
**Towards an automatic solution for flexible quad layout
generation and quad meshing**

Jovana Jezdimirović

Thesis submitted in partial fulfillment of the requirements for the Ph.D
Degree in Engineering Sciences

THESIS COMMITTEE		
Advisor	Jean-François Remacle	<i>UCLouvain</i>
Jury	Jonathan Lambrechts	<i>UCLouvain</i>
	Maxence Reberol	<i>UCLouvain</i>
	Christophe Geuzaine	<i>ULiège</i>
	Franck Ledoux	<i>CEA</i>
	Nicolas Ray	<i>INRIA</i>
Chairperson	Thomas Pardoën	<i>UCLouvain</i>

Louvain-la-Neuve, August 2021

Посвећено Мајкић

ACKNOWLEDGMENTS

I would like to express my profound gratitude to Prof. Jean-François Remacle and Prof. José Miró Julià for providing me with the life-changing opportunities, to the thesis committee for their devotion, time, and suggestions, and dear Euler-ians and Hextreme-rs for making the working environment so pleasant. It has been a privilege and honor.

My dearest thanks to all of my beloved cousins and friends for their heart-warming presence on my life journey, to Philou, for all of his patience, support, widening my horizons and making me happy, and to Cecilija, the light of my days, for her unconditional love and neverending faith in me. I carry you all in my heart.

Jovana Jezdimirović

Захвалница

Желела бих да изразим дубоку захвалност проф. Жон-Франсоа Ремаклу и проф. Хосе Миро Хулији на пруженој шанси; комисији за њихову посвећеност, време и сугестије као и тиму за предивну радну атмосферу. Била је то привилегија и част.

Бескрајно хвала фамилији и пријатељима чије присуство на мом животном путу греје срце; Филу, за сво његово стрпљење, подршку, и ширењу мојих видика као и вољеној Цецилији, светлости мојих дана, за њену безусловну љубав и непрестану веру у мене. Све вас носим у срцу.

Јована Јездимировић

Contents

Contents	i
1 Introduction	1
1 Thesis objective, structure, and contributions	5
2 Motivation and foundations	9
1 Surface representation for numerical purposes	10
2 Taxonomy and classification of quad meshes	15
3 Challenges of generating a block-structured quad mesh	18
4 Are quadrilaterals better than triangles?	21
3 Methods for quad mesh generation	25
1 A brief literature review	26
2 Quad layout for quad meshing	32
3 Automatic generation of a quad layout via cross-fields	36
4 Quad layout and quad mesh generation obtained with Ginzburg-Landau PDE	49
1 Introduction	50
2 Related work	51
3 Purpose of 2D cross-fields for quad mesh generation	52
4 Generating cross-field based on Ginzburg-Landau PDE	54
5 Tracing separatrices	58
6 Examples of multi-block decomposition	63
7 Planar graph embedding of the multi-block decomposition	66
8 Workflow for planar domains	68
9 Workflow for 2D manifolds	72
10 Discussion	74
11 Conclusion	75
12 Appendix	75
5 Elliptic Fekete points obtained by Ginzburg-Landau PDE	79

1	Introduction	80
2	Computational procedure	81
3	Results	82
4	Conclusion	84
6	Flexible quad layout and quad mesh generation: the planar case	85
1	Introduction	86
2	Related work	90
3	First cross-field computation	92
4	Singularity pattern	93
5	Cross-field computation with imposed singularity pattern	95
6	Quad layout: generating and correcting the partitions	105
7	Quad meshing	108
8	Conclusion and Future Work	109
7	Flexible quad layout generation: the 2D manifold case	113
1	Introduction	114
2	Singularity configurations	117
3	Cross-field computation on prescribed singularity configuration	120
4	Quad layout and quad mesh generation	131
5	Conclusion	136
8	The generation of integrable cross-field with anisotropic scaling	137
1	Limit cycle free cross-field formulation	138
2	Integrability condition	141
3	Minimizing integrability error E regarding (θ, H_1, H_2)	146
9	Conclusion	151
1	Perspectives	154
	Bibliography	157

CHAPTER



Introduction

The long-standing desire for a profound understanding of the surrounding physical phenomena has led to numerous scientific breakthroughs, which are mirrored in the development of formalism and the means to predict the behavior of these abstract systems. In the language of mathematical physics, these phenomenons are described through the lenses of a particular entity: the *partial differential equations*, which aid the comprehension of an abstract physical system's life cycle through time and space.

Historically, it was precisely the field of meteorology where, as a direct way to find the solution of the specific partial differential equation (PDE), the *numerical methods* have been suggested for the first time. The work of Lewis Frey Richardson [1] on forecasting in the mid-1920s involved an uncommon angle for that time: “*Weather prediction by numerical process.*” Visualization

of partial differential equations via *vector fields*, which can encode a tremendous amount of information, for this particular field, but in a much broader sense as it will turn out later, is considered a crucial resource. A vector field associates a direction and a magnitude to the given point, where for the visual aid, a velocity of the steady wind can be taken as an example. Based on this, it's no wonder that Richardson's pioneering idea based on approximation was indeed promising. Still, because of the vast computations relying solely on manual efforts, this innovative approach resulted in changing the weather faster than obtaining the numerical results, *i.e.*, being utterly impractical. Yet, not for long.

The true power of numerical methods for solving partial differential equations (PDEs) has been unleashed with the birth of the computer era in the 1950s. It was not long to realize that these methods offer superior physical insight and design over the classical experiments at reduced costs and risks [2]. Indeed, this paradigm shift from experimental to numerical approach allowed remarkable advancement of many fields in science and industry, meteorology being just one of them. Nowadays, Richardson's dream is reality: it is possible to predict the weather for several days in advance with almost an absolute confidence level [3]!

The fast computations, beneficial for various fields, had been supported by the dramatic innovation of computers, and by so, different numerical methods suiting particular phenomena had arisen. One of them was the Finite Element Method (FEM). In the 1930s, Courant–Friedrichs–Lewy's paper [4] presented the formalism of the finite element method, and a bit later, the genuinely outstanding contribution on the matter was given by Argyris [5]. Motivated by the need to solve practical problems in civil engineering, the Earthquake Engineering Research Center of the University of California, Berkeley, became a home of this newly born numerical method during the mid-1950s-1970s [6]. One of the main outcomes of their work was the development of a three-dimensional, linear, static, and dynamic finite element Structural Analysis Program (SAP IV) [7]. It was one of the fastest and the most exhaustive structural analysis programs globally, whose free distribution greatly influenced the further development of FEM in academia and various engineering communities. Many decades after, thanks to the contributions of numerous scientists [8, 9, 10], the FEM has become a standard procedure in aeronautics, automotive design, meteorology, fluid mechanics, soil mechanics, biomechanics, medical imaging, civil engineering, electrical engineering, and electronics [11].

The driving mechanism of FEM in solving PDEs is the double discretization of both the equations and the geometrical model on which the computation is performed. In this manner, the infinite dimensional continuous problem is

thus transformed into a finite dimensional one, allowing actual computations on computers. The *mesh* is the discrete version of the continuous geometrical model. A finite element mesh is a set of elements of simple shape that intersect only at vertices, edges, or faces. Since FEM programs are not usually able to handle non-conforming meshes, *i.e.*, meshes with t-junctions, this thesis will be focused on conformal ones, as illustrated in figure 1.1. Standard elements' types used for FEM calculations are triangles and quadrilaterals (in 2D) and tetrahedra, hexahedra, prisms and pyramids (in 3D). Mesh quality is known to be of paramount importance for obtaining accurate FEM solution [12]. This fact implies the high-quality discretization of often very complex bodies, such as an aircraft, and therefore can be an extremely demanding process.

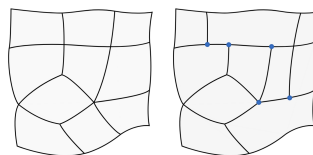


Figure 1.1: Conformal (left) and non-conformal mesh right (picture from [13])

The famous Baker's paper "*Mesh generation: Art or science?*" ([14]) elucidates the history of mesh generation, underlying its decades-long challenges and advances. Although the unglamorous tedious manual efforts are no longer needed to obtain it, due to the development of the automatic algorithms (detailed in [15]), it still represents the major bottleneck in engineering simulations. As estimated by Prof. Hughes, the mesh generation is taking more than 80% of overall analysis time, which imposes an emergency in its further development, especially when considering that it is present in over a million offices per day worldwide [16].

There exist various types of meshes designed in such a way to suit a particular application's demands. However, the common distinction can be based on the level of the structure that mesh exhibits and the type of elements. Among FEM practitioners, there are two distinct streams: the ones focused on triangular/tetrahedral meshes due to the maturity of their generation, which supplies fast and robust algorithms, and the ones that favor block-structured quadrilateral/hexahedral meshes. As presented by many researchers [17, 18, 19, 20], the latter kinds of meshes offer better properties such as numerical stability, alignment with physical features of the object, capturing the anisotropy, the use on multi-structured domains, lower computational costs, and efficient memory usage. Still, the actual process of obtaining block-structured quadrilateral/hexahedral meshes indeed comes with a tremendous challenge: the development of a robust and fast algorithm.

The issue of generating block-structured quadrilateral/hexahedral meshes originates from the strict geometrical and topological constraints that these kinds of meshes must obey, which will be thoroughly outlined in the following Chapter 2 and Chapter 3. Here, it is essential to mention that many works over the years have been dedicated to this particular subject [11, 15, 21], confirming the actual need and the great challenge for its generation. Despite the tremendous amount of invested efforts, there still exists the profound need for the further development of block-structured quadrilateral mesh generation, which will be able to deal successfully with approximation and quad mesh quality, robustness on geometrically inconsistent inputs, the efficiency, interactive control, scalability, and evaluation [21].

The subject of this thesis is precisely targeting the challenging issue of robust and high-quality block-structured quadrilateral mesh generation. In order to reach this goal, the rigorous mathematical background served as the foundation and motivation for developing the corresponding methodology. The exploration of *Ginzburg-Landau PDE* and its related theory, previously known in the field of superconductors, as well as the properties of an auxiliary object *cross-field* (kin of a vector field) hand-in-hand with the use of a *quad layout* concept from [13], turned out to be very fruitful idea for this matter. More specifically, it will be shown that this specific PDE exhibits very advantageous properties for quad mesh generation and that further can even be used to obtain a surprising theoretical result: the Elliptic Fekete points on 2-sphere! At the same time, the related Ginzburg-Landau theory [22] aids the thorough understanding of the behavior of this particular PDE and, by so, crucially assists in the development of the interactive quad mesh generation.

The design and implementation of the methodology have been carried out in the frames of ERC HEXTREME¹ project and the software GMSH. The HEXTREME project is devoted to creating two breakthroughs in the art of mesh generation: the parallelizing of all meshing aspects and the development of robust conformal quadrilateral/hexahedral and boundary layer meshing algorithms. The GMSH software by Geuzaine & Remacle [23] is an open-source 3D finite element mesh generator built around four modules: geometry, mesh, solver, and post-processing, providing a reliable and user-friendly meshing tool. Indeed, the presented working environment offered fruitful soil for guiding the subject of this thesis.

¹European Research Council project: Hexahedral Mesh Generation in Real Time (ERC-2015-AdG-694020)

1 | Thesis objective, structure, and contributions

The objective of this thesis is twofold:

1. To develop the block-structured quad mesher that is automatic and produces high-quality elements.
2. To design the interactive block-structured quad mesher that preserves properties of the first objective.

The manuscript is organized in the following manner:

Chapter 2 depicts the possible manners and the crucial importance of a trustworthy representation of a geometrical object for FEM purposes. The primer, including the topics as surface representations and types of meshes, serves here as a foundation for a deeper understanding of the strict constraints for generating block-structured quad meshes and discussing a long-lasting dilemma: "*Are quadrilaterals better than triangles?*".

Chapter 3 presents the state-of-the-art in quad mesh generation by discussing the most notable methodologies related to the subject of this thesis. It also describes the profound relationship between a quad mesh and a corresponding concept of a quad layout. The notation, generation, topological properties, and a crucial role of an auxiliary object - cross-field for quad mesh/quad layout generation is elaborated in detail.

Chapter 4 represents the reproduction of the paper *Multi-block decomposition and meshing of the 2D domain using Ginzburg-Landau PDE*. A crucial contribution of this particular PDE is developing an automatic mechanism to obtain block-structured quad mesh. The same paper shows that the solutions of Ginzburg-Landau PDE have an adequate distribution of singularities and that, in combination with the use of Crouzeix-Raviart elements, reduce the appearance of limit-cycles, advocating its usage for quad meshing purposes.

Chapter 5 depicts the reproduction of the paper *Elliptic Fekete points obtained by Ginzburg-Landau PDE*. It is shown here that the computation of the cross-field on a 2-sphere with the Ginzburg-Landau model yields a surprising result: the Elliptic Fekete points. The theoretical link between the asymptotic behavior of Ginzburg-Landau energy and Smale's 7th problem is presented, and a very competitive method to obtain the Elliptic Fekete points in an automatic manner is developed.

Chapter 6 represents the reproduction of the paper *Quad layouts with high valence singularities for flexible quad meshing*. The crucial contributions outlined here are developing a formulation and implementation of an automatic algorithm that allows the generation of high-quality block-structured quad mesh

from a given set of singularities, possibly with high valences. A designed correction scheme that repairs limit cycles and non-quadrilateral patches is also presented.

Chapter 7 details the flexible quad layout generation on 2D manifolds relying on strict mathematical background for integrable cross-field generation and isotropic size field. The formulation supports the quad layout construction from a given set of singularities fulfilling the Abel-Jacobi theorem and possibly with high valences.

Chapter 8 considers the issue of limit cycle free cross-field formulation. It outlines the mathematical background for the generation of integrable and anisotropic cross-field and presents the promising preliminary results obtained by the developed numerical schemes.

Finally, the contributions and future perspectives are outlined in **Chapter 9**.

The developed formulations and implementations had been carried out under the supervision of Prof. Jean-François Remacle and had strong connections with the works of Dr. Pierre-Alexandre Beaufort, Dr. Maxence Reberol, and Dr. Alexandre Chemin. For the sake of clarity, my contributions are illustrated in figure 1.2 and figure 1.3.

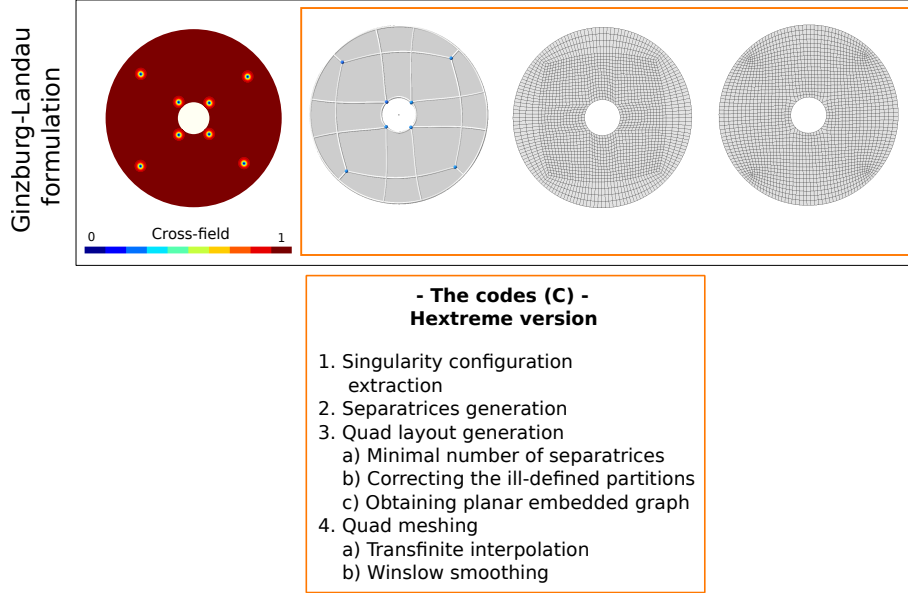


Figure 1.2: Ginzburg-Landau formulation implementation contributions

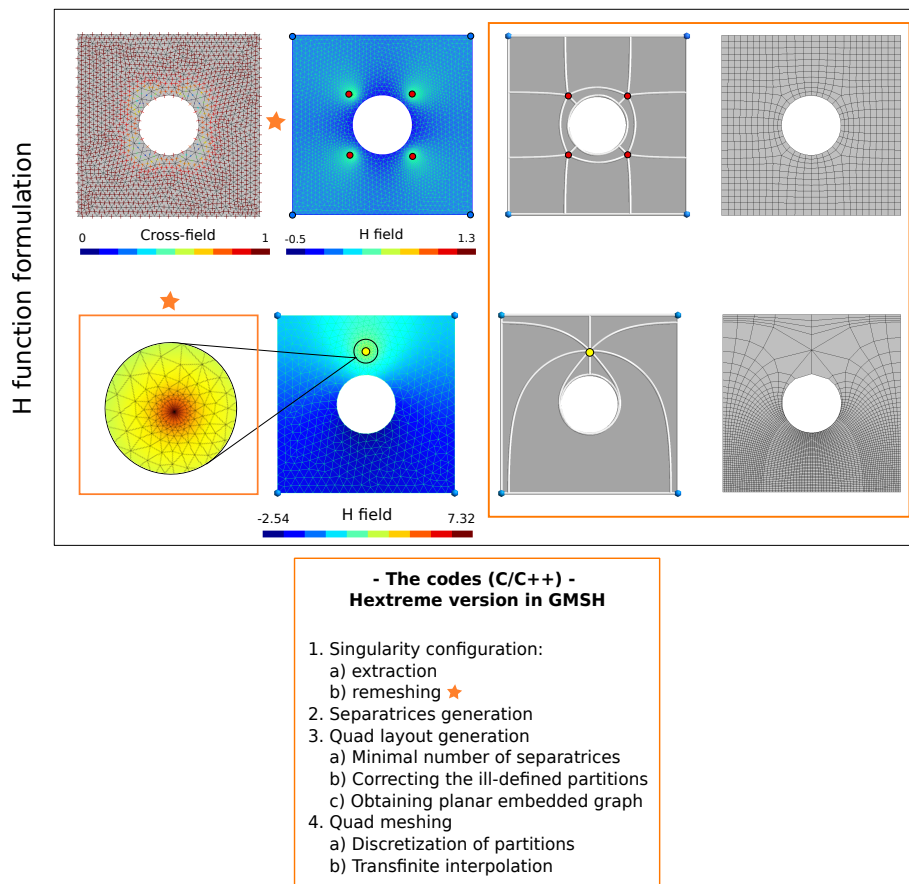


Figure 1.3: H function formulation implementation contributions

CHAPTER



Motivation and foundations

A subject of high interest in FEM is a trustworthy representation of a geometrical object, as an important structure on which the numerical solution is built [15]. There exist various mathematical representations of a geometrical object and its features, each suitable for a particular class of applications. Concentrating on the needs of numerical analysis, specifically FEM, and considering a surface the most prominent object's feature [13], a detailed focus here is given to its reliable representation for this particular application.

More specifically, in this chapter, the type of surfaces of interest and the suitable ways for their representation used in numerical analysis are presented. Afterward, a thorough classification and properties of quadrilateral meshes are introduced. Particular attention is paid to constraints and difficulties for generating a high-quality block-structured quad mesh. Last but not least, the truth about choosing the (mesh) elements with the best performances is

revealed.

1 | Surface representation for numerical purposes

Surfaces represent a special class of topological spaces whose classification is based on characteristics as orientability, Euler characteristic, homeomorphism, and curvature. The following definition specifies the class of surfaces of interest for this thesis, displayed in figure 2.1:

Definition 1.1. A *surface* is an orientable continuous two-dimensional manifold S (with or without boundary ∂S) embedded in $\mathbb{R}^3$.

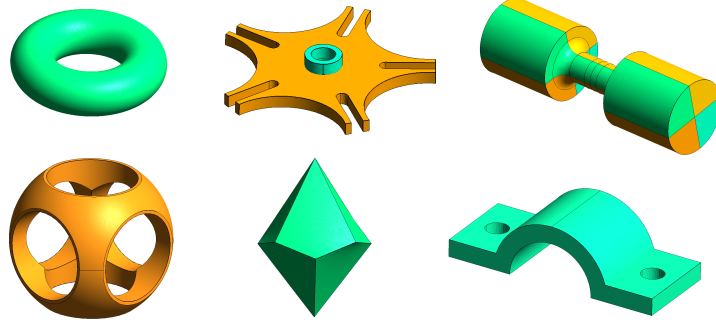


Figure 2.1: Surfaces of interest for numerical analysis

Surfaces can be represented in two different fashions:

- *implicit*: $S = \{x \in \mathbb{R}^3 \mid f(x) = 0\}$, $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ and
- *parametric*: $S = f(\Omega) \subset \mathbb{R}^3$, $f : \Omega \rightarrow S$, $\Omega \subset \mathbb{R}^2$ (illustration in figure 2.2).

A set of surfaces that can be represented in this manner is of negligible size in comparison with the number of complex geometrical shapes for which this representation is not feasible. Nevertheless, it is possible to partition the model S into smaller sub-regions Ω_i , $i \in 1, \dots, n$, $n \in \mathbb{N}$ (see figure 2.3), obtaining a *trimmed* surface representation. Computer-aided design (CAD) systems, on which FEM relies, use precisely this manner of model describing. More specifically, in CAD systems, a boundary representation (BREP) is a method for representing a model as a collection of connected surface elements, which define the boundary between interior and exterior points. For example,

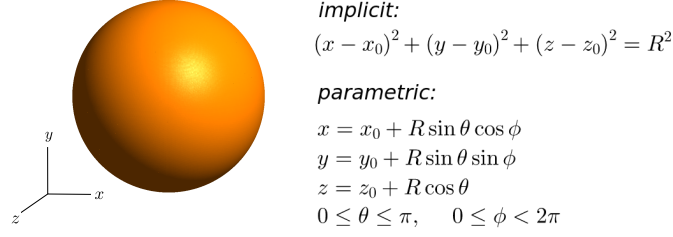


Figure 2.2: Implicit and parametric representation of a sphere

figure 2.3 represents a BREP composed of 12 model surfaces, 20 model curves and 10 model vertices. The graph of model entities' adjacencies defines the topology of its parts, and the trimmed surface is then defined by geometry of each model's face and the trace of its boundary on that geometry.

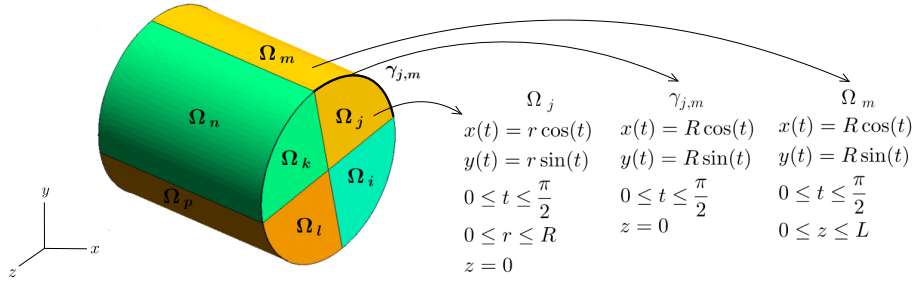


Figure 2.3: Trimmed surface representation of a cylinder

In some CAD systems, *e.g.*, OpenCASCADE and Parasolid, surfaces are built directly via simple primitives (*e.g.*, cubes, spheres, cylinders, and cones), and/or NURBS/Splines networks (see figure 2.4), and by boolean operations. This approach is used as well for surface modifications [24, 25, 26]. Nowadays, modern CAD systems can even handle non-manifold models, composed of, for example, embedded surfaces into volumes or volumes sharing a node face. In order to represent the object exactly (up to a priori given tolerance), it is possible to use an alternative processing pipeline [24]. It starts with the input data, *i.e.*, CAD representation of an object, followed by the removal of topological and geometrical errors, analysis of surface quality, and finally, surface smoothing. In this manner, the optimal result for representing an object is achieved.

After a trustworthy CAD representation of an object is obtained, FEM

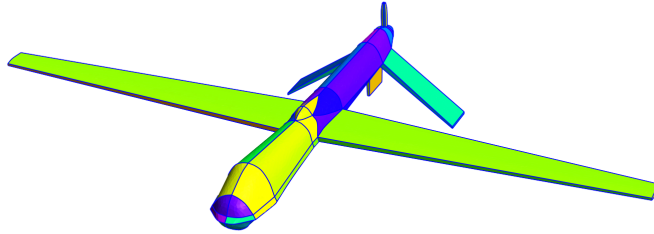


Figure 2.4: CAD of an aircraft. The surface is built from NURBS which respect the geometrical features of the object (picture from [2])

can further proceed with the remaining steps for finding the solution to the particular numerical problem. These steps involve: discretization of the object for simulation, computation of the solution, and post-processing. In order to perform numerical simulation on a particular object, the first step to tackle in FEM is to create a function space to look for the solution. Building an approximation space for FEM solution implies decomposing the object into smaller elements (related to the geometrical shape of reference), *i.e.*, generating *a mesh*. A finite element mesh is a set of elements of simple shape that intersect only at vertices, edges or faces, an illustration in figure 2.5. These elements can be: polygons (triangles and quadrilaterals) for 2D objects, or polyhedrons (tetrahedra, hexahedra, prisms and pyramids) for 3D objects. Since FEM programs are not usually able to handle non-conforming meshes, *i.e.*, meshes with t-junctions, this thesis will be focused on conformal ones. It is crucial that the function space (associated with the mesh) contains a good approximation of the solution and that, at the same time, a mesh approximates well the geometry of the object.

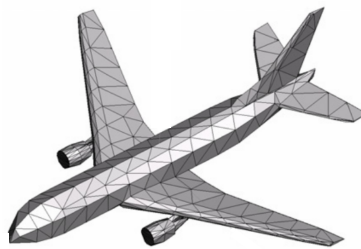


Figure 2.5: A triangulated mesh of an aircraft (picture from [27])

The main sources of errors in a FEM come from: formulation errors, geom-

etry representation errors, solution discretization errors, and numerical errors. Geometry representation errors are related to the CAD modeling of the object. For the cases when CAD workflow respects the above-mentioned geometry processing pipeline (from [24]), the geometrical representation of the object is considered trustworthy. Solution discretization errors are a direct consequence of generating a mesh of poor quality (see figure 2.6). A high-quality mesh respects geometrical features of the object - as boundary shape or curves (*i.e.*, feature lines) and is suitable for a specific numerical simulation. These constraints (coming from geometry and numerical problem) impose the *shape*, *size* and *density* of elements so that the best possible accuracy of the solution can be achieved.

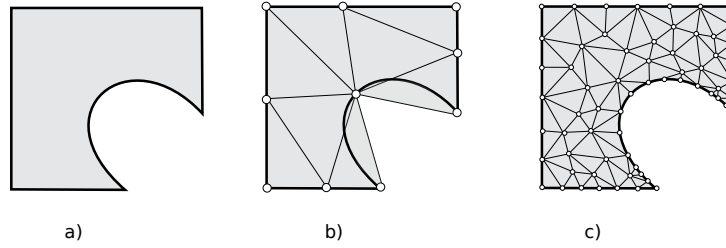


Figure 2.6: Geometry (a), a discretization error due to poor mesh representation (b) and elimination of a discretization error (c)

Above mentioned constraints for generating a mesh are non-negotiable in FEM procedure, and although it is not the subject of this thesis, it is worth mentioning that improvement of the mesh quality, which influences higher accuracy of the solution, can be performed a posteriori. This procedure can involve refinement of the mesh (*h*-version), or, by taking into account that FEM solution is computed by using polynomial interpolation functions over the mesh, fixing the size of the elements and increasing the degree of the polynomial interpolation functions (*p*-version), or by using a hybrid approach (*hp*-version).

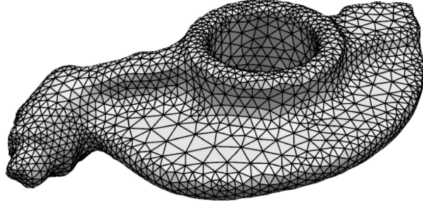
1.1 Types of surface meshes

The building of an approximation space on a surface for computing FEM solution is commonly performed via generating (see figure 2.7):

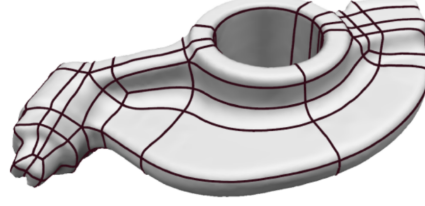
- Triangular meshes (piecewise linear/high-order)
- Nurbs/splines networks (piecewise polynomial)
- Multiblock grids (piecewise regular quad grid) and

- Quad meshes (piecewise bilinear/high-order).

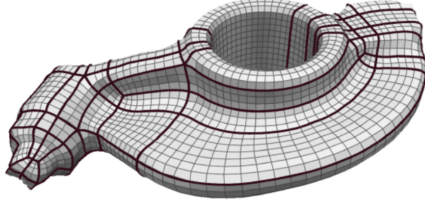
Triangle mesh



Spline domain net



Multiblock grid



Quad mesh

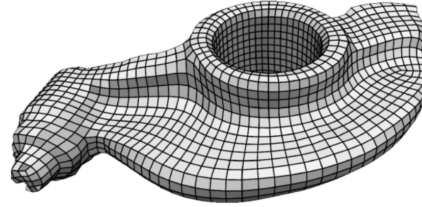


Figure 2.7: Types of meshes (picture from [28])

Triangular meshes have been ubiquitous in numerical analysis applications; its generation is comprehensively documented and there exist numerous robust implementations [15, 21, 29, 30]. Due to the needs of specific applications and aspiration for low computational time and memory cost, other piecewise surface representations have started emanating.

The NURBS networks have recently started emerging in FEM applications, *i.e.*, mesh generation [31, 32, 33]. For this purpose, they are habitually either used for a particular class of surfaces (for example surfaces with Euler characteristic 0 from [34]) or particular application (for instance in fluid dynamics [35, 36] or isogeometric analysis [37]). The main drawback of this type of surface representation is its sustainability in terms of fulfilling numerous geometrical and topological constraints of the specific object [34, 38].

Multi-block grids decompose the domain into polygonal partitions with various shapes and qualities. As defined in [39], “the partitioning of an object’s surface into simple networks of conforming quadrilateral patches” is called a quad layout. The quality of this particular type of surface discretization typically depends on conformity with topological characteristics of the surface [13, 28]. Due to its natural structure and a good trade-off between the number of patches and the level of distortion, a regular refinement of a quad layout leads to generating a high-quality block-structured quad mesh [40]. As stressed out

by [38], an intimate relationship exists between a quad layout and a quad mesh; still, their purposes and properties differ immensely, more details in section 2.2.

A lot of work has been devoted to quad mesh generation because of a large number of tasks that are better suited to quad meshes than to triangular meshes [21]. This trend arises from the need for capturing anisotropic features and decreasing approximation error, which are essential for elastic and elastoplastic analysis [20], spectral and iso-geometric simulations [41] as well for high-order surface modeling purposes [21]. Moreover, a high-quality quad mesh is in its core structured, which enables efficient memory usage (an explanation of this property is detailed in the following section).

In order to meet the requirements for the above-mentioned applications, a quad mesh must obtain certain properties. A detailed explanation of the taxonomy, classification and quality of quad meshes is presented in the following.

2 | Taxonomy and classification of quad meshes

Unless its a hybrid, *i.e.*, *quad-dominant mesh* (a mesh with a majority of quadrilateral elements and a fraction of triangles and/or pentagons), a *quad mesh* is constituted of:

- *vertices* $V = \{v_i \in R^3\}$ which bound each edge,
- oriented *edges* $E = \{e_{ij} \in V \times V\}$ that bound each element and
- distinct quadrilateral *elements* (faces/facets) $F = \{f_{ijkl} \in V^4\}$.

The *valence* of a vertex is the number of its incident edges. The vertex is *regular* if it has valence 4, otherwise *irregular*. The edge with just one incident facet is considered *boundary*; a vertex on the boundary edge is also considered *boundary* (an illustration of the structure is shown in figure 2.8).

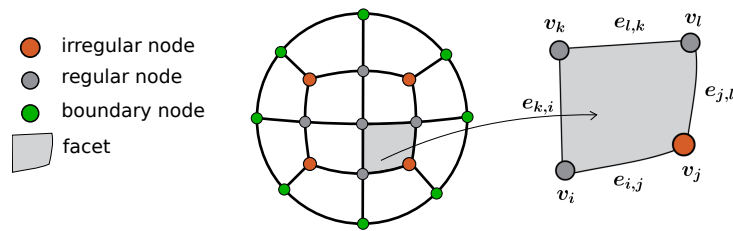


Figure 2.8: Structure of a quad mesh

Based on connectivity among its constituents, a quad mesh can be:

- Conformal or non-conformal
- Structured or unstructured.

A quad mesh is *non-conformal* if there exist at least two faces that share only a part of an edge, otherwise *conformal* (any two faces share either a single vertex, an entire common edge, or nothing), the illustration is shown in figure 2.9. Conforming meshes are preferred in industrial and scientific applications due to the ease of the definition of function spaces, which enables lightweight computations [2], although their generation is a challenging task. Non-conformal meshes are in the literature typically referred to as *T-meshes*, due to the appearance of a node of a face lying down on the interior of an edge (of the other face). It is possible to convert the conformal mesh into the non-conformal one by removing/adding the nodes and edges. On the other hand, the vise-versa process is either computationally very expensive or algorithmically challenging to perform (with the exception of the recent method [42]).

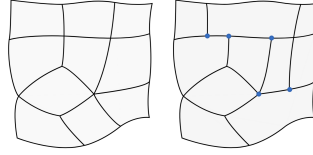


Figure 2.9: Conformal (left) and non-conformal mesh right (picture from [13])

The difference between *structured* and *unstructured* meshes relies mainly on the nature of their respective connectivities. Structured mesh connectivity is implicit, *i.e.*, elements can be automatically stored and accessed using the IJK enumeration scheme, while an unstructured mesh's connectivity is explicit, *i.e.*, each element must be described individually. The regular connectivity of structured meshes is closely related to the notion of topological quality [11], and it also allows making analytical queries on neighboring elements upon their coordinates, which influence low memory usage. A strict alignment of elements that structured meshes exhibit is preferred in various simulations, *e.g.*, boundary layers in computational fluid dynamics and composites in structural dynamics. Taking into account all previously mentioned characteristics of a structured mesh and under the assumption that it is feasible to compute it over a complicated object with the rapid change of element size (underlined as a challenging task by [43]), the utility of structured meshes prevails over unstructured ones.

Among structured quad meshes, it is possible to differentiate the following three subcategories: *all-structured*, *block-structured* and *semi-structured*

meshes, illustrated in figure 2.10. An *all-structured* mesh is a mesh that is structured everywhere, a *block-structured* mesh consists of a limited number of all-structured blocks, while a *semi-structured* does not obtain a high level of structure and typically has a large number of irregular nodes (nodes which are not surrounded by exactly four quadrilaterals) which typically hinder its usability [21].

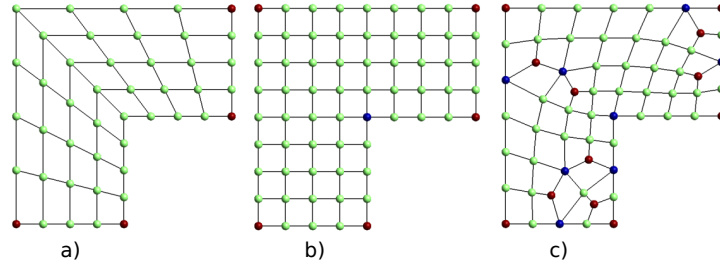


Figure 2.10: A representation of a) all-structured, b) block-structured and c) semi-structured mesh (picture from [44])

2.1 Characteristics of a quad mesh

In order to be used for finite element simulations, a quad mesh must obtain particular attributes, which point out its “sufficient quality“ for solving a specific numerical problem. The *topological validity* defines the quad mesh; *geometric validity* is a requirement for numerical simulations, and it is measured by the value of *scaled Jacobian* [45] which should be strictly positive. A high-quality quad mesh features refer to:

- *The quality of quadrilateral elements*
represents the measure of the deviation of a quadrilateral element from a “perfect square/rectangle“, *i.e.*, each element of a quad mesh should be flat and have corners close to 90 degrees with sides of approximately equal length. This characteristic allows better matrix conditioning, better theoretical error bounds, and better numerical accuracy.
- *Control over anisotropy (orientation, shape, density and element size)*
is of crucial importance for working on specific simulations, *e.g.*, in computational fluid dynamics, with physical problems which are by themselves anisotropic, and complex objects. Varying density and anisotropy of elements allow better adaptation to the specific features of the object (see figure 2.11) and therefore improve the approximation error in numerical simulations.

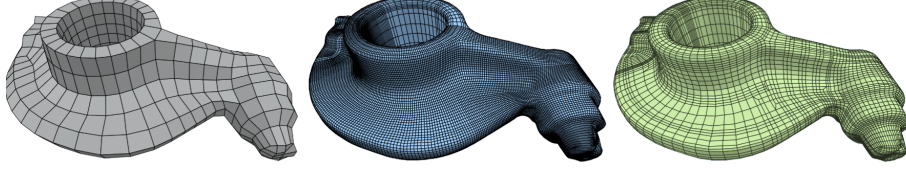


Figure 2.11: Anisotropic refining for capturing object's features (picture from [46])

3 | Challenges of generating a block-structured quad mesh

Creating a high-quality structured quad mesh on an arbitrary surface in an automatic manner is not a trivial task. It implies fulfilling all of the above mentioned attributes and obeying the topological and geometrical characteristics of a surface. More specifically, the *number* and the *valence* of the irregular vertices of a quad mesh are constrained by the *Euler characteristic* χ , which is a topological invariant of the surface.

To better understand this topological constraint, it is helpful to consider an “ideal” all-structured mesh, where every vertex is regular, *i.e.*, has a valence 4. It is possible to show [44] that only surfaces with the Euler characteristic equal to $\chi = 0$ can be paved with an ideal structured mesh, as shown in figure 2.12. In general, structured meshes involve irregular vertices, *i.e.*, vertices with valence $\neq 4$ and therefore the main subject of this thesis will be dealing with block-structured quad meshes.

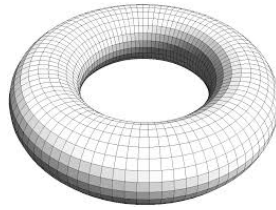


Figure 2.12: An ideal quad mesh on a torus

To show how the topology of the domain constrains the number and valences of irregular nodes, we start from 3.1 theorem. It shows the connection between the number of vertices V , edges E and facets F of an orientable surface S embedded in $\mathbf{R}^3$ with *Euler characteristic* of this surface $\chi(S)$:

Theorem 3.1 (Euler’s polyhedron theorem).

$$V - E + F = \chi(S). \quad (2.1)$$

Further, by exploiting the relations between the number of vertices, edges and facets of a quad mesh, it can be derived [13, 44]:

Theorem 3.2 (Poincaré-Hopf theorem).

$$\sum_{i=1}^N 1 - \frac{\text{valence}(x_i)}{4} = \chi(S). \quad (2.2)$$

Observing the fact that regular nodes do not influence the sum in 3.2, the Poincaré-Hopf theorem demonstrates the emerging of irregular nodes on surfaces with $\chi \neq 0$. A noteworthy remark is that two vertices with $\text{valence} = 5$ and $\text{valence} = 3$ cancel each other out in the sum. This points out that the number of irregular nodes in a quad mesh could be arbitrarily high (see figure 2.13), hindering its usability.

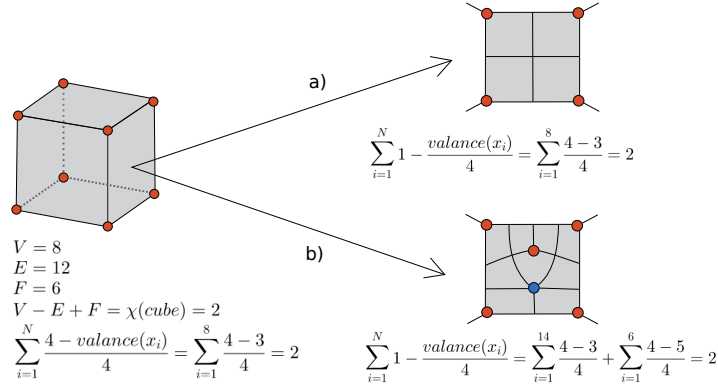


Figure 2.13: Imposing a set of 3–5 irregular nodes without jeopardizing the topology. The minimal number of singularities per cube face a), an increased number of singularities per cube face b). Irregular nodes of $\text{valence} = 3$ are in red and of $\text{valence} = 5$ in blue.

As shown, an overall quad mesh quality is intimately related to the surface on which it is generated. Nevertheless, to be valid, it must also capture another topological, this time intrinsic, characteristic: *inherent global connectivity structures*. This restriction can be explained with Spatial Twist Continuum (STC) theory, developed by [47]. The STC formalism provides a solid ground

for not only generating [48], but also modifying [49, 50] and inspecting the validity of a quad mesh. Its structure relies on two entities: a chord representing a stack of quadrilaterals/hexahedra and a sheet representing a layer of hexahedra [11]. Based on the construction above, a quad mesh can be interpreted as an intertwining of chords. Consequently, the chords' properties and relationships between them dictate the validity of a quad mesh and the possibility of any topological modification, more details in [47, 49, 50]. As already observed [11], the STC theory points out that topological modification of all-quadrilateral (all-hexahedral) mesh is a non-local process and that the problem of its generation is much more constrained than for any other type of FEM meshes.

The topological constraints outlined above do not address all complexities of generating a quad mesh suitable for FEM. The other group of restrictions arises from geometry - as preserving the object's features (boundary and curves shape) or working with a poor representation (which includes, but is not limited to: overlapping patches, gaps between patches or surface intersections [11]), pointing out that the block-structured quad meshing is indeed a very constrained and a challenging task to perform.

4 | Are quadrilaterals better than triangles?

Triangle meshing represents the mature, well-documented, and robust technique used in various engineering fields. On the other hand, quad meshing has been an active research area in the last few decades, due to the constraints outlined earlier. Consequently, the legitimate question arises here: if so much time and effort are invested in generating the latter kind of meshes, do quadrilateral elements indeed offer better performances for numerical simulations compared to triangles?

It will be shown that the nature of the specific numerical problem constrains the choice of a mesh type, *e.g.*, the shape of elements, the level of structure, particular polynomial interpolation. An overview of elements' types of interest is shown in figure 2.14. The illustration of the importance of the choice of the mesh type (on the example of the approximation of a smooth function) is in figure 2.15.

It is vital to note the lack of works comparing the triangular and quadrilateral elements' performances. Nevertheless, available findings and inputs from FEM practitioners offer insight into this topic. A thorough comparison between tetrahedral and hexahedral elements is available in [17, 51, 52].

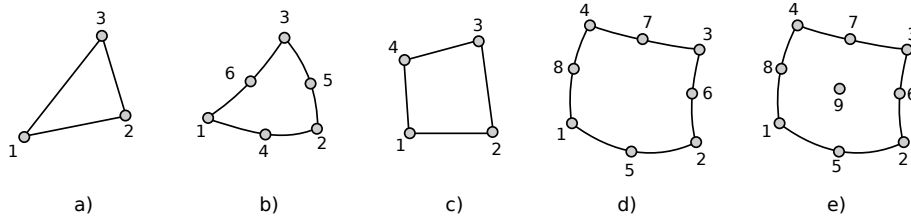


Figure 2.14: An illustration of different polynomial order of triangles and quadrilaterals. Triangle P1 a), triangle P2 b), quadrilateral Q1 c), quadrilateral Q2-serendipity d), quadrilateral Q2 e)

The level of structure. Block-structured quad meshes, by being combinatorially equivalent to regular grids, offer numerous advantages for numerical simulations [51]. As reported [17, 18, 19, 20], these advantages refer to more accurate results, the quality of the solution, decreased computational time, and the use for multi-structured domains. On the other hand, obtaining this kind of mesh typically involves a sophisticated algorithm (which can deal with challenging topology or geometry of the domain) and immensely more elements since structured meshes do not endorse well the rapid change of size [43].

Computational optimization. Quadrilateral elements display tensor-product nature which is easier to manipulate in numerical simulations, and the ability

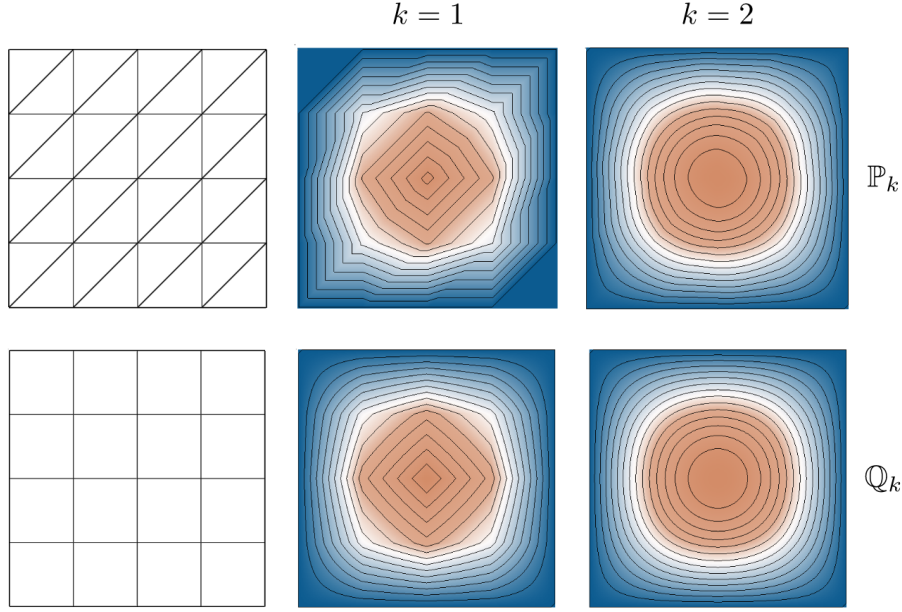


Figure 2.15: The FEM solution for $-\Delta u = 2\pi^2 \sin(\pi x) \sin(\pi y)$, $u = 0$ on $\partial\Omega$. Triangular and quadrilateral mesh on the left; elements of order one in the middle and order two on the right. A picture from [51].

to align with the principal directions of curvature [21, 41, 51]. These features advocate the usage of Q1 elements for anisotropic mesh transformation [41] as well as for fluid mechanics and fluid-structure interaction simulations [51].

