engineering* [online]. 269 pp.471-514.

Kennicott, P. (1995) The initial graphics exchange specification (IGES) Version 5.3. *IGES/PDES Organisation* [online].

Kiendl, J., Bletzinger, K., Linhard, J. and Wüchner, R. (2009) Isogeometric shell analysis with Kirchhoff–Love elements. *Computer Methods in Applied Mechanics and Engineering* [online]. 198 (49-52), pp.3902-3914.

Kiendl, J., Auricchio, F., Beirão da Veiga, L., Lovadina, C. and Reali, A. (2015) Isogeometric collocation methods for the Reissner–Mindlin plate problem. *Computer Methods in Applied Mechanics and Engineering*. 284 pp.489-507.

Kiendl, J., Bazilevs, Y., Hsu, M.-., Wüchner, R. and Bletzinger, K.-. (2010) The bending strip method for isogeometric analysis of Kirchhoff–Love shell structures comprised of multiple patches. *Computer Methods in Applied Mechanics and Engineering* [online]. 199 (37), pp.2403-2416.

Kim, J. and Youn, S. (2012) Isogeometric contact analysis using mortar method. *International Journal for Numerical Methods in Engineering* [online]. 89 (12), pp.1559-1581.

Kim, H., Seo, Y. and Youn, S. (2009) Isogeometric analysis for trimmed CAD surfaces. *Computer Methods in Applied Mechanics and Engineering* [online]. 198 (37), pp.2982-2995.

Ko, K. and Sakkalis, T. (2014) Orthogonal projection of points in CAD/CAM applications: an overview. *Journal of Computational Design and Engineering* [online]. 1 (2), pp.116-127.

Kojić, M. and Bathe, K. (1987) Studies of finite element procedures—Stress solution of a closed elastic strain path with stretching and shearing using the updated Lagrangian Jaumann formulation. *Computers & Structures* [online]. 26 (1-2), pp.175-179.

Krajcinovic, D. and Lemaitre, J. (1987) *Continuum Damage Mechanics: Theory and Applications* [online]. Springer.

Kudela, L., Zander, N., Kollmannsberger, S. and Rank, E. (2016) Smart octrees: Accurately integrating discontinuous functions in 3D. *Computer Methods in Applied Mechanics and Engineering* [online]. 306 pp.406-426.

Lai, W., Yu, T., Bui, T.Q., Wang, Z., Curiel-Sosa, J.L., Das, R. and Hirose, S. (2017) 3-D elasto-plastic large deformations: IGA simulation by Bézier extraction of NURBS. *Advances in Engineering Software* [online]. 108 pp.68-82.

Lee, D. and Lin, A.K. (1986) Generalized Delaunay triangulation for planar graphs. *Discrete & Computational Geometry* [online]. 1 (3), pp.201-217.

Lee, J. and Fenves, G.L. (2001) A return-mapping algorithm for plastic-damage models: 3-D and plane stress formulation. *International Journal for Numerical Methods in Engineering* [online]. 50 (2), pp.487-506.

Leonetti, L., Magisano, D., Liguori, F. and Garcea, G. (2018) An isogeometric formulation of the Koiter's theory for buckling and initial post-buckling analysis of composite shells. *Computer Methods in Applied Mechanics and Engineering* [online]. 337 pp.387-410.

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

Li, X., Zheng, J., Sederberg, T.W., Hughes, T.J.R. and Scott, M.A. (2012) On linear independence of T-spline blending functions. *Computer Aided Geometric Design* [online]. 29 (1), pp.63-76.

Limaiem, A. and Trochu, F. (1995) Geometric algorithms for the intersection of curves and surfaces. *Computers & Graphics* [online]. 19 (3), pp.391-403.

Liu, X., Yang, L., Yong, J., Gu, H. and Sun, J. (2009) A torus patch approximation approach for point projection on surfaces. *Computer Aided Geometric Design* [online]. 26 (5), pp.593-598.

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

Luu, A., Kim, N. and Lee, J. (2015) Isogeometric vibration analysis of free-form Timoshenko curved beams. *Meccanica* [online]. 50 (1), pp.169-187.

Ma, Y.L. and Hewitt, W.T. (2003) Point inversion and projection for NURBS curve and surface: control polygon approach. *Computer Aided Geometric Design* [online]. 20 (2), pp.79-99.

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

Martin, T., Cohen, E. and Kirby, R.M. (2009) Volumetric parameterization and trivariate B-spline fitting using harmonic functions. *Computer Aided Geometric Design*. 26 (6), pp.648-664.