Elastic, aero-elastic, and elastoplastic problems. On these benchmarks, quadrilateral elements are reducing the approximation error, computational time and the number of elements compared to triangles [20, 53]. More specifically, as findings [53] outline, Q1 elements are most suitable for elastic-free vibration analysis; Q2-serendipity for stress analysis, and Q2 for aero-elastic and laminated composite plates analysis.

Modal and stress analysis simulations. Triangular elements can cause mesh locking in bending computations and in simulations on nearly incompressible materials, which is why the authors [53, 54] highly suggest the use of quadrilaterals for these cases.

Miscellaneous. Special finite elements used in the private sector, *e.g.*, com-

mercial software Abaqus FEA, often demand or at least prefer quadrilaterals. The similar stands for practitioners in mechanics deformations, contact problems, computational fluid dynamics (CFD), and PDE's solvers as deal.II library.

The studies mentioned above did not consider non-elliptic PDEs, spectral methods, and numerically challenging problems such as fracture, dynamics shocks, collision response, or multi-physics simulations. The analyses suggested would be of immense importance in thoroughly addressing the question from the title.

CHAPTER



Methods for quad mesh generation

The evolution of quad mesh generation is reflected in the development of methods that can comply with some (or majority) of the restrictions mentioned above. This chapter provides their condensed overview, where special attention is paid to the methods closest to this thesis. Afterward, structural properties of the quad layout and its intimate relationship with the block-structured quad mesh are outlined. This section is followed by a presentation of challenges for generating the quad layout via an auxiliary object - cross-field. The characteristics of cross-fields and their connection with quad meshes are shown in the end.

1 | A brief literature review

The history of quad mesh generation started with algorithms that were in their core manual, yielding great human expertise and being a time-consuming process [11]. Since then, especially in the last several years, numerous algorithms have been developed, and significant progress has been made [21], figure 3.1. This progress is reflected in influential evolution in speed and quality. Nevertheless, quad mesh generation is still a very active research area because its several aspects demand additional efforts to fulfill all requirements of the vast set of its applications. As reported by [21], the most critical areas which need further improvement are approximation and quad mesh quality, robustness on geometrically inconsistent inputs, efficiency, interactive control, scalability, and evaluation.

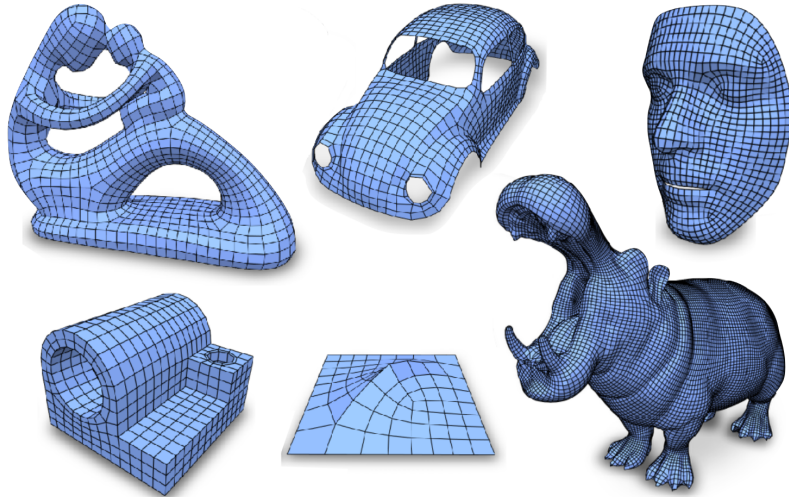


Figure 3.1: High-quality block-structured quad mesh generation (picture from [55])

Bearing in mind that numerous - essentially different approaches have been developed, the classification of quad mesh generation algorithms can be made based on several different criteria. For instance, it can rely on the level of quadrilateral mesh structure that the approach can produce (as presented in [11] and [56]), shown in figure 3.2, or on the essential principle of the algorithm (as in [21]), more details on developed mesh generation methods in [15, 21, 28, 29, 57].

A review of approaches closest to the subject of this thesis is discussed in the following. In particular, a group of methods based on obtaining the quad layout of the domain is presented. This set of approaches aims to partition the

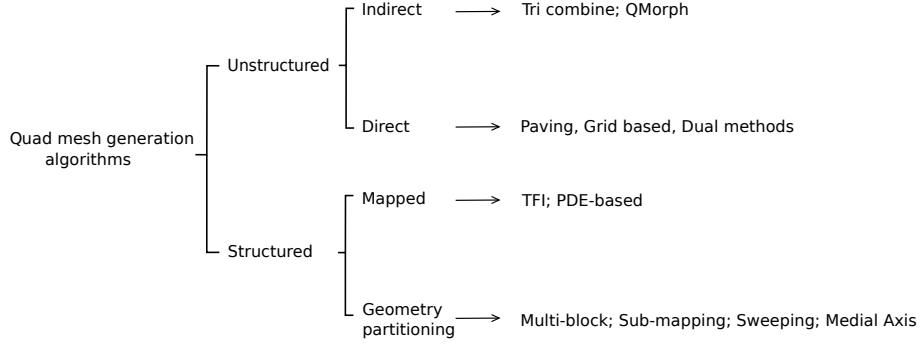


Figure 3.2: Classification of quadrilateral mesh generation methods

geometry so that obtained patches can be trivially refined in order to generate a block-structured quadrilateral mesh of high quality (pipeline illustration in figure 3.3). One way to obtain the partitioning of the domain is by using the driving force of a *cross-field* (see section 3.1) to guide and orient the partitioning of the object. By following the cross-field streamlines emanating from singularities partitions are obtained and at this point their validity and shape can be improved. After, the valid partitions of the geometry are typically individually remeshed by one of the *core (mapped)* algorithms. These algorithms are easy to implement, have low computational cost and work only on simple geometries or, like in this case, patches - which are topologically equivalent to quadrilaterals. The basic idea of core methods is establishing a mapping between computational and physical domain figure 3.4. The most habitually used ones are algebraic (transfinite interpolation TFI) and PDE-based (elliptic, parabolic, hyperbolic) algorithms, the details of their implementation can be found in [15].

The advantages of a quad layout generation are numerous; however, they indeed come with the high price in dealing with complex and time-consuming algorithms, to which an extensive amount of research has been devoted [58]. The family of quad layout methods shares the same underlying principle and the goal, still there are significant differences among its constituents in terms of the cross-field features exploitation and background numerics. Among the developed algorithms, a general distinction can be made among the ones which are, like in the case of this thesis, constructing a cross-field structure that will guide the integral lines emanating from singular points [19, 59, 60, 61, 62, 63]; computing a seamless global parametrization of the domain where integer iso-values of the parameter fields form the sides [55, 64, 65], or the most recently using Riemann surface theory [66, 67].

To discuss the first group, the work of **Fogg et al.** [68] is reviewed. This approach generates smooth cross-fields on triangulated surfaces by using a fast

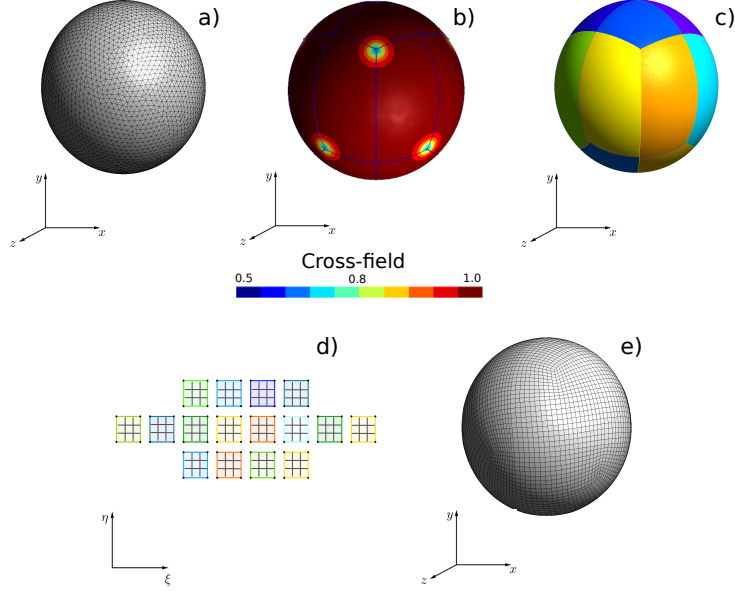


Figure 3.3: The principal workflow of quad layout methods based on cross-field generation. Triangulated mesh of a model a), a cross-field b), extracting patches c), remeshing patches d) and a final block-structured quad mesh e).

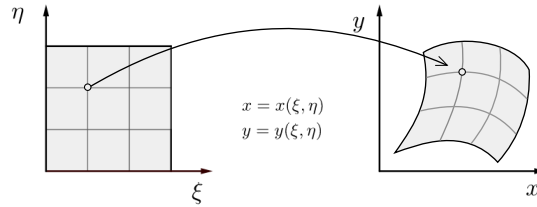


Figure 3.4: Mapping of computational (left) onto physical space (right)

marching algorithm (introduced by [69]). To compute a cross-field, the authors minimize and penalize the Dirichlet energy, so that the variation of directions and element sizes is minimal. Further, the cross-field is set to be aligned with the boundary, so that a conforming quadrilateral mesh can be produced. Partitioning of the geometry is obtained by following the critical streamlines, *i.e.*, *separatrices* of a cross-field from irregular points and boundary corners by using the Runge-Kutta numerical scheme. The authors only show results of 2D domains, on which the obtained multi-block decomposition has a large number of thin patches, which is not suitable for obtaining a high-quality block-structured quad mesh. A significant leap into improving geometry partitioning, has been

taken by **Kowalski et al.** [19], see figure 3.5. Their algorithm has a comparable numerical background to [68] which also does not handle 3D domains, yet it is able to preserve the symmetry of the domain and minimize the number of singular points. The quadrilateral mesh with constant size elements is created by refining each patch by TFI.

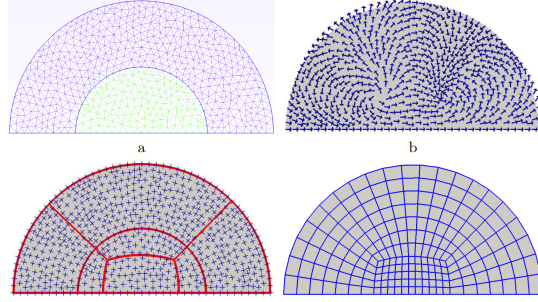


Figure 3.5: An overview of the approach [19]. Triangulated mesh of the geometry with two domains a), a cross-field b), partitioning of the domain c) and a full quadrilateral mesh obtained by TFI d) (picture from [19])

An exceptional trade-off between the minimal number of singular points and distortion of the geometry is accomplished using Ginzburg-Landau energy functional instead of Dirichlet energy. Driven by this idea is the method from **Viertel et al.** [63] which relies on computing a harmonic cross-field based on Ginzburg-Landau PDE, widely known in mathematical physics. The obtained partitions are guaranteed to be four-sided regions, yet with possible T-junctions, which are a direct consequence of limit cycles appearance, *i.e.*, separatrices that don't stop on the boundary or at a singularity, figure 3.6. The authors restrain their work on simple planar Euclidean domains, yet claim that its extendable to surfaces due to the theoretical background from [70].

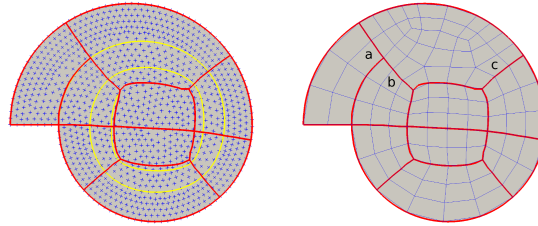


Figure 3.6: A cross-field with a limit cycle (left) and non-conformal partitions (right) (picture from [63])

When it comes to *global parameterization* approaches, their most important contributions for quad meshing purposes, as highlighted by [64], are high

regularity due to explicit control over singularities and smooth distribution of distortion achieved by convex variational formulations. One of the representatives of this set of algorithms is the work of **Ray et al.** [65]. This method aims to construct a globally smooth parameterization obtained from a cross-field (user-defined or based on principal curvatures). It is able to construct quasi-conformal (angle preserving) and, with the aid of introducing more singularities, quasi-geometric (angle and area-preserving) parameterization. The algorithm relies on well-defined and curl-free cross-field, therefore the operations of smoothing and scaling are performed on it. Afterward, iso-parametric curves of the parameterization are penalized to be as tangential as possible to the input vector fields. Patches obtained in this manner are extracted, split in singular areas and finally re-parametrized. This method is able to produce quad-dominant patches, the workflow is shown in figure 3.7. Like [65], a method from **Kälberer et al.** [71], known as *QuadCover*, is focused on constructing a global surface parameterization via cross-field guidance. By defining and optimizing the topological structure of the surface and ensuring that the parameterization is globally closed, this method achieves the level of curvature alignment of [65], while drastically reducing the number of singularities and obtaining all quadrilateral patches.

Although following the global parameterization idea from above, **Campan et al.** [72], in addition, also ensures geometric fidelity and structural simplicity of patches, which accomplishes the task of creating full quadrilateral mesh (the pipeline is shown in figure 3.8). To ensure valid connectivities of patches, the authors introduce the concept of *dual loops*. This idea is based on generating the loops aligned to the cross-field by following the closest directions until singular points are separated from each other. By construction, dual loops obey the following properties: intersect only transversely, no three loops intersect in one point, at least two loop segments bound each region, and each region has disc topology. Dualizing the graph formed in this manner, generates the patches with regular connectivity and good shape. From this point, a final quad mesh is generated trivially. This method’s only drawback is in the local minima since a quad layout is obtained via the greedy process. The same authors later on introduce a novel *quantization* concept, where an integer value is assigned to every constituent of the quad layout [55] so that the global parametrization of an object can be obtained. With clearly defined constraints and properties, the quantization process showed as particularly beneficial for repairing T-meshes into block-structured ones [42].

Bommes et al. [73] base their algorithm on global parameterization by starting from insuring well-defined principal curvature directions. Afterward, a cross-field aligned with curvature is constructed, so that the method can perform well on models with sharp feature edges. The computation of this cross-field relies on an angle field with period jumps of 90 degrees. Consequently,

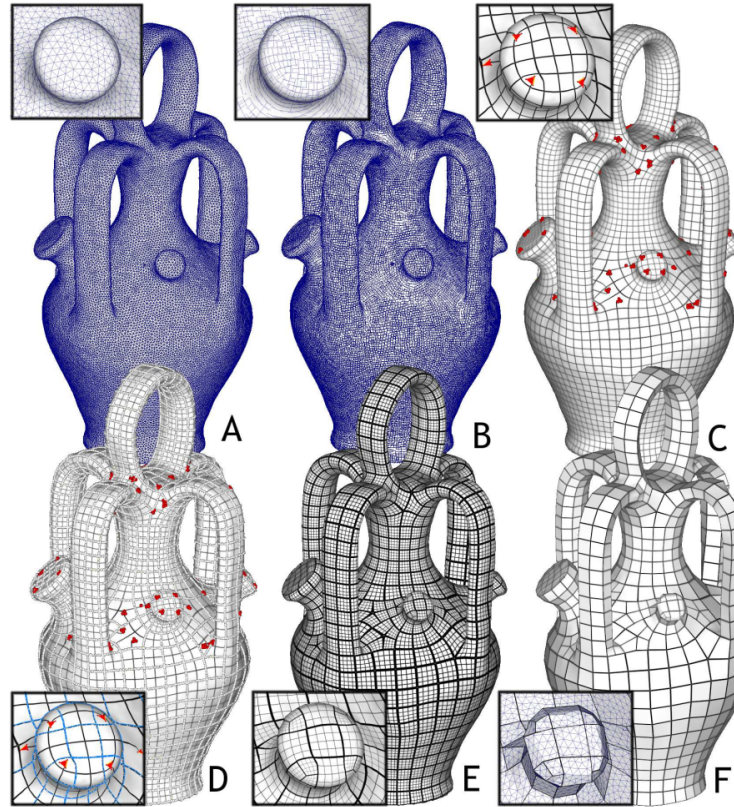


Figure 3.7: The pipeline from Ray et al. Obtained patches (D) from a smoothed cross-field (B) (picture from [65])

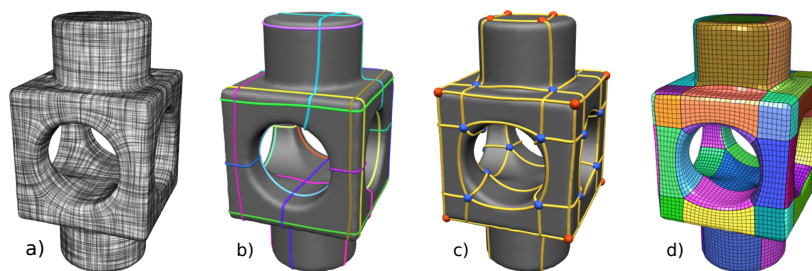


Figure 3.8: Global parameterization via dual loops. A cross-field a), dual loops b), extracted patches c) and full quad mesh d) (picture from [72])

each face is assigned with a real number and each edge with an integer. After that, the algorithm minimizes the energy obtained from angles which are real values and the jumps which are integer values, whence *mixed-integer problem* formulation. A global parameterization with singularities of integer values is generated from these steps, wherefrom a quad mesh is derived, figure 3.9. This algorithm reduces the number of partitions in a quad layout, which in some cases can lead to deviation from the cross-field and finally introduce distortion. An upgrade of this work is a method [64] that uses a global optimization approach instead of a habitual greedy strategy. It overcomes the degeneracies in parameterization caused by the not strictly positive (local) Jacobian determinant of the piecewise linear mapping by using *Mixed-Integer Quadratic Programming* formulation, so that the obtained patches indorse the class of *Integer-Grid Maps*.

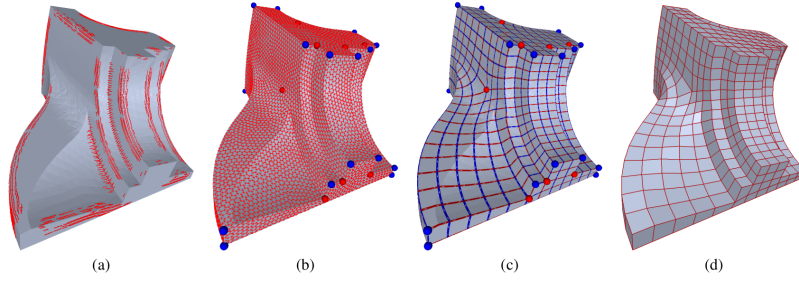


Figure 3.9: Global parameterization via mixed-integer formulation. Alignment constraints a), cross-field b), a globally smooth parameterization c) and quad mesh d) (picture from [73])

Very recently, the correspondence between a quad mesh and a special Riemannian metric with cone singularities, *i.e.*, *quad mesh metric* has been derived by **Chen et al.** [66] and **Lei et al.** [67]. To show this equivalence, the authors use the algebraic-geometric approach and rely on the theoretical background of Riemannian surface properties and meromorphic quartic differentials. In addition, [67] presents Abel-Jacoby conditions, *i.e.*, necessary and sufficient conditions for imposing specific singularity configuration, pointing out that, for instance a quad mesh on a torus with 2 singularities of valence 3 and 5 does not exist.

2 | Quad layout for quad meshing

The quad layout represents an effective structure for parameterization, obtaining subdivision surfaces and shape abstraction, as pointed out by [74]. As shown, partitioning the domain into patches can be performed in different

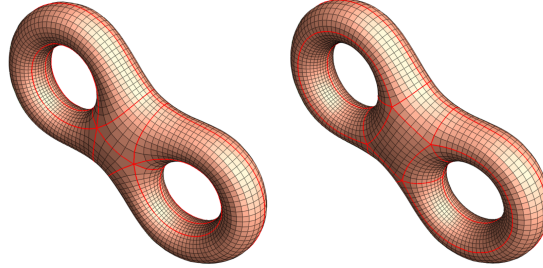


Figure 3.10: A quad mesh with 4 and 8 singularities on a genus 2 surface, a picture from [66]

manners and depending on the driving force of the background numerics, the obtained blocks can be unstructured, ill-shaped or of high quality. The properties as shape, number, level of distortion and structure of obtained partitions greatly influence a final quad mesh. The most desirable features that partitions can acquire are patch simplicity, high-level structure, conformity, rectangularity, and feature and principal direction alignment [28]. This specific class of multi-block grids that fulfills these properties is in literature known as a *quad layout*. Quad layouting of the model enables the generation of multi-block structured meshes which represent, as highlighted by [21], "the most important class in terms of applications" - used in polygonal and high-order modeling, texturing, mesh compression and isogeometric analysis. On the other hand, the algorithm for generation of high-quality quad layouts is very challenging to develop, due to capturing strict inherent structure, geometrical and topological features while solving the optimization problem which includes discrete (number and valence of nodes), combinatorial (layout graph connectivity) and continuous degrees of freedom (embedding of nodes, partitions and partitions' edges), as pointed out in [13].

In order to capture the constraints of the relationship between a quad layout and a quad mesh, the structure and properties of the former are firstly discussed. In the following, a concept and construction of a quad layout graph are introduced, so that refining and modifications of partitions can be deliberated from a more global point of view. Details on obtaining a quad layout from a cross-field can be found in section 3 and additional details on its generation in a recent extensive survey [28].

2.1 Structural properties of a quad layout

A quad layout of a surface consists of patches that are bounded by partition's edges, *i.e.*, *arcs* and nodes. A chain of arcs bounded by two singularities is called a *separatrix*, as shown in figure 3.11. The number of separatrices springing up

from a singularity defines its valence and is always different from 4. Nodes that are obtained by intersection of two separatrices or a separatrix and the boundary are called *intersection nodes*, and their valence is always 4.

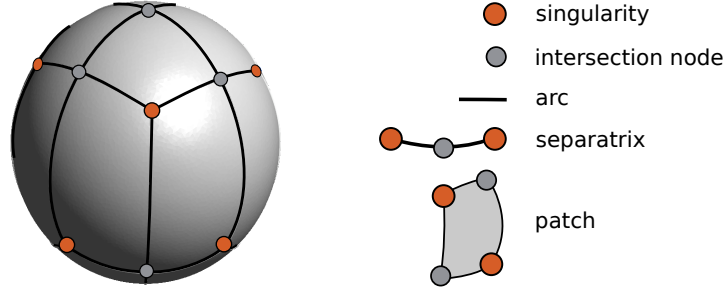


Figure 3.11: An illustration of a quad layout structure

When it comes to quad layout properties, a definition from [28] states:

Definition 2.1. If each arc of a quad layout is part of a separatrix, the layout is *minimal*.

This definition points out that there are no chains of arcs that are cyclic or start and end at the boundary in a minimal layout. Further, a quad layout is *non-conformal* if there exist two patches that share only a part of the arc, otherwise *conformal*. The former case arises from a presence of limit cycle(s) from underlying parameterization whose cuttings produced T-junction(s). Constructing a non-conformal from a conformal quad layout can be accomplished by adding/removing arcs and points. The vice-versa process is usually impossible to achieve on complicated geometries due to the connectivity constraints. However, it is feasible to reduce a limited number of T-junctions from a non-conformal quad layout (as in [74, 75]), so that the underlying partitions are suitable for a specific application.

From a higher point of view, the *quality* of a quad layout is in general assessed based on the specific application. However, as pointed out by [28], it must at least fulfill the geometric fidelity and structural simplicity, *i.e.*, patches should map to planar rectangles with low parametric distortion while the number of patches remains low. This task is hard to fulfill because a small number of patches can produce significant distortions, and only a small amount of state-of-the-art methods can find a good trade-off among these properties.

Depending on an arbitrary surface's geometrical and topological characteristics, its decomposition into partitions can be very complex [28]. To modify a quad layout (for instance remeshing or simplification), the concept of its *planar embedded graph* and its *dual graph* is invoked. An embedded planar graph of a

quad layout is a layout graph $\mathcal{G} = \{V, E\}$ formed from vertices V of the quad layout and its corresponding edges E , drawn (embedded) in the plane without edge crossings. Habitually, connectivity and orientation data are encoded on top of this informations. The dual planar graph $\mathcal{G}^*$ is obtained from a planar graph simply by exchanging the functions of vertices and faces, as illustrated in figure 3.12. Details on the structure and generating a planar graph from a quad layout are presented in Chapter 4 and the thorough theoretical background in [76].

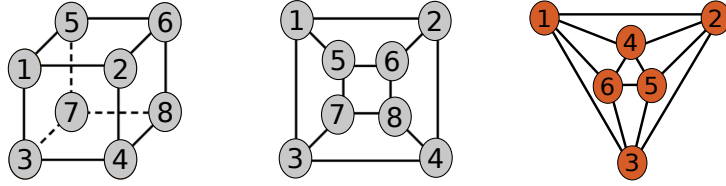


Figure 3.12: A quad layout of a cube (on the left), its planar graph with 8 vertices, 6 facets and 12 edges (in the middle) and the dual graph with 6 vertices, 8 facets and 12 edges (on the right).

When it comes to the merits of layout graph construction for valid *modifications* or obtaining well-defined partitions, the authors of [74] reduce the number of T-junctions by firstly constructing a graph of all potential separatrices, which is further used for obtaining a final layout graph where (by the construction of their algorithm) completeness and orthogonality of separatrices is insured. The method [72] gains the structural simplicity of patches by generating a graph of all cross-field aligned loops, where from, by dualization of its minimal subset a quad layout is extracted. A method [77] constructs a graph defining all possible layouts of the surface derived from a given frame field. By using constrained minimum weight perfect matching on it, this work accomplishes the goal of obtaining valid partitions.

2.2 Quad layout and quad mesh relationship

Taking into account the presented structures of both quad layouts and quadrilateral meshes and the fact that the number and placement of irregular vertices (singularities) are the same in both cases [21], the former can be viewed as a coarse version of the latter one. Despite this structural resemblance, quad layouts convey important data as capturing intrinsic connectivity complexity, which otherwise could not be sustained if the aim of the method was only preserving the quadrilateral shape [78]. The primal role of a quad layout is a trustworthy partitioning of the domain with the minimal number of patches and distortion on top of which additional data can be encoded [28]. On the

other hand, the objective of a quad mesh to ensure the geometrical fidelity of a model's representation. Their quality assessments also differ: quadrilateral mesh quality can be measured locally with different metrics (for instance scaled Jacobian, Hausdorff distance, editability and rigging quality - more details in [21]), while quality of a quad layout represents *"a function of the layout's structural complexity, its topological connectivity and its geometrical embedding"*, as defined by [13].

Due to these core differences, the generation of a quad layout from a quad mesh is usually not suitable and it requires sophisticated algorithms [13]. Albeit it is not the topic of this thesis, its worth mentioning that some applications (typically for entertainment industry purposes) do perform this conversion, by simplifying the structure of a quadrilateral mesh as long as the good patch quality is preserved (more details in [78, 79, 80]) or improving the overall structure by conversion of T-mesh into the conformal quad layout [42].

Starting from a quad layout and by remeshing each patch individually, a block-structured quadrilateral mesh with control on size and (an)isotropy of elements can be trivially obtained. The great challenge in reaching this goal is in ensuring that a quad layout is of high quality (as being conformal, well-shaped, aligned with the model's features). Seeing through the lenses of [80], a quad layout is defined by the network of separatrices which connect cross-field's singularities, in terms of quadrilateral mesh this network is constituted of edges that arise from irregular nodes [74]. This connection points out not only the dependence of a quadrilateral mesh on quad layout properties, but also the remedy for improving the quality of both. More specifically, as highlighted by [74], the solution is in finding a genuine network of separatrices. In the past, this task relied on tedious manual efforts, nowadays (as the work of this thesis) on the suitable algorithm with a strong mathematical background, more details in: Chapter 4, Chapter 6 and Chapter 7.

3 | Automatic generation of a quad layout via cross-fields

The task of generating a quad layout of a surface can be performed via direct algorithms, spectral methods, parameterization approaches or direction field tracing techniques [77]. Cross-fields, the four symmetry direction fields, are ubiquitous for quad mesh generation and are of particular interest for this thesis. As observed by [75], the quality of a quad layout (and a final quad mesh) is in many ways related to the initial cross-field. To show this connection, a discussion on cross-field's representation, computation and characteristics is in the following.

3.1 A 2D cross-field generation

There are many varieties of directional fields: vector and tensor fields, line fields, cross-fields, frame-fields and vector-set fields, which are used in numerical analysis, medical imaging, computer graphics and geometry processing methods, more details in a survey [81]. Their classification can be made based on several criteria as the type of data (scalar, vector, tensor) or temporal and spatial dimension (point, line, surface or volume), as detailed in [82]. Further, depending on an application's objectives, their representation, discretization and optimization algorithms also differ at large. In general, direction fields represent a very broad domain, which surpasses the scope of this work. Therefore, finding all additional information is pointed out to an elaborated mathematical background in [83] and comprehensive visualization techniques in [82, 84, 85].

Cross-fields, the special kind of directional fields, are of high interest for quad meshing purposes because of their tensor nature and ability to guide, scale and orient the quadrilateral elements. In other words, a cross-field can be seen as a set of quads of infinitely small size whose edges are aligned with cross-field branches. Locally, cross-field features impose the orientation and size of quadrilateral elements. Further, it is possible to show that cross-fields have isolated singularities that correspond to the location of irregular vertices of the quad mesh (more details in [44]), which make them a popular auxiliary object on which a high-quality block-structured quad mesh can be built upon.

A 2D cross $\tilde{c}$ is a set of 2 orthogonal vectors and their opposite, *i.e.*, $\tilde{c} = \{\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, -\tilde{\mathbf{u}}, -\tilde{\mathbf{v}}\}$ with $\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}} = 0$. These vectors are called the cross' branches.

A unit 2D cross c is a 2D cross with unit branches, *i.e.*, $c = \{\mathbf{u}, \mathbf{v}, -\mathbf{u}, -\mathbf{v}\}$ with $\mathbf{u} \cdot \mathbf{v} = 0$ and $\|\mathbf{u}\| = \|\mathbf{v}\| = 1$.

Assuming that the domain Ω is a smooth orientable 2D manifold embedded in the Euclidian space E^3 , at each point $\mathbf{X} \in \Omega$ the normal unit vector $\mathbf{n}(\mathbf{X})$ is uniquely defined. A 2D cross-field on Ω is a map $\mathcal{C}_\Omega : \mathbf{X} \in \Omega \rightarrow \tilde{c}(\mathbf{X})$ with the constraint that branches of crosses $\tilde{c}$ belong to the tangent bundle $T\Omega$. In other words:

$$\tilde{\mathbf{u}}(\mathbf{X}) \cdot \mathbf{n}(\mathbf{X}) = \tilde{\mathbf{v}}(\mathbf{X}) \cdot \mathbf{n}(\mathbf{X}) = 0, \forall \mathbf{X} \in \Omega. \quad (3.1)$$

It is possible to define a local orthonormal basis $(\mathbf{t}(\mathbf{X}), \mathbf{T}(\mathbf{X}), \mathbf{n}(\mathbf{X}))$ for all points $\mathbf{X} \in \Omega$. By doing so, a 2D cross $\tilde{c}$ can be defined with 3 scalar fields (θ, l_1, l_2) :

$$\begin{cases} \theta \text{ such as } \tilde{\mathbf{u}} = l_1(\cos(\theta)\mathbf{t} + \sin(\theta)\mathbf{T}) \\ l_1 = \|\mathbf{u}\| \\ l_2 = \|\mathbf{v}\| \end{cases} \quad (3.2)$$

At first glance, it would seem that for quad layout and quad mesh generation, only crosses' orientation is relevant. Well-established methods are

therefore focusing on unit 2D cross-field generation, and in this manner, the problem complexity is reduced as $l_1 = l_2 = 1$. Unit crosses can be defined with one scalar θ , as illustrated in figure 3.13.

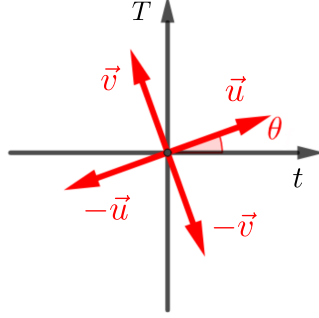


Figure 3.13: Illustration of a 2D cross definition

It is important to note that a unit cross c is not uniquely defined by a scalar θ . Indeed, any $\theta_1 = \theta + k\frac{\pi}{2}$, $k \in \mathbb{Z}$ is also representing c . Using a one to one 2D cross representation allows easing the 2D cross-field generation through a numerical method and can be done through Hertzmann's representation.

The representation of Hertzmann [86] allowing to *uniquely* define a cross is:

$$\mathbf{a} = (a_1, a_2) = (\cos 4\theta, \sin 4\theta), \quad \theta \in [0, \frac{\pi}{2}) \quad (3.3)$$

where $\mathbf{a}$ is invariant by a rotation of $\frac{\pi}{2}$. At the same time, $\mathbf{a}$ can be seen as a complex number as well, where $\mathbf{u}(\theta) = e^{i\theta}$, with a_1 and a_2 as its real and imaginary part and $a_1^2 + a_2^2 = 1$.

The thorough vector calculus and its connection with differential geometry and Riemann metric are elaborated in [81], the special focus here is on the key definitions which are of high importance for quad meshing purposes.

An example of 2D cross-field computed on a torus is given in figure 3.14.

Noteworthy properties of a cross-field $\mathcal{C}_\Omega$ are its branches flux and rotation, which are respectively estimated by computing the divergence $\text{div}(\mathbf{u})$ and the curl $\text{curl}(\mathbf{u})$. Special classes of cross-fields are *divergence-free* and *curl-free* cross-fields.

- A *divergence-free* cross-field is a cross-field $\mathcal{C}_\Omega$ where $\text{div}(\tilde{\mathbf{u}}) = \text{div}(\tilde{\mathbf{v}}) = 0$:

$$\nabla \cdot \tilde{\mathbf{u}} = \frac{\partial \tilde{u}_1}{\partial x} + \frac{\partial \tilde{u}_2}{\partial y} + \frac{\partial \tilde{u}_3}{\partial z} \equiv 0. \quad (3.4)$$

This type of cross-field is area/volume preserving and can be particularly useful for simulation of deformation or mechanics of fluids.

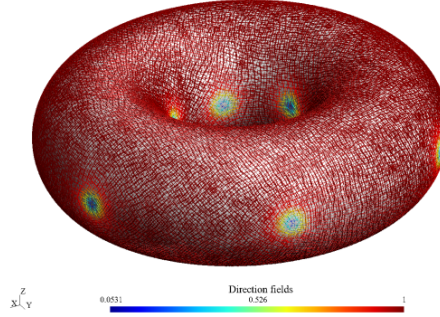


Figure 3.14: A 2D cross-field on a torus. Blue color describes that the cross-field norm is close to zero and red to be close to unity, a picture from [2]. Areas where the crosses' norms collapse to zero are vicinities of singularities.

- A *curl-free* cross-field is a cross-field $\mathcal{C}_\Omega$ which is path independant, *i.e.*, where $\text{curl}(\tilde{\mathbf{u}}) = \text{curl}(\tilde{\mathbf{v}}) = 0$:

$$\nabla \times \tilde{\mathbf{u}} = \nabla \times \tilde{\mathbf{v}} \equiv 0. \quad (3.5)$$

Some works [65] emphasize this type of cross-fields for quad meshing purposes. Since it is not always feasible to generate it, the authors rescale the cross-field to obtain curl correction, leading to high-quality curvature-aligned quad mesh. In this particular case, the parallel transport of vectors is linked to the geodesic curvature [81]. For optimal results, it is also possible to encode additional information on the curvature. Other desirable cross-field properties include fairness, feature alignment, symmetry and control over field topology [81].

Nevertheless, neither *divergence-free* nor *curl-free* cross-field is suitable to generate a valid quad layout and associated quad mesh in the general case. According to Riemannian manifold theory, a cross-field $\mathcal{C}_\Omega$ will generate a valid quad layout and quad mesh if and only if the Lie bracket of its branches $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ vanishes (more details in Chapter 6 and Chapter 7). This requirement coincides with the constraint of a curl-free cross-field in the planar case but takes a different meaning on a 2D manifold.

The considered 2D manifold Ω is embedded in the euclidian space E^3 , natural choice for its Riemannian metric $\langle \cdot, \cdot \rangle_\Omega$ is the restriction of E^3 natural metric $\langle \cdot, \cdot \rangle_{E^3}$ to Ω tangent bundle $T\Omega$. The associated Levi-Civita connection is noted ∇ . The Lie bracket of two vectors $\tilde{\mathbf{u}}, \tilde{\mathbf{v}} \in T\Omega(\mathbf{X})$, $\mathbf{X} \in \Omega$ is defined as:

$$[\tilde{\mathbf{u}}, \tilde{\mathbf{v}}] = \nabla_{\tilde{\mathbf{u}}} \tilde{\mathbf{v}} - \nabla_{\tilde{\mathbf{v}}} \tilde{\mathbf{u}}. \quad (3.6)$$

A cross-field $\mathcal{C}_\Omega$ will generate a valid quad layout if and only if:

$$[\tilde{\mathbf{u}}(\mathbf{X}), \tilde{\mathbf{v}}(\mathbf{X})] = \mathbf{0}, \forall \mathbf{X} \in \Omega. \quad (3.7)$$

This type of cross-field is *integrable* and therefore of high importance in quad meshing, detailed information available in Chapter 6 and 7. Nevertheless, computing such a cross-field with no prior information - such as singularities number, index and location, holonomies - turns out to be a difficult task. Therefore, existing methods for 2D cross-field generation are focusing on generating as smooth as possible unitary cross-field, leaving the *integrability* problem aside and dealing with it in the cross-field post-processing stage. The downside of not taking into account cross-field integrability is the appearance of limit-cycles, making the generated quad layout invalid, detailed in section 3.3.

3.1.1 Formulating the cross-field generation as a non-linear problem

The computation of a unitary cross-field typically involves either a non-linear or integer-valued task. The first group, which is of interest here, represents an optimization problem involving the solving of a particular partial differential equation (PDE) by using FEM, typically over a triangulation of domain Ω . However, there exist different discretizations: face-based, edge-based (*e.g.*, Crouzeix-Raviart elements [44]) and vertex-based (*e.g.*, Lagrange P1 elements [63]), as well as the numerous formulations to fulfill this task. It will be shown in the following that the choice of specific PDE and background numerics is of high importance since it directly affects the results in practice.

The formulations of divergence and curl outlined above satisfy Stokes' theorem and can be used to derivate the Dirichlet energy, more details in [87]. Here, precisely the simple minimization of Dirichlet energy:

$$E^D(\mathbf{a}) = \frac{1}{2} \int_{\Omega} |\nabla \mathbf{a}|^2 dv \quad (3.8)$$

with the appropriate boundary conditions is used as a first approach to generate a cross-field. Although simple, this approach is not adequate for obtaining a quad layout [60], see figure 3.15. On the other hand, it can be particularly useful as a starting point for a more elaborate formulation. Namely, by adding an additional penalty term to ensure that the norm of the cross stays as close to unity as possible, a new functional with promising results for quad meshing can be developed:

$$E^{GL}(\mathbf{a}) = E^D + E^P = \frac{1}{2} \int_{\Omega} |\nabla \mathbf{a}|^2 + \frac{1}{4\epsilon^2} \int_{\Omega} (|\mathbf{a}|^2 - 1)^2. \quad (3.9)$$

The second term is scaled by a coherence length ϵ , which represents a mesh size and influences the position of singularities. The functional E^{GL} itself is known in the literature [22] as Ginzburg-Landau functional and it is reported to yield very competitive results, in terms of a good trade-off between the number of singularities and distortion [44, 60, 63]. More details on Ginzburg-Landau theory and its application in meshing are in Chapter 4.

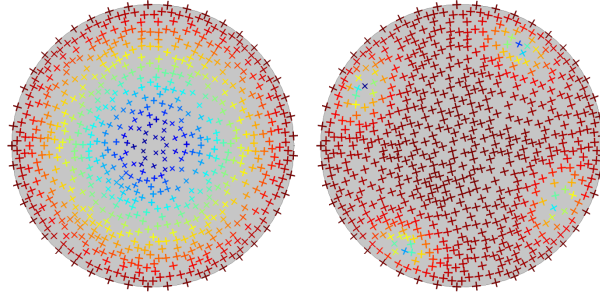


Figure 3.15: A 2D cross-field obtained via: minimizing Dirichlet energy (on the left) and minimizing Ginzburg-Landau energy (on the right). Blue color describes that the cross-field norm is close to zero and red to be close to unity, a picture from [2].

Ginzburg-Landau PDE represents a very beneficial tool for quad-meshing purposes, specifically due to the adequate singularity configuration [60]. Although the number of singularities is not minimal, their symmetrical location aids the reduction of overall distortion, more details in Chapter 4. The corresponding theory, coming from the domain of superconductors and superfluids [22], explains the singularity behavior: their stability, corresponding energies, and locations, unlike other formulations that assess the cross-field quality only by following Poincaré-Hopf theorem [2]. More specifically, the asymptotic behavior of Ginzburg-Landau energy, thoroughly elaborated in [22], points out that the singularities carry the overall energy and for this matter only singularities of valence 3 and 5 are supported by this model, while their number is bounded, *i.e.*, $N = 4\chi$, more details in Chapter 5. This theoretical remark highly suggests using this particular model to design cross-fields due to the low number and valence of singular points.

The benefits of the Ginzburg-Landau model for meshing purposes indeed come with high computational costs [44]. The authors of [63] bypass this difficulty by using a generalization of the Merriman-Bence-Osher (MBO) algorithm [88]. Their method relies on repeating the implicit diffusion step and pointwise renormalization as long as threshold convergence is not achieved, and by doing so, efficiently approximate local minimizers for the Ginzburg-Landau energy, more details in Chapter 6.

3.1.2 Formulating the cross-field generation as a linear problem

The downsides of generating a cross-field by minimizing Ginzburg-Landau energy or using a mixed integer formulation [64] are that the location of obtained singularities depends on the mesh used for the problem discretization (see Chapter 6). Singularities are repulsed from the refined regions and attracted to the coarser ones. As vicinity of singularities are areas where we would need a more precise discretization due to the presence of high gradients of the cross-field representation, this behavior is highly undesirable. Furthermore, the cross-field generation is computationally expensive due to the non-linear nature of these formulations.

In order to bypass these difficulties, methods to compute a cross-field with an imposed singularity configuration - possibly with high valences - wherefrom a high-quality cross-field and finally a quad mesh can be obtained have been developed [61, 67]. The one presented in Chapter 6 and Chapter 7 relies on the following constraints:

- branches $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ have to be of the same norm, and
- $[\tilde{\mathbf{u}}, \tilde{\mathbf{v}}] = \mathbf{0}$.

These conditions for the 2D cross-field generation, demand the existence of a conformal mapping between the resulting quad mesh and the domain Ω . Relying on these constraints and working with an imposed singularity configuration, it is possible to introduce a dual scalar field $H = \log \|\tilde{\mathbf{u}}\|$. This field is continuous on Ω , endorsing the reformulation of the 2D cross-field generation problem as a diffusion problem on H . The detailed theoretical foundation for this approach is available in [22, 89], and computational details in Chapter 6 (planar domains) and Chapter 7 (2D manifold surfaces).

Computing a cross-field with an imposed set of singularities allows obtaining a mesh independent solution, the *integrability* condition is now taken into account in the formulation and the cross-field generation problem becomes linear. This decrease in problem complexity comes from the fact that in this case, singularity locations and indices are known and imposed by the user. An example of the quad layout and quad mesh generated using the dual scalar function H is presented in figure 3.16.

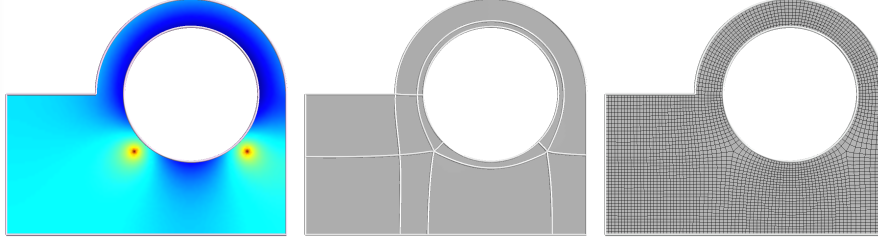


Figure 3.16: From left to right: dual scalar field H , quad layout obtained and resulting quad mesh.

3.2 A 2D cross-field topology

The cross-field's topological degrees of freedom are determined by the *turning numbers* of a homology basis of cycles [62], which measure the number of rotations of the cross-field along loops. Turning numbers also generalize the notion of *singularities* (equation (3.10)), an important cross-field characteristic [90], representing points on the domain where the norm l of the associated cross tends to zero. A cross-field has a finite number of singularities, but it is not an arbitrary set. Further, it is possible to calculate the identifier of such a point, *i.e.*, its *index* by computing the number of times the cross-field coils around it. More specifically, let γ be an infinitesimal curve around singularity X_i not containing any other singularity, the index of X_i is then estimated as:

$$\text{index}(X_i) = \frac{1}{2\pi} \oint_{\gamma} d\theta_f, \quad \theta_f = \arctan\left(\frac{u_1}{u_2}\right), \quad u_1 = \cos(4\theta), u_2 = \sin(4\theta), \quad (3.10)$$

where $2\pi = \oint_{\gamma} d\theta$, an illustration in figure 3.17.

It can be shown that the number of singularities and their respective indices correspond exactly to the topology of the given surface [62, 81, 91]. The Poincaré-Hopf theorem states exactly this relationship:

$$\sum_{i=1}^N \text{index}(X_i) = \chi(\Omega), \quad X_i \in \mathcal{S}. \quad (3.11)$$

This finding indicates that singular points will emerge on domains with Euler characteristic $\chi \neq 0$. Pictorially, as *The hairy ball theorem* [92] points out, it is possible to comb a torus without tufts ($\chi = 0$), which is not the case with the sphere ($\chi = 2$) though.

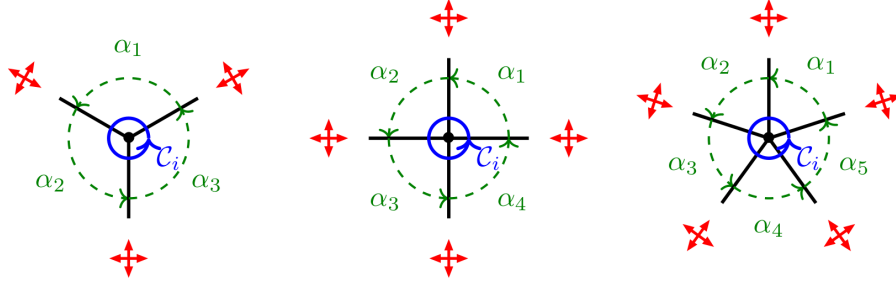


Figure 3.17: Illustration of singularities of index $\frac{1}{4}$, 0 and $-\frac{1}{4}$ (from left to right), a picture from [2].

By following the Euler polyhedron and Poincaré-Hopf theorems, it can be derived that the singularities of a cross-field represent the irregular points of the quad mesh and quad layout [13, 44]. Moreover, their configuration (in terms of their locations and number) greatly influences the quality of a quad layout and, finally, a block-structured quad mesh [21].

An index of singularity is related to the number of integral lines, *i.e.*, *separatrices* that spring out from this point, also known as its *valence*. These two concepts are related via:

$$valence(X_i) = 4 - 4 * index(X_i), \quad X_i \in \mathcal{S}. \quad (3.12)$$

The typical valence of singularities appearing in cross-fields is 3 or 5, where a point with valence 4 is a regular point obtained via orthogonal crossings of separatrices (an illustration in figure 3.18). The observation of equation 3.11 points out that doublets of singularities of valence 3 and 5 do not influence the final sum, which means that the cross-field may exhibit many irregular points, and by so highly degrade the quality of the final quad mesh. This remark highlights the need for control over singularity configuration via adequate formulation for cross-field computation.

The practice of imposing singularity locations and valences has been prevalent in the past when the partitioning of the domain was performed manually; nowadays, it is possible to do so in an automatic manner (see Chapter 6 and Chapter 7). By following the Poincaré-Hopf theorem it appears that the only constraint to be fulfilled is assuring that the number of singularities and their indices match the corresponding sum in equation 3.11. Nevertheless, it is not the case. The generation of a conformal quadrangulation is under the constraint of a combination of choices of holonomies and singularities' indices [93]. These additional constraints come from the assumption $\|\tilde{\mathbf{u}}\| = \|\tilde{\mathbf{v}}\|$. As an illustration of this surprising fact, an example of imposing only two singularities

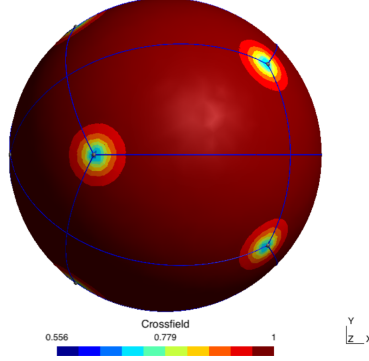


Figure 3.18: Cross-field singularities of valence 3 on the sphere, blue curves represent separatrices, a picture from [2].

on a torus $\chi = 0$ can be used. As already proved [67, 94, 95, 96], imposing a pair of valence 3 – 5 singularities, although fulfilling the constraints of Poincaré-Hopf theorem, is not feasible in this case. These works use different manners for proving this claim, the closest one to quad meshing purposes is the method [67], which uses the profound link between Riemann surface theory and quad meshing. The authors derive the *Abel-Jacobi theorem* and corresponding algorithm to give for the first time the necessary and sufficient conditions for imposing singularity configuration.

3.3 The separatrices network

To generate a quad mesh from a cross-field it is possible to use explicit and parametrization methods, more details in [21]. The former group of methods relies on tracing the integral lines emanating from singularities of the cross-field, *i.e.*, *separatrices* to partition the domain, as it is the case here.

Separatrices represent curves that are consistent with the underlying cross-field and are obtained via the propagation of integral lines emanating from singularities. As a formal definition from [19] states:

Let $\gamma(s)$, $s \in [0, 1]$ be a parametrization of a C^1 curve defined over Ω , with $\partial\gamma(s)$ its derivative and $\mathbf{u}_i$ orthogonal branches of a cross $\mathbf{u}$. Then γ is a *separatrix* if:

$$\forall s \in [0, 1], \quad \exists i \in [1, \dots, 4], \quad \partial\gamma(s) \times \mathbf{u}_i(\partial\gamma(s)) = 0. \quad (3.13)$$

These integral curves are a cross-field vital feature since their network determines the intrinsic connectivity complexity of a quad layout, and finally a

quad mesh [21]. The possible behavior of separatrices of a cross-field is explained via Poincaré-Bendixson theorem for manifolds [97], in quad meshing sense, their desirable properties include being aligned with cross-field, generating only orthogonal crossings and being as short as possible.

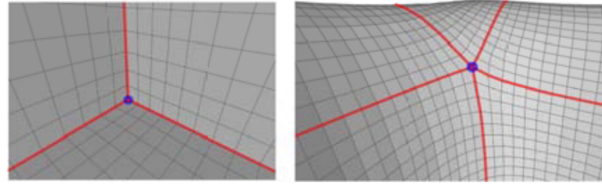


Figure 3.19: Illustration of separatrices springing out from a singularity of valence 3 on the left and valence 5 on the right, a picture from [21].

Obtaining a quad layout by simply tracing these curves is a challenging problem [74, 98, 99], since it may not necessarily provide a good solution [79, 80]. This issue typically arises from re-entrant corners and misalignment of singularities [100], which cause a global problem: thin partitions, non-quadrilateral patches or limit cycles in a final quad layout, as illustrated in figure 3.20.

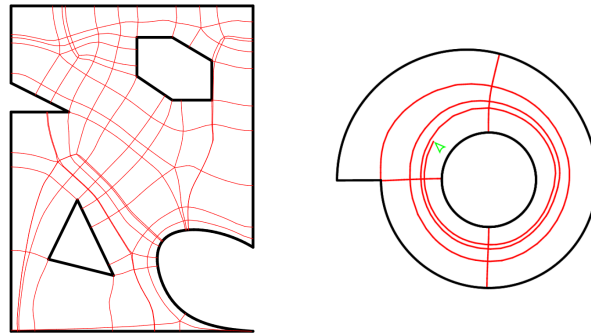


Figure 3.20: Illustration of thin partitions and non-quadrilateral patches on the left, a picture from [68]. A limit cycle appearance on the right, a picture from [59].

Of notable interest here are *limit cycles* which represent separatrices that coil up inside the domain, and by so not ending up neither at the singularity nor orthogonally to the boundary, more details in [83]. This type of separatrices is, for quad meshing purposes, typically stopped at the first (or last) orthogonal intersection, leaving behind a quad layout with T-junctions.

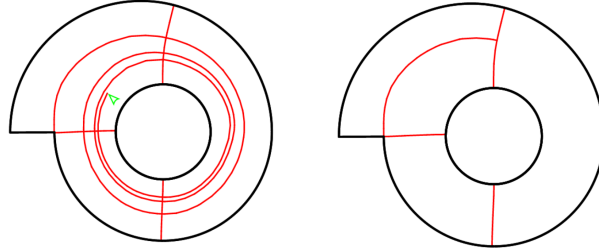


Figure 3.21: A limit cycle and reparation of a quad layout involving appearance of a T-junction, a picture [59].

The issues mentioned above highly inhibit the generation of a quad layout, and heterogeneous methods are developed to bypass these difficulties by finding the genuine network of separatrices, *i.e.*, the minimal number of simply connected integral lines. Some of the most notable approaches include graph-based techniques [74, 80, 101], collapse operation techniques [98, 100], separatrices snapping with the given tolerance around singularities methods [19, 60, 68], various quantization techniques [55] and their hybrid versions in order to obtain the best results. Novel criteria to obtain the minimal number of separatrices and reparation schemes of ill-defined patches for obtaining the genuine set of separatrices are detailed in Chapter 4, Chapter 6 and Chapter 7.

The majority of methods outlined above are relying on repairing the integral curves network, a work [77] suggests instead improving cross-field properties *a priori*. Namely, the authors suggest that curl-free cross-field are particularly useful in avoiding the existence of limit cycles. The thesis adopts this line of thinking and points out the importance of cross-field integrability and improving its accuracy around singular points (see Chapter 4, 6 and 7).

CHAPTER

4

Quad layout and quad mesh generation obtained with Ginzburg-Landau PDE

Numerous theoretical and experimental results proved the benefits of Ginzburg-Landau's theory in the area of superconductors and superfluids [102, 103, 104, 105]. The strict mathematical background presented in [22] has supported exploring its advantages for quad meshing purposes. More specifically, the motivation for its use comes from analyzing the asymptotic regime of energy minimization, pointing out the adequate type, position, and number of singularities. The following chapter details the quad meshing pipeline based on Ginzburg-Landau theory¹.

¹Jezdimirović, J., Chemin, A., and Remacle, J. F. (2019) *Multi-block decomposition and meshing of 2D domain using Ginzburg-Landau PDE.*, Proceedings of the 28th International Meshing Roundtable

Abstract

An in-depth method to generate a quad layout of the arbitrary 2D domain using 2D cross-fields solution of Ginzburg-Landau partial differential equation (PDE) is presented. It relies on the parameterization of decomposition of the domain, obtained by using particular PDE to generate direction fields, appropriate number, and localization of singular points and their separatrices. We have proved that solutions of particular PDE imply locally integrable vector fields and have the adequate distribution of singularities, advocating its usage. Quad layout graph was generated by the separatrices and extraordinary vertices of the domain (singularities, corners, and separatrices intersections), and obtained partitions were parameterized/remeshed. As a result, a mechanism to obtain block-structured quad mesh in an automatic manner is developed.

Keywords: Ginzburg-Landau PDE, cross-fields, quad layout, block-structured quad mesh

1 | Introduction

Block-structured quad meshes offer numerous advantages for mesh generation in general, as reported by an increasing number of authors [17, 18, 19, 20]. Some of the crucial improvements refer to: numerical stability, quality of the solution and computational time, efficient use of advanced vector extensions (AVX) of modern microprocessors, development of efficient/optimal preconditioners, dramatically reduced memory footprint and the use for multi-structured domains.

The goal of the method presented here is to obtain automatic solution for the issue of quad layout generation and quad meshing of 2D domains starting from a specific partial differential equation (PDE).

We consider as input a triangulated surface Ω . Ginzburg-Landau PDE is then solved on Ω which allowed computation of a cross-field $\tilde{C}_\Omega$ [44]. The cross-field is then used for computing a quad layout of Ω which is a global parameterization of the domain. In our approach, singularities are precisely located at first, lifting of the cross-field is then computed and used to obtain domain separatrices. This stage leads to a domain decomposition that is used to build a finite element quadrilateral mesh with the use of an elliptic smoother. Figure 4.1 presents an overview of the methodology.

Our method has specificities/advantages with respect to existing methods. First, we give a proof of integrability of unitary cross-field, justifying its usage for domain partitioning. Quad layout is generated to be aligned with the

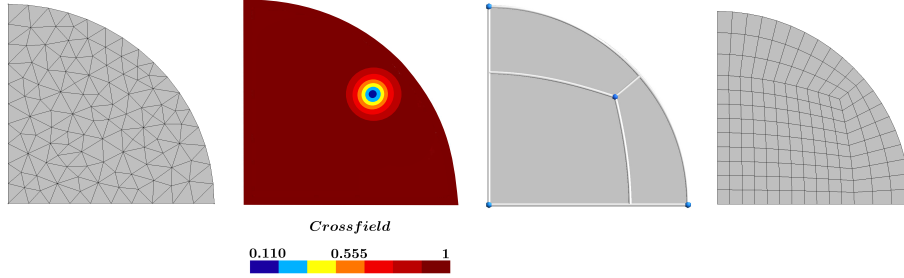


Figure 4.1: Simplicial mesh on Ω , computed cross-field, generated separatrices and final quad mesh (from left hand side to right hand side respectively)

cross-field. Control over adding/reducing the number of partitions for creation/discarding of boundary layers is presented. The numerical methodology used avoids the problem of limit cycles in the reported examples from [59, 63]. Finally, the true outcome of this paper is a fully functional block-structured mesh generator that is available in Gmsh [23], the open source finite element mesh generator.

2 | Related work

Numerous methods for field designed parameterization/remeshing have been developed in the past decades, thorough overviews given in [21, 28]. One of the research lines generally relies on cut graph methodology, where cross-field construction is followed by continuous parameterization, with differences on integer rounding [55, 64]. When it comes to developments in correlation with our work, similarly to [19, 63], we used a PDE based approach for quad layout generation purpose. In our case, more in depth developed procedures with extension to surfaces and specificities of the chosen PDE are shown. To avoid possible misalignments in the resulting parameterization, contrary to [98], the quad layout remained aligned with the cross-field. The techniques used for computing the cross-field are from [44] and they may seem to be similar to [65], although there are many differences. Some of the crucial divergencies are: the finite scheme relies on a Ginzburg-Landau PDE and not on the guidance field; the degrees of freedom are the real and imaginary parts of a vector field and not two angles defining the parameterization; the penalty factor is not constant, but it is governed by the mesh size. Following the idea of the importance of improving integrability of a cross-field by [65, 106], we have shown that, for a given cross-field, it is always possible to find a scaling scalar function allowing to obtain a locally integrable cross-field. This demonstration legitimates the

usage of 2D cross-fields to generate quad layouts and finally block-structured quad meshes.

3 | Purpose of 2D cross-fields for quad mesh generation

A 2D cross $\mathbf{c}$ is defined as a set of 4 orthogonal vectors of norm l , $\mathbf{c} = \{\mathbf{u}_k\}_{k \in \llbracket 1,4 \rrbracket}$. With a given 2D orthonormal basis $(\mathbf{x}, \mathbf{y})$, $\mathbf{u}_k = l \cos(\theta + k\frac{\pi}{2})\mathbf{x} + l \sin(\theta + k\frac{\pi}{2})\mathbf{y}$, represented in figure 4.2.

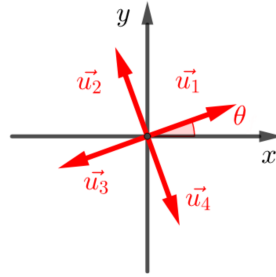


Figure 4.2: Cross definition

On a 2D domain Ω , for each point $\mathbf{x} \in \Omega$ it is possible to define a cross $\mathbf{c}(\mathbf{x})$. A 2D cross-field on Ω is defined as the set $\mathcal{C}_\Omega = \{\mathbf{c}(\mathbf{x}), \mathbf{x} \in \Omega\}$.

In order to highlight the use of 2D cross-fields for quad mesh generation, we will focus on a 2D domain Ω conformal to the unit square. Let's $\mathcal{U}$ be the planar unit square, $\mathcal{F}$ a conformal transformation and $\Omega = \mathcal{F}(\mathcal{U})$ (figure 4.3).

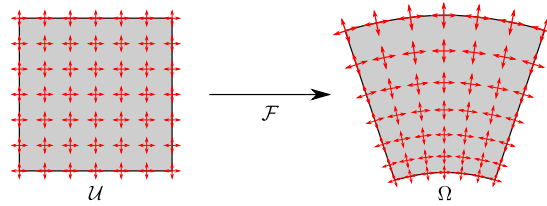


Figure 4.3: Conformal transformation between unitary square and physical domain with associated cross-fields

Let's define $\mathcal{C}_\mathcal{U}$ as a uniform cross-field of norm 1 aligned with the principal axis of $\mathcal{U}$ laying in the tangent space of $\mathcal{U}$, and $\tilde{\mathcal{C}}_\Omega$ the image of $\mathcal{C}_\mathcal{U}$ by $\mathcal{F}$. $\tilde{\mathcal{C}}_\Omega$

is a representation of the jacobian of $\mathcal{F}$ and in the following we will define $\mathcal{C}_\Omega$ as the normalized jacobian of $\mathcal{F}$.

It is possible to generate a quad layout of Ω by tracing integral lines of $\tilde{\mathcal{C}}_\Omega$, represented in figure 4.4, which are identical to integral lines of $\mathcal{C}_\Omega$.

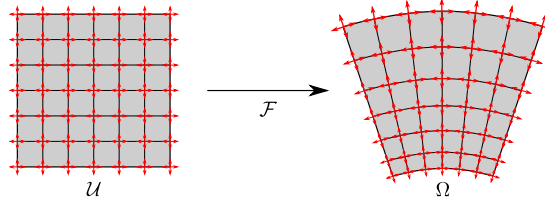


Figure 4.4: Quadrilateral multi-block decomposition of domain Ω obtained by propagating integral lines of $\mathcal{C}_\Omega$

Unfortunately, the conformal transformation $\mathcal{F}$ is usually unknown, and computing it is a challenging process. Instead, we will focus on computing the normalized jacobian $\mathcal{C}_\Omega$ of $\mathcal{F}$, knowing that on the boundary $\partial\Omega$ one direction of the normalized jacobian has to be aligned with the normal of $\partial\Omega$ (figure 4.5). Therefore, we are looking for a normalized 2D cross-field $\mathcal{C}_\Omega$ on Ω , as smooth as possible and having one direction aligned to $\partial\Omega$'s normal on the boundary.

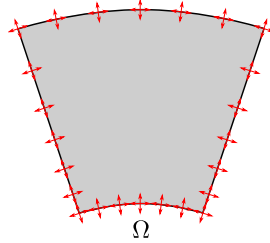


Figure 4.5: Reference problem for 2D cross-field computation

The quad layout is then obtained by tracing $\mathcal{C}_\Omega$'s integral lines. As we only computed a normalized cross-field, which correspond to a normalized jacobian, one could wonder if it exists a conformal transformation $\mathcal{F}$ such as its normalized jacobian is equal to $\mathcal{C}_\Omega$. To show that it is the case, we will show that for a given normalized cross-field $\mathcal{C}_\Omega$, it is easy to build a corresponding 2D cross-field with the same orientation and non uniform norms $\tilde{\mathcal{C}}_\Omega$ which is integrable.

Assuming we know a normalized 2D cross-field $\mathcal{C}_\Omega$, $\mathcal{C}_\Omega = \{\mathbf{c}(\mathbf{x}), \mathbf{x} \in \Omega\}$ with:

$$\begin{aligned} \mathbf{c} &= \{\mathbf{u}_k\}_{k \in [1,4]} \\ \mathbf{u}_k &= \begin{bmatrix} \cos(\theta(\mathbf{x}) + k\frac{\pi}{2}) \\ \sin(\theta(\mathbf{x}) + k\frac{\pi}{2}) \end{bmatrix}, \mathbf{x} \in \Omega \end{aligned} \quad (4.1)$$

which is completely defined by the function θ . We are looking for $\tilde{\mathcal{C}}_\Omega = \{\tilde{\mathbf{c}}(\mathbf{x}), \mathbf{x} \in \Omega\}$ with :

$$\begin{aligned} \tilde{\mathbf{c}} &= \{\tilde{\mathbf{u}}_k\}_{k \in [1,4]} \\ \tilde{\mathbf{u}}_k &= l(\mathbf{x}) \begin{bmatrix} \cos(\theta(\mathbf{x}) + k\frac{\pi}{2}) \\ \sin(\theta(\mathbf{x}) + k\frac{\pi}{2}) \end{bmatrix}, \mathbf{x} \in \Omega, \end{aligned} \quad (4.2)$$

where θ is known, such that $\tilde{\mathcal{C}}_\Omega$ is integrable.

A 2D cross-field $\tilde{\mathcal{C}}_\Omega$ is integrable if and only if for $\forall \mathbf{x} \in \Omega$, the Lie bracket of 2 orthogonal branches $(\tilde{\mathbf{u}}_1, \tilde{\mathbf{u}}_2)$ is equal to $\mathbf{0}$:

$$[\tilde{\mathbf{u}}_1, \tilde{\mathbf{u}}_2] = \nabla_{\tilde{\mathbf{u}}_2} \tilde{\mathbf{u}}_1 - \nabla_{\tilde{\mathbf{u}}_1} \tilde{\mathbf{u}}_2 = \nabla \tilde{\mathbf{u}}_1 \cdot \tilde{\mathbf{u}}_2 - \nabla \tilde{\mathbf{u}}_2 \cdot \tilde{\mathbf{u}}_1 = \mathbf{0}. \quad (4.3)$$

As detailed computation in Appendix 12.2 shows, for $l \neq 0$, $\tilde{\mathcal{C}}_\Omega$ is integrable if l verifies:

$$\nabla(\log(l)) = \begin{bmatrix} \theta_{,y} \\ -\theta_{,x} \end{bmatrix} \text{ on } \Omega. \quad (4.4)$$

As θ is known, it is easy to compute l , with a multiplicative constant, and build an integrable cross-field $\tilde{\mathcal{C}}_\Omega$. As crosses of $\mathcal{C}_\Omega$ and $\tilde{\mathcal{C}}_\Omega$ have the same orientation, integral lines of these 2 cross-fields are identical. This justifies the usage of integral lines of a normalized cross-field $\mathcal{C}_\Omega$ for generating a quad layout.

Note that $H = \log(l)$ is defined in [22] as a Green's function that is proven, contrary to θ , to be continuous everywhere in the domain except at singular points where H blows up as $\log r$ which means that l tends linearly to 0 at the vicinity of singularities.

The partitioning of a domain Ω will be obtained in 2 steps. First, a normalized 2D cross-field $\mathcal{C}_\Omega$ is generated on Ω , then integral lines of $\mathcal{C}_\Omega$ are computed to obtain the quad layout.

4 | Generating cross-field based on Ginzburg-Landau PDE

4.1 2D crosses representation

As presented in the previous section, it is possible to completely define a cross with an angle θ . However, symmetries of the cross lead to the fact that θ and

$\theta + k\frac{\pi}{2}$, $k \in \mathbb{Z}$ define the same cross, thus this representation is not unique. It is possible to uniquely define a cross using the following representation:

$$\mathbf{u} = (\cos 4\theta, \sin 4\theta), \quad \theta \in [0, \frac{\pi}{2}). \quad (4.5)$$

In this representation, $\mathbf{u}$ is invariant by a rotation of $\frac{\pi}{2}$, thus represents a “cross” with 4 orthogonal branches, as shown in figure 4.6 and figure 4.7.

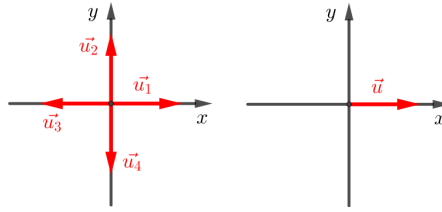


Figure 4.6: Reference cross (left) and its representation with 2D vector $\mathbf{u}$ (right)

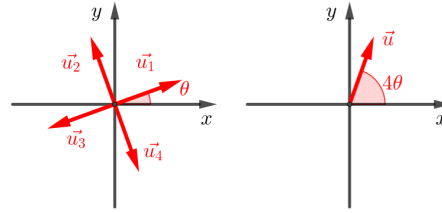


Figure 4.7: Rotation of reference cross by θ (left) and its representation with 2D vector $\mathbf{u}$ (right)

4.2 Computing 2D cross-field through partial differential equations

The idea of computing a cross-field $\mathbf{u} = (u_1, u_2)$ is to force crosses to be aligned with the boundaries of Ω and to propagate those crosses inside Ω using a PDE that eventually produces smooth cross-fields. In what follows, we propose a series of formulations that produce quite different results in practice.

The first approach is to choose $\mathbf{u} \in (H_1(\Omega))^2$ and simply minimize the Dirichlet energy:

$$E^D(\mathbf{u}) = \frac{1}{2} \int_{\Omega} |\nabla \mathbf{u}|^2 dv$$

with appropriate boundary conditions. Minimizing E^D is equivalent to solve $\nabla^2 \mathbf{u} = 0$. The main issue of that simple approach is that u_1 and u_2 are going to rapidly become equal to zero away from the boundaries. This is simply due to the mean value property of harmonic functions. When $u_1 = u_2 = 0$, the cross direction $\theta = \frac{1}{4} \text{atan2}(u_2, u_1)$ is undefined. Even though cross-fields issued from this naive approach are not suitable for obtaining the quad layout, they will be used as a starting point for other methods.

The main drawback of the first “naive” approach that has just been presented is that $\mathbf{u}$ actually leaves S^1 away from boundaries and is not a cross anymore. There are two possible options to force $\mathbf{u}$ to stay in S^1 : (i) choosing $\mathbf{u} \in S^1$ *explicitly* or (ii) choosing $\mathbf{u} \in (H_1(\Omega))^2$ and penalize $\mathbf{u}$ away from S^1 .

The first approach thus consists in choosing $\mathbf{u} \in S^1$ explicitly and writes $\mathbf{u}(\theta) = e^{i\theta}$. In this case, $\mathbf{u}$ is a complex number with u_1 and u_2 as its real and imaginary part and $u_1^2 + u_2^2 = 1$. In this case,

$$\min_{\mathbf{u} \in S^1} \int_{\Omega} |\nabla \mathbf{u}|^2 dv$$

is equivalent to

$$\min_{\theta \in H^1/Q} E^{\theta} = \int_{\Omega} |\nabla \theta|^2 dv.$$

Here, θ lives in the quotient space H^1/Q where Q is the group of symmetries of the square. Angle θ thus does not live in a linear space and minimizing E^{θ} is not strictly equivalent to solve $\nabla^2 \theta = 0$. Solutions to that problem have already been provided by using a mixed integer approach [73]. It is also possible to write an explicit smoother that averages θ locally. This method provides exploitable results when the θ 's are initiated by minimizing E^D using the first naive approach.

Another approach is to propose an alternative energy, namely the Ginzburg-Landau energy functional:

$$E^{GL}(\mathbf{u}) = E^D + E^P = \frac{1}{2} \int_{\Omega} |\nabla \mathbf{u}|^2 + \frac{1}{4\epsilon^2} \int_{\Omega} (|\mathbf{u}|^2 - 1)^2, \quad (4.6)$$

where the parameter ϵ has a dimension of length and in literature is known as coherence length [22]. Energy (4.6) contains two terms: the standard Dirichlet energy and a penalization. The only minimizer of the Ginzburg-Landau functional is solution of:

$$\nabla^2 \mathbf{u} + \frac{1}{\epsilon^2} \mathbf{u} (|\mathbf{u}|^2 - 1)^2 = 0.$$

The only vector field that satisfies both $\nabla^2 \mathbf{u} = 0$ and $|\mathbf{u}| = 1$ is the constant vector field. Consequently, whenever $\epsilon \neq 0$, the Ginzburg-Landau formulation

cannot force $\mathbf{u} \in S^1$ everywhere and unit vectors can only exist if they don't actually "turn", *i.e.*, if θ is constant.

A mixed approach consists in choosing $\mathbf{u} \in H_1(\Omega, S^1)$, *i.e.*, choosing $\mathbf{u} \in S^1$ while keeping both its components. In this context, it can be shown [22] that the only minimizer of the Dirichlet energy E^D is solution of

$$\nabla^2 \mathbf{u} + \mathbf{u} |\nabla \mathbf{u}|^2 = 0.$$

This formulation is strictly equivalent to $\nabla^2 \theta = 0$. Yet, it requires to choose $\mathbf{u} \in S^1$ *a priori* which is of course not easy. One can penalize $\mathbf{u}$ away from S^1 by using a penalization like in the Ginzburg-Landau case and solve

$$\nabla^2 \mathbf{u} + \mathbf{u} |\nabla \mathbf{u}|^2 + \frac{1}{\epsilon^2} \mathbf{u} (|\mathbf{u}|^2 - 1)^2 = 0.$$

The smoothness term of this energy functional minimizes the gradient of the cross-field and a penalty term makes its norm close to unity.

Further on, 2D cross-fields will be computed by minimizing energy functional defined by the equation 4.6. Crouzeix-Raviart finite elements are used for the interpolation and a Newton-Raphson scheme for solving the nonlinear problem (computational details in [44]), obtained result is shown in figure 4.8.

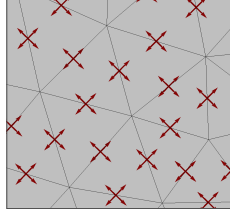


Figure 4.8: Generated crosses on triangulated surface Ω

4.3 Cross-field topology

For the orientable surface Ω with genus g and b as the number of connected components of $\partial\Omega$, the Euler characteristic χ of Ω is an integer:

$$\chi = 2 - 2g - b. \quad (4.7)$$

Considering a mesh on Ω with n nodes, n_e edges and n_f facets, the Euler formula states:

$$\chi = n - n_e + n_f. \quad (4.8)$$

According to equation (4.7) and (4.8) the mesh on the surface with Euler characteristic $\chi \neq 0$ will have irregular vertices. As shown in [44], these irregular vertices of the mesh are corresponding to singular points obtained by the cross-field. Moreover, their number and type depend on Euler characteristic χ . The type of critical point x_i is defined by its index $index(x_i)$ and can be found directly by computing:

$$index(x_i) = \frac{1}{2\pi} \oint_{C_i} d\theta \quad (4.9)$$

where θ is an angular reference and C_i is a closed curve on the surface Ω enclosing only one singularity: x_i . For a given quad mesh, a vertex x_i with valance v_i , where v_i represents the number of facets in the mesh adjacent to the x_i , the integral (4.9) is evaluated as:

$$index(x_i) = \frac{4 - v_i}{4}.$$

Therefore, vertices with index $0, \frac{1}{4}, -\frac{1}{4}$ are respectively adjacent to 4, 3 and 5 quadrangular elements. Dependency of number and type of singular points on Ω with Euler characteristic χ is given by Poincaré-Hopf theorem:

$$\sum_i index(x_i) = \chi(\Omega). \quad (4.10)$$

According to [22], the result of the minimization of the Ginzburg-Landau energy (4.6) in $2D$ supports coexistence of $index(x_i) = \pm \frac{1}{4}$, as shown in figure 4.9.

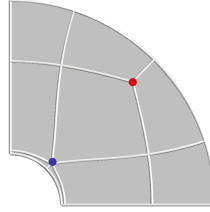


Figure 4.9: Coexistence of positive singularity (red) and negative singularity (blue) in $2D$

In a special case, following [107], the result of (4.6) for $\mathbf{u} \in S^1/C_N$, corresponds to the elliptic Fekete points on a sphere, illustration in figure 4.10 and more details on this subject in Chapter 5.

5 | Tracing separatrices

As already reported [19, 59, 63, 98] the tracing of separatrices of computed cross-field accomplishes the goal of domain partitioning. This section describes

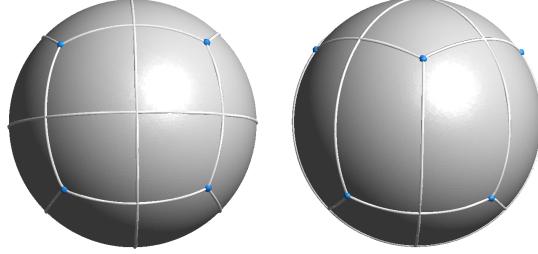


Figure 4.10: Singular points forming an anticube on the sphere

procedures for generating the separatrices on arbitrary 2D domain Ω . Our algorithm is divided into three stages: (i) the initiation of separatrices on a locally small neighborhood containing one singular point, (ii) the propagation of separatrices on the whole domain Ω and (iii) a post processing stage that allows to obtain the minimal number of separatrices.

5.1 Initialization of separatrices inside the critical elements

A triangular element $C_i, i \in [1; n]$ of the mesh is considered critical if a singular point is located at its vertex, edge or area, [19, 107]. For the sake of consistency with the numerical scheme used (*i.e.* Crouzeix-Raviart finite elements), singular points $S_i, i \in [1; n]$ will be located on the middle of the edges where cross-field vanishes. By traversing all edges of the mesh and finding the locations with the smallest $\|\mathbf{u}\|$, the critical elements are marked and singular points extracted.

For the purpose of separatrices propagation on each critical element C_i , we iterated over each edge of C_i to find points where the cross-field is aligned with a singularity S_i , *i.e.*, fulfilling 4.11 or 4.12, an illustration in figure 4.11. Finding values of a cross-field in these points is done by linear interpolation, similarly as in [19].

$$\theta_{P_i} = \theta_{P_i S_i} \pm \alpha \quad (4.11)$$

$$\theta_{Q_i} = \theta_{Q_i S_i} \pm \alpha \quad (4.12)$$

with α representing the tolerance criteria.

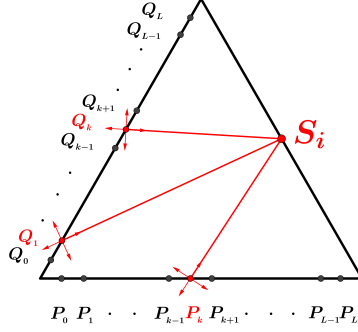


Figure 4.11: Obtaining separatrices on C_i

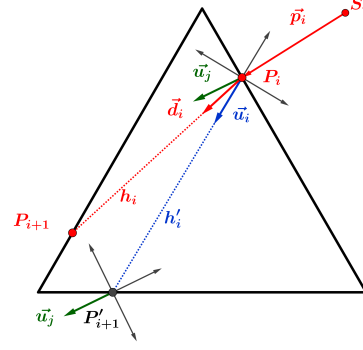


Figure 4.12: Separatrices propagation method

5.2 Propagation of separatrices on the whole domain Ω

In order to obtain a decomposition of the domain Ω , separatrices are propagated through a finite number of elements of triangulation $T = T_i - \{C_1, \dots, C_n\}$ following the adequate direction of the cross-field until stopping criteria is fulfilled. We used the propagation scheme described in [19] relying on Heun's (a variation of Runge-Kutta 2) numerical scheme:

$$P'_{i+1} = P_i + h'_i \cdot \mathbf{u}_i(P_i) \quad (4.13)$$

$$\mathbf{d}_i = \frac{\mathbf{u}_i(P_i) + \mathbf{u}_j(P'_{i+1})}{2} \quad (4.14)$$

$$P_{i+1} = P_i + h_i \cdot \mathbf{d}_i \quad (4.15)$$

where P_i is no-singular point on critical element C_i ; $\mathbf{u}_i$ and $\mathbf{u}_j$ are cross-field directions; h'_i and h_i represent the mesh size dependent step and P_{i+1} is computed point. More detailed, as shown in figure 4.12, the algorithm aims to compute the point P_{i+1} , where S_i represents a critical point, and P_i is derived based on information about the direction u_i which is the closest one to the input direction $\mathbf{p}_i = \overrightarrow{S_i P_i}$. Further on, the information about the direction $\mathbf{u}_j$, at the cross at point P'_{i+1} is used to obtain direction $\mathbf{d}_i$ generating point P_{i+1} and allowing further propagation in the same manner.

5.3 Stopping criteria for separatrices propagation

Separatrices generated in the manner described above are traced until they reached critical patch E_{pi} (figure 4.13) or the boundary $\partial\Omega$. For the computational purposes we defined a critical patch E_{pi} around each singular point S_i , which represents a set of triangles with a locally small distance r from S_i .

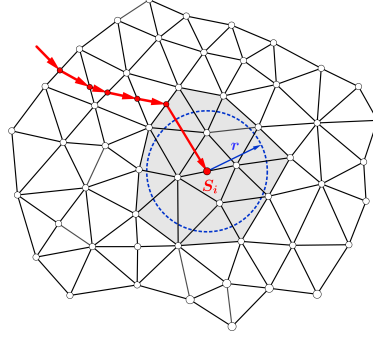


Figure 4.13: Critical patch

5.4 Cleaning the redundant separatrices

Presented method for separatrices propagation is not generating a minimal number of separatrices needed for multi-block decomposition of Ω . This result is a direct consequence of allowing the propagation of the same separatrix from two different singular points (as shown in figure 4.14).

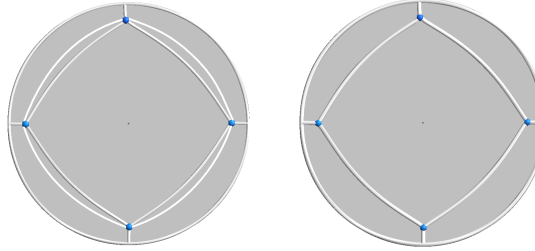


Figure 4.14: Quad layout of a circle: separatrices before (left) and after cleaning (right)

In order to obtain the minimal number of separatrices, we developed the procedure, detailed in algorithm 1, to determine and discard redundant ones. Two or more separatrices are defined as redundant if they have: identical beginnings, identical endings and intersect the same set of separatrices. An example for two redundant separatrices is given in figures 4.15-4.16. In red are represented two singular points, in black generated separatrices and in blue and green two separatrices meeting the criterion for defining redundant separatrices exposed previously.

Figure 4.15 is a case where the set of intersected separatrices is empty. The patch generated by redundant separatrices (in white) is made of two edges and can be removed by deleting one of the redundant separatrices without modifying the type of adjacent blocks (in grey).

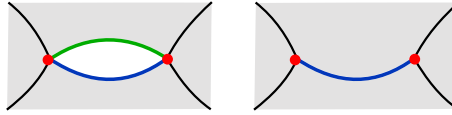


Figure 4.15: An empty set of separatrices' intersection (before and after)

Figure 4.16 is a case where the set of intersected separatrices is not empty. Redundant separatrices create the quadrangular blocks and two triangular blocks (in white). Deleting one of the redundant separatrices removes all blocks in white (including triangular blocks) without modifying the type of adjacent blocks (in grey).

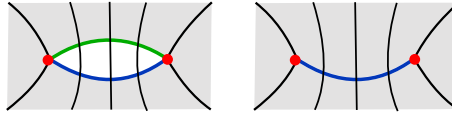


Figure 4.16: Not an empty set of separatrices' intersection (before and after)

Therefore, for each group of redundant separatrices a random one is chosen to be kept and all the other (redundant) separatrices are removed.

Algorithm 1 Obtain minimal number of separatrices

```

while There is a non-traversed separatrix  $i$  do
    Determine beginning and ending of separatrix  $i$ 
    Associate these data to separatrix  $i$  attributes
end while
Define groups by gathering separatrices with the same beginning and the
same ending
while There is a non-traversed separatrix  $i$  do
    Determine which separatrices from other groups separatrix  $i$  is intersect-
    ing
end while
Find redundant separatrices
Discard copies
    
```

The reason for taking into account separatrices intersections is demonstrated in figures 4.17-4.18: separatrices number 1 and 2 have the same beginnings and endings (belong to the same group), but they are intersecting different sets of separatrices (as for separatrices 3 and 4).

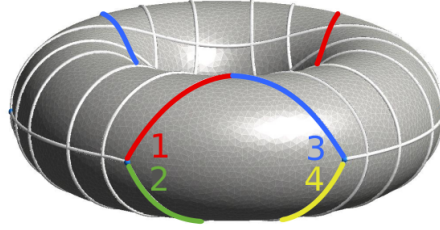


Figure 4.17: Quad layout of a torus

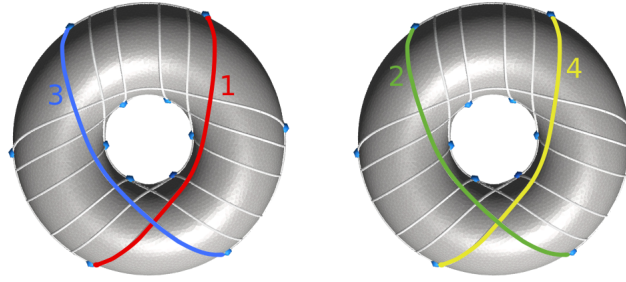


Figure 4.18: Overview of intersecting separatrices

6 | Examples of multi-block decomposition

In this section we present the results obtained by applying Ginzburg-Landau PDE for the purpose of generating a quad layout of the given domain Ω . By the definition of separatrices, each partition is smooth inside which allows meshing with quad elements. The quality of the mesh generated in this manner is expected to be high, due to the proven torsion free properties of a generated cross-field.

In the following, we demonstrate the dependency of generated quad layout on:

- geometrical properties of the domain, figure 4.19

- the type of boundary conditions imposed, figure 4.20
- the value of coherence length parameter ϵ , figure 4.21.

Depending on requirements, we can add (remove) a boundary layer (as shown in figures 4.20) by imposing weak or strong boundary conditions. Using different values for global coherence length ϵ , we can generate different quad layouts of the same domain. As shown in figure 4.21, both of them are valid.

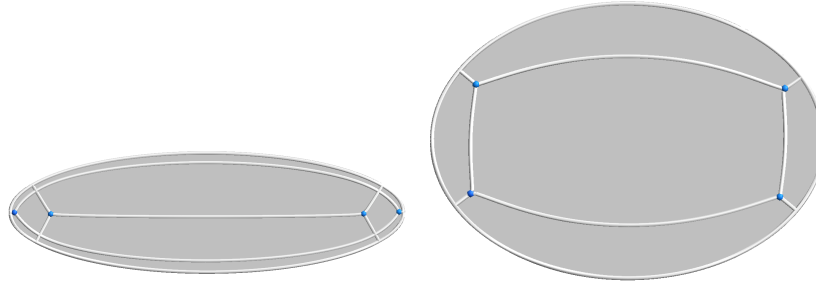


Figure 4.19: Quad layout dependence on geometrical characteristics of the domain. Eccentricity of the ellipse on the left is $e = 0.71$ and on the right is $e = 0.95$.

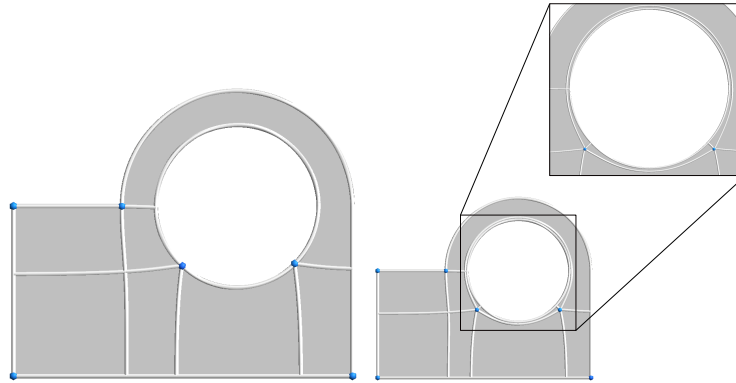
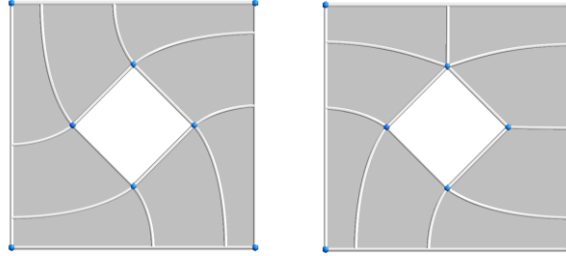
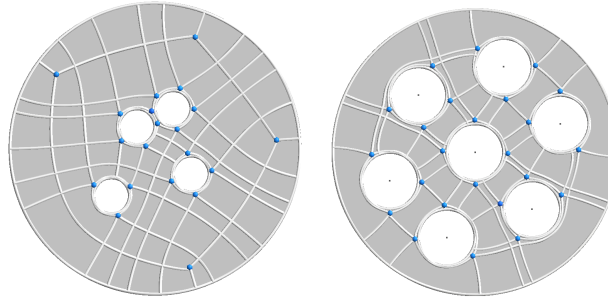


Figure 4.20: Quad layout with imposed weak boundary conditions on the left and with imposed strong boundary conditions on the right

We have chosen a few examples (figures 4.22-4.24) to demonstrate abilities of our algorithm for creating the quad layout of topologically and/or geometrically challenging domains.

Figure 4.21: Coherence length $\epsilon = 0.001$ (left) and $\epsilon = 0.01$ (right)Figure 4.22: Euler characteristic $\chi = -3$ on the left and $\chi = -6$ on the right

Following [44], generating cross-field using Ginzburg-Landau functional has many desirable properties for meshing purposes. Recent reports on cross-field generation [59, 63], pointed out the existence of the limit cycle, defined as one or more separatrices failure to converge to a singular point or a boundary, which prevents generating the quad layout. The decompositions we obtained, on reported domains by [59, 63], are shown in figures 4.25 and 4.26.