Marussig, B. and Hughes, T.J. (2018) A review of trimming in isogeometric analysis: challenges, data exchange and simulation aspects. *Archives of Computational Methods in Engineering* [online]. 25 (4), pp.1059-1127.

Matheron, M. (1980) Spline et Krigeage leur Equivalence Formelle, Rapport N-667, Centre de Geostatistiques, Fontainbleau, Ecole des Mines, Paris. l'. [online].

May, S., Vignollet, J. and De Borst, R. (2015) The role of the Bézier extraction operator for T-splines of arbitrary degree: linear dependencies, partition of unity property, nesting behaviour

and local refinement. *International Journal for Numerical Methods in Engineering* [online]. 103 (8), pp.547-581.

Morgenstern, P. (2016) Globally Structured Three-Dimensional Analysis-Suitable T-Splines: Definition, Linear Independence and m-graded local refinement. *SIAM Journal on Numerical Analysis* [online]. 54 (4), pp.2163-2186.

Nagarajan, P. (2019) *Matrix Methods of Structural Analysis* [online]. Boca Raton: Taylor & Francis, a CRC title, part of the Taylor & Francis imprint, a member of the Taylor & Francis Group, the academic division of T&F Informa, plc.

Nagel, R.N., Nagel, R.N., Braithwaite, W.W. and Kennicott, P.R. (1980) *Initial Graphics Exchange Specification Iges Version 1.0* [online].

Nagy, A.P., Abdalla, M.M. and Gürdal, Z. (2010) Isogeometric sizing and shape optimisation of beam structures. *Computer Methods in Applied Mechanics and Engineering* [online]. 199 (17-20), pp.1216-1230.

Nagy, A.P. and Benson, D.J. (2015) On the numerical integration of trimmed isogeometric elements. *Computer Methods in Applied Mechanics and Engineering* [online]. 284 pp.165-185.

Neto, E.A.d.S., Peric, D. and Owen, D.R.J. (2008) *Computational Methods for Plasticity: Theory and Applications* [online]. Chichester, West Sussex: Wiley.

Anon. (1971) *Abhandlungen Aus Dem Mathematischen Seminar Der Universität Hamburg* [online]. Springer.

Oh, Y., Kim, Y., Lee, J., Kim, M. and Elber, G. (2012) Efficient point-projection to freeform curves and surfaces. *Computer Aided Geometric Design* [online]. 29 (5), pp.242-254.

Oñate, E., Díez, P., Zárate, F. and Larese, A. (2008) *Introduction to the Finite Element Method.Lectures*. 1st ed. Erasmus Mundos.

Parvizian, J., Düster, A. and Rank, E. (2007) Finite cell method. *Computational Mechanics* [online]. 41 (1), pp.121-133.

Piegl, L.A. and Tiller, W. (2001) Parametrization for surface fitting in reverse engineering. *Computer-Aided Design* [online]. 33 (8), pp.593-603.

Piegl, L. and Tiller, W. (1996) *The NURBS Book*. 2nd ed. Springer-Verlag.

Pratt, M.J., Anderson, B.D. and Ranger, T. (2005) Towards the standardized exchange of parameterized feature-based CAD models. *Computer-Aided Design* [online]. 37 (12), pp.1251-1265.

Anon. (2006) *Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling* [online].

Provatidis, C.G. (2005) Analysis of box-like structures using 3-D Coons' interpolation. *Communications in Numerical Methods in Engineering*. 21 (8), pp.443-456.

Ruess, M., Schillinger, D., Bazilevs, Y., Varduhn, V. and Rank, E. (2013) Weakly enforced essential boundary conditions for NURBS-embedded and trimmed NURBS geometries on the basis of the

finite cell method. *International Journal for Numerical Methods in Engineering* [online]. 95 (10), pp.811-846.

Ruess, M., Schillinger, D., Oezcan, A.I. and Rank, E. (2014) Weak coupling for isogeometric analysis of non-matching and trimmed multi-patch geometries. *Computer Methods in Applied Mechanics and Engineering* [online]. 269 pp.46-71.

Ruppert, J. (1995) A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J.Algorithms* [online]. 18 (3), pp.548-585.

Schillinger, D., Dede, L., Scott, M.A., Evans, J.A., Borden, M.J., Rank, E. and Hughes, T.J. (2012) An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces. *Computer Methods in Applied Mechanics and Engineering* [online]. 249 pp.116-150.