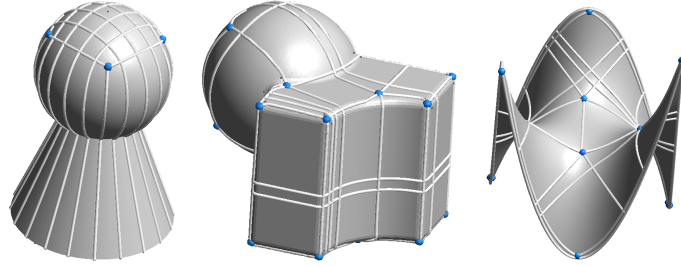


Figure 4.23: Quad layout of manifolds

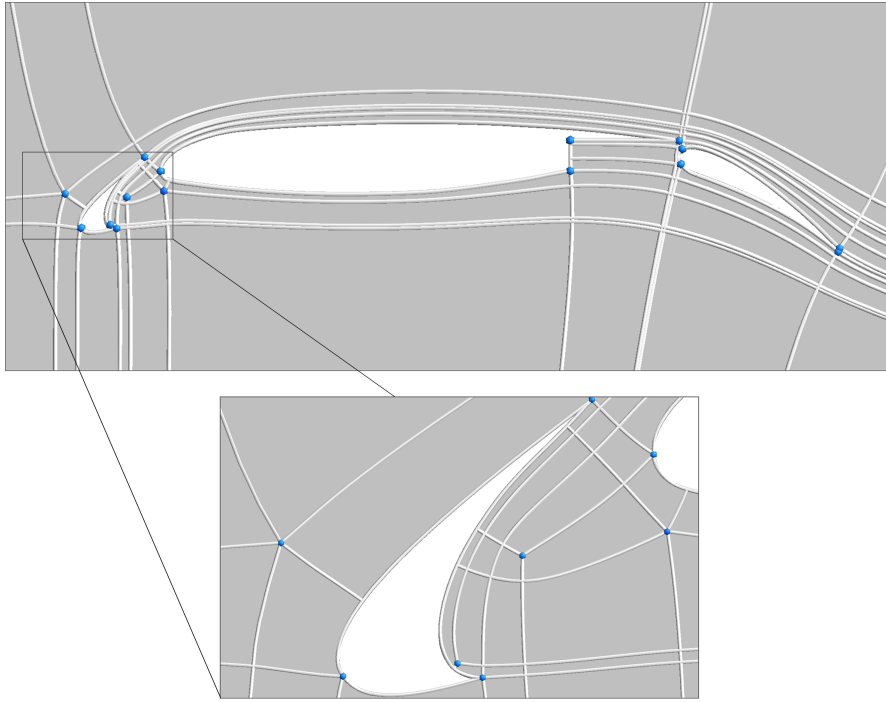


Figure 4.24: Quad layout of three component wing

7 | Planar graph embedding of the multi-block decomposition

For the purpose of the efficient and practical representation of complex domains Ω , according to [28], we will generate the planar graph embedding based on its

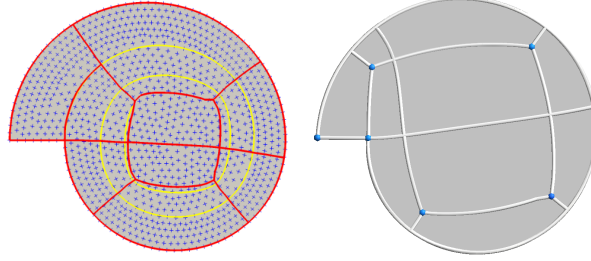


Figure 4.25: Multi-block decomposition by [63] (left) and by our approach (right)

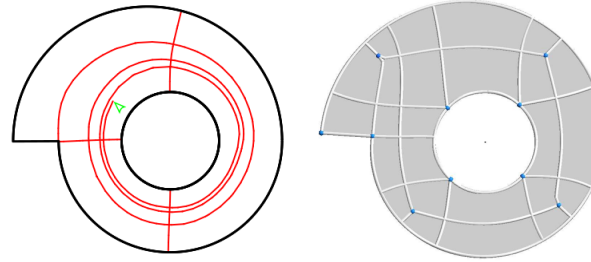


Figure 4.26: Multi-block decomposition by [59] (left) and by our approach (right)

multi-block decomposition (figure 4.27). To do so, we will use extraordinary vertices T_i , $i \in \{1, \dots, n\}$, defined as singular points, corners and intersections between separatrices and separatrices with $\partial\Omega$. These vertices are further on used for creating the graph $\Gamma = \{V, E\}$ defined by vertices $V = \{T_1, T_2, \dots, T_n\}$ and corresponding edges $E = \{\{T_1, T_2\}, \{T_2, T_3\}, \dots, \{T_j, T_n\}\}$. For the purpose of meeting well-defined connectivity (figure 4.28) data information of orientation is associated with each extraordinary vertex T_i : T_m, T_n, T_p , *e.g.*, listing of neighbors in counter-clockwise direction, wherefrom graph edges E are derived. Information on orientation and data structure ensured that each edge will be traversed only once and all quads $Q_i = \{T_i, T_j, T_k, T_l\}$ will be extracted. The used algorithm 2, showed below, is the adaptation of work described by [76].

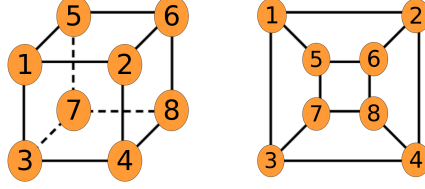


Figure 4.27: Illustration of a multi-block decomposition (left) and its planar graph embedding (right)

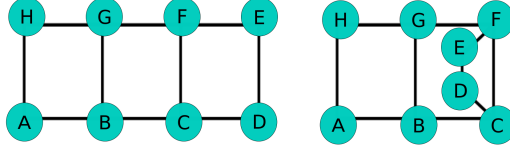


Figure 4.28: Importance of orientation data: connected (left) and disconnected graph (right)

Algorithm 2 Generate the graph

Compute extraordinary vertices T_i
Associate orientation data to each T_i
while There is a non-traversed oriented edge $\{T_i T_j\}$ **do**
 Take the first edge $T_i T_j$
 Take T_k - the counter-clock oriented neighbor of T_j which is before T_i
 Take T_t - the counter-clock oriented neighbor of T_k which is before T_j
 Define the quad $Q_l = \{T_i, T_j, T_k, T_t\}$
end while

The obtained patches $Q = Q_1 \cup Q_2 \cup \dots \cup Q_n$, by the definition of separatrices and singular points, have a smooth cross-field inside, allowing further on parameterization/remeshing.

8 | Workflow for planar domains

In case of planar domain Ω , partitions defined by a planar graph embedding, are directly used for applying algebraic (1) and afterward (2) elliptic grid generation (figure 4.29). The result obtained is high quality block-structured quad mesh. Theoretical and computational details on these methods are given in the following sections.

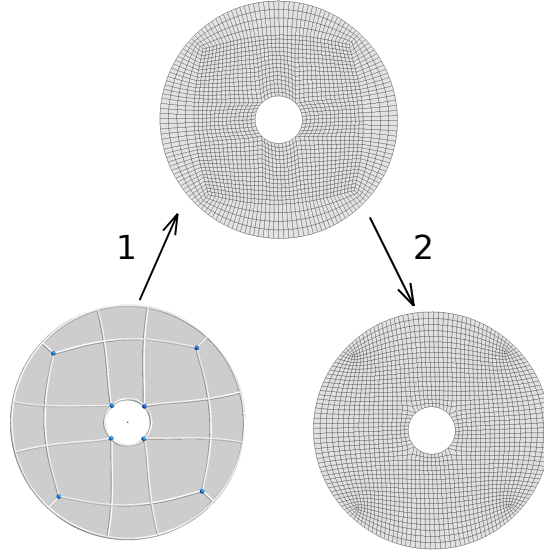
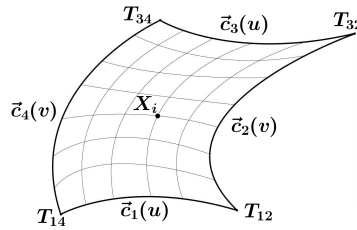


Figure 4.29: Illustration of the pipeline for the planar cases

8.1 Transfinite bilinear interpolation

For the aim of refinement of each block, transfinite interpolation (TFI) is used as a, according to [15, 19, 108], computationally efficient algebraic grid generation technique. The grid obtained with this procedure is structured, conforming the $\partial\Omega$ and has controlled grid spacing. For a given physical domain Q_i , defined with parameterized curves $\mathbf{c}_1(u)$, $\mathbf{c}_3(u)$, $\mathbf{c}_2(v)$ and $\mathbf{c}_4(v)$ (shown in figure 4.30), the position of point $\mathbf{X}_i(u, v)$ in the given domain is defined by the equation 4.16:

Figure 4.30: TFI grid points on physical domain Q_i

$$\begin{aligned}
 \mathbf{X}_i(u, v) &= (1 - v) \cdot \mathbf{c}_1(u) \\
 &+ v \cdot \mathbf{c}_3(u) + (1 - u) \cdot \mathbf{c}_2(v) + u \cdot \mathbf{c}_4(v) \\
 &- [(1 - u) \cdot (1 - v) \cdot \mathbf{T}_{12} \\
 &+ u \cdot v \cdot \mathbf{T}_{34} \\
 &+ u \cdot (1 - v) \cdot \mathbf{T}_{14} \\
 &+ (1 - u) \cdot v \cdot \mathbf{T}_{32}].
 \end{aligned} \tag{4.16}$$

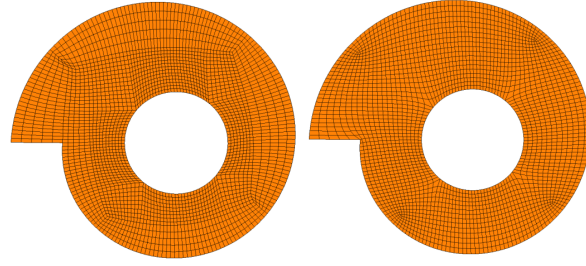


Figure 4.31: TFI mesh (left) and mesh smoothing (right)

As a result, a block-structured quad mesh $M_t = (V, E, Q)$, with vertices V , edges E and corresponding quadrilaterals Q is generated (figure 4.31 left). In order to obtain better orthogonality and improve overall quality of elements in the quad mesh obtained, bilinear TFI has been used as a step towards implementing a PDE based meshing technique (figure 4.31 right).

8.2 Grid smoothing

To ensure robustness and computational efficiency of a PDE based algorithm, which can be adopted for an unstructured mesh, our approach followed the work described in [109]. This work represents the Winslow smoothing [110] on 2D unstructured mesh and its based on solving the second-order nonlinear elliptic partial differential equations:

$$\begin{cases} g_{22}x_{\xi\xi} - 2g_{12}x_{\xi\eta} + g_{11}x_{\eta\eta} = 0 \\ g_{22}y_{\xi\xi} - 2g_{12}y_{\xi\eta} + g_{11}y_{\eta\eta} = 0 \end{cases} \tag{4.17}$$

with g_{11} , g_{12} and g_{22} computed as:

$$\begin{cases} g_{11} = x_{\xi}x_{\xi} + y_{\xi}y_{\xi} \\ g_{12} = x_{\xi}x_{\eta} + y_{\xi}y_{\eta} \\ g_{22} = x_{\eta}x_{\eta} + y_{\eta}y_{\eta}. \end{cases} \tag{4.18}$$

For the sake of completeness, the algorithm and its implementation are in detail explained in the Appendix 12.1.

In order to determine the quality of a quad mesh, the measure of quadrilateral element quality $\eta(q)$ of the element q with angles α_i , is computed as in [111]:

$$\eta(q) = \max\left(1 - \frac{2}{\pi} \max\left(\left|\frac{\pi}{2} - \alpha_i\right|\right), 0\right). \quad (4.19)$$

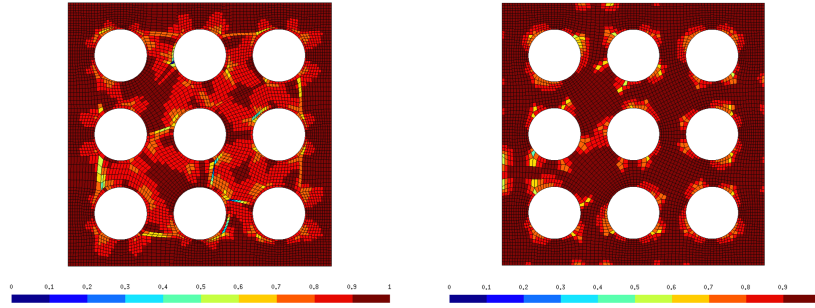


Figure 4.32: Quality of a mesh generated using TFI on the left and the quality of a mesh generated using Winslow smoother on the right

Improvements of the mesh quality using Winslow smoother are shown in a few examples in the table below. Notations used for values of the minimum and average quality of elements are respectively η_ω and $\bar{\eta}$, where $\eta_{0.9}$ represents the percentage of elements in the mesh whose quality is greater than 0.9 and h is the mesh size.

Figure	h	Algorithm	Mesh quality		
			η_ω	$\bar{\eta}$	$\eta_{0.9}$
Fig 4.32	0.01	TFI	0.01	0.90	0.59
		Winslow	0.47	0.94	0.80
Fig 4.22 left	0.05	TFI	0.00	0.94	0.82
		Winslow	0.52	0.93	0.79
Fig 4.22 right	0.01	TFI	0.14	0.92	0.73
		Winslow	0.48	0.93	0.80

Table 4.1: Comparison of TFI and Winslow mesh quality

9 | Workflow for 2D manifolds

We will now suppose that the domain Ω we are interested in is a 2D manifold. Workflow presented previously for planar domains has to be adapted for such domains. Indeed, the quad layout can be obtained in the same manner, but the algebraic grid generation and the elliptic grid smoothing will generate points not belonging to Ω in the general case. To be able to follow the grid generation procedure, first a parameterization of the domain Ω is needed. Figure 4.33 points out the main steps of the workflow.

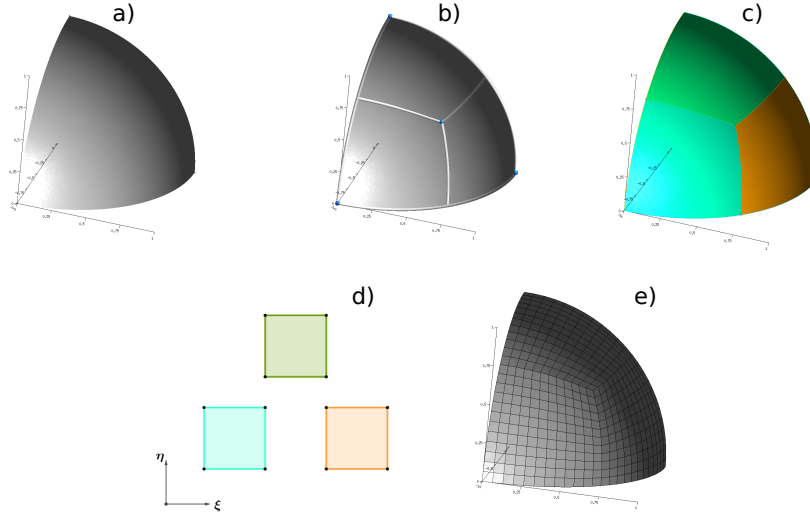


Figure 4.33: Illustration of the pipeline for the manifold cases a-e)

9.1 Parametrization of the domain

There are many ways to parameterize arbitrary 2D manifolds Ω [112, 113, 114] and they usually require to split Ω in a finite number n of subdomains $\Omega_i, i \in [1, n]$. Then each Ω_i is parametrized independently from each other. Such methods could be used to further generate grids only if boundaries of these subdomains correspond to the edges of the quad layout. The choice made here is to define each quadrilateral of the quad layout as a subdomain Ω_i and parameterize it independently following the method proposed in [114] (figure 4.34). The parameterization relies on mean value coordinates [115] which guarantees a one-to-one parameterization of each subdomain. For most of the obtained quad layouts, each partition can be parameterized with only

one atlas, but it can happen that it is necessary to split it in subdomains in order to get a proper parameterization. These kinds of cases are handled by the methodology proposed in [114].

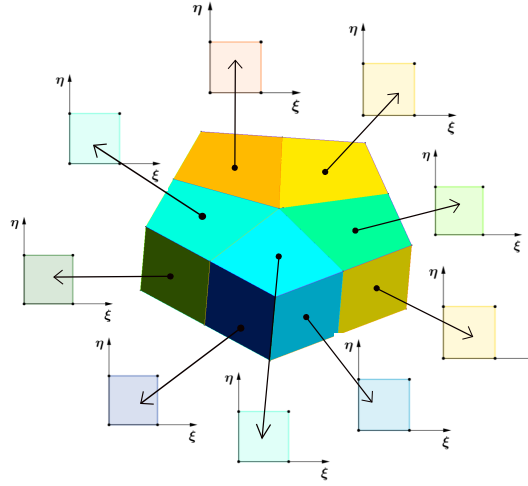


Figure 4.34: Illustration of the proposed parameterization

Once each subdomain Ω_i is parameterized, it is possible to generate a first grid on each Ω_i with an algebraic method and then use an elliptic smoother in order to obtain a high quality block-structured quad mesh.

It is important to note that, due to the choice of independent parameterization of each partition, it is not possible anymore as in the planar case, to optimize the position of separatrices and extraordinary vertices with the elliptic smoother.

9.2 Modification of original workflow

The workflow for 2D manifold is obtained by modifying the workflow for planar cases through adding a parameterization step, as presented in figure 4.33. First, the quad layout is generated and then each partition is parameterized independently. On each one of them, an initial grid is created in the parametric space, smoothed with Winslow smoother [109] and then mapped back to Ω in the physical space.

10 | Discussion

Due to the behavior of the cross-field at the vicinity of singular points, as well as our algorithm for creating critical patches, a triangular block can appear (figures 4.35 - 4.36). This issue is resolved by further propagation (figure 4.35), or, in cases when neither one of separatrices from the triangular block can not be further propagated, removal of the separatrix (figure 4.36).

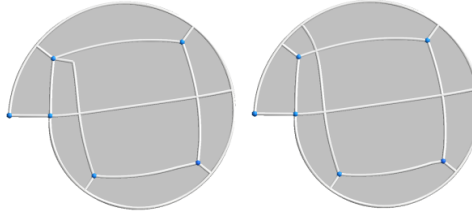


Figure 4.35: Before and after applying the algorithm for triangular block cleaning (repropagating)

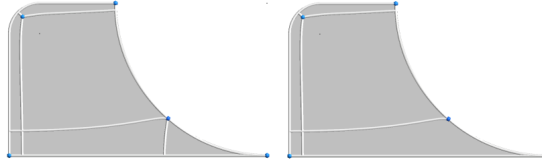


Figure 4.36: Before and after applying the algorithm for triangular block cleaning (removing)

When it comes to the computational cost of our method, it is subordinated to the number of elements of the triangulation. For most of the geometries, time for obtaining multi-block decomposition varied from less than one up to a few seconds with the same addition for parameterization/remeshing step. Concerning computational time for cross-field generation, the current numerical scheme to solve the Ginzburg-Landau PDE described in [44] is not competitive with existing cross-fields generators. Addressing this issue is presented in Chapter 6.

The future directions of the work include: optimization of performance time (detailed in Chapter 6), reported problem of limit cycles existence (Chapter 6, 7 and 8) and meshing with varying elements' sizes, as shown in figure 4.37 (detailed in Chapter 6, 7 and 8).

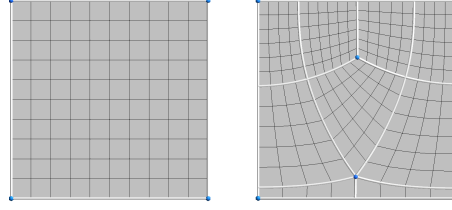


Figure 4.37: Uniform (left) and non-uniform quad mesh (right)

11 | Conclusion

This paper presents a contribution in developing an algorithm for generating a quad layout and block-structured quad mesh of manifolds. The specific interest in using cross-fields for these purposes is pointed out from a mathematical point of view. Choosing Ginzburg-Landau PDE is advocated by fair distribution of singular points - a crucial asset for a parameterization [81, 116]. To the best of our knowledge, the algorithms to obtain the minimal number of separatrices and the proof of local integrability of 2D cross-fields have not been exposed before. Last but not least, we demonstrated how to use our method to create block-structured quad meshes in an automatic manner and made it available in Gmsh [23], the open source finite element mesh generator.

12 | Appendix

12.1 Winslow smoother

Using the finite difference discretization, equations 4.17 can be written, for each node n of the mesh, as:

$$D_n(\mathbf{x}) = G_{22}D_{\xi\xi}(\mathbf{x}_n) - 2G_{12}D_{\xi\eta}(\mathbf{x}_n) + G_{11}D_{\eta\eta}(\mathbf{x}_n) = 0, \quad (4.20)$$

where G_{11} , G_{12} and G_{22} are:

$$\begin{cases} G_{11} = D_\xi(\mathbf{x}_n) \cdot D_\xi(\mathbf{x}_n) \\ G_{12} = D_\xi(\mathbf{x}_n) \cdot D_\eta(\mathbf{x}_n) \\ G_{22} = D_\eta(\mathbf{x}_n) \cdot D_\eta(\mathbf{x}_n), \end{cases} \quad (4.21)$$

with values D_ξ , D_η , $D_{\xi\xi}$, $D_{\xi\eta}$ and $D_{\eta\eta}$ depending on the valance v_n of the node n .

By defining logical space with: $\xi = \cos \theta_m$, $\eta = \sin \theta_m$ at each node, where

$$\theta_m = \frac{2\pi \cdot m}{v_n},$$

the following equations (4.22 - 4.23) are derived:
Approximations for $v_n = 4$:

$$\left\{ \begin{array}{l} D_\xi(\mathbf{x}_n) = \frac{2}{v_n} \sum_{m=0}^{v_n-1} (\mathbf{x}_m - \mathbf{x}_n) \cos \theta_m \\ D_\eta(\mathbf{x}_n) = \frac{2}{v_n} \sum_{m=0}^{v_n-1} (\mathbf{x}_m - \mathbf{x}_n) \sin \theta_m \\ D_{\xi\xi}(\mathbf{x}_n) = \frac{4}{v_n} \sum_{m=0}^{v_n-1} (\mathbf{x}_m - \mathbf{x}_n) \cos^2 \theta_m \\ D_{\xi\eta}(\mathbf{x}_n) = \frac{2}{v_n} \sum_{m=0}^{v_n-1} (\widehat{\mathbf{x}}_m - \mathbf{x}_n) \cos \widehat{\theta}_m \sin \widehat{\theta}_m \\ D_{\eta\eta}(\mathbf{x}_n) = \frac{4}{v_n} \sum_{m=0}^{v_n-1} (\mathbf{x}_m - \mathbf{x}_n) \sin^2 \theta_m \end{array} \right. \quad (4.22)$$

where:

$$\widehat{\theta}_m = \frac{2\pi \cdot (m + \frac{1}{2})}{v_n}$$

and $\widehat{\mathbf{x}}_m$ are associated with the diagonal nodes, as shown in figure 4.38.

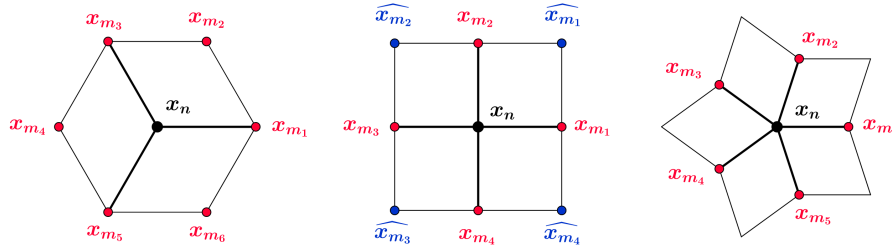


Figure 4.38: Dependency on v_n of $\mathbf{x}_n$ used for approximation of 3, 4 and 5 valent nodes (left to right)

Considering 3-valent nodes, equations demonstrated below are considering 6 neighbouring nodes (figure 4.38) and therefore the valent degree $v_n = 3$ rises up to $v_n = 6$, more detailed in [109].

Approximations for $v_n = 3$ and $v_n \geq 5$:

$$\left\{ \begin{array}{l} D_\xi(\mathbf{x}_n) = \frac{2}{v_n} \sum_{m=0}^{v_n-1} (\mathbf{x}_m - \mathbf{x}_n) \cos \theta_m \\ D_\eta(\mathbf{x}_n) = \frac{2}{v_n} \sum_{m=0}^{v_n-1} (\mathbf{x}_m - \mathbf{x}_n) \sin \theta_m \\ D_{\xi\xi}(\mathbf{x}_n) = \frac{2}{v_n} \sum_{m=0}^{v_n-1} (\mathbf{x}_m - \mathbf{x}_n) (4 \cos^2 \theta_m - 1) \\ D_{\xi\eta}(\mathbf{x}_n) = \frac{8}{v_n} \sum_{m=0}^{v_n-1} (\mathbf{x}_m - \mathbf{x}_n) \cos \theta_m \sin \theta_m \\ D_{\eta\eta}(\mathbf{x}_n) = \frac{2}{v_n} \sum_{m=0}^{v_n-1} (\mathbf{x}_m - \mathbf{x}_n) (4 \sin^2 \theta_m - 1) \end{array} \right. \quad (4.23)$$

To solve the nonlinear problem $D_n(\mathbf{x}) = 0$, for each node n from M_t where boundary nodes are fixed, Picard iterations are performed. In order to do so, the following notations are defined: M_q^k as the current quad mesh, $\mathbf{x}^k$ the coordinates of its nodes, $\mathbf{x}^{k+1}$ the coordinates of nodes after one smoothing iteration and M_q^{k+1} the corresponding quad mesh. By computing the values G_{11}^k , G_{12}^k and G_{22}^k evaluated on every node of M_q^k , we can define:

$$D_n^k(\mathbf{x}) = G_{22}^k D_{\xi\xi}(\mathbf{x}_n^{k+1}) - 2G_{12}^k D_{\xi\eta}(\mathbf{x}_n^{k+1}) + G_{11}^k D_{\eta\eta}(\mathbf{x}_n^{k+1})$$

for all nodes n . This system of n nonlinear equations is put under the form $D^k \cdot \mathbf{x}^{k+1}$, where D^k depends only on $\mathbf{x}^k$. Finding $\mathbf{x}^{k+1}$ is done by solving $D^k \cdot \mathbf{x}^{k+1} = \mathbf{0}$. Performed computational steps are described in algorithm 3.

Algorithm 3 Solving $D_n(\mathbf{x}) = 0$ with Picard iterations

```

Define convergence criteria  $c$ 
 $k = 0$ 
 $M_0^k = M_t$ 
while  $\|D^k \cdot \mathbf{x}^{k+1}\|_\infty > c$  do
  Compute  $G_{11}^k, G_{12}^k, G_{22}^k$  for all nodes of  $M_q^k$ 
  Build  $D^k$ 
  Solve  $D^k \mathbf{x}^{k+1} = \mathbf{0}$ 
  Generate  $M_q^{k+1}$  from  $\mathbf{x}^{k+1}$ 
   $k = k + 1$ 
end while
Write new smoothed mesh  $M_w = M_q^k$ 

```

12.2 Local integrability of the cross-field

Starting from the definition of 2D cross-field $\tilde{C}_\Omega$ with the non uniform norms (equation 4.2) and replacing the corresponding values in equation (4.3), we obtain the result computed in (4.24), which shows that for $l \neq 0$, $\tilde{C}_\Omega$ is integrable if l verifies the condition (4.4).

$$\begin{aligned}
 [\tilde{\mathbf{u}}_1, \tilde{\mathbf{u}}_2] &= \begin{bmatrix} l_{,x} \cdot \cos \theta - l \cdot \sin \theta \cdot \theta_{,x} & l'_{,y} \cdot \cos \theta - l \cdot \sin \theta \cdot \theta_{,y} \\ l_{,x} \cdot \sin \theta + l \cdot \cos \theta \cdot \theta_{,x} & l'_{,y} \cdot \sin \theta + l \cdot \cos \theta \cdot \theta_{,y} \end{bmatrix} \cdot \begin{bmatrix} -l \cdot \sin \theta \\ l \cdot \cos \theta \end{bmatrix} \\
 &- \begin{bmatrix} -l_{,x} \cdot \sin \theta - l \cdot \cos \theta \cdot \theta_{,x} & -l'_{,y} \cdot \sin \theta - l \cdot \cos \theta \cdot \theta_{,y} \\ l_{,x} \cdot \cos \theta - l \cdot \sin \theta \cdot \theta_{,x} & l'_{,y} \cdot \cos \theta - l \cdot \sin \theta \cdot \theta_{,y} \end{bmatrix} \cdot \begin{bmatrix} l \cdot \cos \theta \\ l \cdot \sin \theta \end{bmatrix} \\
 &= \begin{bmatrix} -l \cdot l_{,x} \cdot \sin \theta \cdot \cos \theta + l^2 \cdot \sin^2 \theta \cdot \theta_{,x} + l \cdot l_{,y} \cdot \cos^2 \theta - l^2 \cdot \sin \theta \cdot \cos \theta \cdot \theta_{,y} \\ -l \cdot l_{,x} \cdot \sin^2 \theta - l^2 \cdot \sin \theta \cdot \cos \theta \cdot \theta_{,x} + l \cdot l_{,y} \cdot \sin \theta \cdot \cos \theta + l^2 \cdot \cos^2 \theta \cdot \theta_{,y} \end{bmatrix} \\
 &- \begin{bmatrix} -l \cdot l_{,x} \cdot \sin \theta \cdot \cos \theta - l^2 \cdot \cos^2 \theta \cdot \theta_{,x} - l \cdot l_{,y} \cdot \sin^2 \theta - l^2 \cdot \sin \theta \cdot \cos \theta \cdot \theta_{,y} \\ l \cdot l_{,x} \cdot \cos^2 \theta - l^2 \cdot \sin \theta \cdot \cos \theta \cdot \theta_{,x} + l \cdot l_{,y} \cdot \sin \theta \cdot \cos \theta - l^2 \cdot \sin^2 \theta \cdot \theta_{,y} \end{bmatrix} \\
 &= \begin{bmatrix} l^2 \cdot \theta_{,x} + l \cdot l_{,y} \\ l^2 \cdot \theta_{,y} - l \cdot l_{,x} \end{bmatrix} = l^2 \begin{bmatrix} \theta_{,x} + \frac{l_{,y}}{l} \\ \theta_{,y} - \frac{l_{,x}}{l} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
 \end{aligned} \tag{4.24}$$

CHAPTER



Elliptic Fekete points obtained by Ginzburg-Landau PDE

The computation of the cross-field on a 2-sphere with the Ginzburg-Landau model yields a surprising result. Namely, the singularities' locations do not form a cube as expected, but an anticube. The reason for such an unusual result lies in the asymptotic behavior of Ginzburg-Landau energy, and as it will be shown in this chapter¹, this result corresponds to the solution of Smale's 7th problem [117], *i.e.*, the determination of the minimum potential energy location of N points x_i , $i = 1, \dots, N$ on the 2-sphere.

¹Jezdimirović, J., Chemin, A., Beaufort, P. A., and Remacle, J. F. (2017) *Elliptic Fekete points obtained by Ginzburg-Landau PDE*. Proceedings of the 26th International Meshing Roundtable

Abstract

The Ginzburg-Landau model exhibits the existence of a logarithmic force between singularities: attractive in the case when their indices are of the opposite sign, otherwise repulsive. It will be shown that minimization of this specific energy on a unit 2-sphere locates the singularities on vertices of the two squares of side $\frac{1}{\sqrt{3}}$, which are tilted by $\frac{\pi}{4}$ degrees around their common axis, and by so forming an anticube, corresponding to the solution of Smale's 7th problem.

The specific case that is treated here is the Fekete problem where points repel each other with a force that depends on the logarithm of their respective distance. This chapter presents a method to compute Fekete points by solving a partial differential equation that is usually used for computing cross-fields. This approach is quite uncommon and quite competitive for N large.

Keywords: Smale's 7th problem, Fekete problem, minimal energy, PDEs

1 | Introduction

In a famous paper [117], Steve Smale details eighteen open mathematical problems for the 21st century. Smale gives there an update for the twenty-three open problems that David Hilbert presented at the 2nd international mathematical congress in 1900.

Smale's 7th problem is about distribution of points on the 2-sphere S^2 . Consider a set of distinct points $\mathcal{X}_N = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ that all belong to S^2 . We use notation $|\mathbf{x}_i - \mathbf{x}_j|$ for the geodesic distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. The energy of point set $\mathcal{X}_N$ is written as:

$$E(\mathcal{X}_N) = - \sum_{i \neq j} \log |\mathbf{x}_i - \mathbf{x}_j|. \quad (5.1)$$

The problem is to find $\mathcal{X}_N$ that minimizes $E(\mathcal{X}_N)$. The optimal points are called elliptic Fekete points. Another way of looking at the problem is to maximize the function:

$$W(\mathcal{X}_N) = (\exp E_1(\mathcal{X}_N))^{-1} = \left(\prod_{i \neq j} |\mathbf{x}_i - \mathbf{x}_j|^{-1} \right)^{-1} = \prod_{i \neq j} |\mathbf{x}_i - \mathbf{x}_j|. \quad (5.2)$$

There exists an abundant literature on that problem, both on the theoretical side and on the algorithmic side [117, 118, 119, 120]. In this paper, we show that it is possible to find Fekete points by solving a partial differential equation (PDE) on the sphere. The PDE is essentially the one that is used for computing direction fields in the context of quadrilateral mesh generation [44].

1.1 Ginzburg-Landau PDE

In this section, we aim at showing the profound link between Fekete points and the minimization of the following energy functional, namely Ginzburg-Landau's functional:

$$E_\epsilon(\mathbf{u}) = \frac{1}{2} \int_G |\nabla \mathbf{u}|^2 + \frac{1}{4\epsilon^2} \int_G (|\mathbf{u}|^2 - 1)^2. \quad (5.3)$$

Here, $\mathbf{u}$ represents an element of the quotient space S^1/C_N where S^1 is the circle group and C_N is the group of N -fold symmetry, *i.e.*, rotational symmetry of order N . Computing the first variation of equation (4.6) allows to express this minimization problem as the solution of the following PDE:

$$\nabla^2 \mathbf{u} + \frac{1}{\epsilon^2} (1 - |\mathbf{u}|^2) \mathbf{u} = 0. \quad (5.4)$$

According to Alberti-Ambrosio-Cabr  theorem for stable solutions of elliptic PDE's [121], there is a unique minimizer for the given energy $E_\epsilon(\mathbf{u})$ when $\epsilon \rightarrow 0$. Moreover, it is also possible to show that [22]:

$$\lim_{\epsilon \rightarrow 0} E_\epsilon = 2N\pi \log(1/\epsilon) + W(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{2N}) + O\left(\frac{1}{|\log \epsilon|}\right). \quad (5.5)$$

Here, the $2N$ points $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{2N}\}$ are singular points, *i.e.*, points where $\mathbf{u}$ vanishes. It is indeed possible to show that those points are Fekete points (note here that our approach is only able to compute Fekete points of even orders).

The first term of equation (5.5), namely $2N\pi \log(1/\epsilon)$ is called singular core energy: energy indeed blows up when $\epsilon \rightarrow 0$. This term being totally independent of the position of the critical points, it has no influence on their locations. The third term of equation (5.5) goes to zero when $\epsilon \rightarrow 0$, it has no influence as well. The only term of equation (5.5) that is finite when $\epsilon \rightarrow 0$ and that depends on the position of critical points is the second one, W , the renormalized energy:

$$W(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) = -\pi \sum_{i \neq j} \log |\mathbf{x}_i - \mathbf{x}_j|. \quad (5.6)$$

The position of the $2N$ singular points that result of the minimization of the Ginzburg-Landau energy (4.6) for $u \in S^1/C_N$ thus correspond to the Fekete points of order $2N$.

2 | Computational procedure

A finite element method (FEM) has been used to approximate solution of Ginzburg-Landau PDE. A conformal triangulation $\cup_i \Omega_i = \Omega$ over domain $\Omega =$

$S^2 \in \mathbb{R}^3$ is generated. In order to handle N -folds symmetry issues, a N -fold symmetry vector field $\mathbf{u} = (u_1, u_2)$:

$$(u_1, u_2) = (\cos N\alpha, \sin N\alpha), \quad \alpha \in (0, 2\pi), \quad (5.7)$$

is chosen as the unknown vector. Details on the finite element procedure that has been used can be found in [44]. The principal ingredients are:

- The use of non conforming Crouzeix-Raviart's triangular finite elements to discretize $\mathbf{u}$,
- A Newton-Raphson's scheme to solve the non linear system of equations,
- Choice of the coherence length ϵ of the order of magnitude of mesh size but way smaller than the distance between singular points.

2.1 Localization of singular points

According to the Hairy ball theorem of algebraic topology and following [81], a vector field V has a singularity at a certain point $\mathbf{x}_i$ if it vanishes or is not defined at this point. Therefore, singularities are detected as the zeros of piecewise linear interpolation of the representation cross-field $\mathbf{u}$ obtained at the triangle element edges of the mesh. In order to localize these singular points on an element size area, elements with the smallest vector norms are extracted. A triangular element, in this sense, is considered to be singular if contains a singular edge. By traversing all existing triangle elements of a mesh in the given domain Ω , regions with finite number of triangles are obtained. For the purpose of accurate localization of singular points, every region is refined and singular triangular elements with singular points, on the middle of their mutual edges, are extracted.

The result on a unit 2-sphere is shown in figure 5.1 and it consists of 8 singular points whose locations form an anticube. For special cases of 4, 6 and 12 singular points obtained via the same methodology, it can be shown that their positions actually form Platonic solids, *i.e.*, respectively tetrahedron, octahedron and icosahedron. In the latter cases, the obtained partitions are triangular, which leads to a very important remark: the cross-fields can be used in triangular meshing as well!

3 | Results

According to many authors [117, 120, 122], there is a variety of computational techniques for obtaining log-optimal configurations: equal area points method

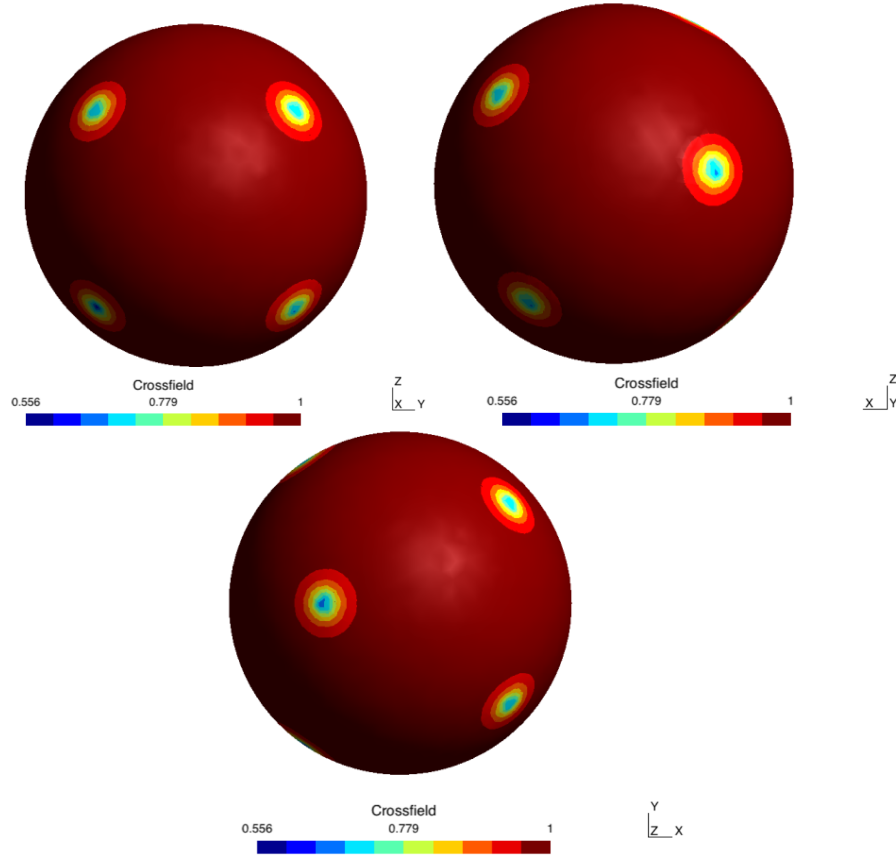


Figure 5.1: Fekete points on a unit 2-sphere, a picture from [2].

[123], spiral points algorithm [120], hexagonal tilings [122], force method approach [124], etc. Drawbacks of above-mentioned algorithms, as restriction for N large, are bypassed by recent contributions of Cluster Clonetroop, FinisTerra challenge and MareNostrum numerical experiments [124]. Both theoretical and computer-aided approaches have mainly relied on geometrical properties of the domain, classical optimization algorithms, combinatorial optimization methods and ODE integrators [124]. Yet, to the best of our knowledge, elliptic Fekete points have never been obtained by a solution of a PDE. Moreover, in comparison with up-to-date calculations for electrostatic potential energy minimum (EPEM) equation (5.2), as shown in figure 5.2, the use of Ginzburg-Landau energy (GLE) equation (4.6) is giving competitive results.

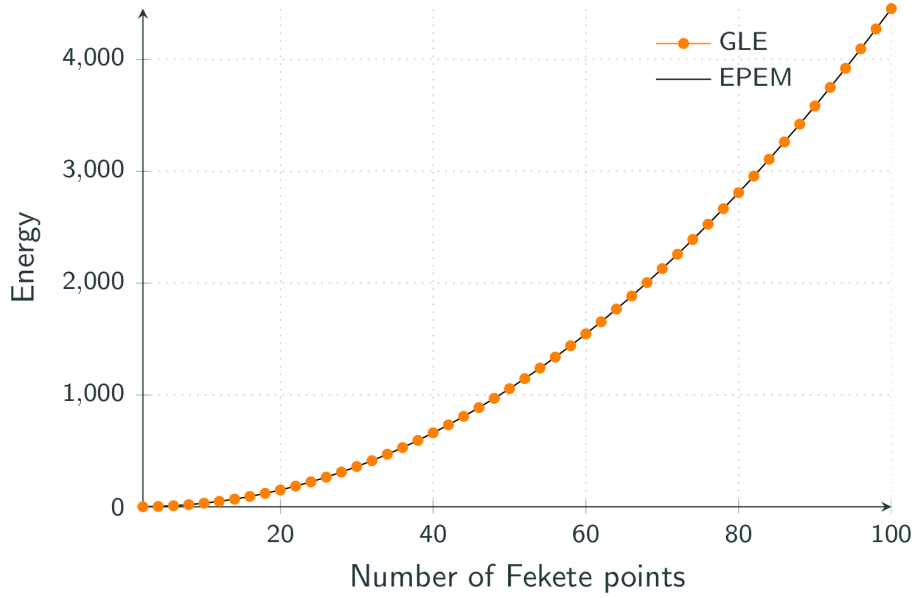


Figure 5.2: Preliminary results.

4 | Conclusion

This paper presents a contribution in developing a novel approach for generating elliptic Fekete configurations. According to the asymptotic behavior of Ginzburg-Landau functional, the number and localization of singular points obtained by the solution of the cross-field $\mathbf{u}$ correspond to Smale's 7th problem. The algorithm used the FEM for solving elliptic Ginzburg-Landau PDE in order to determine these points in an automatic way. As a result, log-optimal configurations, on the unit sphere $S^2 \in \mathbb{R}^3$, were obtained in a robust and efficient way. Depending on requirements, the future work would be addressed to finding reliable solutions for different dimensional cases, topologies and values of N in order to provide more general results on computational cost, convergence and stability of the proposed approach.

CHAPTER

6

Flexible quad layout and quad mesh generation: the planar case

This chapter¹ aims to answer the issues of the presented numerical scheme for quad meshing based on Ginzburg-Landau PDE in Chapter 4 for planar domains. More specifically, decreasing the computational time for cross-field generation, reported problems of limit cycles existence, and numerical inaccuracies around singularities are addressed. The MBO algorithm [63] is used to locate singularities and determine their valences, and the integrable cross-field with isotropic scaling (from the obtained singularity configuration) is generated with the computation of scalar field H , corresponding to the branches' scaling, from related Ginzburg-Landau literature [22].

¹Jezdimirović, J., Chemin, A., Reberol, M., Henrotte, F., and Remacle, J. F. (2021) *Quad layouts with high valence singularities for flexible quad meshing*. Proceedings of the 29th International Meshing Roundtable

Abstract

A novel algorithm that produces a quad layout based on imposed set of singularities is proposed. In this paper, we either use singularities that appear naturally, *e.g.*, by minimizing Ginzburg-Landau energy, or use as an input user-defined singularity pattern, possibly with high valence singularities that do not appear naturally in cross-field computations. The first contribution of the paper is the development of a formulation that allows computing a cross-field from a given set of singularities through the resolution of two linear PDEs. A specific mesh refinement is applied at the vicinity of singularities to accommodate the large gradients of cross directions that appear in the vicinity of singularities of high valence. The second contribution of the paper is a correction scheme that repairs limit cycles and/or non-quadrilateral patches. Finally, a high quality (elements edges being as close to orthogonal as possible) block-structured quad mesh is generated from the quad layout and per-partition parametrization.

Keywords: quad layout, quad meshing, high valence singularities

1 | Introduction

Quad meshing is a discipline that has been developed for different purposes by two distinct communities. Computer graphics and engineering analysis communities have indeed produced extensive research in the subject for decades, leading to both unstructured and structured approaches of quad meshing. A recent survey [28] comprehensively discusses the issue of quad meshing from manual to fully automatic generation, along with the respective advantages and drawbacks of the different methods.

In a block-structured quad mesh, most of the vertices are regular. A vertex v is regular if it has the optimal number n_v of adjacent quads: $n_v = 2$ if the vertex is on the boundary $\partial\Omega$ of the domain Ω to be meshed, and $n_v = 4$ if it is an internal vertex. Our approach for block-structured quad meshing relies on the computation of an auxiliary object, called a cross-field, as a guide to orient the quadrangular elements. Cross-fields have isolated singularities that can be shown to correspond to the location of irregular vertices of the quad mesh [44]. Using this useful structure of cross-fields has become popular to generate block-structured quad meshes [19, 21, 59, 60, 63]. More specifically, generating block-structured quad meshes requires the construction of a *quad layout*. A quad layout is the partitioning of an object's surface into simple networks of conforming quadrilateral patches [39]. Figure 6.1 shows examples of quad layouts for a simple 2D domain.

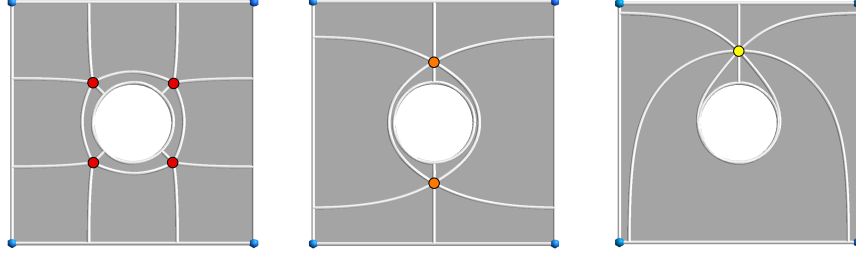


Figure 6.1: Three quad layouts of a simple domain defined by a square minus a disk. Colored points are the irregular corners of the layouts.

Two main approaches exist to extract a quad layout from the singular structure of a cross-field. The first approach consists of computing a seamless global parameterization of the domain where integer iso-values of the parameter fields form the sides, or the separatrices, of the quad layout [55, 64, 65]. This approach is implicit in the sense that it does not explicitly compute separatrices from the cross-field. The second approach is explicit. It is the one used in this paper. It consists of directly tracing the separatrices of the quad layout, starting at the singularities of the cross-field. A singularity of valence 5 corresponds to a vertex v with $n_v = 5$ adjacent quads, so that 5 separatrices can be started at this vertex v . Separatrices then follow integral lines of the cross-field, eventually ending on $\partial\Omega$ or on another singularity.

The main drawback of the explicit approach is related to its lack of geometrical robustness. Assuming that singularities are located at vertices of our mesh, the orientation of the cross-field is thus varying abruptly at the vicinity of a singular vertex v . That bad representation of the cross-field can have dramatic consequences. In many cases, the right number of separatrices cannot even be traced starting at v . Bad cross representation around singularities is also the cause of tangential crossings, *i.e.*, separatrices that should meet at a singular vertex but that miss the rendez-vous and continue further, eventually forming an infinite loop.

Mesh refinement around singularities would then seem to be a reasonable idea. Disappointingly, it is not. Cross-field smoothers have the bad habit to push singularities away from refined zones, because the energies used in cross-field models are unbounded (in the continuous case) at singularities, so that locating singularities in coarse region allows finding better minima (in the discrete case). Standard mesh adaptation thus fails here, because the solution does not converge in terms of energy. Figure 6.2 shows an example of that typical behavior. Figure 6.2 (left) shows a cross-field computed on a uniform mesh. Mesh refinement is then applied close to those singularities, and the

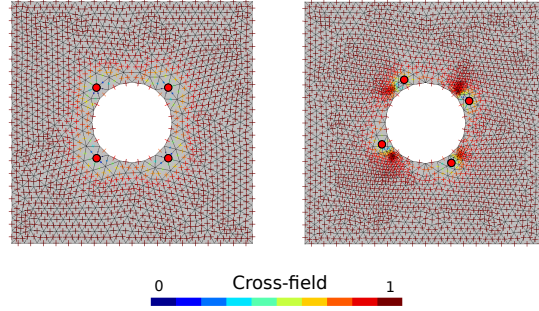


Figure 6.2: Displacement of singular points due to a cross-field mesh dependency. Cross-field computation follows the work from [44].

cross-field solver is then applied once again with the refined mesh. On figure 6.2 (right), one can observe that the singularities have moved with respect to their position after the first cross-field computation. The tendency to escape refined regions is so strong that the solution is actually breaking the symmetry of the problem.

In order to avoid this issue, our cross-field computation is implemented as a two-step process. A first cross-field is computed in a standard fashion by means of minimizing a nonlinear energy (the Ginzburg-Landau energy functional in our case). The *singularity pattern* of the cross-field is extracted from this first computation and then injected as a constraint for a second cross-field computation on an adapted mesh. The formulation for computing this second cross-field is different. Here, the singularity pattern is taken as an input and a cross-field is computed by means of solving two linear systems. Singularity positions *are thus prescribed* so mesh adaptation can be performed. The local structure of a cross-field at a singularity is essentially radial: the mesh that is used in this second step thus involves specific *bicycle spokes patterns* at singularities (see section 4). With that *ad hoc* refinement, the singular structure of the cross-field is perfectly captured. Another advantage of our approach is the possibility of moving singularities, adding or removing some, while still respecting topological constraints. In this work, we use that advantage for correcting or adapting the quad layout when necessary, *e.g.*, to fix non-quadrilateral partitions or to avoid limit cycles.

The full pipeline of our block-structured quad mesher consists of four steps and is described in figure 6.3.

Step 1: compute a first cross-field using standard non linear minimization procedure that allows to find a singularity pattern, *i.e.*, singularity positions and indices.

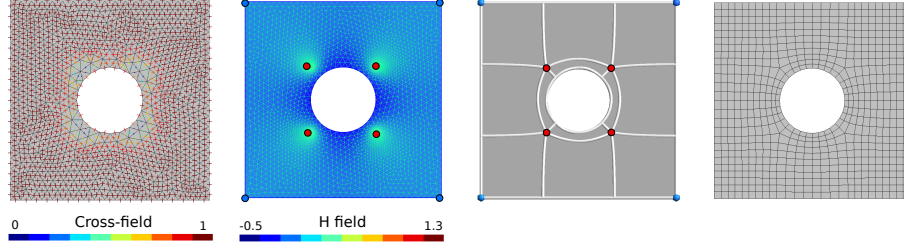


Figure 6.3: Illustration of 4-steps pipeline

Step 2: compute a second cross-field with the prescribed singularity pattern of Step 1 on a refined mesh.

Step 3: compute a neat quad layout decomposition without infinite loops or t-junctions. Quad layout partitioning is performed on the accurate cross-field of Step 2 by applying the separatrice tracing scheme presented in [60].

Step 4: generate a full quadrilateral mesh.

Note that Step 1 can be skipped if a singularity pattern is provided by the user. Figure 6.4 shows this reduced 3-step process with a very unique singularity of very high valence 8!

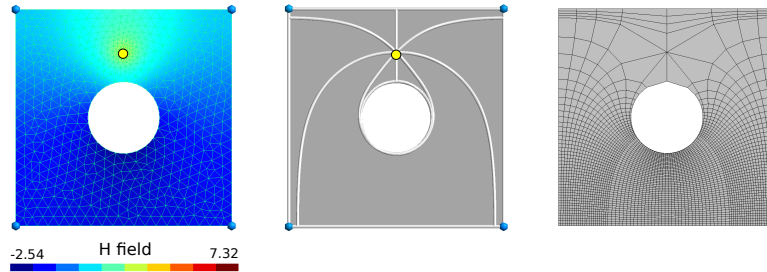


Figure 6.4: Illustration of the reduced 3-steps pipeline

The two-step cross-field computation delivers thus an accurate mesh independent cross-field, figure 6.3, from which one can then continue with a neat quad layout decomposition, and eventually a full quadrilateral mesh, figure 6.3 and 6.4. Quad layout partitioning is performed on the final (accurate) cross-field by applying the separatrice tracing scheme presented in [60]. The final step is the generation of the quadrilateral mesh by following the parameterization imposed by the cross-field (section 7).

The steps of the proposed approach are detailed in Algorithm 4. The whole pipeline has been implemented as a fully functional module in Gmsh [23], the open source finite element mesh generator. Although the pipeline can be fully automated, a certain level of interactivity is possible. The user is indeed given the opportunity to modify the singularity pattern and to adapt the location and index of singularities, as long as it corresponds to the topology of the domain. The modified singularity valence can be as high as six or eight, figure 6.4.

Algorithm 4 Generate all quad mesh

INPUT: FILE in **.geo** format

OUTPUT: FILE in **.msh** format

- 1: Non-linear cross-field computation on Ω (section 3)
 - 2: Extract singularity pattern (section 4)
 - 3: Compute cross-field with imposed singularities (section 5)
 - 4: Obtain and correct quad layout (section 6)
 - 5: Generate quad mesh on Ω (section 7)
-

2 | Related work

Due to the fact that some of the techniques used in our approach are customary in the meshing and computational geometry communities, we will focus in this article on the most notable trends and contrasts/resemblances with existing works. For the sake of comprehensiveness, bibliographic references for our algorithm are indicated in corresponding sections.

2.1 Cross-field generation for meshing purposes

For quad meshing purposes, authors are generally relying on either explicit quadrangulations or parameterization/remeshing techniques [73]. The group of parameterization approaches is enriched with numerous methods [28], where rough dissemination could be made with two generalized flows: cross-field guidance algorithms [19, 59, 60, 63] and cut-graph techniques [55, 64]. Our approach follows the first flow with the difference of using two independent successive cross-fields, which is unusual in related work on cross-field generation [62, 81]. Similarly to [60, 65, 106], we aim at obtaining a locally integrable cross-fields, such that for each point X inside a patch holds:

$$\nabla \times \mathbf{u}_1 = 0, \quad \nabla \times \mathbf{u}_2 = 0, \quad (6.1)$$

where $\mathbf{u}_1$ and $\mathbf{u}_2$ represent two orthogonal branches of the cross defined in point X .

Stressed out by [125], a major compromise to be done in the computation of cross-fields is between efficient design of vector field and control over singularities. Our algorithm runs fully automatic which frees the user of painful task to explicitly specify all singularities on complex domains [126]. At the same time, due to the independent steps of the algorithm, we are also able to fulfill the aim of generating a fast smooth cross-field that respects a user-defined singularity patterns as in [62, 73].

2.2 Singularities with high valences

High-order singularities' patterns play an important role in numerous scientific applications [90, 127, 128, 129, 130]. Initially, high-order singularities' definition, extraction and visualization were relying on mathematical background of Clifford algebra, typically motivated by applications for fluid mechanics or electrostatics [90, 127, 130]. Some more recent papers [129, 130] are focused on their role in vector field generation. In our case, the application of high-order singular points is exploited for its benefits for quad meshing purposes, specifically obtaining a quad layout and inducing important element size gradients in the refined block-structured quad mesh.

When it comes to imposed singularity configuration, it is essential to note that not all of the topologically correct singularity patterns will be valid for quad meshing. The work [67] presents the sufficient and necessary conditions for valid singularity configuration, where both the singularities' valences and holonomy are taken into account. Some of the most interesting findings on the non-existence of conformal quadrangulations can be found in [67, 94, 95, 96].

2.3 Quad layout partition guided by cross-field

Representation of geometrically or topologically complex domains requires their decomposition into blocks, preferably quadrilateral. A wide range of existing methods is described in the survey [28]. Works analogous to ours, *i.e.*, cross-field based approaches, are decomposing the domain either by generating separatrix graph obtained from numerical integration of streamlines [19, 59, 60, 63, 100] or by constructing edge maps [98, 99, 131] and motorcycle graph [101] using the isolines of an underlying parameterization. As already reported by authors working on the first group of approaches, generated quad layout commonly requires repairing of invalid configurations caused by the appearance of limit cycles and thin blocks. Some of the efficient methods for solving these issues are [49, 98, 100, 132] as well as the quantization method [55]. Generating and correcting the quad layout, in our case, shares some common points with [55, 60, 100], exact differences with respect to these works will be outlined in the corresponding sections.

3 | First cross-field computation

A 2D cross $\mathbf{c}$ is a set of 4 orthogonal vectors of the same norm l , *i.e.*, $\mathbf{c} = \{\mathbf{u}_k, |\mathbf{u}_k| = 1, k \in [1, 4]\}$. On a planar 2D domain Ω , a Cartesian basis $\{\mathbf{x}, \mathbf{y}\}$ can be defined, and the angular orientation of the cross $\mathbf{c}$ can be simply represented by an angle θ in that basis, since one has $\mathbf{u}_k = l \cos(\theta + k\frac{\pi}{2})\mathbf{x} + l \sin(\theta + k\frac{\pi}{2})\mathbf{y}$, as depicted in figure 6.5. This angular representation θ of the cross is however determined only up to an additive factor $k'\frac{\pi}{2}$ with $k' \in \mathbb{Z}$.

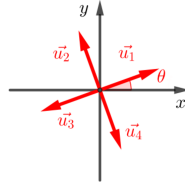


Figure 6.5: 2D cross definition as a set of four orthogonal vectors

A 2D cross-field, now, is a map $\mathcal{C}_\Omega : \mathbf{X} \in \Omega \rightarrow \mathbf{c}(\mathbf{X})$, and the standard approach to compute a smooth boundary-aligned cross-field is to minimize the Dirichlet energy

$$\min_{\mathcal{C}_\Omega} \int_{\Omega} \|\nabla \mathcal{C}_\Omega\|^2 \quad (6.2)$$

subject to the boundary condition $\mathbf{c}(\mathbf{X}) = \mathbf{g}(\mathbf{X})$ on $\partial\Omega$, where $\mathbf{g}$ is a given function.

The indetermination on θ mentioned above prevents from using it directly as unknown for the finite element formulation of the problem. In order to have a unique (symmetry invariant) and continuous cross representation, the alternative vector representation $\mathbf{v}(\mathbf{X}) = (\cos(4\theta), \sin(4\theta))$ is preferred [86], which leaves us with a vector field to solve for.

This vector field is continuous, but also subject to the nonlinear constraint $|\mathbf{v}(\mathbf{X})| = 1$. An elegant way to deal with this constraint is to minimize a Ginzburg-Landau type functional, which penalizes local vector values that deviate from the unit norm [44]. Direct solving of the non-linear Ginzburg-Landau problem is however rather expensive. Another approach is the MBO algorithm proposed in [63]. It is a faster and simpler alternative that provably converges to the same results on uniform meshes. The main idea of the MBO algorithm is to alternate a heat diffusion problem, that performs the Laplacian smoothing of the vector field, with a projection back onto the space of unit norm cross-fields, until convergence is achieved. Each step of MBO algorithm is thus given by

$$\frac{\partial \mathbf{v}}{\partial t} = \alpha \nabla^2 \mathbf{v} \quad \text{then} \quad \mathbf{v} \leftarrow \frac{\mathbf{v}}{\|\mathbf{v}\|}. \quad (6.3)$$

As in [133], we start with large diffusion (*e.g.* $\alpha = (0.1 * \text{bbox diagonal})^2$) and we progressively decrease until we reach the small diffusivity coefficient associated with the mesh size (*e.g.* $\alpha = (\text{edge min})^2$). This strategy considerably accelerates convergence as the low frequencies are solved before the larger ones, as in a multi-grid approach. To further accelerate the computation, we use a small number of diffusivity coefficients (*e.g.*, five to ten) and we apply the MBO step until convergence for each diffusivity coefficient. As the diffusion linear system is not changing for a given diffusivity coefficient, we can factorize the linear system one time and reuse it until convergence. In the end, the computational cost is approximately one linear system solve per diffusivity coefficient, *i.e.*, five to ten in total, which is very reasonable to accurately solve the cross-field non-linear problem (equation 7.2).

It is important to note that the aim of this first cross-field computation is to locate singularities. Therefore, any cross-field computation method could be used here instead, as for example the one described in [134], which could prove to be faster.

4 | Singularity pattern

Singularities play an important role for meshing purposes and their properties are well documented in the literature [19, 59, 60, 63, 100]. Obviously, cross-fields have a close parenthood with vector fields and, like any direction field (a normalized vector field, a cross-field, $\dots$, more details in [62]), they are subject to the topological constraints of the Poincaré-Hopf theorem. Let us consider, for instance, a smooth unit norm vector field $\mathbf{u}$ on a surface Ω . The local value of this vector field can be defined as the rotation by a certain angle θ with respect to a local angular reference around the local normal vector to the surface. This natural geometric representation of the vector field has however a number of issues. It is only defined up to an additive angle $2k\pi, k \in \mathbb{Z}$. It is also discontinuous wherever the value of θ jumps from 0 to 2π or reversely. Finally, as customary in differential geometry, θ values at distant points are incommensurable unless a parallel transport rule is added, because they refer to different local normal vectors and different local reference positions.

Still, in an infinitesimal neighborhood V_X of a point $X \in \Omega$, the parallel transport is superfluous because both the normal vector and the reference angular reference can be considered constant. The smooth vector field $\mathbf{u}$ then induces a map $C^1 \rightarrow C^1$, where C^1 is the unit circle, for any (infinitesimal) closed curve $C \subset V_X$. This map follows the turns done by $\mathbf{u}$ as the curve C is traveled once. As $\mathbf{u}$ is smooth, the number of turns is necessarily an integer number, called the *index* of $\mathbf{u}$ at X , $\text{index}_X(\mathbf{u}) \in \mathbb{Z}$. The point X is regular if the index is zero, and it is a singularity of index $\text{index}_X(\mathbf{u})$ otherwise.

The principle is similar for a cross-field $\mathcal{C}_\Omega$ except that the map is $C^1 \rightarrow C^1/O$ where O stands for the symmetry group of the cross, and that the index is now an integer multiple of $\frac{1}{4}$. If one notes $S_j, j = 1, \dots, N$, the singularities of the cross-field, the Poincaré-Hopf theorem states that

$$\sum_{j=1}^N \text{index}_{\mathcal{C}_\Omega}(S_j) = \chi(\Omega), \quad (6.4)$$

where $\chi(\Omega)$ is the Euler characteristic of the (not necessarily planar) surface Ω . The indices of the singularities will be noted

$$k_j \equiv \text{index}_{\mathcal{C}_\Omega}(S_j) = \frac{t}{4}, \quad t \in \mathbb{Z} \quad (6.5)$$

and we shall also use the related concept of valence defined by

$$k_j = \frac{4 - \text{valence}(S_j)}{4}, \quad \text{valence}(S_j) = 4 - 4k_j.$$

In practice, we use a simple way to find the singularity location by computing the winding number in a similar fashion as in [63], originally from [135]. The corresponding valence is then determined by detecting the number of separatrices concurring at the singularity [60]. When the cross-field is computed on a domain with a coarse mesh around singularities (of valence five or higher), numerical inaccuracies can be expected in the determination of the cross-field orientations, which will finally lead to an invalid quad layout generation. To prevent this, the mesh is refined in the form of bicycle spokes in the vicinity of the identified singularities, so that the large gradients of cross directions are well accommodated, as shown in figure 6.6.

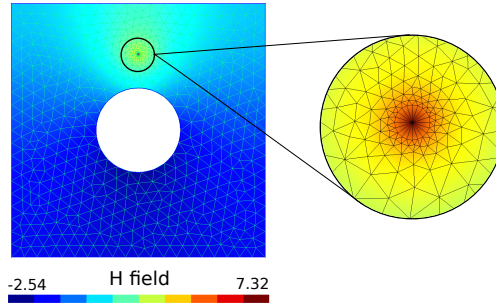


Figure 6.6: Mesh refinement around singularity of valence 8

An adaptation of the singularity pattern can be performed at this stage of the pipeline. With the two-step approach, the user is indeed offered the

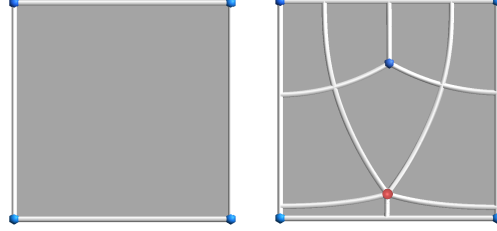


Figure 6.7: Two quad layouts of the same region. The addition of a 3-5 pair to the singularity pattern, in the second example, allows imposing a specific size map on the region.

opportunity before the second cross-field computation to modify the location of singularities, or even to place additional singularities, provided the final singularity pattern still fulfills the Poincaré-Hopf topological constraint (7.1). For instance, additional pairs of singularities of valence 3 and 5 do not modify the result of the summation in (7.1). Such 3-5 pairs can therefore be added freely to the singularity pattern in order, for instance, to allow a less distorted quad layout or to impose a specific size map, as shown in figure 6.7.

In practice, imposing a singularity pattern simply consists of encoding valences data for vertices of the initial mesh and ensuring a sufficient mesh refinement in their respective vicinities.

It will also be shown at the end of the following section that not all Poincaré-Hopf valid singularity patterns will be suitable for obtaining a high quality quad layout. Mispositioning of singularities or imposing inadequate valences may lead in practice to non-quadrilateral patches, since the developed formulation is under constraint of Abel-Jacobi theorem [67].

5 | Cross-field computation with imposed singularity pattern

The nonlinear problem described in section 3 is able to reveal a singularity pattern for the domain Ω under analysis. The computed cross-field $\mathbf{v}$ is however rather inaccurate. It is indeed continuous and singularities are areas in this continuous field where the norm deviation from unity fails to be ruled out by the penalty term. This inaccuracy does not heavily affect the singularity pattern, because singularity indices are integer quantities and their location needs only to be known approximately. However, it jeopardizes the accurate subsequent tracing of separatrices and thus justifies the construction of a second more accurate cross-field with a truly nonlinear field representation, and whose

computation is based on the singularity pattern that has been extracted from the first computation (section 4).

Let the singularity pattern be defined as the set $\mathcal{S} = \{S_j, j \in [1, N]\}$ of N singularities located at $\mathbf{X}_j$ and of respective index k_j . A cross-field $\mathcal{C}_\Omega$ matching this singularity pattern must fulfill three constraints:

- if $\mathbf{X}$ is on $\partial\Omega$, at least one branch of $\mathcal{C}_\Omega(\mathbf{X})$ is tangent to $\partial\Omega$,
- if $\mathbf{X}$ is a singularity of $\mathcal{C}_\Omega$ then $\mathbf{X} \in \mathcal{S}$,
- if $\mathbf{X} = S_j$ then $\text{index}(\mathcal{C}_\Omega(\mathbf{X})) = k_j$.

As in this paper only the *planar case* is considered, $\Omega \subset \mathbb{C}$ can be regarded as a subset of the complex plane. We shall note $\mathbf{n}$ the unit normal vector to the complex plane, $\mathbf{g}$ the interior unit normal vector to the boundary $\partial\Omega$, and $\mathbf{T} = \mathbf{g} \times \mathbf{n}$ the unit vector tangent to $\partial\Omega$.

The method presented in this section relies on the computation of a dual scalar field H defined in section 5.1, continuous on the whole domain Ω . This approach is similar as the one detailed in [89].

5.1 Computation of the H field

We are looking for a holomorphic function

$$\begin{aligned} \mathcal{F} : \quad \mathcal{P} \subset \mathbb{C} &\rightarrow \mathbb{C} \\ \mathbf{z} = U + iV &\mapsto \mathbf{P} = x + iy \end{aligned} \quad (6.6)$$

where $\mathcal{P}$ is a parametric space. As $\mathcal{F}$ and $\mathcal{P}$ are unknown, solving directly this problem is proved to be a difficult task. Instead, the focus is on finding the 2×2 jacobian matrix of $\mathcal{F}$

$$J_{\mathcal{F}}(\mathbf{z}) = (\partial_U \mathcal{F}(\mathbf{z}), \partial_V \mathcal{F}(\mathbf{z})) \equiv (\tilde{\mathbf{u}}, \tilde{\mathbf{v}}), \quad (6.7)$$

where $\tilde{\mathbf{u}}, \tilde{\mathbf{v}}$ are the column vectors of J . The function $\mathcal{F}$ being holomorphic, the columns of J have the same norm $\|\tilde{\mathbf{u}}\| = \|\tilde{\mathbf{v}}\| \equiv e^H$ and are orthogonal to each other, $\tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}} = 0$, so that one can write

$$J_{\mathcal{F}}(\mathbf{z}) = e^H(\mathbf{u}, \mathbf{v}) \quad , \quad \mathbf{n} = \mathbf{u} \wedge \mathbf{v}$$

$\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ are 2 branches of the cross-field we are looking for, and $\tilde{\mathbf{u}}$ can be represented by a complex function

$$f : \Omega \rightarrow \mathbb{C} \quad , \quad f = e^{H+i\theta}. \quad (6.8)$$

The computation of the cross-field is rephrased into the computation of two real functions, $H : \Omega \rightarrow \mathbb{R}$ and $\theta : \Omega \rightarrow \mathbb{R}$, from which the first branch $\tilde{\mathbf{u}} = f = e^{H+i\theta}$ can be deduced, and the second branch of the cross-field is simply $\tilde{\mathbf{v}} = i\tilde{\mathbf{u}} = if$.

By construction, $\tilde{\mathbf{u}}$ being a partial derivative of a holomorphic function, f is holomorphic. As equation (6.8) shows, H and θ are the real and imaginary parts of a complex logarithmic function. Indeed,

$$\begin{aligned} \log(f) &= \ln(|f|) + i \arg(f) \\ &= H + i(\theta + 2k\pi), k \in \mathbb{Z}. \end{aligned} \quad (6.9)$$

The complex logarithm being a holomorphic function, both H and θ are harmonic real functions. The H field, which is the real part of the logarithm, is continuous on $\Omega \setminus \mathcal{S}$, *i.e.*, on the domain Ω from which the pointwise singularities $\mathcal{S}_j$ have been excluded. The θ field, on the other hand, is multi-valued and a *branch cut* (see section 5.2.1) has to be additionally defined on $\Omega \setminus \mathcal{S}$ to have it single-valued, and make its computation possible. Finally H and θ obey the Cauchy-Riemann equations

$$\nabla\theta = \mathbf{n} \times \nabla H, \quad (6.10)$$

which shall allow to obtain θ , once H has been computed.

The computation of the scaling field H , with regard to boundary curvature and singularities indices, is performed first. As shown in [22], it is the solution of the partial differential problem on Ω

$$\nabla^2 H = 2\pi \sum_{j=1}^N \frac{k_j}{4} \delta(\mathbf{X}_j). \quad (6.11)$$

The boundary condition $\mathbf{T}(s) = (\cos \theta(s), \sin \theta(s))$ ensures that the cross, represented by $\theta(s)$, is tangent to the boundary $\partial\Omega$, represented by the tangent vector $\mathbf{T}(s)$, with s a curvilinear coordinate on $\partial\Omega$. This entails by derivation

$$\partial_s \mathbf{T} = (-\sin \theta, \cos \theta) \partial_s \theta = \mathbf{g} \partial_s \theta.$$

On the other hand, one knows from the differential geometry of surfaces that $\partial_s \mathbf{T} \equiv \kappa_g \mathbf{g}$ with κ_g the geodesic curvature of the curve $\partial\Omega$. One has thus by identification

$$\kappa_g = \partial_s \theta = \mathbf{T} \cdot \nabla \theta = (\mathbf{n} \times \nabla H) \cdot \mathbf{T} = -\mathbf{g} \cdot \nabla H,$$

where equation (6.10) has been used.

Summing up, the computation of H consists of solving the linear boundary value problem

$$\begin{cases} \nabla^2 H &= 2\pi \sum_{j=1}^N \frac{k_j}{4} \delta(\mathbf{X}_j) & \text{in } \Omega \\ -\mathbf{g} \cdot \nabla H &= \kappa_g & \text{on } \partial\Omega. \end{cases} \quad (6.12)$$

As the boundary condition is of Neumann type on the entire boundary, a solution H exists if the condition

$$\int_{\partial\Omega} \kappa_g \, d\partial\Omega = \int_{\Omega} 2\pi \sum_{j=1}^N \frac{k_j}{4} \delta(\mathbf{X}_j) \, d\Omega \quad (6.13)$$

is verified, which is obviously the case, as the Gauss-Bonnet theorem states the first identity in

$$\int_{\partial\Omega} \kappa_g \, d\partial\Omega = 2\pi\chi(\Omega) = \int_{\Omega} 2\pi \sum_{j=1}^N \frac{k_j}{4} \delta(\mathbf{X}_j) \, d\Omega \quad (6.14)$$

whereas the second identity is the Poincaré-Hopf theorem.

The solution H of the boundary value problem equation (6.12) is however known up to an arbitrary additive constant, which is not harmful as only ∇H is needed to determine θ . It is a linear problem solved using the finite element method on a triangulation of Ω with order 1 Lagrange elements. Once H is known, one proceeds with the determination of θ , as explained in the next section, to complete the cross-field $(f, \iota f)$.

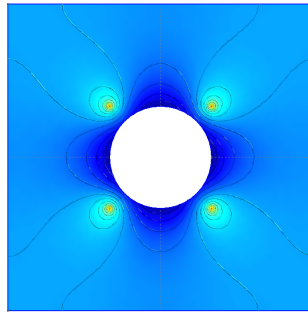


Figure 6.8: H field obtained for 4 singularities of valence 3 on corners, and 4 singularities of valence 5 inside the domain.

5.2 Computation of the θ field

As the curl of the right-hand side of equation (6.10)

$$\nabla \times (\mathbf{n} \times \nabla H) = (\nabla^2 H)\mathbf{n} = \mathbf{0}$$

vanishes in $\Omega \setminus \mathcal{S}$, one knows that the left-hand side is indeed a gradient, and that there exists a scalar field θ verifying the differential equation (6.10). This function is however multi-valued and a branch cut has to be determined to make it single-valued, and make its computation possible.

5.2.1 Generate the branch cut

A branch cut is a set $\mathcal{L}$ of curves of a domain Ω that do not form any closed loop and that cut the domain in such a way that it is impossible to find any closed loop in $\Omega \setminus \mathcal{L}$ that encloses one or several singularities, or an internal boundary. As we already have a triangulation of Ω , the branch cut $\mathcal{L}$ is in practice simply a set of edges of the triangulation. The method presented here is based on [73].

The nodes and the edges of a triangulation of a domain Ω can be regarded as a graph $\mathcal{G}$. A spanning tree $\mathcal{T}$ is a sub-graph of $\mathcal{G}$ that links all vertices of $\mathcal{G}$ with the minimum number of edges. A spanning tree can be generated by a Breadth-first-search algorithm. One starts from an arbitrary vertex v , and adds to the spanning tree $\mathcal{T}$ all edges (v, v_i) that are adjacent to v and that connect vertices v_i that have not yet been visited by the algorithm. The procedure is continued recursively with all newly considered vertices v_i until running out of unvisited vertices. There exist many equivalent spanning trees for a given mesh. Their shape can be improved by modifying the Breadth-first-search algorithm so that vertices which are the closest to the singularities or to the boundaries are processed in priority. A spanning tree built this way is presented in red in figure 6.9.

This spanning tree forbids however closed curves around all nodes of the mesh, which is an unnecessarily strong restriction, as we only want to forbid closed curves around the singularities. If we call hanging edge of $\mathcal{T}$ an edge whose leaf node v_l is not a singularity (a leaf node is a node of the graph adjacent to exactly one edge), the branch cut is the sub-graph of $\mathcal{T}$ obtained by successively removing hanging edges until no hanging edges is left in the spanning tree. The result of this subtraction is the branch cut $\mathcal{L}$ needed for the computation of θ . It is depicted in black in figure 6.9.

5.2.2 Solve the Cauchy-Riemann equations

Once a branch cut $\mathcal{L}$ is available, the field θ can be computed by solving the linear Cauchy-Riemann equations (6.10). The chosen boundary condition

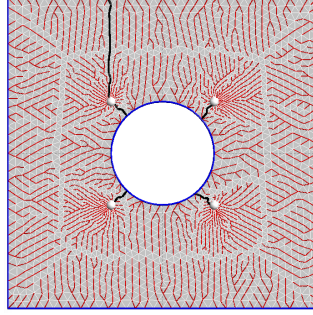


Figure 6.9: $\partial\Omega$ is represented in blue, in white the singularities, in red are edges of the spanning tree $\mathcal{T}$, and in black the edges of the branch cut $\mathcal{L}$.

consists of fixing the angle θ at *one* arbitrary point $\mathbf{P} \in \partial\Omega$ so that $\mathcal{C}_\Omega(\mathbf{P})$ has one of its branches collinear with $\mathbf{T}(\mathbf{P})$. The problem can be rewritten as the well-posed equation (6.15) and is solved using finite element method on the triangulation Ω_T with order one Crouzeix-Raviart elements. This kind of elements have proven to be more efficient for cross-field representation as underlined in [60].

$$\begin{cases} \nabla\theta &= \mathbf{n} \times \nabla H \text{ in } \Omega \\ \theta(\mathbf{P}) &= \theta_{\mathbf{P}} \text{ for an arbitrary } \mathbf{P} \in \partial\Omega \\ \theta &\text{ discontinuous on } \mathcal{L} \end{cases} \quad (6.15)$$

Figure 6.10 presents results for 3 different types of singularities. As expected, H and θ isolines are orthogonal, these two scalar fields being harmonic conjugate.

Once H and θ scalar fields are computed on Ω , the cross-field $\mathcal{C}_\Omega$ can be retrieved for all $\mathbf{X} \in \Omega$:

$$\mathbf{c}(\mathbf{X}) = \{\mathbf{u}_k = e^{H(\mathbf{X}) + jk\theta(\mathbf{X})}, k \in [1, 4]\} \quad (6.16)$$

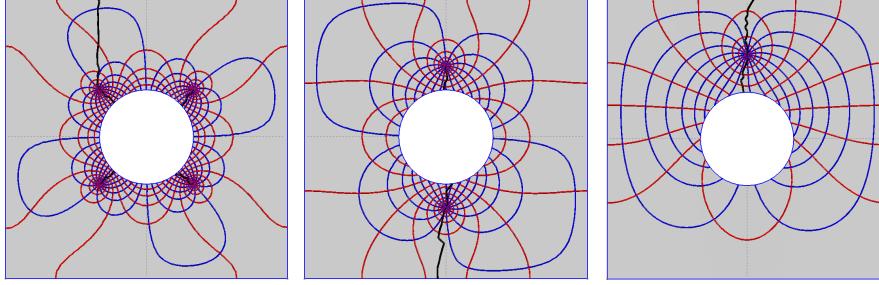


Figure 6.10: H isolines (in red) and θ isolines (in blue) for different singularities configurations. The black line represent the cutgraph. All cases have valence 3 singularities imposed at corners (for a total of 4 singularities in each case). The extra singularities which are added inside the domain are: 4 singularities of valence 5 (left figure), 2 singularities of valence 6 (center figure), 1 singularity of valence 8 (right figure).

5.2.3 Non quad-meshable cross-fields

We showed in the previous section that if the imposed singularity pattern respects the condition on Euler's characteristic, then there exists a unique scalar function H (up to an additive constant) verifying equation (6.12) and a unique scalar function θ verifying equation (6.15). This is leading us to a unique cross-field solution $\mathcal{C}_\Omega$ to our problem.

However, we never showed that the obtained cross-field $\mathcal{C}_\Omega$ is actually respecting the following and mandatory condition:

- $\forall \mathbf{X} \in \partial\Omega$, at least one branch of $\mathcal{C}_\Omega(\mathbf{X})$ is tangent to $\partial\Omega$

Indeed, problem (6.15) is well-posed and only ensures this condition for a given $\mathbf{P} \in \partial\Omega$. Adding extra boundary conditions for θ will lead to make problem (6.15) ill-posed. Therefore, when domain Ω is not simply connected, it is possible that obtained cross-field $\mathcal{C}_\Omega$ is not tangent on parts of $\partial\Omega$.

When this happens, it means that there is no holomorphic function $\mathcal{F}$ (equation 6.6) matching all conditions imposed (*i.e.* singularity pattern and tangency to domain boundaries). Figure 6.11 shows two different singularity patterns where only the singularity location differs for a same domain Ω . The domain Ω considered is an annulus, and a set of 4 index -1 and 4 index 1 singularities is imposed. Singularity pattern for the example on the left leads to a quad decomposition of the domain and singularity pattern for the example on the right does not. Cross-field obtained for example on the right does not respect

the tangency condition on Ω interior boundary. There exists no holomorphic transformation matching at the same time singularity pattern and cross-field tangency to Ω boundaries.

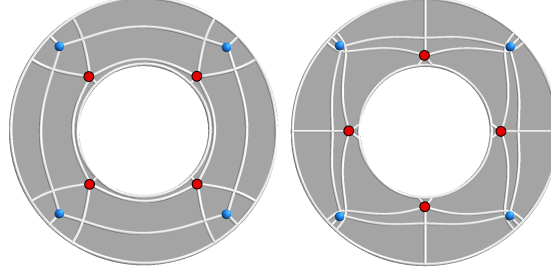


Figure 6.11: Highlighting of singularity patterns for which no holomorphic transformation matching all imposed conditions exists.

The issue mentioned above emanates from the fact that the developed formulation is under constraint of having singularity configuration fulfilling the Abel-Jacobi conditions presented in [67]. More specifically, imposing Poincaré-Hopf's valid singularity configuration ensures that the flat metric, *i.e.*, the H field can be obtained, and if this singularity configuration also verifies the holonomy condition, the cross-field will be aligned with the boundary and consistent across the cut graph. The algorithm for valid singularity configurations [67] is developed for closed 2D manifolds, illustration of invalid singularity configuration on closed 2D manifold in figure 6.12 and its topologically equivalent flat 2D domain in figure 6.13.

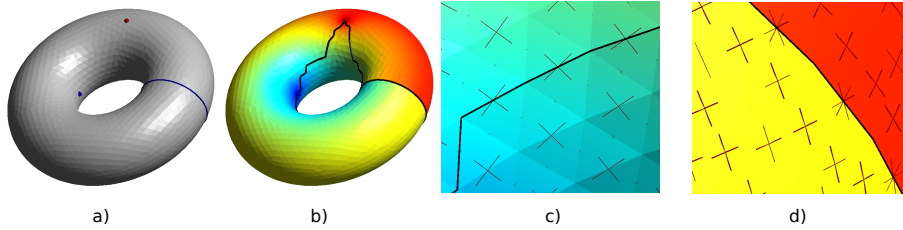


Figure 6.12: Imposing a 3 – 5 singularity configuration on a torus with the boundary marked in blue a), marking the boundary and the cut graph in black b), consistent cross-field across the cut graph c), and cross-field not aligned with the boundary d)

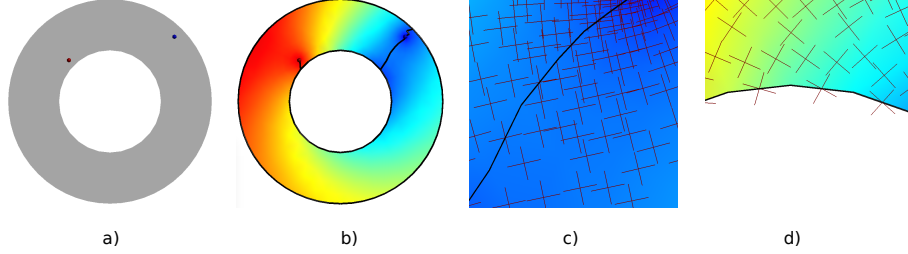


Figure 6.13: Imposing a 3 – 5 singularity configuration on an annulus a), marking the boundary and the cut graph in black b), consistent cross-field across the cut graph c), and cross-field not aligned with the boundary d)

5.3 Computing the per-partition parameterization

Obtaining a global parameterization comes down to computing real values functions $U(x, y)$ and $V(x, y)$ defined in equation (6.6), from the known smooth jacobian of the mapping $\mathcal{F} : J_{\mathcal{F}}(\mathbf{z}) = (\partial_U \mathcal{F}(\mathbf{z}), \partial_V \mathcal{F}(\mathbf{z})) \equiv (\tilde{\mathbf{u}}, \tilde{\mathbf{v}})$ equation (6.7). Establishing U and V is equivalent to determining $\mathcal{F}^{-1}$. We know that $\forall \mathbf{P} = \mathcal{F}(\mathbf{z}) \in \Omega \setminus \mathcal{S}$, $J_{\mathcal{F}}$ corresponds to two branches of the cross-field.

We also know that $J_{\mathcal{F}^{-1}}$ jacobian of $\mathcal{F}^{-1}$ is, $\forall \mathbf{P} \in \Omega \setminus \mathcal{S}$, such as:

$$\begin{cases} J_{\mathcal{F}^{-1}}(\mathbf{P}) &= (\partial_x \mathcal{F}^{-1}(\mathbf{P}), \partial_y \mathcal{F}^{-1}(\mathbf{P})) \\ &= J^{-1}(\mathcal{F}^{-1}(\mathbf{P})) \\ &\equiv (\tilde{\mathbf{u}}, \tilde{\mathbf{v}})^T. \end{cases} \quad (6.17)$$

which, by using the equation (6.7), gives:

$$\begin{cases} \tilde{\mathbf{u}} &= e^{-H} \mathbf{u} = \nabla U \\ \tilde{\mathbf{v}} &= e^{-H} \mathbf{v} = \nabla V. \end{cases} \quad (6.18)$$

Now, the parameterization can be simply obtained by solving:

$$\begin{cases} \nabla U = \tilde{\mathbf{u}} \\ \nabla V = \tilde{\mathbf{v}}, \end{cases} \quad (6.19)$$

using a finite element method with order one Lagrange elements.

We mentioned earlier that for the problem to be well-posed, $J_{\mathcal{F}}$ has to be smooth, meaning that $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ have to be smooth. This implies that a lifting of the cross-field $\mathcal{C}_{\Omega}$ has to be done on Ω allowing discontinuities of $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}})$ across $\mathcal{L}$. Once this operation is done, it is possible to determine $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}})$ and compute U and V by solving the equation (6.19). By construction, U and V will be

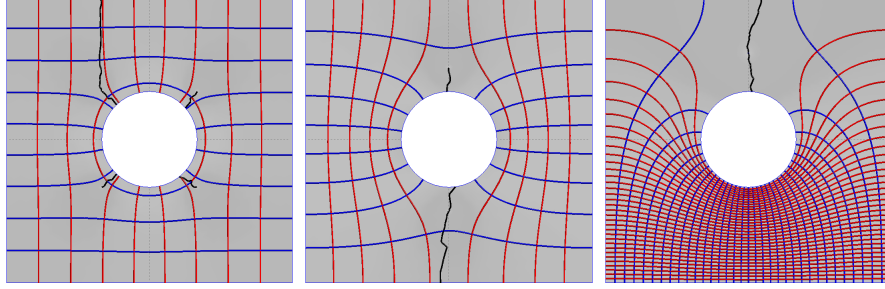


Figure 6.14: U isolines (in red) and V isovalues (in blue) for different singularities configurations. The black line represents the cutgraph.

discontinuous across $\mathcal{L}$, and their isolines will be in all points tangent to the cross-field $\mathcal{C}_\Omega$.

It is important to note that for this global parameterization computation, no conditions are imposed on the cut graph $\mathcal{L}$. Therefore, nothing guarantees correspondence of U and V isolines across $\mathcal{L}$. In principle, this global parameterization is not enough by itself to generate a conforming quadrilateral mesh of a domain Ω . Some works bypass these difficulties [64, 71, 73] by constraining parametric coordinates on boundaries and on $\mathcal{L}$, however it is not the direction followed in our approach.

The goal here is to obtain per-partition parameterization, starting from the construction of a conformal quad layout of Ω (section 6) and followed by the computation of independent parameterizations for each of the resulting quadrilateral partitions. The per-partition parameterization computed in this manner is aligned with smooth cross-field (singularities can only be located on corners of the partitions). To achieve this goal, the first step is to extract corresponding triangles of each patch. Due to the mutual separatrices' points, neighboring partitions contain the overlapping triangles, as shown in figure 6.15.

An important remark is that no cut graph is needed here to perform a lifting of the underlying cross-field to obtain smooth couple of vector fields $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}})$. As no cut graph is used, all isolines of U and V will be continuous inside the patch, figure 6.16.

Based on [60] for locally integrable cross-fields, note that on each partition holds:

$$\begin{cases} \nabla \times e^{-H}u = 0 \\ \nabla \times e^{-H}v = 0, \end{cases} \quad (6.20)$$

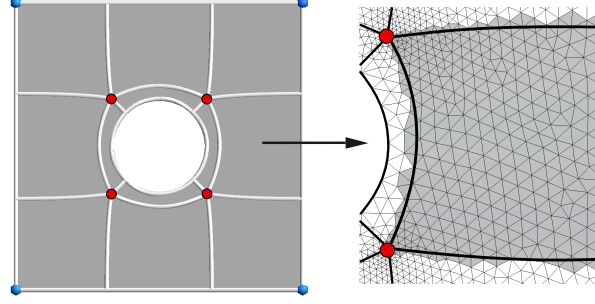
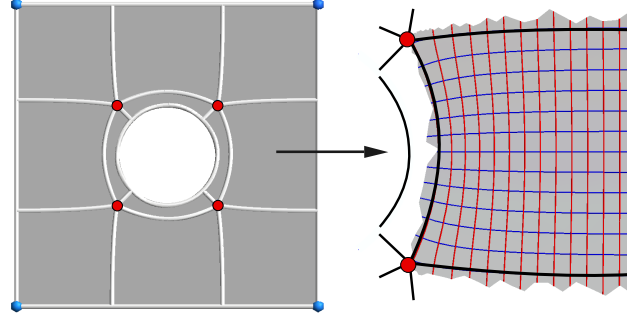


Figure 6.15: Extracted triangulated partition for parameterization

Figure 6.16: U isolines (in red) and V isolines (in blue) on extracted partition

where from $(\tilde{u}, \tilde{v})$ is a locally curl-free cross-field. Thus, the resulting parameterization (U, V) is *conformal* on each patch [65]. Following [136], a *conformal* mapping is also *harmonic*. Radó-Kneser-Choquet (RKC) theorem for planar harmonic mappings (details in [136, 137]) states:

If $f : \mathcal{S} \rightarrow \mathbb{R}^2$ is harmonic and maps the boundary $\partial\mathcal{S}$ homeomorphically into the boundary $\partial\mathcal{Q}$ of some convex region $\mathcal{Q} \subset \mathbb{R}^2$, then f is *one-to-one*.

In our case, the parametric (image) region is in the shape of a rectangle, thus, our parameterization (U, V) is per-partition *bijective*.

6 | Quad layout: generating and correcting the partitions

In order to reach the ultimate goal of generating a block-structured quad mesh, we must ensure that the quad layout contains only conformal quadrilateral partitions. To do so, a special attention is paid on obtaining the minimal

number of separatrices, existence of limit cycles and non-quadrilateral partitions. Although the quad mesh generation here relies on integrable cross-field, it is essential to note that it is under the constraint of Abel-Jacobi conditions. Therefore, misplacing singularities' locations and numerical inconsistencies can cause separatrix to miss the singularity and loop around the domain. In the following, we present ways to modify all ill-defined partitions, such that the overall quad layout topology is not jeopardized.

6.1 Obtaining the partitions

To obtain the quad layout, the separatrix-tracing algorithm is performed on retrieved cross-field from H function. Initiation of separatrices in singular areas follows the method from [60], while the tracing of separatrices in the smooth parts of domain relies on Heun's (a variation of Runge-Kutta 2) numerical scheme.

Our algorithm doesn't have an order of tracing the separatrices - they are all generated simultaneously and independently, similarly to *motorcycle graph* algorithm. Contrary to *motorcycle graph's* stopping criteria, in our case, the tracing of separatrices is finished as soon as they reach the boundary or encounter a singular point. The number of generated separatrices in this manner is not appropriate for a quad layout, due to the separatrices' initiation process which allows tracing the same separatrix from two different singular points, as shown in figure 6.17. To obtain the valid quad layout, criteria for gaining the minimal number of separatrices are adopted from [60].