Schillinger, D. and Ruess, M. (2015) The Finite Cell Method: A review in the context of higher-order structural analysis of CAD and image-based geometric models. *Archives of Computational Methods in Engineering* [online]. 22 (3), pp.391-455.

Schmidt, R., Wuchner, R. and Bletzinger, K. (2012) Isogeometric analysis of trimmed NURBS geometries. *Computer Methods in Applied Mechanics and Engineering* [online]. 241 pp.93-111.

Scholz, F. and Jüttler, B. (2019) Numerical integration on trimmed three-dimensional domains with implicitly defined trimming surfaces. *Computer Methods in Applied Mechanics and Engineering* [online]. 357 pp.112577.

Scott, M.A., Borden, M.J., Verhoosel, C.V., Sederberg, T.W. and Hughes, T.J. (2011) Isogeometric finite element data structures based on Bézier extraction of T-splines. *International Journal for Numerical Methods in Engineering* [online]. 88 (2), pp.126-156.

Scott, M.A., Li, X., Sederberg, T.W. and Hughes, T.J.R. (2012) Local refinement of analysis-suitable T-splines. *Computer Methods in Applied Mechanics and Engineering* [online]. 213-216 pp.206-222.

Sederberg, T., Cardon, D., Finnigan, G., North, N., Zheng, J. and Lyche, T. (2004) T-spline simplification and local refinement. *ACM Transactions on Graphics (TOG)* [online]. 23 (3), pp.276-283.

Selimovic, I. (2006) Improved algorithms for the projection of points on NURBS curves and surfaces. *Computer Aided Geometric Design* [online]. 23 (5), pp.439-445.

Sevilla, R., Fernández-Méndez, S. and Huerta, A., 2008. NURBS-enhanced finite element method (NEFEM). *International Journal for Numerical Methods in Engineering*, *76*(1), pp.56-83.

Sevilla, R., Fernández-Méndez, S. and Huerta, A., 2011. NURBS-enhanced finite element method (NEFEM). *Archives of Computational Methods in Engineering*, *18*(4), p.441.

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

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

Shewchuk, J.R. (2009) General-dimensional constrained Delaunay and constrained regular triangulations, I: Combinatorial properties. In: (2009) *Twentieth Anniversary Volume:* [online]. Springer, pp.1-58.

Shih, A.M., Yu, T., Gopalsamy, S., Ito, Y. and Soni, B. (2005) Geometry and mesh generation for high fidelity computational simulations using non-uniform rational B-splines. *Applied Numerical Mathematics*. 55 (3), pp.368-381.

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

Si, H. (2008) Adaptive tetrahedral mesh generation by constrained Delaunay refinement. *International Journal for Numerical Methods in Engineering* [online]. 75 (7), pp.856-880.

Si, H. and Shewchuk, J.R. (2014) Incrementally constructing and updating constrained Delaunay tetrahedralizations with finite-precision coordinates. *Engineering with Computers* [online]. 30 (2), pp.253-269.

Simo, J. and Taylor, R. (1986) A return mapping algorithm for plane stress elastoplasticity. *International Journal for Numerical Methods in Engineering* [online]. 22 (3), pp.649-670.

Singh, S.K., Singh, I.V., Mishra, B.K. and Bhardwaj, G. (2018) Analysis of cracked plate using higher-order shear deformation theory: Asymptotic crack-tip fields and XIGA implementation. *Computer Methods in Applied Mechanics and Engineering*. 336 pp.594-639.

Skytt, V. and Haenisch, J. (2013) Extension of ISO 10303 with isogeometric model capabilities. *Iso Tc* [online]. 184 .

Stroud, I. (2006) *Boundary Representation Modelling Techniques* [online]. Springer Science & Business Media.

Surazhsky, V., Alliez, P. and Gotsman, C. (2003) *Isotropic Remeshing of Surfaces: A Local Parameterization Approach* [online].

Temizer, I., Wriggers, P. and Hughes, T. (2012) Three-dimensional mortar-based frictional contact treatment in isogeometric analysis with NURBS. *Computer Methods in Applied Mechanics and Engineering* [online]. 209 pp.115-128.

Teschemacher, T., Bauer, A., Oberbichler, T., Breitenberger, M., Rossi, R., Wüchner, R. and Bletzinger, K. (2018) Realization of CAD-integrated shell simulation based on isogeometric B-Rep analysis. *Advanced Modeling and Simulation in Engineering Sciences* [online]. 5 (1), pp.19.