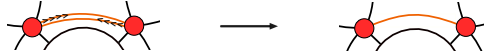


Figure 6.17: Illustration of doubled separatrix and removal of the duplicate

In case when Abel-Jacobi conditions are not fulfilled, the next step is to establish the existence of limit cycle(s) in the set of the generated separatrices. In more details, we define a separatrix which is transpassing the other separatrices more than once as a *possible* limit cycle. An *authentic* limit cycle is a possible limit cycle which has more far-flung intersections than its pair, or a separatrix not reaching the boundary orthogonally. An authentic limit cycle is then cut at the closest orthogonal intersection with another separatrix, as shown in figure 6.18. By doing so, simplifying the quad layout is accomplished (due to the disappearance of the thin chords), however the obtained partitions can contain T-junctions, which will be modified later on.

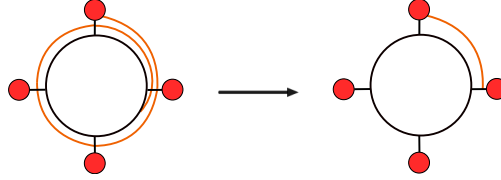


Figure 6.18: Illustration of an authentic limit cycle treatment

6.2 Valid quad layout

If the quad layout contains T-junctions (generated from cutting the authentic limit cycles) or non-quadrilateral partitions, it's no longer suitable for block-structured quad meshing. More specifically, a high-quality quad mesh, according to [21], must obtain not only the adequate configuration of singularities (in terms of their number and placement) but also respect the connectivity constraint of being conforming and purely quadrilateral, which the previous issues hinder.

The appearance of T-junctions in our algorithm has already been elucidated, which is not the case with the non-quadrilateral blocks though. In order to examine (and later on correct) the latter case, it's important to mention that our algorithm deals with planar geometries on which, as stressed out by [21] and [62], degenerate singularities (of valence 1 - *singlet* or of valence 2 - *doublet*) may appear. For instance, the typical placement of singularities of valence 2 is on corners of convex boundaries [21] or on very sharp/obtuse angles due to the number of cross-field's rotations around the local normal, as the exact formulas for singularity index in [63] show. While these types of singularities do not endanger topological validity of the obtained quad layout, they can cause problems for the block-structured quad meshing by forming non-quadrilateral blocks, as shown in figure 6.20.

6.3 Correcting the partitions

When the partition contains a T-junction, the idea is to find the closest neighbor (which is not the boundary vertex or another T-junction) and, by knowing the parameterization of these blocks, to merge these two separatrices in the parametric space and correct all patches affected by this action, figure 6.19. This process is iteratively repeated until the quad layout is T-junction free. Following the definition of separatrices and singular points, each of the newly obtained partitions have smooth cross-field inside allowing further on parameterization.

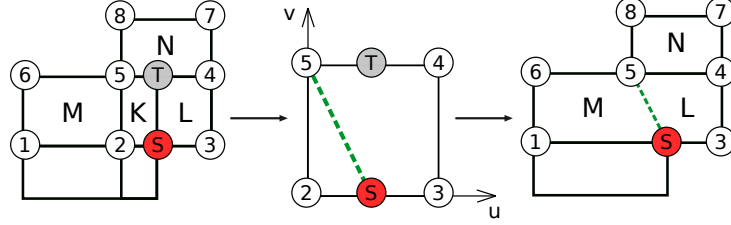


Figure 6.19: Modifying patches containing T-junctions and affected partitions

Correcting the triangular partitions is performed by forcing the singularity of valence 2 on the boundary into splitting in two singularities of valence 3, so that the topology of the overall obtained partitions remains valid (figure 6.20). The position of the imposed singularity is determined simply by finding the barycenter of the partition. Final quad layout is obtained by splitting the triangular patch into three quadrilateral partitions and correcting all patches influenced by the former action.

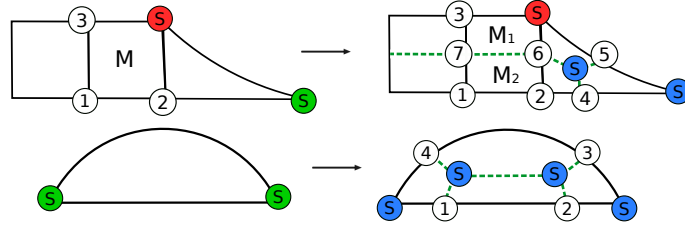


Figure 6.20: Modifying the patch containing a singularity of valence 2 and affected partitions

7 | Quad meshing

To generate a final quad mesh of the model, we start by remeshing each layout's partition with bilinear transfinite interpolation (TFI) in the parametric space, while respecting the discretizations of layout's edges implied by the field H . This step is followed by a computation of a per-partition parameterization aligned with the cross-field (as detailed in section 5.3). Finally, generating a quad mesh of pleasing quality and intact singularities' positions (examples in section 8) is obtained by mapping TFI mesh onto the physical space. At this point, it is indeed possible to further optimize vertices' placements via smoothing (*e.g.* Winslow from [109, 110]), in the cases when the application

allows/encourages moving singularities away from their initial positions. For the sake of completeness, details of the above-mentioned steps are elucidated in the following.

In order to remesh each patch of the quad layout, the chords' data has been extracted. Keeping in mind that the quad layout is now composed of only conforming quadrilaterals, it is possible to consistently discretize all its edges by following the size map implied by H :

$$s = e^H, \quad (6.21)$$

where s represents the mesh size. This step tends to generate a uniform quad mesh, unless a specific size map (via implying an adequate configuration of singularities) is imposed by the user. With obtained discretization of edges, a quad mesh in parametric space is trivially created by remeshing each partition with bilinear transfinite interpolation [15].

As the parameterization is now defined, the mapping of the quad mesh from parametric onto physical space is achieved by simply finding the triangle T_M in the parametric space to which the point $M(U, V)$ belongs. Afterward, the computing of coordinates $M(x, y, z)$ is done by assuming that there exists a *linear mapping* from physical space X to parametric space P inside the triangle T_M .

8 | Conclusion and Future Work

We presented an algorithm that can effectively generate a conformal quad layout from a naturally appearing as well as user-imposed set of singularities, as shown in figure 6.21. The latter case is possible under constraints of the topological correctness and existence of conformal mapping for imposed singularity pattern. In particular, the developed method relies on the computation of two cross-fields. The first one is computed solving a non-linear system and is used to locate singularities and obtain their valence, the second one is computed on a refined triangular mesh to accommodate high gradients and improve cross-field representation around singularities. It is obtained by solving two linear systems. With the correction scheme for limit cycles and non-quadrilateral partitions, we can produce a conformal quad layout and, finally, a block-structured quad mesh. Remeshing of each patch relies on constructing a per-partition parametrization and following the isolines from the second cross-field. A quad mesh obtained in this manner preserves singularity placement and is of pleasing quality, as shown in Table 6.1. Once the quad layout obtained, the computational cost of our quad meshing algorithm depends on the target number of quads or the approximate length of the edge imposed by the user, which takes

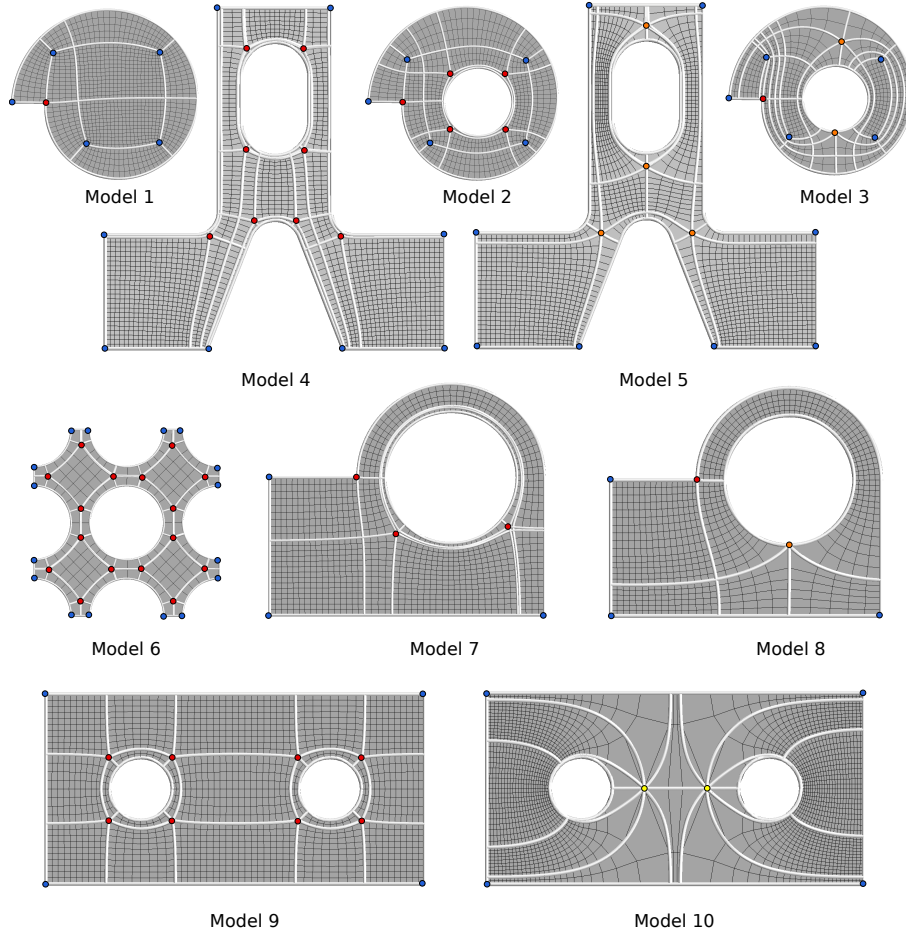


Figure 6.21: Examples of quad meshes generated from natural and user-imposed singularity configurations. Singularities of valence 3 are marked in blue, of valence 5 in red, of valence 6 in orange and of valence 8 in yellow.

less than a second in the presented examples. At last, our novel pipeline exhibits interactive design while being simple, automatic and available in the Gmsh software.

Attractive directions for future work include, although are not limited to extending the algorithm for computations on various surfaces (more details in Chapter 7), as well as its adaptation for working on varying user-prescribed element sizes. In the first case, the extension of the algorithm includes a different

computational procedure for the H function and an updated scheme for dealing with limit cycles and T-junctions. If the anisotropic quad mesh is required, two scalings for the cross-field must be defined instead of only one like in our case (more details in Chapter 8).

Geometry	Edge length	Mesh quality		
		$\bar{\eta}$	η_{ω}	τ
Figure 6.3	0.05	0.96	0.74	88.16
Figure 6.4	0.02	0.98	0.40	98.39
Model 1	0.05	0.96	0.34	93.38
Model 2	0.05	0.94	0.47	89.44
Model 3	0.05	0.94	0.52	89.19
Model 4	0.20	0.98	0.76	96.93
Model 5	0.20	0.98	0.63	96.94
Model 6	0.05	0.88	0.70	48.91
Model 7	0.02	0.98	0.78	98.78
Model 8	0.02	0.97	0.51	98.28
Model 9	0.02	0.98	0.74	96.36
Model 10	0.02	0.97	0.37	95.03

Table 6.1: The quality of obtained meshes: average quality of elements under $\bar{\eta}$, the worst element's quality under η_{ω} and percentage of elements with quality higher than 0.9 under τ . Element's quality is adopted from [111], and is a measure of the deviation of element's angle from $\frac{\pi}{2}$.

CHAPTER



Flexible quad layout generation: the 2D manifold case

This chapter presents the generation of integrable cross-field with isotropic scaling on 2D manifolds, relying on the computation of scalar field H from related Ginzburg-Landau literature [22]. The strict mathematical background from Chapter 6 underwent several changes due to curvatures; nevertheless, the concept itself and quad meshing pipeline, previously detailed only for planar domains, remained intact. The developed formulation also supports imposing singularity configuration, possibly with high valences, under the constraint of fulfilling the Abel-Jacobi theorem. The preliminary results and discussion on the developed formulation are elucidated in the following.

1 | Introduction

Numerous methods for surface parametrization/representation have been developed for a large number of applications, more details in surveys [21, 28, 136]. In cases when a shape exhibits complex topological or geometrical characteristics, it is necessary to split it into disk-alike partitions, the special case of partitioning it into a simple connected network of conformal quadrilateral partitions is called the *quad layout* [13]. The latter manner of surface representation is a subject of great interest in meshing and computer graphics communities, due to the providing of a wide range of benefits for engineering and entertainment industry purposes [21, 28, 58].

The advantages of a quad layout generation indeed come with the high price of dealing with complex and time-consuming algorithms, to which an extensive amount of research has been devoted [58]. Among the developed algorithms, a general distinction can be made among the ones which are: computing a seamless global parametrization of the domain where integer iso-values of the parameter fields form the sides [55, 64, 65], using Riemann surface theory [66, 67], or like in our case, constructing a *cross-field* structure which will guide the integral lines emanating from *singularities* [19, 59, 60, 61, 62, 63].

The latter group of methods relies on the computation of an auxiliary object: a *cross-field*, to partition the domain and orient the quadrilateral elements. This structure represents the set of four perpendicular directions associated to each point of Ω and is typically obtained via minimization of a nonlinear energy, *e.g.*, the Dirichlet energy, Ginzburg-Landau energy functional [60] or Laplacian smoothing [63], more details in [81, 86]. In the following, it will be shown that we are in fact generating two cross-fields, which is a very unusual action in related engineering fields. The motivation for designing such an object is to be able to force both singularities' locations and valences in refined regions for more accurate numerical results.

In our case, the first cross-field computation follows the MBO algorithm [63], due to the appealing results and time efficiency, while the second cross-field requires solving only two linear systems and is based on the theory of Ginzburg-Landau PDE [22]. Similarly to [60, 65], we generate curl-free cross-fields, as *integrability* contributes greatly to accurate surface parameterizations [106].

Although leaning on heterogeneous approaches, all the above mentioned methods share the common challenge: dealing with the inevitable singularity configuration. A *singularity* appears where a cross-field vanishes and it represents an irregular vertex of a quad layout / quad mesh [44], *i.e.*, a vertex which doesn't have exactly four adjacent quadrilaterals. It is important to note that the number and valence (the number of adjacent quadrilaterals) of the singular structure are constrained by Euler characteristic χ , which is a topological

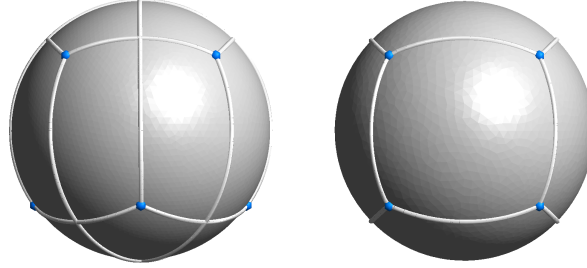


Figure 7.1: Two quad layouts of a sphere. Colored points are the irregular corners of the layouts of valence 3, which form an anti-cube on the left and a cube on the right.

invariant of a surface, making a quad layout generation a very constrained problem. Moreover, as already observed [58, 63, 64], a sub-optimal number or location of singularities have severe consequences: causing undesirable thin partitions, large distortion, not adequate number and/or tangential crossings of separatrices as well as limit cycles. Indeed, singularity configuration has a dramatically important role in both quad layout and quad mesh generation.

Our approach offers a contribution regarding this issue through the concept of user-imposed singularity pattern in order to gain direct control over their number, location, and valence. More specifically, the present work focuses on extending the algorithm [61], now for generating a quad layout and finally a block-structured quad mesh on an orientable and compact 2-manifold surface, possibly with boundaries. As in the case of planar domains, detailed in Chapter 6, the user is enabled to use either naturally appearing singularities, obtained by solving a non-linear problem, or to impose its own singularity configuration, possibly with high valences, illustrated in figure 7.18. Making sure that the high gradients of the cross-field around singularities are well resolved, we also impose a mesh refinement around them, section 2.

1.1 The algorithm

The complete pipeline of our block-structured quad mesher consists of four steps, illustrated in figure 7.2 and it follows the idea of [61], although there exist theoretical differences in computing the second cross-field, which will be elucidated in the following (section 3), allowing the expansion of the algorithm [61] on 2-manifolds.

Step 1: compute a first cross-field using a standard non-linear minimization procedure that allows finding a singularity pattern, *i.e.*, position and indices of singularities (see [61]).

Step 2: compute a second cross-field with the prescribed singularity pattern of Step 1 on an adapted mesh, by solving only two linear systems (section 3).

Step 3: compute a quad layout on the accurate cross-field of Step 2 by applying the separatrix tracing scheme presented in [60] and upgraded scheme for limit cycles handling from [61].

Step 4: eventually generate a full block-structured quadrilateral mesh (section 4).

Note that Step 1 can be skipped if the user provides a singularity configuration, figure 7.3 shows this reduced 3-step process.

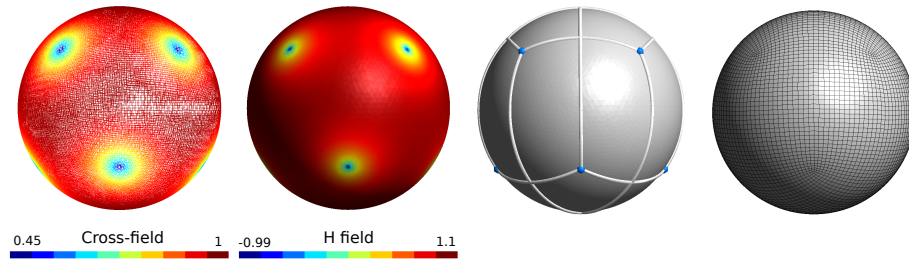


Figure 7.2: Illustration of the 4-step algorithm. Quad mesh on a 2-sphere with natural singularity configuration coming from solving a non-linear problem, *i.e.*, by using the MBO algorithm from [63].

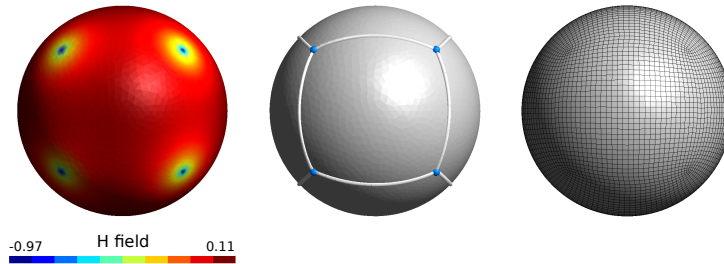


Figure 7.3: Illustration of the 3-step algorithm. Quad mesh on a 2-sphere with imposed singularity configuration.

Obtained quad mesh in this manner is isotropic, in case when the application demands an anisotropic one, two sizing fields must be computed instead of one like in our case. Quad mesh in figure 7.2 has average quality 0.96, the worst element's quality is 0.42, and the percentage of elements with the quality higher than 0.9 is 92%. Similarly, the quad mesh in figure 7.3 has average quality

0.97, the worst element's quality is 0.47, and the percentage of elements with a quality higher than 0.9 is 96%. The estimation of the element's quality as a measure of the element's angle deviation from $\frac{\pi}{2}$ relies on the evaluation presented in [111].

It is interesting to note that these two singularity configurations carry different Ginzburg-Landau energies. Although both configurations are composed of 8 singular points of valence 3, the overall energy also considers their locations (more details in [22]). By tilting the base squares of the singularity configuration around their common axis from 0 to $\frac{\pi}{2}$ degrees, it is possible to show that the minimum energy corresponds to a tilting angle of $\frac{\pi}{4}$, *i.e.*, the anticube configuration, as shown in figure 7.4. A significant remark is that both singularity configurations are valid for quad meshing despite the energy differences.

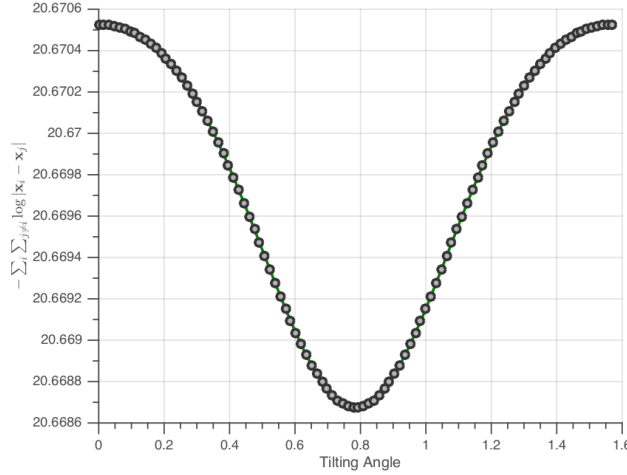


Figure 7.4: Ginzburg-Landau renormalized energy evaluation and the tilting angle (ranging from 0 to $\frac{\pi}{2}$ degrees) of singularity configuration on the sphere, a picture from [2]

2 | Singularity configurations

The number and valance of singularities of the cross-field are under the topological constraints of the Poincaré-Hopf theorem. More specifically, for the set of singularities $\mathcal{S} = \bigcup S_k$, $k = 1, \dots, N$ and a surface Ω , this theorem states:

$$\sum_{j=1}^N \frac{4 - \text{valence}(S_k)}{4} = \chi(\Omega), \quad (7.1)$$

where $\chi(\Omega)$ is the Euler characteristic of Ω .

It can also be shown that singularities represent the irregular vertices of the quad layout, subsequently a quad mesh [44]. Due to this intimate relationship, the generation of a quad layout and finally a quad mesh are constrained in the same manner.

When it comes to naturally obtained singularity configuration, their locations and valences are in our algorithm determined by following the standard procedure, *i.e.*, computing the winding number and the number of integral lines emanating from each of them (see details in [61, 63]). On the other hand, our method as well endorses the user-imposed singularity pattern, *i.e.*, the case when their number, locations and valences are imposed by the user. In practice, this procedure, simply consists of encoding valence data for vertices of the initial mesh and ensuring mesh refinement around them, as illustrated in figure 7.7. Nevertheless, not all user-imposed singularity configurations fulfilling the equation (7.1) will be valid for quad meshing. The central cause for this lies in the fact that a combination of choices of valences and holonomy (turning numbers) is not arbitrary [93]. Indeed, the topologically correct singularity configuration respecting the equation (7.1) represents a *necessary* condition, but not the *sufficient* one for reaching the aim of generating a conformal quadrilateral mesh.

The topic of the non-existence of conformal quandragulations has been explored for some time. The most exciting findings [67, 93, 94, 95, 96], are related to proving that there exists no conformal quad mesh on a torus with only one valence 3 and one valence 5 singular vertices, figure 7.5! Interestingly enough, it can be shown that this is not the case with imposing a singularity doublet with 2 – 6 valences on the same surface or two valence 3 and two valence 5 singularities, figure 7.6. Recently, authors of [67] presented for the first time the *Abel-Jacobi conditions* which are taking into the account both topological correctness of both imposed singularity configuration and holonomy for closed 2D manifolds.

It is important to stress out that our algorithm yields representative results in cases when a singularity pattern is obtained naturally, which was not previously the case with all imposed singularity sets. As shown in the previous Chapter 6, competitive results in the latter case are typically constrained on working on surfaces that are simply connected. Now, following [67], it is possible to determine and impose the adequate singularity configuration, and therefore, our method could be used on a much broader set of models and singularity configurations.

The quality of a conformal quad mesh does not depend only on obtaining the adequate singularity configuration. Considering that the large gradients of cross directions are typically not well accommodated on a coarse mesh around

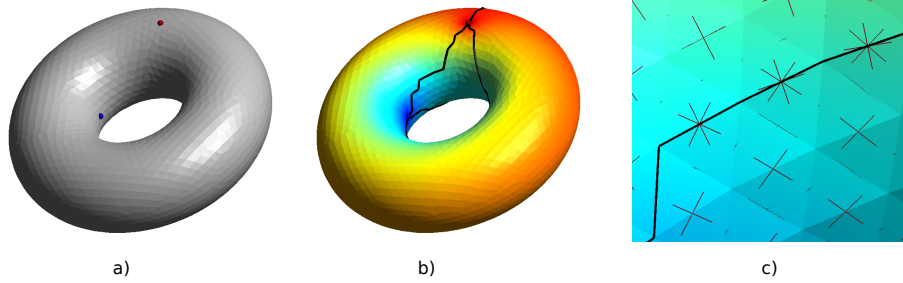


Figure 7.5: Imposing 3 – 5 set of singularities on the torus a), a cut graph in black b) and cross-field inconsistencies along the cut graph c) illustrating that singularity configuration is not valid for quad meshing

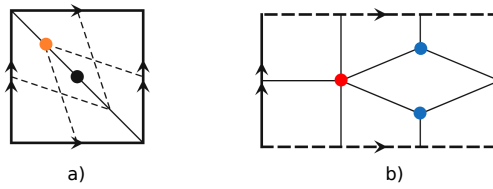


Figure 7.6: Illustrations of conformal quadtangulations on the torus with 2–6 valences (in orange and black) under a) and two valence 3 and two valence 5 (in blue and red) singularities under b) (a picture from [96])

singularities, the optimal results may not be reached. To surpass this challenge, our block-structured quad mesher imposes a special mesh refinement around singular points (both naturally obtained or user-imposed), as shown in figure 7.7. In this manner, it is possible to compute all integral lines emanating from singularities and therefore obtain a genuine network of separatrices.

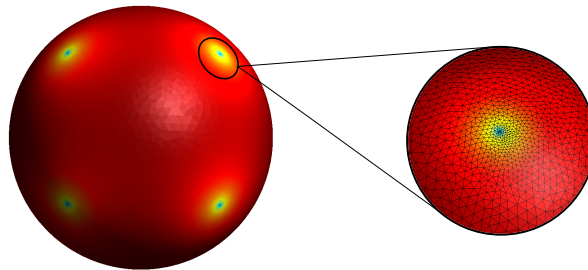


Figure 7.7: Mesh refinement around singularity of valence 3

3 | Cross-field computation on prescribed singularity configuration

We define a 2D cross $\mathbf{c}$ as a set of 2 unit coplanar orthogonal vectors and their opposite, *i.e.*, $\mathbf{c} = \{\mathbf{u}, \mathbf{v}, -\mathbf{u}, -\mathbf{v}\}$ with $\mathbf{u} \cdot \mathbf{v} = 0$, $|\mathbf{u}| = |\mathbf{v}| = 1$ and $\mathbf{u}, \mathbf{v}$ are coplanar. These vectors are called cross' branches.

A 2D cross-field on a 2D manifold $\mathcal{M}$, now, is a map $\mathcal{C}_{\mathcal{M}} : \mathbf{X} \in \mathcal{M} \rightarrow \mathbf{c}(\mathbf{X})$, and the standard approach to compute a smooth boundary-aligned cross-field is to minimize the Dirichlet energy:

$$\min_{\mathcal{C}_{\mathcal{M}}} \int_{\mathcal{M}} \|\nabla \mathcal{C}_{\mathcal{M}}\|^2 \quad (7.2)$$

subject to the boundary condition $\mathbf{c}(\mathbf{X}) = \mathbf{g}(\mathbf{X})$ on $\partial\mathcal{M}$, where $\mathbf{g}$ is a given function.

The classical boundary condition for cross-field computation is that $\forall \mathbf{P} \in \partial\mathcal{M}$, with $\mathbf{T}(\mathbf{P})$ a unit tangent vector to $\mathcal{M}$ at $\mathbf{P}$, one branch of $\mathbf{c}(\mathbf{P})$ has to be colinear to $\mathbf{T}(\mathbf{P})$. In the general case, there exists no smooth cross-field matching this boundary condition. The cross-field will present a finite number of singularities S_j , located at $\mathbf{X}_j$ and of index k_j . We define a singularity pattern as the set $\mathcal{S} = \{S_j, j \in [1, N], N \in \mathbb{Z}\}$.

In this section a method to compute a cross-field $\mathcal{C}_{\mathcal{M}}$ matching a given singularity pattern $\mathcal{S}$ is developed. In other words, we are looking for $\mathcal{C}_{\mathcal{M}}$ such as:

$$\left\{ \begin{array}{l} \cdot \text{ if } \mathbf{X} \text{ belongs to } \partial\mathcal{M}, \text{ at least one branch of } \mathcal{C}_{\mathcal{M}} \text{ is tangent to } \partial\mathcal{M}, \\ \cdot \text{ singularities of } \mathcal{C}_{\mathcal{M}} \text{ are matching the given singularity pattern } \mathcal{S} \\ \quad \text{(same location and indices)}. \end{array} \right. \quad (7.3)$$

Before developing the method to compute such a cross-field, a few operators on 2D manifold have to be defined.

3.1 Curvature and Levi-Civita connection on the 2D manifold

Let E^3 be the Euclidean space equipped with a Cartesian coordinates system $\{x^i, i = 1, 2, 3\}$, and $\mathcal{M}$ the 2D domain to be meshed. We are assuming $\mathcal{M}$ to be an oriented two-dimensional manifold embedded in E^3 . We note $\mathbf{n}(\mathbf{X})$ the unit normal to $\mathcal{M}$ at $\mathbf{X} \in \mathcal{M}$. It is assumed that the normal field $\mathbf{n}$ is smooth and that the Gaussian curvature K is defined and smoothed on $\mathcal{M}$.

If $\gamma(s)$ is a curve on $\mathcal{M}$ parametrized by arc length, the Darboux frame is the orthonormal frame defined by

$$\mathbf{T}(s) = \gamma'(s) \quad (7.4)$$

$$\mathbf{n}(s) = \mathbf{n}(\gamma(s)) \quad (7.5)$$

$$\mathbf{t}(s) = \mathbf{n}(s) \times \mathbf{T}(s) \quad (7.6)$$

One then has the differential structure

$$d \begin{pmatrix} \mathbf{T} \\ \mathbf{t} \\ \mathbf{n} \end{pmatrix} = \begin{pmatrix} 0 & \kappa_g & \kappa_n \\ -\kappa_g & 0 & \tau_r \\ -\kappa_n & -\tau_r & 0 \end{pmatrix} \begin{pmatrix} \mathbf{T} \\ \mathbf{t} \\ \mathbf{n} \end{pmatrix} ds \quad (7.7)$$

where κ_g is the geodesic curvature of the curve, κ_n the normal curvature of the curve, and τ_r the relative torsion of the curve. $\mathbf{T}$ is the unit tangent, $\mathbf{t}$ the tangent normal and $\mathbf{n}$ the unit normal.

Arbitrary vector fields V and $W \in E^3$ can be expressed as

$$\mathbf{V} = V^i \mathbf{E}_i, \quad \mathbf{W} = W^i \mathbf{E}_i$$

in the natural basis vectors $\{\mathbf{E}_i, i = 1, 2, 3\}$ of this coordinate system, and we shall note

$$\langle \mathbf{V}, \mathbf{W} \rangle = V^i W^j \delta_{ij}, \quad \|\mathbf{V}\| = \sqrt{\langle \mathbf{V}, \mathbf{V} \rangle}$$

the Euclidean metric and the associated norm for vectors. The Levi-Civita connection on E^3 in Cartesian coordinates is trivial (all Christoffel symbols vanish), and one has

$$\nabla_{\mathbf{V}}^E \mathbf{W} = (\nabla_{\mathbf{V}} W^i) \mathbf{E}_i.$$

The Levi-Civita connection on the Riemannian submanifold $\mathcal{M}$, now, is not a trivial one. It is the orthogonal projection of $\nabla_{\mathbf{V}}^E$ in the tangent bundle $T\mathcal{M}$, so that one has

$$\nabla_{\mathbf{V}} \mathbf{W} = P_{T\mathcal{M}}[\nabla_{\mathbf{V}}^E \mathbf{W}] = (\nabla_{\mathbf{V}} W^i) P_{T\mathcal{M}}[\mathbf{E}_i] \quad (7.8)$$

where $P_{T\mathcal{M}} : E^3 \mapsto T\mathcal{M}$ is the orthogonal projection operator on $T\mathcal{M}$.

3.2 Conformal mapping

We are looking for a conformal mapping

$$\begin{aligned} \mathcal{F} : \quad \mathcal{P} &\rightarrow \mathcal{M} \subset E^3 \\ \mathbf{P} = (\xi, \eta) &\mapsto \mathbf{X} = (x^1, x^2, x^3) \end{aligned} \quad (7.9)$$

where $\mathcal{P}$ is a parametric space. As finding $\mathcal{F}$ right away is a difficult problem, one focus instead on finding the 3×2 jacobian matrix of $\mathcal{F}$

$$J(\mathbf{P}) = (\partial_\xi \mathcal{F}(\mathbf{P}), \partial_\eta \mathcal{F}(\mathbf{P})) \equiv (\tilde{\mathbf{u}}(\mathbf{P}), \tilde{\mathbf{v}}(\mathbf{P})), \quad (7.10)$$

where $\tilde{\mathbf{u}}, \tilde{\mathbf{v}} \in T\mathcal{M}$ are the columns vectors of J . The mapping $\mathcal{F}$ being conformal, the columns of $J(\mathbf{P})$ have same norm $L(\mathbf{P}) \equiv \|\tilde{\mathbf{u}}(\mathbf{P})\| = \|\tilde{\mathbf{v}}(\mathbf{P})\|$ and are orthogonal to each other, $\tilde{\mathbf{u}}(\mathbf{P}) \cdot \tilde{\mathbf{v}}(\mathbf{P}) = 0$, so that one can write

$$J = L(\mathbf{u}, \mathbf{v}) \quad , \quad \mathbf{n} = \mathbf{u} \wedge \mathbf{v}$$

where $(\mathbf{u}, \mathbf{v}, \mathbf{n})$ forms a set of 3 orthonormal basis vectors. A triplet $(\mathbf{u}, \mathbf{v}, \mathbf{n})$ can be represented by means of the Euler angles (ψ, γ, ϕ) and with the shorthands $s_\phi \equiv \sin \phi$ and $c_\phi \equiv \cos \phi$, as

$$\mathbf{u} = \begin{pmatrix} -s_\phi s_\psi c_\gamma + c_\phi c_\psi \\ s_\phi c_\psi c_\gamma + s_\psi c_\phi \\ s_\phi s_\gamma \end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix} -s_\phi c_\psi - s_\psi c_\phi c_\gamma \\ -s_\phi s_\psi + c_\phi c_\psi c_\gamma \\ s_\gamma c_\phi \end{pmatrix}, \quad \mathbf{n} = \begin{pmatrix} s_\psi s_\gamma \\ -s_\gamma c_\psi \\ c_\gamma \end{pmatrix} \quad (7.11)$$

in the vector basis of E^3 . The projection operator $P_{T\mathcal{M}}$ introduced in (7.8) then simply amounts in disregarding the component along $\mathbf{n}$ of vectors.

For a vector field $\mathbf{w}$ defined on $\mathcal{M}$, one can write by derivation of (7.11)

$$\begin{aligned} \nabla_{\mathbf{w}}^E \mathbf{u} &= \mathbf{v} \nabla_{\mathbf{w}} \phi + s_\phi \mathbf{n} \nabla_{\mathbf{w}} \gamma + (c_\gamma \mathbf{v} - s_\gamma c_\phi \mathbf{n}) \nabla_{\mathbf{w}} \Psi \\ \nabla_{\mathbf{w}}^E \mathbf{v} &= -\mathbf{u} \nabla_{\mathbf{w}} \phi + c_\phi \mathbf{n} \nabla_{\mathbf{w}} \gamma + (-c_\gamma \mathbf{u} + s_\gamma s_\phi \mathbf{n}) \nabla_{\mathbf{w}} \Psi \\ \nabla_{\mathbf{w}}^E \mathbf{n} &= -(s_\phi \mathbf{u} + c_\phi \mathbf{v}) \nabla_{\mathbf{w}} \gamma + s_\gamma (c_\phi \mathbf{u} - s_\phi \mathbf{v}) \nabla_{\mathbf{w}} \Psi \end{aligned}$$

and hence, using (7.8), the expression of the covariant derivatives on the sub-manifold $\mathcal{M}$

$$\begin{aligned} \nabla_{\mathbf{w}} \mathbf{u} &= \mathbf{v} \nabla_{\mathbf{w}} \phi + c_\gamma \mathbf{v} \nabla_{\mathbf{w}} \Psi \\ \nabla_{\mathbf{w}} \mathbf{v} &= -\mathbf{u} \nabla_{\mathbf{w}} \phi - c_\gamma \mathbf{u} \nabla_{\mathbf{w}} \Psi. \end{aligned}$$

This allows writing the Lie bracket

$$\begin{aligned} [\mathbf{u}, \mathbf{v}] &= \nabla_{\mathbf{u}} \mathbf{v} - \nabla_{\mathbf{v}} \mathbf{u} \\ &= -(\mathbf{u} \nabla_{\mathbf{u}} \phi + \mathbf{v} \nabla_{\mathbf{v}} \phi) - c_\gamma (\mathbf{u} \nabla_{\mathbf{u}} \Psi + \mathbf{v} \nabla_{\mathbf{v}} \Psi), \end{aligned} \quad (7.12)$$

which will be used in the upcoming section.

It is important to note that $\mathbf{u}$ and $\mathbf{v}$ are 2 branches of the cross-field $\mathcal{C}_{\mathcal{M}}$ we are looking for.

3.3 Integrability condition

The mapping $\mathcal{F}$, now, defines a conformal parametrization of $\mathcal{M}$ if the columns of J commute as vector fields, *i.e.*, if the differential condition

$$0 = [\tilde{\mathbf{u}}, \tilde{\mathbf{v}}] = \nabla_{\tilde{\mathbf{u}}} \tilde{\mathbf{v}} - \nabla_{\tilde{\mathbf{v}}} \tilde{\mathbf{u}} = [L\mathbf{u}, L\mathbf{v}] \quad (7.13)$$

is verified. Developing the latter expression and posing for convenience $L = e^H$ gives

$$0 = \mathbf{v} \nabla_{\mathbf{u}} H - \mathbf{u} \nabla_{\mathbf{v}} H + [\mathbf{u}, \mathbf{v}],$$

and then

$$\begin{cases} \nabla_{\mathbf{u}} H &= -\langle \mathbf{v}, [\mathbf{u}, \mathbf{v}] \rangle \\ \nabla_{\mathbf{v}} H &= \langle \mathbf{u}, [\mathbf{u}, \mathbf{v}] \rangle \end{cases} \quad (7.14)$$

which after substitution of (7.12)

$$\begin{cases} \nabla_{\mathbf{u}} H &= \nabla_{\mathbf{v}} \phi + c_{\gamma} \nabla_{\mathbf{v}} \psi \\ -\nabla_{\mathbf{v}} H &= \nabla_{\mathbf{u}} \phi + c_{\gamma} \nabla_{\mathbf{u}} \psi \end{cases} \quad (7.15)$$

In order to obtain the boundary value problem for H , partial differential equation (*PDE*) governing H have to be expressed on $\partial\mathcal{M}$ as well as on the interior of $\mathcal{M}$ without Euler angles (ψ, θ, ϕ) which are unknown.

3.4 H PDE on the boundary

As the boundary $\partial\mathcal{M}$ is represented by curves on $\mathcal{M}$, it is possible to parametrize them by arc length and thus associate for each $\mathbf{X} \in \partial\mathcal{M}$ a Darboux frame $(\mathbf{T}(\mathbf{X}), \mathbf{t}(\mathbf{X}), \mathbf{n}(\mathbf{X}))$. As we are looking for a cross-field $\mathcal{C}_{\mathcal{M}}$ fulfilling conditions 7.3, the triplet $(\mathbf{u}, \mathbf{v}, \mathbf{n})$ can be identified as $(\mathbf{T}(\mathbf{X}), \mathbf{t}(\mathbf{X}), \mathbf{n}(\mathbf{X}))$. One then has:

$$\partial_s T = \kappa_g \mathbf{t} + \kappa_n \mathbf{n} \equiv \nabla_{\mathbf{u}} \mathbf{u} = \mathbf{v} \nabla_{\mathbf{u}} \phi + s_{\phi} \mathbf{n} \nabla_{\mathbf{u}} \gamma + (c_{\gamma} \mathbf{v} - s_{\gamma} c_{\phi} \mathbf{n}) \nabla_{\mathbf{u}} \psi$$

where from follows

$$\begin{cases} \kappa_g &= \nabla_{\mathbf{u}} \phi + c_{\gamma} \nabla_{\mathbf{u}} \psi \\ \kappa_n &= s_{\phi} \nabla_{\mathbf{u}} \gamma - s_{\gamma} c_{\phi} \nabla_{\mathbf{u}} \psi \end{cases} \quad (7.16)$$

Using 7.15 it becomes:

$$\nabla_{\mathbf{t}} H = -\kappa_g \quad (7.17)$$

This is a comforting result as one ends up with the boundary condition that matches exactly the one, $-\mathbf{t} \cdot \nabla H = \kappa_g$, found in the planar case [61].

3.5 H PDE in smooth region on the interior of M

To find the PDE governing H , let's assume the jacobian J is smooth (and therefore H) in a vicinity $\mathcal{V}$ of $\mathbf{X} \in M$.

We choose $\mathcal{U} \subset \mathcal{V}$ such as $\mathbf{X} \in \mathcal{U}$, $\partial\mathcal{U}$ such as unit tangent vector $\mathbf{T}_0$ to $\partial\mathcal{U}_0$ verifies $\mathbf{T}_0 = \mathbf{v}$, $\mathbf{T}_1$ to $\partial\mathcal{U}_1$ verifies $\mathbf{T}_1 = \mathbf{u}$, $\mathbf{T}_2$ to $\partial\mathcal{U}_2$ verifies $\mathbf{T}_2 = -\mathbf{v}$, $\mathbf{T}_3$ to $\partial\mathcal{U}_3$ verifies $\mathbf{T}_3 = -\mathbf{u}$.

Thus we have a submanifold $\mathcal{U} \subset M$ on which h is smooth, and such as $\partial\mathcal{U} = \partial\mathcal{U}_0 \cup \partial\mathcal{U}_1 \cup \partial\mathcal{U}_2 \cup \partial\mathcal{U}_3$. Darboux frames of $\partial\mathcal{U}$ are (figure 7.8):

$$\left\{ \begin{array}{ll} (\mathbf{T}, \mathbf{t}, \mathbf{n}) = (\mathbf{v}, -\mathbf{u}, \mathbf{n}) & \text{on } \partial\mathcal{U}_0 \\ (\mathbf{T}, \mathbf{t}, \mathbf{n}) = (-\mathbf{u}, -\mathbf{v}, \mathbf{n}) & \text{on } \partial\mathcal{U}_1 \\ (\mathbf{T}, \mathbf{t}, \mathbf{n}) = (-\mathbf{v}, \mathbf{u}, \mathbf{n}) & \text{on } \partial\mathcal{U}_2 \\ (\mathbf{T}, \mathbf{t}, \mathbf{n}) = (\mathbf{u}, \mathbf{v}, \mathbf{n}) & \text{on } \partial\mathcal{U}_3 \end{array} \right. \quad (7.18)$$

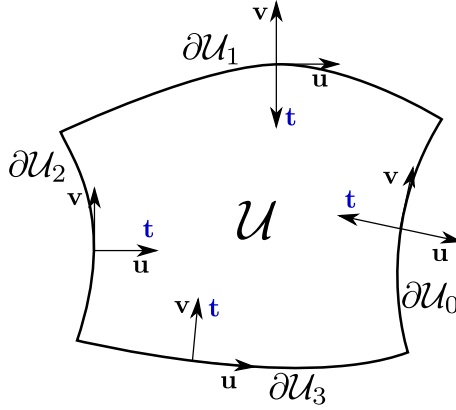


Figure 7.8: Vicinity of $\mathbf{X}$ considered.

In order for $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}})$ to be a local coordinate system, we recall equation (7.17) demonstrated in section 3.4:

$$\kappa_g = -\nabla_{\mathbf{t}} H, \text{ with } \mathbf{t} = \mathbf{n} \wedge \mathbf{T} \quad (7.19)$$

and the divergence theorem stating that:

$$\int_{\partial\mathcal{U}} \nabla_{\mathbf{t}} H = - \int_{\mathcal{U}} \Delta H. \quad (7.20)$$

Applying the Gauss-Bonnet theorem on $\mathcal{U}$ leads to:

$$\int_{\mathcal{U}} K \, d\mathcal{U} + \int_{\partial\mathcal{U}} \kappa_g \, dl + 4 \frac{\pi}{2} = 2\pi\chi(\mathcal{U})$$

where K and $\chi(\mathcal{U})$ are respectively the Gaussian curvature and Euler characteristic of $\mathcal{U}$. As $\chi(\mathcal{U}) = 1$ and using equations (7.19) and (7.20), it becomes:

$$\int_{\mathcal{U}} K \, d\mathcal{U} = - \int_{\mathcal{U}} \Delta H \, d\mathcal{U} \quad (7.21)$$

which holds for any chosen $\mathcal{U}$. Hence, there is:

$$\Delta H = -K, \text{ if } J \text{ is smooth.} \quad (7.22)$$

For general cases, it is impossible for J to be smooth everywhere. Indeed, let's assume $\mathcal{M}$ to be with smooth boundary $\partial\mathcal{M}$ (*i.e.* with no corners) and of Euler characteristic $\chi(\mathcal{M}) = 1$. If we assume J is smooth everywhere, it becomes:

$$\begin{cases} \int_{\mathcal{M}} K \, d\mathcal{M} + \int_{\partial\mathcal{M}} \kappa_g \, dl &= 0 \\ 2\pi\chi(\mathcal{M}) &= 2\pi \end{cases} \quad (7.23)$$

which is not in accordance with the Gauss-Bonnet theorem. Therefore, J has to be singular somewhere in $\mathcal{M}$.

The goal being the building of a usable parametrization of $\mathcal{M}$, *i.e.*, being able to use this parametrization to build a quad mesh of $\mathcal{M}$, we will allow J to be singular on a finite number N of points $\mathbf{S}_j$, $j \in [0, N-1]$. We will show that this condition is sufficient for this problem always to have a unique solution.

3.6 H PDE at singular points

For now, we know boundary conditions for H (equation (7.17)) and local equation in smooth regions (equation (7.22)). The only thing left is to determine a local PDE governing H at singular points $\{\mathbf{S}_j\}$. We define k_j the index of singularity $\mathbf{S}_j$.

For this, we are making two reasonable assumptions:

$$\begin{cases} \Delta H(\mathbf{S}_j) &= -K(\mathbf{S}_j) + \alpha_j \delta(\mathbf{S}_j) & \text{where } \alpha_j \text{ is a constant} \\ & & \text{and } \delta \text{ is Dirac distribution} \\ k_i &= k_j \Rightarrow \alpha_i = \alpha_j. \end{cases} \quad (7.24)$$

We consider the disk $\mathcal{M}$ represented in figure (7.9) with 4 singularities $\mathbf{S}_j, j \in [0, 3]$ of index $k_j = 1$.

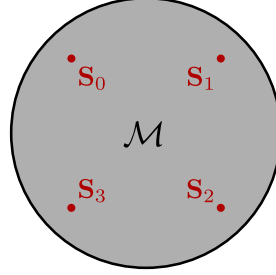


Figure 7.9: The disk with 4 singularities of valence 3

The Gauss-Bonnet theorem states that:

$$\int_{\mathcal{M}} K \, d\mathcal{M} + \int_{\partial\mathcal{M}} \kappa_g \, dl = 2\pi\chi(\mathcal{M}).$$

Replacing K and κ_g by their values in equations (7.17) and (7.22), and using hypothesis (7.24) we have:

$$\begin{aligned} \int_{\mathcal{M}} -\Delta H \, d\mathcal{M} + \sum_i \alpha_i \delta(\mathbf{S}_i) \, d\mathcal{M} + \int_{\partial\mathcal{M}} -\nabla_{\mathbf{t}} H \, dl &= 2\pi\chi(\mathcal{M}) \\ \Rightarrow \int_{\mathcal{M}} -\Delta H \, d\mathcal{M} - \int_{\partial\mathcal{M}} \nabla_{\mathbf{t}} H \, dl + \alpha \int_{\mathcal{M}} \sum_i \delta(\mathbf{S}_i) \, d\mathcal{M} &= 2\pi \\ \Rightarrow \alpha &= 2\pi \frac{1}{4} \end{aligned}$$

For singularity of index 1 we have:

$$\Delta H(\mathbf{S}_j) = -K(\mathbf{S}_j) + 2\pi \frac{1}{4} \delta(\mathbf{S}_j).$$

Using the same idea, we can generalize the following:

$$\Delta H(\mathbf{S}_j) = -K(\mathbf{S}_j) + 2\pi \frac{k_j}{4} \delta(\mathbf{S}_j). \quad (7.25)$$

3.7 Boundary value problem for H

To sum up, equations governing H on $\mathcal{M}$ are:

$$\begin{cases} \Delta H &= -K + 2\pi \frac{k_j}{4} \delta(\mathbf{S}_j) & \text{on } \mathcal{M} \\ \nabla_{\mathbf{t}} H &= -\kappa_g & \text{on } \partial\mathcal{M} \end{cases} \quad (7.26)$$

This problem is well-posed and admits a unique solution to an arbitrary additive constant. A triangulation $\mathcal{M}_T$ of the manifold $\mathcal{M}$ is generated and problem 7.26 is solved using a finite element formulation with order 1 Lagrange elements. Once H is determined, the next step is to retrieve J orientation,

detailed in the next section. The fact that H is only known up to an additive constant is not harmful as only ∇H will be needed to retrieve J orientation.

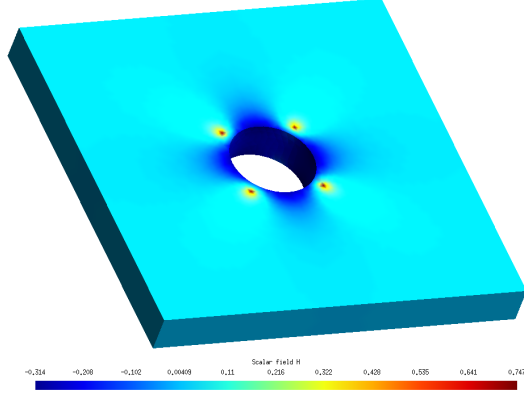


Figure 7.10: H function obtained on a closed manifold.

3.8 Retrieving crosses orientation from H

In order to get an orientation at a given point $\mathbf{X} \in \mathcal{M}$, a local reference basis in $\mathbf{X}$ is needed.

Let's define arbitrary local basis $(\mathbf{u}_\mathbf{X}, \mathbf{v}_\mathbf{X}, \mathbf{n})$ for every $\mathbf{X} \in \mathcal{M}$ such as euler angles (ψ, γ, ϕ) associated are $\mathcal{C}^1$ on $\mathcal{M}$. Orientation θ is defined as $\mathbf{u} = \mathcal{R}_{\theta, \mathbf{n}}(\mathbf{u}_\mathbf{X})$ and $\mathbf{v} = \mathcal{R}_{\theta, \mathbf{n}}(\mathbf{v}_\mathbf{X})$. Equation (7.14) impose that:

$$\begin{cases} \nabla_{\mathbf{u}} H &= \nabla_{\mathbf{v}}(\phi + \theta) + c_\gamma \nabla_{\mathbf{v}} \psi \\ \nabla_{\mathbf{v}} H &= -\nabla_{\mathbf{u}}(\phi + \theta) - c_\gamma \nabla_{\mathbf{u}} \psi \end{cases} \quad (7.27)$$

which is equivalent to:

$$\begin{cases} \nabla_{\mathbf{u}_\mathbf{X}} H &= \nabla_{\mathbf{v}_\mathbf{X}}(\phi + \theta) + c_\gamma \nabla_{\mathbf{v}_\mathbf{X}} \psi \\ \nabla_{\mathbf{v}_\mathbf{X}} H &= -\nabla_{\mathbf{u}_\mathbf{X}}(\phi + \theta) - c_\gamma \nabla_{\mathbf{u}_\mathbf{X}} \psi \end{cases} \quad (7.28)$$

and eventually gives:

$$\begin{cases} \nabla_{\mathbf{u}_\mathbf{X}} \theta &= -\nabla_{\mathbf{v}_\mathbf{X}} H - \nabla_{\mathbf{u}_\mathbf{X}} \phi - c_\gamma \nabla_{\mathbf{u}_\mathbf{X}} \psi \\ \nabla_{\mathbf{v}_\mathbf{X}} \theta &= \nabla_{\mathbf{u}_\mathbf{X}} H - \nabla_{\mathbf{v}_\mathbf{X}} \phi - c_\gamma \nabla_{\mathbf{v}_\mathbf{X}} \psi \end{cases} \quad (7.29)$$

which is linear in θ .

We can show that there always exists a scalar field θ verifying (7.29). θ exists if and only if we have:

$$\nabla_{\mathbf{u}_\mathbf{X}} \nabla_{\mathbf{v}_\mathbf{X}} \theta - \nabla_{\mathbf{v}_\mathbf{X}} \nabla_{\mathbf{u}_\mathbf{X}} \theta = 0 \quad (7.30)$$

Using equations 7.29 we obtain:

$$\begin{aligned}
& \nabla_{\mathbf{u}_X} \nabla_{\mathbf{v}_X} \theta - \nabla_{\mathbf{v}_X} \nabla_{\mathbf{u}_X} \theta \\
&= \Delta H + \nabla_{\mathbf{v}_X} (c_\gamma \nabla_{\mathbf{u}_X} \psi) - \nabla_{\mathbf{u}_X} (c_\gamma \nabla_{\mathbf{v}_X} \psi) \\
&= -K + \nabla_{\mathbf{v}_X} (c_\gamma \nabla_{\mathbf{u}_X} \psi) - \nabla_{\mathbf{u}_X} (c_\gamma \nabla_{\mathbf{v}_X} \psi)
\end{aligned} \tag{7.31}$$

We know that, for 2D manifolds embedded in $\mathbb{R}^3$, the Gaussian curvature K is equal to the jacobian of the Gauss map of the manifold [138]. We have:

$$\begin{cases} \nabla_{\mathbf{u}_X} \mathbf{n} &= s_\gamma \nabla_{\mathbf{u}_X} \psi \begin{pmatrix} c_\psi \\ s_\psi \\ 0 \end{pmatrix} - \nabla_{\mathbf{u}_X} \gamma \begin{pmatrix} -s_\psi c_\gamma \\ c_\psi c_\gamma \\ s_\gamma \end{pmatrix} \\ \nabla_{\mathbf{v}_X} \mathbf{n} &= s_\gamma \nabla_{\mathbf{v}_X} \psi \begin{pmatrix} c_\psi \\ s_\psi \\ 0 \end{pmatrix} - \nabla_{\mathbf{v}_X} \gamma \begin{pmatrix} -s_\psi c_\gamma \\ c_\psi c_\gamma \\ s_\gamma \end{pmatrix} \end{cases} \tag{7.32}$$

Therefore we also have:

$$K = s_\gamma (\nabla_{\mathbf{v}_X} \psi \nabla_{\mathbf{u}_X} \gamma - \nabla_{\mathbf{u}_X} \psi \nabla_{\mathbf{v}_X} \gamma) \tag{7.33}$$

Using equation (7.33) in equation (7.31) we get:

$$\begin{aligned}
& -K + \nabla_{\mathbf{v}_X} (c_\gamma \nabla_{\mathbf{u}_X} \psi) - \nabla_{\mathbf{u}_X} (c_\gamma \nabla_{\mathbf{v}_X} \psi) \\
&= -K + c_\gamma \nabla_{\mathbf{v}_X} \nabla_{\mathbf{u}_X} \psi - s_\gamma \nabla_{\mathbf{v}_X} \gamma \nabla_{\mathbf{u}_X} \psi - c_\gamma \nabla_{\mathbf{u}_X} \nabla_{\mathbf{v}_X} \psi + s_\gamma \nabla_{\mathbf{u}_X} \gamma \nabla_{\mathbf{v}_X} \psi \\
&= 0
\end{aligned} \tag{7.34}$$

As equation (7.30) is verified, we know that there exists a scalar field θ verifying the equation (7.29), and therefore that our problem has a solution which is unique.

In order to solve equation (7.29), it is needed to define a smooth global basis $(\mathbf{u}_X, \mathbf{v}_X, \mathbf{n})$ on $\mathcal{M}$. This is possible by generating a branch cut $\mathcal{L}$, as defined below, and computing a smooth global basis $(\mathbf{u}_X, \mathbf{v}_X, \mathbf{n})$ on $\mathcal{M}$ allowing discontinuities across $\mathcal{L}$.

A branch cut is a set $\mathcal{L}$ of curves of a domain M that do not form any closed loop and that cut the domain in such a way that it is impossible to find any closed loop in $M \setminus \mathcal{L}$ that encloses one or several singularities, or an internal boundary. As we already have a triangulation of M , the branch cut $\mathcal{L}$ is in practice simply a set of edges of the triangulation.

The branch cut is generated with the method described in [61] which is based on [73]. An example of generated branch cut is presented in figure 7.11.

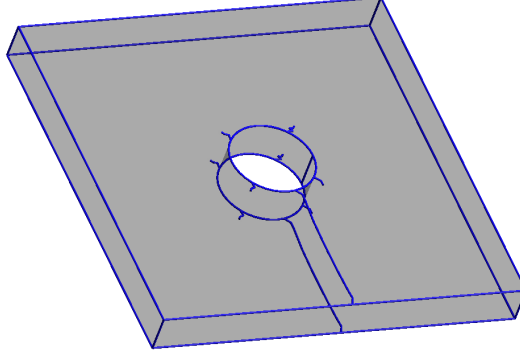


Figure 7.11: Edges of the branch cut $\mathcal{L}$ are represented in blue. There exists no closed loop in $M \setminus \mathcal{L}$ enclosing one or several singularities.

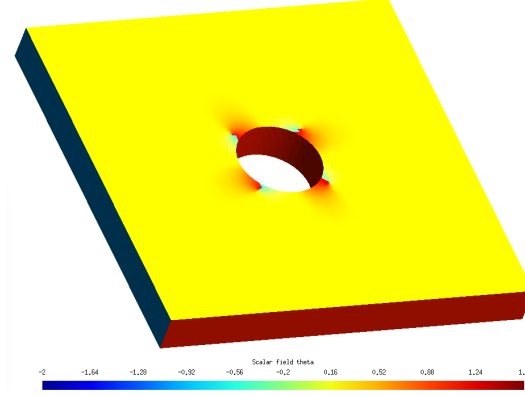
Once a branch cut $\mathcal{L}$ is available, the field θ can be computed by solving the linear equations (7.29). With equations 7.29, θ is known up to an additive constant. For the problem to be well-posed, θ value has to be imposed at one point of domain $\mathcal{M}$. The chosen boundary condition consists in fixing the angle θ at *one* arbitrary point $\mathbf{X}_{BC} \in \partial\mathcal{M}$ so that $\mathcal{C}_{\mathcal{M}}(\mathbf{X}_{BC})$ has one of its branches collinear with $\mathbf{T}(\mathbf{X})$. The problem can be rewritten as the well-posed equation (7.35) and is solved using the finite element method on the triangulation $\mathcal{M}_T$ with order one Crouzeix-Raviart elements. This kind of elements has proven to be more efficient for cross-field representation as underlined in [60].

$$\begin{cases} P_{T\mathcal{M}}(\nabla\theta) &= P_{T\mathcal{M}}(\mathbf{n} \times \nabla H - \nabla\phi - c_\gamma \nabla\psi) \text{ in } \mathcal{M} \\ \theta(\mathbf{X}_{BC}) &= \theta_{\mathbf{X}_{BC}} \text{ for an arbitrary } \mathbf{X}_{BC} \in \partial\mathcal{M} \\ \theta &\text{ discontinuous on } \mathcal{L} \end{cases} \quad (7.35)$$

It is important to note that for (equation 7.35) to be well-posed, the θ value can only be imposed on a *single* point. A consequence is that if $\mathcal{M}$ has more than one boundary ($\partial\mathcal{M} = \partial\mathcal{M}_1 \cup \partial\mathcal{M}_2 \cup \dots \cup \partial\mathcal{M}_n$), the resulting cross-field is guaranteed to be tangent to the boundary $\partial\mathcal{M}_i$ such as $\mathbf{X}_{BC} \in \partial\mathcal{M}_i$, which does not necessarily hold for all boundaries $\partial\mathcal{M}_j$ for $j \neq i$, as detailed in section 3.8.1.

Once H and θ scalar fields are computed on $\mathcal{M}$, the cross-field $\mathcal{C}_{\mathcal{M}}$ can be retrieved for all $\mathbf{X} \in \mathcal{M}$:

$$\mathbf{c}(\mathbf{X}) = \{\mathbf{u}_k = \mathcal{R}_{\theta+k\frac{\pi}{2}, \mathbf{n}}(\mathbf{u}_{\mathbf{X}}), k \in \llbracket 0, 3 \rrbracket\} \quad (7.36)$$

Figure 7.12: Scalar field θ obtained from H represented figure 7.10.

3.8.1 Non quad-meshable cross-fields

Problem (7.35) is well-posed. Suppose the imposed singularity pattern respects the condition on Euler's characteristic. In that case, there exists a unique scalar function H (up to an additive constant) verifying equation (7.26) and a unique scalar function θ verifying equation (7.35), which is leading to a unique cross-field solution $\mathcal{C}_M$ for presented a problem.

Nevertheless, it is not proven that the obtained cross-field $\mathcal{C}_M$ is actually respecting the following and mandatory condition:

- if $\mathbf{X}$ belongs to ∂M , at least one branch of $\mathcal{C}_M$ is tangent to ∂M ,

Indeed, the problem (7.35) is well-posed and only ensures this condition for a given $\mathbf{P} \in \partial M$. Imposing additional boundary conditions for θ will make the problem (7.35) ill-posed. Therefore, when domain M is not simply connected, it is possible that obtained cross-field $\mathcal{C}_M$ is not tangent on some parts of ∂M .

This issue arises from the non-existence of holomorphic function $\mathcal{F}$ (equation 6.6) matching all conditions imposed (*i.e.*, singularity pattern and tangency to domain boundaries). For $\mathcal{F}$ to exist, the singularity pattern must fulfill all Abel-Jacobi conditions presented in [67]. Otherwise, the resulting cross-field will not be boundary aligned, hence generating non-quadrilateral partitions or limit cycles.

4 | Quad layout and quad mesh generation

The following section gives a brief description of the computational process and necessary changes of criteria from [61] for the minimal number of separatrices and limit cycles handling on 2-manifolds. Afterward, examples of quad layouts are shown, and a discussion on results and possible improvements is presented.

4.1 Obtaining the genuine network of separatrices

Separatrices represent integral lines emanating from singular points and tracing these curves, in our case, followed the standard numerical Runge-Kutta 2 scheme (more details in [60]). Their network indeed partitions the domain into patches, unfortunately there are no guarantees that those partitions would be suitable for block-structured quad meshing, which is well documented in the literature [19, 21, 55, 59, 63, 67, 93]. The main causes for ill-defined partitions (non-quadrilateral or thin patches) are typically related to not obtaining the *minimal number* of separatrices as well as the appearance of *limit cycles*, *i.e.* separatrices which are not converging to any singularity or a boundary. However, it is possible to establish criterias for obtaining the genuine network of these integral curves and to ultimately repair the ill-defined patches. For these particular matters, our quad layout generation relies on the works [60, 61]. It is important to note here that the above mentioned algorithms deal only with planar domains, which is not the case here. As it will be elucidated in the following, the additional criteria are imposed on the previously developed methods to suit the 2-manifold cases.

The first issue to be tackled here is to adopt all feature lines of the model that are imposed as boundary conditions into the family of existing separatrices, wherefrom the network with the minimal number of these integral curves can be defined. As already established by [60], two (or more) separatrices are *identical* if they have the same beginning and ending as well as the same set of intersected separatrices. On 2-manifold cases, this criteria demands additional conditions. As illustrated in figure 7.13, separatrices γ_a and γ_b should be, by the previous criteria, defined as identical and by doing so, the removal of one of them would be required. Proceeding in this manner would mean obtaining the ill-defined patches, leaving the dramatic consequences upon the quad mesh generation. Motivated by the results from the practice, we impose an additional requirement to former criteria for separatrices to be identical: the necessity of moving towards the same direction, while not looping on itself, illustrations below.

The decreasing of separatrices number in this manner simplifies the connectivity complexity of the obtained quad layout. Nevertheless, there would typically still be many partitions due to the appearance of separatrices that

revolve around the domain, figure 7.15. In order to generate as simple as possible quad layout, dealing with this kind of separatrices' behavior is necessary, and it is detailed in the following section 4.2.

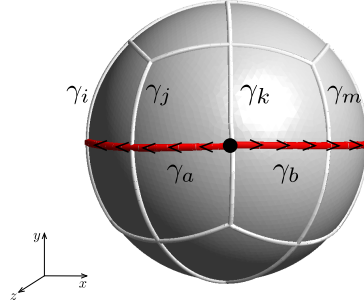


Figure 7.13: The network of separatrices on 2-sphere. Separatrices γ_a and γ_b have the same beginning and ending (feature node marked as a black circle) as well as the same set of intersecting separatrices ($\gamma_i, \gamma_j, \gamma_k, \gamma_m$), yet they are not identical.

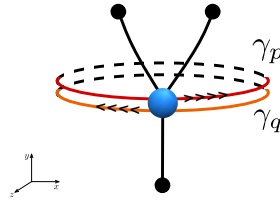


Figure 7.14: Illustration of the importance of loops consideration

4.2 Limit cycles on 2-manifolds

The findings on separatrices behavior from [83] and Poincaré-Bendixson theorem [97] point out the possibility of limit cycles' presence on 2D manifolds. Although the quad mesh generation here relies on integrable cross-field, it is essential to note that it is under the constraint of Abel-Jacobi conditions, as in the planar case (Chapter 6). Therefore, misplacing singularities' locations and numerical inconsistencies can cause separatrix to miss the singularity and loop around the domain, figure 7.15, generating a very fine quad layout that resembles a quad mesh in practice.

Although the algorithm follows the steps from Chapter 6, it is essential to note that the detection of limit cycles and correction scheme for ill-defined

partitions (non-quadrilateral patches or partitions containing t-junction) must be redefined. The reasons for imposing these redefinitions are related to the more complex behavior of separatrices on 2-manifolds than planar domains, especially when it comes to 2-manifolds without boundaries.

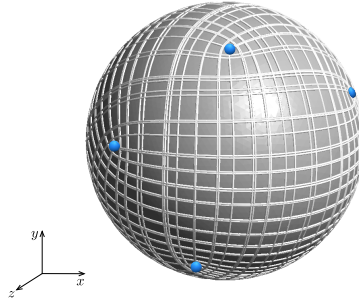


Figure 7.15: Illustration of limit cycles on a 2D manifold

After a careful examination of separatrices behavior on 2-manifold cases, the results from practice indicate that the separatrix is a limit cycle if it respects limit cycle criteria from [61] and it is not a loop (the singularity of its emanating is the same as the one where separatrix ends). Limit cycles defined in this manner can not be stopped at the first orthogonal intersection with another separatrix, as is the case in [61], since it would be impossible to generate a quad layout. Instead, on 2-manifolds cases, limit cycles are stopped at the first intersection with a feature line. Obtained quad layout in this manner is not conformal, and typically ill-defined partitions can not be corrected as in the planar case due to the connectivity complexities of the obtained quad layout.

4.3 Obtaining conformal quad layout

The partitioning of the domain obtained in the previously presented manner typically consists of a finite number of non-quadrilateral patches. Many correction schemes have been developed [139, 140, 141, 142], nevertheless the solution is typically not optimal because many new singularities are introduced on each of the ill-defined partitions. The method recently presented in [143] relies on the algorithm for quadrangulation of N -polygons from [144] and introduces only *one* singularity per N -polygon partition, as presented in figure 7.16, while enabling the generalizations for non-quadrilateral partitions with large N . This method aims in the quad-tessellation of N -polygon, by subdividing all of its edges and applying one step of Catmull-Clark subdivision [145]. The applicability of this approach is constrained by fulfillment of inequalities 7.37, corresponding illustration in figure 7.17. At this point, complex base of obtained

partitioning can be updated so that all quadrilateral conformal partitions are well defined for remeshing.

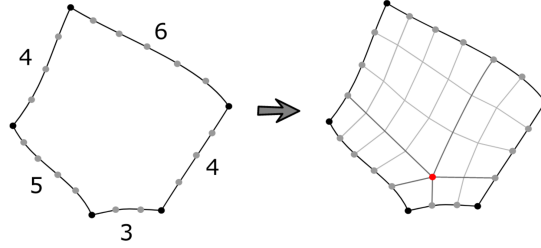


Figure 7.16: An example of quadrangulation of a pentagon, a picture from [144]

$$\begin{aligned}
 N = 3, \quad \forall i, \quad e_i &\leq e_{i+1} + e_{i+2} \\
 N = 5, \quad \forall i, \quad e_i + e_{i+1} + e_{i+4} &\geq e_{i+2} + e_{i+3} \\
 N = 6, \quad \forall i, \quad e_i &\leq e_{i+2} + e_{i+4}.
 \end{aligned} \tag{7.37}$$

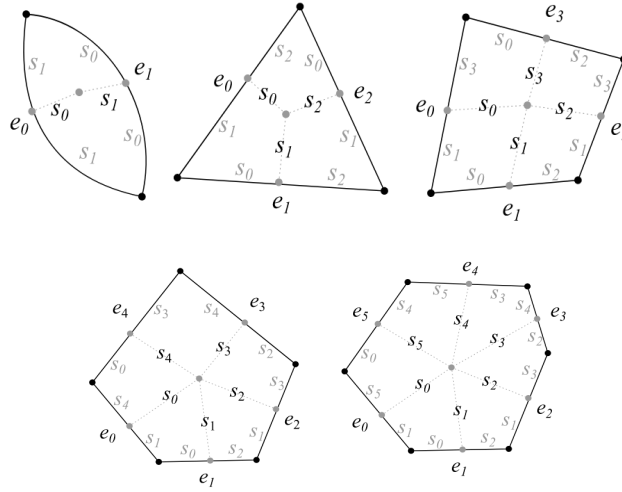


Figure 7.17: Quadrangulations of non-quadrilateral patches, a picture from [144]

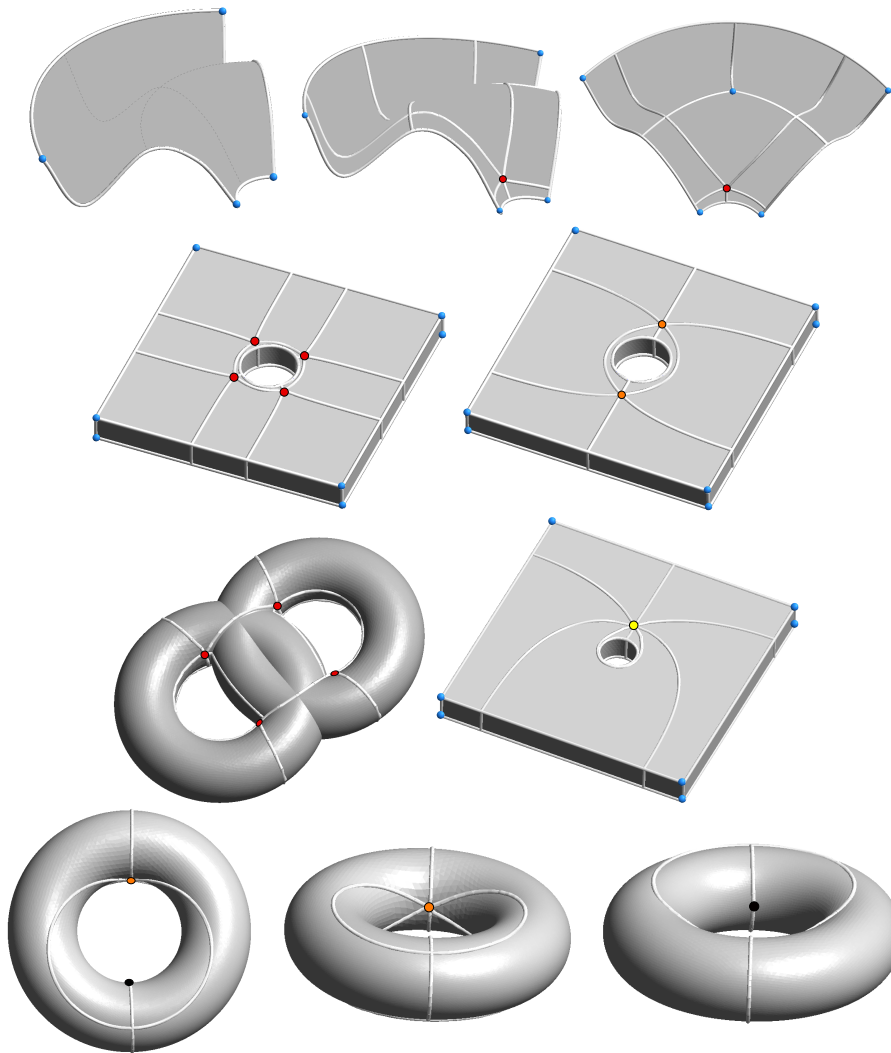


Figure 7.18: Preliminary results. Obtained quad layouts with singularities of valence 2 are marked in black, of valence 3 in blue, of valence 5 in red, of valence 6 in orange and of valence 8 in yellow.

4.4 Quad meshing algorithm

The generating of a final quad mesh relies on the algorithm presented in Chapter 6. More specifically, the layout's partitions are remeshed with bilinear transfinite interpolation (TFI) in the parametric space while respecting the discretizations of the layout's edges implied by the field H . Since there exists a per-partition parametrization (section 3), remeshed partitions are mapped back onto the physical space by assuming that there exists a linear mapping from physical to parametric space. The quad mesh obtained in this manner is isotropic, and depending on the properties of the final quad layout, can be conformal or non-conformal. The meshing time depends on the target number of quads or the approximate length of the edge imposed by the user, and it takes a negligible time compared to the rest of the pipeline.

5 Conclusion

This work proposes a novel algorithm to obtain flexible quad layout and quad mesh generation from naturally or user-imposed singularity configuration. It relies on the extension of the planar concept presented in Chapter 6 for 2D manifolds. The specific changes made upon the original pipeline, as the mathematical foundations and calculus of scalar field H and dealing with challenging separatrices' behavior, are presented. Although the algorithm not necessarily generates a conformal quad layout, a manner to overcome this issue is outlined, allowing the obtaining of conformal isometric quad mesh.

The shown quad layout examples present the proof of concept of the proposed method, and the improving overall robustness and computational time are underway. The future works include the automatic generation of anisotropic quad meshes by computing two independent scalar fields (more details in Chapter 8) and comparing the quality of elements between isotropic and anisotropic versions of the algorithm.

CHAPTER



The generation of integrable cross-field with anisotropic scaling

The integrability condition, along with computing only one scaling field H , imposes the strict constraint on singularity locations and configurations, *i.e.*, the need for fulfilling the *Abel Jacobi theorem* from [67]. In practice, the mispositioning of singularities leads to generating a cross-field that is not aligned with all boundaries, and consequently, to the appearance of limit cycles. In order to bypass this limitation, a new mathematical formulation relying on an integrable cross-field with an anisotropic scaling has been developed. As the initial results demonstrate, this formulation allows the computation of an integrable boundary-aligned cross-field on an imposed set of singularities that do not respect the location's condition from the Abel-Jacobi theorem.

1 | Limit cycle free cross-field formulation

Limit cycles represent a central stepping stone in the field-guided quad meshing algorithms [60, 63], causing the appearance of thin partitions and non-quadrilateral patches. In order to resolve these problems, the associated works typically rely on improving the obtained partitioning of the domain *a posteriori*, using quantization based [42, 55] or quad-tessellation of N-polygon based algorithms [139, 140, 141, 142, 143], which principally leads to not obtaining the optimal solution since many new singularities could be introduced. The work of [77] suggests solving limit cycles issue *a priori* by improving the cross-field quality, *i.e.*, its integrability. Nevertheless, to the best author's knowledge, the limit cycle free cross-field generation represents an open problem.

Motivated by the need for generating a limit cycle free cross-field, a new formulation with an imposed singularity configuration considering its integrability *a priori* is developed, as presented in section 3.1.2. Although this method yields very competitive results in terms of more accurate solution at singularities' vicinity, *i.e.*, finding the exact number of separatrices, and low computational costs (as detailed in Chapter 6 and 7), its flexibility is limited. Indeed, the imposing of both the integrability condition and isotropic scaling of crosses' branches ($||\tilde{\mathbf{u}}|| = ||\tilde{\mathbf{v}}||$) leads to the constraint on the valid singularity pattern, *i.e.*, fulfilling the Abel Jacobi theorem [67]. In practice, the mispositioning of singularities leads to generating a cross-field that is not aligned with all boundaries, leading to the generation of non-quadrilateral partitions and limit cycles.

This issue imposes the need for developing a new cross-field formulation on the imposed set of singularities, which considers the integrability, while relaxing the condition on isotropic scaling of crosses' branches. Therefore, two sizing fields $L_1 = ||\tilde{\mathbf{u}}||$ and $L_2 = ||\tilde{\mathbf{v}}||$ are introduced. As it will be shown in the following, this setting presents promising results in generating an integrable and boundary-aligned cross-field on the imposed set of singularities, even when their positions are not fulfilling all Abel Jacobi conditions, illustrated in figure 8.1.

In order to properly define this new formulation for a 2D cross-field generation along with *integrability* conditions, the mathematical context is settled in the following.

Let E^3 be the Euclidean space equipped with a Cartesian coordinates system $\{x^i, i = 1, 2, 3\}$, and Ω triangulated 2D domain. We are assuming Ω to be an oriented two-dimensional manifold embedded in E^3 . We note $\mathbf{n}(\mathbf{X})$ the unit normal to Ω at $\mathbf{X} \in \Omega$. It is assumed that the normal field $\mathbf{n}$ is smooth and that there exists a smooth orthonormal local basis $(\mathbf{t}(\mathbf{X}), \mathbf{T}(\mathbf{X}), \mathbf{n}(\mathbf{X}))$ for all $\mathbf{X}$ in Ω .

Arbitrary vector fields V and $W \in E^3$ can be expressed as:

$$\mathbf{V} = V^i \mathbf{E}_i \quad , \quad \mathbf{W} = W^i \mathbf{E}_i$$

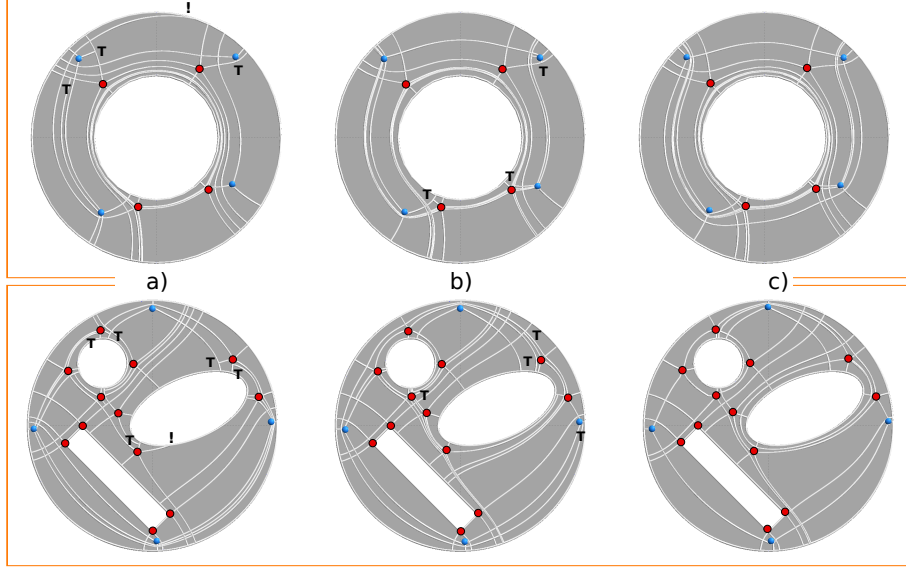


Figure 8.1: Obtained quad layouts on an imposed set of singularities that do not respect the location's condition from the Abel-Jacobi theorem. Quad layouts obtained using the integrable cross-field with isotropic scaling are not aligned with boundaries a), quad layouts obtained with imposing the θ value are boundary aligned but not integrable and demonstrate the presence of t-junctions (marked in **T**) b), and quad layouts obtained using the integrable cross-field with an anisotropic scaling c)

in the natural basis vectors $\{\mathbf{E}_i, i = 1, 2, 3\}$ of this coordinate system, and we shall note

$$\langle \mathbf{V}, \mathbf{W} \rangle = V^i W^j \delta_{ij} \quad , \quad \|\mathbf{V}\| = \sqrt{\langle \mathbf{V}, \mathbf{V} \rangle}$$

the Euclidean metric and the associated norm for vectors. The Levi-Civita connection on E^3 in Cartesian coordinates is trivial (all Christoffel symbols vanish), and one has:

$$\nabla_{\mathbf{V}}^E \mathbf{W} = (\nabla_{\mathbf{V}} W^i) \mathbf{E}_i.$$

The Levi-Civita connection on the Riemannian submanifold Ω , now, is not a trivial one. It is the orthogonal projection of $\nabla_{\mathbf{V}}^E$ in the tangent bundle $T\Omega$, such that:

$$\nabla_{\mathbf{V}} \mathbf{W} = P_{T\Omega}[\nabla_{\mathbf{V}}^E \mathbf{W}] = (\nabla_{\mathbf{V}} W^i) P_{T\Omega}[\mathbf{E}_i] \quad (8.1)$$

where $P_{T\Omega} : E^3 \mapsto T\Omega$ is the orthogonal projection operator on $T\Omega$.

The triple $(\mathbf{t}, \mathbf{T}, \mathbf{n})$ can be represented by means of the Euler angles (ψ, γ, ϕ) and with the shorthands $s_\phi \equiv \sin \phi$ and $c_\phi \equiv \cos \phi$, as

$$\mathbf{t} = \begin{pmatrix} -s_\phi s_\psi c_\gamma + c_\phi c_\psi \\ s_\phi c_\psi c_\gamma + s_\psi c_\phi \\ s_\phi s_\gamma \end{pmatrix}, \quad \mathbf{T} = \begin{pmatrix} -s_\phi c_\psi - s_\psi c_\phi c_\gamma \\ -s_\phi s_\psi + c_\phi c_\psi c_\gamma \\ s_\gamma c_\phi \end{pmatrix}, \quad \mathbf{n} = \begin{pmatrix} s_\psi s_\gamma \\ -s_\gamma c_\psi \\ c_\gamma \end{pmatrix} \quad (8.2)$$

in the vector basis of E^3 .

As defined previously, a 2D cross $\tilde{c}(\mathbf{X})$, $\mathbf{X} \in \Omega$ is the set

$$\tilde{c}(\mathbf{X}) = \{\tilde{\mathbf{u}}(\mathbf{X}), \tilde{\mathbf{v}}(\mathbf{X}), -\tilde{\mathbf{u}}(\mathbf{X}), -\tilde{\mathbf{v}}(\mathbf{X})\},$$

respecting the following conditions:

$$\begin{cases} \tilde{\mathbf{u}}(\mathbf{X}) \cdot \tilde{\mathbf{v}}(\mathbf{X}) &= 0 \\ \tilde{\mathbf{u}}(\mathbf{X}) \cdot \mathbf{n}(\mathbf{X}) &= 0 \\ \tilde{\mathbf{v}}(\mathbf{X}) \cdot \mathbf{n}(\mathbf{X}) &= 0. \end{cases} \quad (8.3)$$

We define:

$$L_1 = \|\tilde{\mathbf{u}}\|, \quad L_2 = \|\tilde{\mathbf{v}}\| \quad (8.4)$$

and

$$\begin{aligned} \mathbf{u} &= \frac{\tilde{\mathbf{u}}}{\|\tilde{\mathbf{u}}\|} \\ \mathbf{v} &= \frac{\tilde{\mathbf{v}}}{\|\tilde{\mathbf{v}}\|} \end{aligned} \quad (8.5)$$

where $(\mathbf{u}, \mathbf{v}, \mathbf{n})$ can be seen as a rotation of $(\mathbf{t}, \mathbf{T}, \mathbf{n})$ among the direction $\mathbf{n}$. Therefore, a 2D cross $\tilde{c}(\mathbf{X})$, $\mathbf{X} \in \Omega$ can be defined with the help of a scalar field θ and the local manifold basis $(\mathbf{t}, \mathbf{T}, \mathbf{n})$ as:

$$\mathbf{u} = c_\theta \mathbf{t} + s_\theta \mathbf{T}, \quad \mathbf{v} = -s_\theta \mathbf{u} + c_\theta \mathbf{v} \quad (8.6)$$

where $(\mathbf{u}, \mathbf{v}, \mathbf{n})$ can also be expressed using the Euler angles (ψ, γ, ϕ) and θ as:

$$\mathbf{u} = \begin{pmatrix} -s_{\theta+\phi} s_\psi c_\gamma + c_{\theta+\phi} c_\psi \\ s_{\theta+\phi} c_\psi c_\gamma + s_\psi c_{\theta+\phi} \\ s_{\theta+\phi} s_\gamma \end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix} -s_{\theta+\phi} c_\psi - s_\psi c_{\theta+\phi} c_\gamma \\ -s_{\theta+\phi} s_\psi + c_{\theta+\phi} c_\psi c_\gamma \\ s_\gamma c_{\theta+\phi} \end{pmatrix}, \quad \mathbf{n} = \begin{pmatrix} s_\psi s_\gamma \\ -s_\gamma c_\psi \\ c_\gamma \end{pmatrix} \quad (8.7)$$

The projection operator $P_{T\Omega}$ introduced in equation (8.1) then simply amounts in disregarding the component along $\mathbf{n}$ of vectors.

For a vector field $\mathbf{w}$ defined on Ω , one can write by derivation of (8.2):

$$\begin{aligned} \nabla_{\mathbf{w}}^E \mathbf{u} &= \mathbf{v} \nabla_{\mathbf{w}}(\theta + \phi) + s_{\theta+\phi} \mathbf{n} \nabla_{\mathbf{w}} \gamma + (c_\gamma \mathbf{v} - s_\gamma c_{\theta+\phi} \mathbf{n}) \nabla_{\mathbf{w}} \Psi \\ \nabla_{\mathbf{w}}^E \mathbf{v} &= -\mathbf{u} \nabla_{\mathbf{w}}(\theta + \phi) + c_{\theta+\phi} \mathbf{n} \nabla_{\mathbf{w}} \gamma + (-c_\gamma \mathbf{u} + s_\gamma s_{\theta+\phi} \mathbf{n}) \nabla_{\mathbf{w}} \Psi \\ \nabla_{\mathbf{w}}^E \mathbf{n} &= -(s_{\theta+\phi} \mathbf{u} + c_{\theta+\phi} \mathbf{v}) \nabla_{\mathbf{w}} \gamma + s_\gamma (c_{\theta+\phi} \mathbf{u} - s_{\theta+\phi} \mathbf{v}) \nabla_{\mathbf{w}} \Psi \end{aligned}$$

and hence, using (8.1), the expression of the covariant derivatives on the sub-manifold Ω is:

$$\begin{aligned}\nabla_{\mathbf{w}}\mathbf{u} &= \mathbf{v}\nabla_{\mathbf{w}}(\theta + \phi) + c_\gamma\mathbf{v}\nabla_{\mathbf{w}}\Psi \\ \nabla_{\mathbf{w}}\mathbf{v} &= -\mathbf{u}\nabla_{\mathbf{w}}(\theta + \phi) - c_\gamma\mathbf{u}\nabla_{\mathbf{w}}\Psi.\end{aligned}$$

This allows writing the Lie bracket

$$\begin{aligned}[\mathbf{u}, \mathbf{v}] &= \nabla_{\mathbf{u}}\mathbf{v} - \nabla_{\mathbf{v}}\mathbf{u} \\ &= -(\mathbf{u}\nabla_{\mathbf{u}}(\theta + \phi) + \mathbf{v}\nabla_{\mathbf{v}}(\theta + \phi)) - c_\gamma(\mathbf{u}\nabla_{\mathbf{u}}\psi + \mathbf{v}\nabla_{\mathbf{v}}\psi),\end{aligned}\quad (8.8)$$

which will be used in the upcoming section.

It is important to note that $\mathbf{u}$ and $\mathbf{v}$ are the two unit branches of the cross-field $\mathcal{C}_\Omega$ we are looking for.

2 | Integrability condition

As explained previously, a cross-field $\mathcal{C}_\Omega$ is integrable if and only if $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ commute under the Lie Bracket. In other words, the condition:

$$0 = [\tilde{\mathbf{u}}, \tilde{\mathbf{v}}] = \nabla_{\tilde{\mathbf{u}}}\tilde{\mathbf{v}} - \nabla_{\tilde{\mathbf{v}}}\tilde{\mathbf{u}} = [L_1\mathbf{u}, L_2\mathbf{v}] \quad (8.9)$$

has to be verified. Developing the latter expression and posing for convenience $L_1 = e^{H_1}$ and $L_2 = e^{H_2}$, it becomes:

$$0 = \mathbf{v}\nabla_{\mathbf{u}}H_2 - \mathbf{u}\nabla_{\mathbf{v}}H_1 + [\mathbf{u}, \mathbf{v}],$$

and then

$$\begin{cases} \nabla_{\mathbf{u}}H_2 &= -\langle \mathbf{v}, [\mathbf{u}, \mathbf{v}] \rangle \\ \nabla_{\mathbf{v}}H_1 &= \langle \mathbf{u}, [\mathbf{u}, \mathbf{v}] \rangle \end{cases} \quad (8.10)$$

which after the substitution of (8.8) gives:

$$\begin{cases} \nabla_{\mathbf{u}}H_2 &= \nabla_{\mathbf{v}}\theta + \nabla_{\mathbf{v}}\phi + c_\gamma\nabla_{\mathbf{v}}\psi \\ -\nabla_{\mathbf{v}}H_1 &= \nabla_{\mathbf{u}}\theta + \nabla_{\mathbf{u}}\phi + c_\gamma\nabla_{\mathbf{u}}\psi \end{cases} \quad (8.11)$$

It is important to note that the three scalar fields (θ, H_1, H_2) are completely defining the cross-field $\mathcal{C}_\Omega$, as (ψ, γ, ϕ) are known, since they are defining the local manifold basis $(\mathbf{t}, \mathbf{T}, \mathbf{n})$.

From the equations 8.11, we can define the cross-field $\mathcal{C}_\Omega$ integrability error E as:

$$E^2(\theta, H_1, H_2) = \int_{\Omega} (\nabla_{\mathbf{u}}H_2 - \nabla_{\mathbf{v}}\theta - \nabla_{\mathbf{v}}\phi - c_\gamma\nabla_{\mathbf{v}}\psi)^2 + (\nabla_{\mathbf{v}}H_1 + \nabla_{\mathbf{u}}\theta + \nabla_{\mathbf{u}}\phi + c_\gamma\nabla_{\mathbf{u}}\psi)^2 d\Omega \quad (8.12)$$

The problem of generating a limit cycle free cross-field can therefore be reduced at finding three scalar fields (θ, H_1, H_2) verifying $E(\theta, H_1, H_2) = 0$.

The process of solving this problem presents several difficulties. First, $(\theta, \psi, \gamma, \phi)$ are the multivalued functions. This kind of difficulty is commonly encountered in cross-field generation and is tackled here by cutting the domain Ω along a generated cut graph. Then, minimizing E regarding (θ, H_1, H_2) is an ill-posed problem. Indeed, there are no constraints on $\nabla_{\mathbf{u}} H_1$ and $\nabla_{\mathbf{v}} H_2$. This is the main obstacle for generating an integrable 2D cross-field with an anisotropic scaling.

A simple approach to solve this problem is proposed here. In order to do so, it is needed to:

- be able to generate a boundary aligned cross-field matching the imposed singularity pattern,
- compute (H_1, H_2) minimizing E for an imposed $\bar{\theta}$,
- compute θ minimizing E for an imposed $(\bar{H}_1, \bar{H}_2)$.

The final resolution algorithm proposed in section 3 allows finding a local minimum for E around an initialization (θ^0, H_1^0, H_2^0) .

2.1 Local manifold basis generation and θ initialization

As shown previously, generating a boundary-aligned cross-field using the H function will fail if Ω is not simply connected, and if singularity locations are not matching Abel-Jacobi conditions, which is typically the case. The aim here is to generate a boundary-aligned cross-field fitting an imposed singularity pattern without taking the integrability into consideration.