Thai, C.H., Nguyen-Xuan, H., Nguyen-Thanh, N., Le, T., Nguyen-Thoi, T. and Rabczuk, T. (2012) Static, free vibration, and buckling analysis of laminated composite Reissner–Mindlin plates using NURBS-based isogeometric approach. *International Journal for Numerical Methods in Engineering* [online]. 91 (6), pp.571-603.

Timoshenko, S. (1983) *History of Strength of Materials: With a Brief Account of the History of Theory of Elasticity and Theory of Structures* [online]. Courier Corporation.

Anon. (1998) *Imr* [online].

Uhm, T. and Youn, S. (2009) T-spline finite element method for the analysis of shell structures. *International Journal for Numerical Methods in Engineering* [online]. 80 (4), pp.507-536.

Varduhn, V., Hsu, M., Ruess, M. and Schillinger, D. (2016) The tetrahedral finite cell method: higher-order immersogeometric analysis on adaptive non-boundary-fitted meshes. *International Journal for Numerical Methods in Engineering* [online]. 107 (12), pp.1054-1079.

Verhoosel, C.C., Scott, M., Borst, d.,R René and Hughes, T. (2011) An isogeometric approach to cohesive zone modeling. *International Journal for Numerical Methods in Engineering* [online]. 87 (1-5), pp.336-360.

Verhoosel, C.C., Scott, M., Hughes, T. and Borst, d.,R René (2011) An isogeometric analysis approach to gradient damage models. *International Journal for Numerical Methods in Engineering* [online]. 86 (1), pp.115-134.

Vuong, A., Giannelli, C., Jüttler, B. and Simeon, B. (2011) A hierarchical approach to adaptive local refinement in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering* [online]. 200 (49-52), pp.3554-3567.

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

Wang, X. and Qian, X. (2014) An optimization approach for constructing trivariate B-spline solids. *Computer-Aided Design*. 46 (1), pp.179-191.

Weeger, O., Wever, U. and Simeon, B. (2013) Isogeometric analysis of nonlinear Euler–Bernoulli beam vibrations. *Nonlinear Dynamics* [online]. 72 (4), pp.813-835.

Xia, S., Wang, X. and Qian, X. (2015) Continuity and convergence in rational triangular Bézier spline based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering* [online]. 297 pp.292-324.

Xia, S. and Qian, X. (2017) Isogeometric analysis with Bézier tetrahedra. *Computer Methods in Applied Mechanics and Engineering*. 316 pp.782-816.

Xu, F., Schillinger, D., Kamensky, D., Varduhn, V., Wang, C. and Hsu, M. (2016) The tetrahedral finite cell method for fluids: Immersogeometric analysis of turbulent flow around complex geometries. *Computers & Fluids* [online]. 141 pp.135-154.

Xu, G., Mourrain, B., Duvigneau, R. and Galligo, A. (2013a) Analysis-suitable volume parameterization of multi-block computational domain in isogeometric applications. *Computer-Aided Design*. 45 (2), pp.395-404.

Xu, G., Mourrain, B., Duvigneau, R. and Galligo, A. (2013b) Optimal analysis-aware parameterization of computational domain in 3D isogeometric analysis. *Computer-Aided Design*. 45 (4), pp.812-821.

Anon. (2009) *Computer Graphics Forum* [online]. Wiley Online Library.

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

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

Yin, S., Yu, T., Bui, T.Q., Xia, S. and Hirose, S. (2015) A cutout isogeometric analysis for thin laminated composite plates using level sets. *Composite Structures* [online]. 127 pp.152-164.

Yusuf, O., Zhao, G., Wang, W. and Onuh, S. (2015) Simulation based on trivariate nurbs and isogeometric analysis of a spur gear. *Strength of Materials* [online]. 47 (1), pp.19-28.

Zhang, Y., Wang, W. and Hughes, T.J.R. (2013) Conformal solid T-spline construction from boundary T-spline representations. *Computational Mechanics*. 51 (6), pp.1051-1059.

Zhang, Y., Wang, W. and Hughes, T.J.R. (2012) Solid T-spline construction from boundary representations for genus-zero geometry. *Computer Methods in Applied Mechanics and Engineering*. 249-252 pp.185-197.

Zienkiewicz, O.C. and Taylor, R.L. (2000) *The Finite Element Method. Volume 1. the Basis*. 5th ed. Butterworth-Heinemann.

Zuo, B., Huang, Z., Wang, Y. and Wu, Z. (2015) Isogeometric analysis for CSG models. *Computer Methods in Applied Mechanics and Engineering* [online]. 285 pp.102-124.