This problem has already been tackled in the existing literature. The method presented here is inspired by the work presented in [73].

As exposed earlier, in order to completely define a unitary cross-field $\mathcal{C}_\Omega$ with a scalar field θ it is needed to define a smooth global basis $(\mathbf{t}, \mathbf{T}, \mathbf{n})$ on Ω . This is possible by generating a branch cut $\mathcal{L}$ and computing a smooth global basis $(\mathbf{t}, \mathbf{T}, \mathbf{n})$ on Ω allowing discontinuities across $\mathcal{L}$.

A branch cut is a set $\mathcal{L}$ of curves of a domain M that do not form any closed loop and that cut the domain in such a way that it is impossible to find any closed loop in $\Omega \setminus \mathcal{L}$ that encloses one or several singularities, or an internal boundary. As we are working on a triangulation of Ω , the branch cut $\mathcal{L}$ is in practice simply a set of edges of the triangulation.

The branch cut is generated using the method described in Chapter 6, based on [73]. A local basis $(\mathbf{t}, \mathbf{T}, \mathbf{n})$ on Ω can be generated with any cross-field

method. Such local basis will be smooth and will not show any singularities, as discontinuities are allowed across the cut graph $\mathcal{L}$ and no boundary alignment is required.

Once the cut graph $\mathcal{L}$ and the local basis $(\mathbf{t}, \mathbf{T}, \mathbf{n})$ are generated, it is possible to compute θ only if:

- θ values on $\partial\Omega$ are known,
- θ jump values across $\mathcal{L}$ are known.

These can be found using methods described in [73], or can be deduced from a low computational cost cross-field generation using method detailed in section 3.1.2, Chapter 6 and 7.

2.2 Computing (H_1, H_2) from imposed $\bar{\theta}$

For a given $\bar{\theta}$, it is possible to find (H_1, H_2) minimizing E . It is important to note that, in general, there does not exist a couple (H_1, H_2) such as $E = 0$. Minimizing E with imposed $\bar{\theta}$ is finding the couple (H_1, H_2) for which integrability error is minimal.

The problem to solve is the following: Find $(\bar{H}_1, \bar{H}_2)$ such as

$$E(\bar{\theta}, \bar{H}_1, \bar{H}_2) = \min_{(H_1, H_2) \in (\mathcal{C}^1(\Omega))^2} E(\bar{\theta}, H_1, H_2) \quad (8.13)$$

Let's define $\mathcal{S} = \{(\bar{H}_1, \bar{H}_2) \mid (\bar{H}_1, \bar{H}_2) \text{ verifies 8.13}\}$.

For the problem to be well-posed, a necessary condition is to have 2 independant scalar equations involving ∇H_1 , and the same for ∇H_2 . We can note that in our case, there are no constraints on $\nabla_{\mathbf{u}} H_1$ and $\nabla_{\mathbf{v}} H_2$. Therefore, there is only 1 scalar equation involving ∇H_1 , and 1 scalar equation involving ∇H_2 . As a consequence, the problem we are looking to solve is ill-defined.

As this problem is ill-defined, $\mathcal{S}$ will not be a singleton. In the general case, there will be more than one solution to problem 8.13. This can be highlighted with the simple example illustrated in figure 8.2.

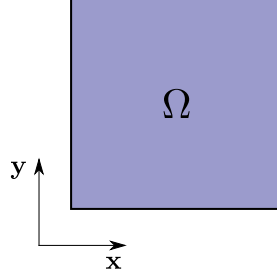


Figure 8.2: Planar square.

In this case, the unitary frame field $\mathcal{C}_\Omega$ obtained with common methods is:

$$\mathcal{C}_\Omega = \{c(\mathbf{X}) = \{\mathbf{x}, \mathbf{y}, -\mathbf{x}, -\mathbf{y}\}, \mathbf{X} \in \Omega\} \quad (8.14)$$

which is equivalent to:

$$\bar{\theta} = 0. \quad (8.15)$$

As in this case Ω is planar, we also have:

$$\psi = \gamma = \phi = 0. \quad (8.16)$$

Equation 8.11 becomes:

$$\begin{cases} \nabla_{\mathbf{x}} H_2 &= 0 \\ -\nabla_{\mathbf{y}} H_1 &= 0 \end{cases} \quad (8.17)$$

which gives:

$$\begin{cases} H_1(x, y) &= f(x), \forall (x, y) \in \Omega, \forall f \in \mathcal{C}^1(\mathbb{R}) \\ H_2(x, y) &= g(y), \forall (x, y) \in \Omega, \forall g \in \mathcal{C}^1(\mathbb{R}) \end{cases} \quad (8.18)$$

Knowing this, we finally have $\mathcal{S} = (C^1(\mathbb{R}))^2$. There is an infinity of solutions, confirming the fact that problem 8.13 is ill-defined.

The solution we could expect to obtain for quad meshing purposes would be:

$$\mathcal{S} = \{(H_1, H_2) = (0, 0)\}, \quad (8.19)$$

which is equivalent to $(L_1, L_2) = (1, 1)$.

Therefore, problem (8.13) has to be regularized in order to reduce the solution space. One way to achieve this goal is to add a constraint on the (H_1, H_2) fields we are looking for. A natural one is to look for (H_1, H_2) verifying equation (8.13) and being as smooth as possible.

With this constraint, the problem to solve becomes:

$$\begin{aligned} &\text{Find } (\bar{H}_1, \bar{H}_2) \in \mathcal{S} \text{ such as} \\ &\int_{\Omega} \|\nabla \bar{H}_1\|^2 + \|\nabla \bar{H}_2\|^2 \, d\Omega = \min_{(H_1, H_2) \in \mathcal{S}} \int_{\Omega} \|\nabla H_1\|^2 + \|\nabla H_2\|^2 \, d\Omega \quad (8.20) \end{aligned}$$

Adding this constraint transforms the linear problem 8.13 into a non-linear one 8.20. The solving is performed in the following manner:

Algorithm 5 Regularized solver for (H_1, H_2)

```

k = 0
initial guess  $H_1^0, H_2^0$ 
compute  $\epsilon^0 = E(\bar{\theta}, H_1^0, H_2^0)$ 
while  $\epsilon^k < \epsilon^{k-1}$  do
    k = k + 1
    find  $(H_1^k, H_2^k)$  minimizing:
     $(E(\theta, f_1, f_2) + \int_{\Omega} \|\nabla f_1 - \nabla H_1^{k-1}\|^2 + \|\nabla f_2 - \nabla H_2^{k-1}\|^2 \, d\Omega),$ 
     $(f_1, f_2) \in (\mathcal{C}^1(\Omega))^2$ 
    compute  $\epsilon^k = E(\bar{\theta}, H_1^k, H_2^k)$ 
end

```

Applying the algorithm 5 is leading to an E 's local minimum $(\bar{\theta}, \bar{H}_1, \bar{H}_2)$ close to $(\bar{\theta}, H_1^0, H_2^0)$.

2.3 Computing θ from $(\bar{H}_1, \bar{H}_2)$

For an imposed couple $(\bar{H}_1, \bar{H}_2)$, it is possible to find θ minimizing E . The problem to solve is formalized as:

$$\text{Find } \bar{\theta} \in \mathcal{C}^1(\Omega) \text{ such as } E(\bar{\theta}, \bar{H}_1, \bar{H}_2) = \min_{\theta \in \mathcal{C}^1(\Omega)} E(\theta, \bar{H}_1, \bar{H}_2) \quad (8.21)$$

This problem is non-linear too. Indeed, $\nabla_{\mathbf{v}} H_1$ and $\nabla_{\mathbf{u}} H_2$ are showing a non-linear dependence regarding θ .

The algorithm used to solve 8.21 is the following:

Algorithm 6 Solver for θ

```

 $k = 0$ 
initial guess  $\theta^0$ 
deduce  $(\mathbf{u}^0, \mathbf{v}^0)$  from  $\theta^0$ 
compute  $\epsilon^0 = E(\theta^0, \bar{H}_1, \bar{H}_2)$ 
while  $\epsilon^k < \epsilon^{k-1}$  do
     $k = k + 1$ 
    find  $\theta^k$  minimizing:
        
$$E^k(f, \bar{H}_1, \bar{H}_2) = \int_{\Omega} (\nabla_{\mathbf{u}^k} \bar{H}_2 - \nabla_{\mathbf{v}^k} f - \nabla_{\mathbf{v}^k} \phi - c_{\gamma} \nabla_{\mathbf{v}^k} \psi)^2 +$$


$$(\nabla_{\mathbf{v}^k} \bar{H}_1 + \nabla_{\mathbf{u}^k} f + \nabla_{\mathbf{u}^k} \phi + c_{\gamma} \nabla_{\mathbf{u}^k} \psi)^2 d\Omega, f \in \mathcal{C}^1(\Omega)$$

    compute  $\epsilon^k = E(\bar{\theta}, H_1^k, H_2^k)$ 
end

```

Solver 6 is leading to a E 's local minimum $(\bar{\theta}, \bar{H}_1, \bar{H}_2)$ close to $(\theta^0, \bar{H}_1, \bar{H}_2)$.

3 | Minimizing integrability error E regarding (θ, H_1, H_2)

Using the three steps exposed previously, it is possible to find a local minimum in the vicinity of an initialization (θ^0, H_1^0, H_2^0) following the scheme:

Algorithm 7 Solver for (θ, H_1, H_2)

```

 $k = 0$ 
initial guess  $\theta^0$  using method presented section 2.1
compute  $(H_1^0, H_2^0)$  from  $\theta^0$  using solver 5
compute  $\epsilon^0 = E(\theta^0, H_1^0, H_2^0)$ 
while  $\epsilon^k < \epsilon^{k-1}$  do
     $k = k + 1$ 
    compute  $\theta^k$  from  $(H_1^{k-1}, H_2^{k-1})$  using solver 6
    compute  $(H_1^k, H_2^k)$  from  $\theta^k$  using solver 5
    compute  $\epsilon^k = E(\theta^k, H_1^k, H_2^k)$ 
end

```

For the example presented in figure 8.3, a set of four index $\frac{1}{4}$ and four index $-\frac{1}{4}$ singularities is imposed. As singularities' locations are not fulfilling the Abel-Jacobi condition, a cross-field generated using dual H function will not be boundary aligned, and a cross-field generated using method presented in section 2.1 will not be integrable and therefore will generate limit cycles.

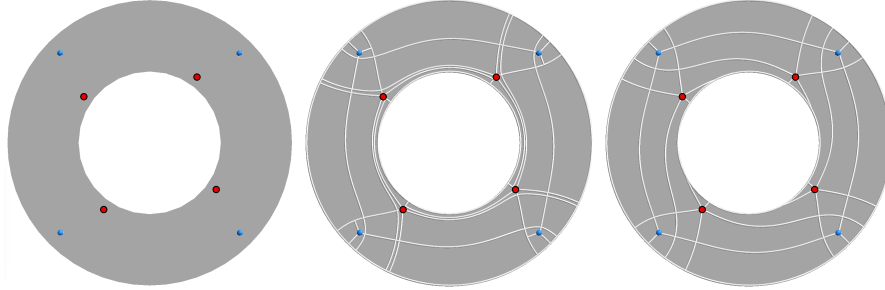


Figure 8.3: From left to right: singularity pattern imposed (index $\frac{1}{4}$ in blue and $-\frac{1}{4}$ in red), quad layout obtained from a cross-field generated with dual H function, and a quad layout obtained with method presented section 2.1. For the sake of clarity, limit cycles are cut upon their first orthogonal intersection, therefore creating T-junctions.

The method presented here is applied to compute an integrable boundary-aligned cross-field. Figure 8.4 represents the cross-field used as an initial guess and figure 8.5 the one obtained at solver 7 convergence.

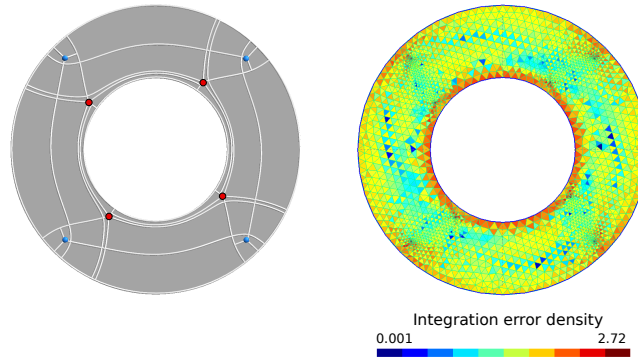


Figure 8.4: Left: quad layout obtained at initialization. Right: integration error density on Ω . The total integration error is $E = 0.307898$.

This shows that generating a limit cycle free 2D cross-field can be done by solving 8.11. Nevertheless, problem 8.11 is highly non-linear and ill-defined; solving it turns out to be difficult. The method proposed here works well when initialization is not far from an integrable solution, as we are looking for a local minimum in the vicinity of this initialization.

If singularity pattern leads to an initialization far from such a solution, the method exposed here does not converge up to the desired solution, by getting

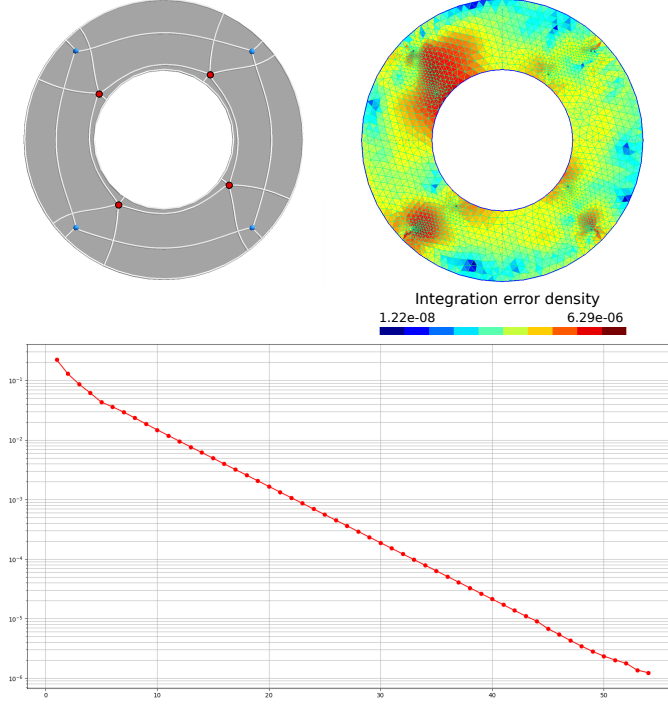


Figure 8.5: Up left: quad layout obtained at convergence. Up right: integration error density on Ω . Down: integration error regarding solver 7 iteration number. The total integration error at convergence is $E = 1.45639e - 06$.

stuck in a local minimum $(\bar{\theta}, \bar{H}_1, \bar{H}_2)$ which does not satisfy $E(\bar{\theta}, \bar{H}_1, \bar{H}_2) = 0$. This is the case on the following example. Figure 8.6 shows the singularity pattern imposed and the matching quad layout we would like to obtain.

Figure 8.6 is the cross-field from the method presented in section 2.1 used as initialization. Applying solver 7 to this example gives results presented in figure 8.7. The obtained quad layout is closer to the desired one (figure 8.6) but still exhibits limit cycles due to the fact that solver 7 is stuck in a local minimum for E .

These results highlight the fact that solver 7 is not robust enough to generate an integrable (*i.e.* limit cycles free) boundary-aligned 2D cross-field in the general case, but robust enough if initialization is not too far from the desired solution. This finding suggests a profound need for developing an automated scheme for optimizing the singularity configuration to satisfy the Abel-Jacobi theorem not only for 2D closed manifolds (as presented in [146]), but also for

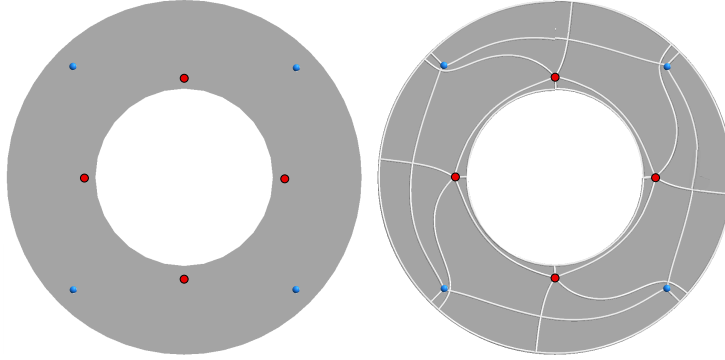


Figure 8.6: Left: singularity pattern imposed (index $\frac{1}{4}$ in blue and $-\frac{1}{4}$ in red). Right: ideal quad layout to obtain.

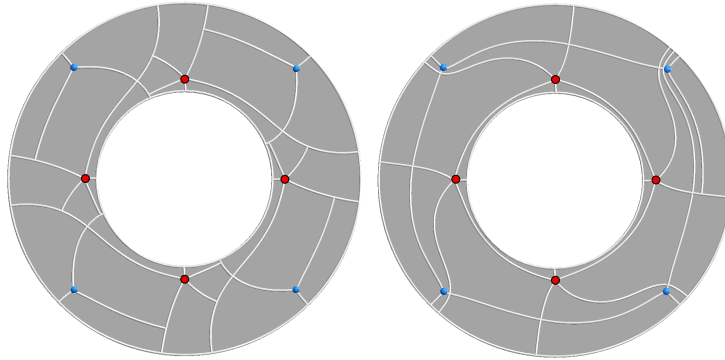


Figure 8.7: Left: quad layout obtained at initialization, the total integration error is $E = 0.842169$. Right: quad layout obtained at convergence, the total integration error is $E = 0.013597$

planar domains and surfaces with boundaries.

CHAPTER



Conclusion

Mesh generation represents a unique field, regarded both as an art and science [14, 15]. Although this process heavily relies on mathematical foundations, it requires a certain level of craftsmanship since there is no unique mathematical path to follow. Its generation has evolved considerably from unglamorous and tedious manual efforts to an automated methodology through the decades. As elaborated in a famous book “*Handbook of Grid Generation*“ [15], and more recently in [147], numerous approaches have been developed to address this exciting topic. These works represent the testimony of the severe difficulties on the journey of accomplishing the goal mentioned above, regardless of the chosen method or background numerics used. At the same time, they point out the profound need for further improvements in reliability, robustness, and complete automatization [147], making mesh generation an advancing field.

As outlined in Chapter 2, there exist various types of meshes, each of them suiting a particular group of applications. When it comes to capturing anisotropic features and decreasing approximation error, which is essential for elastic and elastoplastic analysis [20], spectral and iso-geometric simulations [41], high-order surface modeling purposes [21], and many demanding numerical problems, the *quad meshes* represent an indispensable tool [17, 18, 53]. As reviewed in Chapter 3, there is no lack of available algorithms targeting this specific topic. Nevertheless, many important questions still remain unanswered, *e.g.*, how to deal successfully with approximation and quad mesh quality, robustness on geometrically inconsistent inputs, efficiency, interactive control, scalability, and evaluation [21].

Regarding the particular issue of *block-structured quad mesh* generation, the objective of this thesis was twofold: (1) develop the block-structured quad mesher that is automatic and produces high-quality elements, and (2) design the interactive block-structured quad mesher that preserves properties of the first objective.

In order to accomplish these goals, a *field-guided method* allowing control over the orientation and the size of quad elements is explored. This path implied using an auxiliary object: the *cross-field*, including its topology, *i.e.*, *singular points* hand in hand with the *quad layout concept*, which yielded two research directions. The first one was finding the most promising *Partial Differential Equation* (PDE) model for the cross-field computation. The main advantages of the proposed Ginzburg-Landau formulation are the thorough mathematical background and adequate singularities' positions and valences, which are highly advocated for quad meshing purposes (Chapter 4 and Chapter 5). Nevertheless, due to the nature of this particular PDE, the solution is mesh dependant, *i.e.*, singularities are repelled from the refined regions inhibiting the finding of the accurate number of separatrices, and it generates high computational costs. These facts, being both theoretically and practically explored, have shifted the focus towards generating a cross-field with an imposed set of singularities, relying on a dual scalar field H . This new formulation is well established in the corresponding Ginzburg-Landau theory [22] and is in its core linear. The H field showed quite competitive results in computational costs, mesh independency of singularities' locations, the improvement of numerical results, and enabling interactive quad mesh design. It was precisely the characteristics of this scalar field that enabled working with *high valence singularities*, an advanced topic in quad meshing, and tackling the critical question of *non-quad meshable* singularity configurations (Chapter 6, Chapter 7 and Chapter 8).

The fact that a cross-field's properties are intimately related to quad mesh generation [21, 81] served as a motivation for the second research direction, *i.e.*, working on its *integrability*. As detailed in Chapter 3, the Riemannian manifold theory shows that a cross-field generates a valid quad layout, and

consequently, a conformal block-structured quad mesh, if and only if it is *integrable*. The existing methods are leaving the integrability problem aside and dealing with it in the post-processing stage, which influences the appearance of coiled-up integral lines, *i.e.*, *limit cycles*, leaving dramatic consequences on the quad layout and associated quad mesh. Here, the integrability condition was considered *a priori*, *i.e.*, in the cross-field formulation. By doing so, the quality of the final result, *i.e.*, a quad mesh, was increased directly, and the cross-field generation problem had become linear, as detailed in Chapter 6 (for planar case) and Chapter 7 (for 2D manifold case), allowing the decrease in overall problem complexity.

The integrability condition, along with computing only one scaling field H , imposed the strict constraint on singularity locations and configurations, *i.e.*, the need for fulfilling the *Abel Jacobi theorem* from [67]. In order to bypass this limitation, a new mathematical formulation relying on anisotropic scaling has been developed, detailed in Chapter 8, allowing the computation of an integrable and boundary-aligned cross-field on imposed set of singularities, while not respecting the location's condition from Abel Jacobi theorem. As presented, the initial results were promising; nevertheless, to accomplish the goal of obtaining a conformal quad layout using the presented separatrix tracing scheme, the development of an automated singularity configuration optimization fulfilling the Abel-Jacobi theorem is of utmost importance.

The research journey of this thesis relied on the strict mathematical background, yielding the cross-field generation which is quad meshable and integrable, both with isotropic and anisotropic scaling and computational costs as low as solving only two linear systems. Presented advances went hand-in-hand with flexibility on imposing singularity configurations, improving numerical inconsistencies of the cross-field in singularities vicinity and reparation schemes for ill-defined partitions. Unfortunately, the presented algorithm does not guarantee to conformally quad mesh all possible domains, as figure 9.1 illustrates, yet the formulations' characteristics would greatly benefit current state-of-the-art methods. The robustness issue of the presented method originates from choosing the separatrix tracing scheme instead of methods less sensitive to fulfilling the Abel-Jacobi conditions, *e.g.*, point generation [148] or global parametrization approach [65].

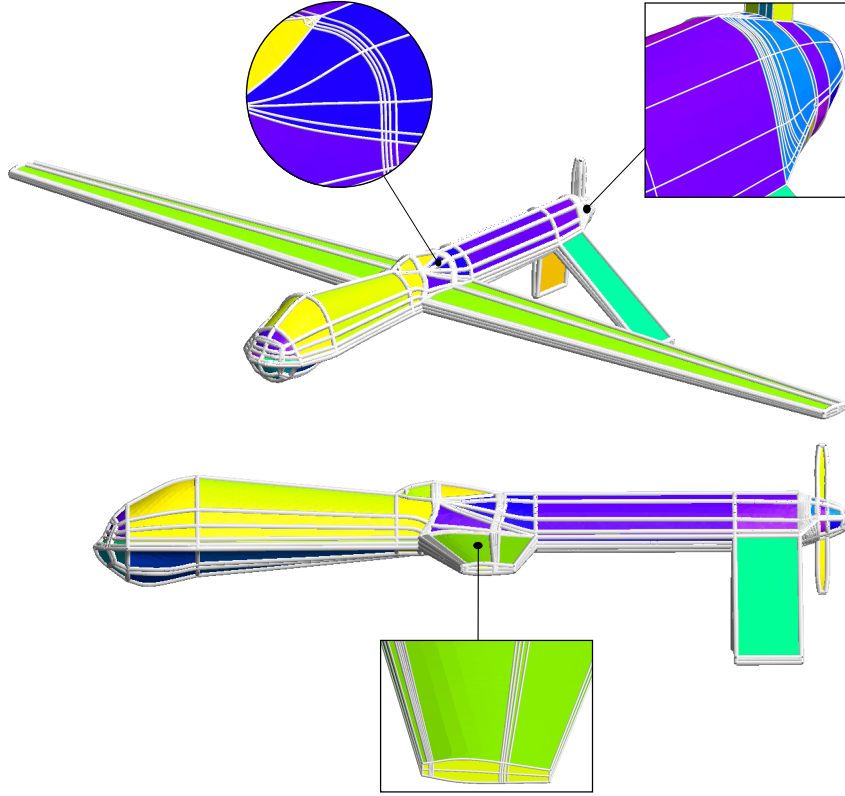


Figure 9.1: The sub-optimal singularity configuration causing the appearance of limit cycles and triangular partitions in the aircraft quad layout

1 | Perspectives

1.1 Extension of the algorithm to 3D

The focus of this thesis was on obtaining a surface quad layout by using 2D cross-field structure, wherefrom a quad mesh was generated. The straightforward extension of the proposed method to hexahedral meshing implies generating a hex layout relying on a 3D frame field. Although a 3D frame field generation has been fairly explored [149, 150, 151, 152], the control over its topology, *i.e.*, singularity configurations, represents the principal bottleneck in the current hex meshing state-of-the-art. As explained in [153]: “*It remains an open problem to give a sufficient condition on singularity graphs compatible to non-degenerate hexahedral meshing.*” On the other hand, as extensive surveys on hex meshing algorithms [40, 154] point out, it is possible in some cases to

use surface quad mesh for generating a hex mesh, nevertheless, the developed algorithms, *e.g.*, [155, 156], typically can deal only with specific topologies, and as already concluded in [157]: “*Generating hexahedral meshes of valid topology and practical geometric properties from quadrilateral surface meshes of arbitrary genus and possibly self-intersecting loops of quadrilaterals is unresolved in general.*” Unfortunately, a similar conclusion stands for generating hex layout from a quad layout, except for some particular cases [13].

1.2 Generating an anisotropic quad mesh

The existence of adaptive quad meshing algorithms is crucial for some fields, *e.g.*, computational fluid dynamics, dealing with anisotropic physical problems, and consequently using anisotropic quad meshes. In order to gain control over alignment, density, and anisotropy of the elements’ distribution, some works rely on a specific metric [158, 159], two sizing fields [160], the curvature tensor field [161], or a non-orthogonal and non-unit-length frame fields [162]. In this thesis, the shape of quadrilateral elements is influenced by the isotropic sizing field H (Chapter 6 and Chapter 7), resulting in generating the isotropic quad mesh. At the same time, it is possible to compute two independent sizing fields (Chapter 8), accomplishing the task of obtaining the anisotropic quad mesh. Future work would include comparing the quality of elements between isotropic and anisotropic versions of the algorithm.

1.3 Generating a quad mesh respecting the user-imposed size map

The field-guided methods, in general, support a significant amount of user control and imposing the mesh size is just one of them. Typically, the information on user-prescribed target quad size or edge length, *i.e.*, a size field, is not considered in the cross-field computation step, rather during the remeshing process, inducing considerable distortions or the algorithm failure originating from geometrical or topological inconsistencies of the input mesh. The method presented in [163] reformulates the construction of the cross-field in a scale-aware manner to target both the prescribed sizing field and dealing with ill-behaved mesh inputs.

By using the integrable cross-field formulation relying on two sizing fields H_1 and H_2 , presented in Chapter 8, it would be possible to take into account the anisotropic size field to guide the cross-field generation. The size field obtained from the generated cross-field would not precisely match the one prescribed by the user, but would be as close as possible with the singularity pattern chosen

for the cross-field generation.

Some of the other exciting perspectives for future works include comparing the differences in quality of an anisotropic version of the method presented in this thesis with its original isotropic version, improving the robustness, and decreasing the overall computational time of the presented algorithm.

Bibliography

- [1] Lewis F Richardson. *Weather Prediction by Numerical Process*. Cambridge University Press, 1922.
- [2] Pierre-Alexandre Beaufort. “Parameterizations for mesh generation: discrete atlases, crossfields and 3D frames”. PhD thesis. UCL-Université Catholique de Louvain, 2019.
- [3] Peter Lynch. “The origins of computer weather prediction and climate modeling”. In: *Journal of computational physics* 227.7 (2008), pp. 3431–3444.
- [4] Richard Courant, Kurt Friedrichs, and Hans Lewy. “Über die partiellen Differenzengleichungen der mathematischen Physik”. In: *Mathematische annalen* 100.1 (1928), pp. 32–74.
- [5] John H Argyris and Sydney Kelsey. *Energy theorems and structural analysis*. Vol. 60. Springer, 1960.
- [6] Ray W Clough and Edward L Wilson. “Early finite element research at Berkeley”. In: *Fifth US National Conference on Computational Mechanics*. 1999, pp. 1–35.
- [7] Edward L Wilson, KJ Bathe, FE Peterson, and HH Dovey. “SAP—A structural analysis program for linear systems”. In: *Nuclear Engineering and Design* 25.2 (1973), pp. 257–274.
- [8] J Babushka. *A. K. Aziz-Survey lectures on the mathematical foundations of the finite elements method. The mathematical formulations of the finite elements method with application to partial differential equations*. 1972.
- [9] Susanne Brenner and Ridgway Scott. *The mathematical theory of finite element methods*. Vol. 15. Springer Science & Business Media, 2007.
- [10] Gilbert Strang and George J Fix. “An analysis of the finite element method”. In: (1973).
- [11] Amaury Johnen. “Indirect quadrangular mesh generation and validation of curved finite elements”. PhD thesis. Université de Liège, Liège, Belgique, 2016.

- [12] Ivo Babuška and A Kadir Aziz. “On the angle condition in the finite element method”. In: *SIAM Journal on numerical analysis* 13.2 (1976), pp. 214–226.
- [13] Marcel Campen. “Quad Layouts—Generation and Optimization of Conforming Quadrilateral Surface Partitions”. PhD thesis. 2014.
- [14] Timothy J Baker. “Mesh generation: Art or science?” In: *Progress in Aerospace Sciences* 41.1 (2005), pp. 29–63.
- [15] Joe F Thompson, Bharat K Soni, and Nigel P Weatherill. *Handbook of grid generation*. CRC press, 1998.
- [16] TJR Hughes, JA Cottrell, and Y Bazilevs. “Consider a Spherical Cow—Conservation of Geometry in Analysis: Implications for Computational Methods in Engineering”. In: *IMA Hot Topics Workshop: Compatible Spatial Discretizations for Partial Differential Equations*. 2004.
- [17] Steven E Benzley, Ernest Perry, Karl Merkley, Brett Clark, and Greg Sjaardama. “A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis”. In: *Proceedings, 4th international meshing roundtable*. Vol. 17. Sandia National Laboratories Albuquerque, NM. 1995, pp. 179–191.
- [18] Jesse Chan, Zheng Wang, Axel Modave, Jean-Francois Remacle, and Tim Warburton. “GPU-accelerated discontinuous Galerkin methods on hybrid meshes”. In: *Journal of Computational Physics* 318 (2016), pp. 142–168.
- [19] Nicolas Kowalski, Franck Ledoux, and Pascal Frey. “A PDE based approach to multidomain partitioning and quadrilateral meshing”. In: *Proceedings of the 21st international meshing roundtable*. Springer, 2013, pp. 137–154.
- [20] Jason F Shepherd and Chris R Johnson. “Hexahedral mesh generation constraints”. In: *Engineering with Computers* 24.3 (2008), pp. 195–213.
- [21] David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Cláudio T Silva, Marco Tarini, and Denis Zorin. “Quad Meshing.” In: *Eurographics (STARs)*. 2012, pp. 159–182.
- [22] Fabrice Bethuel, Haïm Brezis, Frédéric Hélein, et al. *Ginzburg-landau vortices*. Vol. 13. Springer, 1994.
- [23] Christophe Geuzaine and Jean-François Remacle. “Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities”. In: *International journal for numerical methods in engineering* 79.11 (2009), pp. 1309–1331.
- [24] Leif P Kobbelt, Stephan Bischoff, Mario Botsch, Kilja Kähler, Christian Rössl, Robert Schneider, and Jens Vorsatz. *Geometric modeling based on polygonal meshes*. Vol. 1. Max-Planck-Institut für Informatik, 2000.

- [25] Gerald Farin. *Curves and surfaces for computer-aided geometric design: a practical guide*. Elsevier, 2014.
- [26] Les Piegl and Wayne Tiller. *The NURBS book*. Springer Science & Business Media, 2012.
- [27] Soji Yamakawa and Kenji Shimada. “Polygon crawling: feature edge extraction from a general polygonal surface for mesh generation”. In: *Engineering with Computers* 26.3 (2010), pp. 249–264.
- [28] Marcel Campen. “Partitioning surfaces into quadrilateral patches: a survey”. In: *Computer Graphics Forum*. Vol. 36. 8. Wiley Online Library. 2017, pp. 567–588.
- [29] Pascal Jean Frey and Paul-Louis George. *Mesh generation: application to finite elements*. Iste, 2007.
- [30] Steven J Owen. “A survey of unstructured mesh generation technology.” In: *IMR* 239 (1998), p. 267.
- [31] Soo-Won Chae and Ki-Youn Kwon. “Quadrilateral mesh generation on trimmed NURBS surfaces”. In: *KSME international journal* 15.5 (2001), pp. 592–601.
- [32] Klaus Höllig. *Finite element methods with B-splines*. SIAM, 2003.
- [33] Ruben Sevilla, Luke Rees, and Oubay Hassan. “The generation of triangular meshes for NURBS-enhanced FEM”. In: *International Journal for Numerical Methods in Engineering* 108.8 (2016), pp. 941–968.
- [34] Maximilian Harmel, Roger A Sauer, and David Bommers. “Volumetric mesh generation from T-spline surface representations”. In: *Computer-Aided Design* 82 (2017), pp. 13–28.
- [35] Amir Ebrahimi-Fizik, Esmail Lakzian, and Ali Hashemian. “Entropy generation analysis of wet-steam flow with variation of expansion rate using NURBS-based meshing technique”. In: *International Journal of Heat and Mass Transfer* 139 (2019), pp. 399–411.
- [36] Yongjie Zhang, Yuri Bazilevs, Samrat Goswami, Chandrajit L Bajaj, and Thomas JR Hughes. “Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow”. In: *Computer methods in applied mechanics and engineering* 196.29-30 (2007), pp. 2943–2959.
- [37] Yuri Bazilevs, Victor M Calo, John A Cottrell, John A Evans, Thomas Jr R Hughes, S Lipton, Michael A Scott, and Thomas W Sederberg. “Isogeometric analysis using T-splines”. In: *Computer Methods in Applied Mechanics and Engineering* 199.5-8 (2010), pp. 229–263.
- [38] Mario Botsch. “High quality surface generation and efficient multiresolution editing based on triangle meshes”. In: *Ausgezeichnete Informatikdissertationen 2005* (2006).

- [39] Marcel Campen and Leif Kobbelt. “Quad layout embedding via aligned parameterization”. In: *Computer Graphics Forum*. Vol. 33. 8. Wiley Online Library. 2014, pp. 69–81.
- [40] Ted Blacker. “Meeting the challenge for automated conformal hexahedral meshing”. In: *9th international meshing roundtable*. 2000, pp. 11–20.
- [41] Eduardo F D’azevedo. “Are bilinear quadrilaterals better than linear triangles?” In: *SIAM Journal on Scientific Computing* 22.1 (2000), pp. 198–217.
- [42] Leif Kobbelt Max Lyon Marcel Campen. “Quad Layouts via Constrained T-Mesh Quantization”. In: *Eurographics* 40 (2021).
- [43] M Bern and P Plassmann. *Handbook of Computational Geometry Edited by JR Sack and J. Urritia*. 2000.
- [44] Pierre-Alexandre Beaufort, Jonathan Lambrechts, François Henrotte, Christophe Geuzaine, and Jean-François Remacle. “Computing cross fields A PDE approach based on the Ginzburg-Landau theory”. In: *Procedia engineering* 203 (2017), pp. 219–231.
- [45] Patrick M Knupp. “Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part II—a framework for volume mesh optimization and the condition number of the Jacobian matrix”. In: *International Journal for numerical methods in engineering* 48.8 (2000), pp. 1165–1185.
- [46] M Lyon, D Bommers, and L Kobbelt. “Cost Minimizing Local Anisotropic Quad Mesh Refinement”. In: *Computer Graphics Forum*. Vol. 39. 5. Wiley Online Library. 2020, pp. 163–172.
- [47] Peter Murdoch, Steven Benzley, Ted Blacker, and Scott A Mitchell. “The spatial twist continuum: A connectivity based method for representing all-hexahedral finite element meshes”. In: *Finite elements in analysis and design* 28.2 (1997), pp. 137–149.
- [48] Timothy J Tautges, Ted Blacker, and Scott A Mitchell. “The whisker weaving algorithm: a connectivity-based method for constructing all-hexahedral finite element meshes”. In: *International Journal for Numerical Methods in Engineering* 39.19 (1996), pp. 3327–3349.
- [49] Joel Daniels, Cláudio T Silva, Jason Shepherd, and Elaine Cohen. “Quadrilateral mesh simplification”. In: *ACM transactions on graphics (TOG)* 27.5 (2008), pp. 1–9.
- [50] Matthew L Staten, Steven Benzley, and Michael Scott. “A methodology for quadrilateral finite element mesh coarsening”. In: *Engineering with Computers* 24.3 (2008), pp. 241–251.

- [51] Maxence Reberol. “Maillages hex-dominants: génération, simulation et évaluation”. PhD thesis. Université de Lorraine, 2018.
- [52] Teseo Schneider, Yixin Hu, Xifeng Gao, Jeremie Dumas, Denis Zorin, and Daniele Panozzo. “A Large Scale Comparison of Tetrahedral and Hexahedral Elements for Finite Element Analysis”. In: *arXiv preprint arXiv:1903.09332* (2019).
- [53] Mohamed Mahran, Adel ELsabbagh, and Hani Negm. “A comparison between different finite elements for elastic and aero-elastic analyses”. In: *Journal of advanced research* 8.6 (2017), pp. 635–648.
- [54] Thomas JR Hughes. “The Finite Element Method Prentice-Hall”. In: *New Jersey (USA)* (1987).
- [55] Marcel Campen, David Bommers, and Leif Kobbelt. “Quantized global parametrization”. In: *Acm Transactions On Graphics (tog)* 34.6 (2015), pp. 1–12.
- [56] Steve Owen. “An introduction to mesh generation algorithms”. In: *Sandia National laboratories 14th International Meshing Roundtable, San Diego, USA* (2005).
- [57] Daniel SH Lo. *Finite element mesh generation*. CRC Press, 2014.
- [58] Kendrick M Shepherd, René R Hiemstra, and Thomas JR Hughes. “The Quad Layout Immersion: A Mathematically Equivalent Representation of a Surface Quadrilateral Layout”. In: *arXiv preprint arXiv:2012.09368* (2020).
- [59] Harold J Fogg, Cecil G Armstrong, and Trevor T Robinson. “Automatic generation of multiblock decompositions of surfaces”. In: *International Journal for Numerical Methods in Engineering* 101.13 (2015), pp. 965–991.
- [60] Jovana Jezdimirovic, Alexandre Chemin, and Jean François Remacle. “Multi-block decomposition and meshing of 2D domain using Ginzburg-Landau PDE”. In: (2019).
- [61] Jovana Jezdimirović, Alexandre Chemin, Maxence Reberol, François Henrotte, and Jean François Remacle. “Quad layouts with high valence singularities for flexible quad meshing”. In: *Proceedings of the 29th Meshing Roundtable* (2021).
- [62] Nicolas Ray, Bruno Vallet, Wan Chiu Li, and Bruno Lévy. “N-symmetry direction field design”. In: *ACM Transactions on Graphics (TOG)* 27.2 (2008), pp. 1–13.
- [63] Ryan Viertel and Braxton Osting. “An approach to quad meshing based on harmonic cross-valued maps and the Ginzburg–Landau theory”. In: *SIAM Journal on Scientific Computing* 41.1 (2019), A452–A479.

- [64] David Bommes, Marcel Campen, Hans-Christian Ebke, Pierre Alliez, and Leif Kobbelt. “Integer-grid maps for reliable quad meshing”. In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), pp. 1–12.
- [65] Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. “Periodic global parameterization”. In: *ACM Transactions on Graphics (TOG)* 25.4 (2006), pp. 1460–1485.
- [66] Wei Chen, Xiaopeng Zheng, Jingyao Ke, Na Lei, Zhongxuan Luo, and Xianfeng Gu. “Quadrilateral mesh generation I: Metric based method”. In: *Computer Methods in Applied Mechanics and Engineering* 356 (2019), pp. 652–668.
- [67] Na Lei, Xiaopeng Zheng, Zhongxuan Luo, Feng Luo, and Xianfeng Gu. “Quadrilateral mesh generation II: Meromorphic quartic differentials and Abel–Jacobi condition”. In: *Computer Methods in Applied Mechanics and Engineering* 366 (2020), p. 112980.
- [68] Harold J Fogg, Cecil G Armstrong, and Trevor T Robinson. “Multi-block decomposition using cross-fields”. In: *Proceedings of adaptive modelling and simulation, Lisbon* (2013), pp. 254–267.
- [69] James Albert Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*. Vol. 3. Cambridge university press, 1999.
- [70] Ko-Shin Chen and Peter Sternberg. “Dynamics of Ginzburg-Landau and Gross-Pitaevskii vortices on manifolds”. In: *arXiv preprint arXiv: 1211.6780* (2012).
- [71] Felix Kälberer, Matthias Nieser, and Konrad Polthier. “Quadcover-surface parameterization using branched coverings”. In: *Computer graphics forum*. Vol. 26. 3. Wiley Online Library. 2007, pp. 375–384.
- [72] Marcel Campen, David Bommes, and Leif Kobbelt. “Dual loops meshing: quality quad layouts on manifolds”. In: *ACM Transactions on Graphics (TOG)* 31.4 (2012), pp. 1–11.
- [73] David Bommes, Henrik Zimmer, and Leif Kobbelt. “Mixed-integer quadrangulation”. In: *ACM Transactions On Graphics (TOG)* 28.3 (2009), pp. 1–10.
- [74] Nico Pietroni, Enrico Puppo, Giorgio Marcias, Roberto Scopigno, and Paolo Cignoni. “Tracing field-coherent quad layouts”. In: *Computer Graphics Forum*. Vol. 35. 7. Wiley Online Library. 2016, pp. 485–496.
- [75] Ashish Myles, Nico Pietroni, Denis Kovacs, and Denis Zorin. “Feature-aligned T-meshes”. In: *ACM Transactions on Graphics (TOG)* 29.4 (2010), pp. 1–11.

- [76] Gunnar Brinkmann, Brendan D McKay, et al. “Fast generation of planar graphs”. In: *MATCH Commun. Math. Comput. Chem* 58.2 (2007), pp. 323–357.
- [77] Faniry H Razafindrazaka, Ulrich Reitebuch, and Konrad Polthier. “Perfect matching quad layouts for manifold meshes”. In: *Computer Graphics Forum*. Vol. 34. 5. Wiley Online Library. 2015, pp. 219–228.
- [78] Stefano Nuvoli, Alex Hernandez, Claudio Esperança, Riccardo Scateni, Paolo Cignoni, and Nico Pietroni. “QuadMixer: layout preserving blending of quadrilateral meshes”. In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), pp. 1–13.
- [79] David Bommes, Timm Lempfer, and Leif Kobbelt. “Global structure optimization of quadrilateral meshes”. In: *Computer Graphics Forum*. Vol. 30. 2. Wiley Online Library. 2011, pp. 375–384.
- [80] Marco Tarini, Enrico Puppo, Daniele Panozzo, Nico Pietroni, and Paolo Cignoni. “Simple quad domains for field aligned mesh parametrization”. In: *Proceedings of the 2011 SIGGRAPH Asia Conference*. 2011, pp. 1–12.
- [81] Amir Vaxman, Marcel Campen, Olga Diamanti, Daniele Panozzo, David Bommes, Klaus Hildebrandt, and Mirela Ben-Chen. “Directional field synthesis, design, and processing”. In: *Computer Graphics Forum*. Vol. 35. 2. Wiley Online Library. 2016, pp. 545–572.
- [82] Andrea Brambilla, Robert Carnecky, Robert Peikert, Ivan Viola, and Helwig Hauser. “Illustrative flow visualization: State of the art, trends and challenges”. In: *Visibility-oriented Visualization Design for Flow Illustration* (2012).
- [83] Daniel Asimov. *Notes on the topology of vector fields and flows*. Tech. rep. Technical report, NASA Ames Research Center, 1993. RNR-93-003, 1993.
- [84] Robert S Laramee, Helwig Hauser, Lingxiao Zhao, and Frits H Post. “Topology-based flow visualization, the state of the art”. In: *Topology-based methods in visualization*. Springer, 2007, pp. 1–19.
- [85] Frits H Post, Benjamin Vrolijk, Helwig Hauser, Robert S Laramee, and Helmut Doleisch. “Feature extraction and visualisation of flow fields.” In: *Eurographics (STARs) 2* (2002).
- [86] Aaron Hertzmann and Denis Zorin. “Illustrating smooth surfaces”. In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000, pp. 517–526.
- [87] Fernando de Goes, Mathieu Desbrun, and Yiyang Tong. “Vector field processing on triangle meshes”. In: *ACM SIGGRAPH 2016 Courses*. 2016, pp. 1–49.

- [88] Barry Merriman, James K Bence, and Stanley J Osher. “Motion of multiple junctions: A level set approach”. In: *Journal of Computational Physics* 112.2 (1994), pp. 334–363.
- [89] Guy Bunin. “A continuum theory for unstructured mesh generation in two dimensions”. In: *Computer Aided Geometric Design* 25.1 (2008), pp. 14–40.
- [90] Tobias Günther and Holger Theisel. “The state of the art in vortex extraction”. In: *Computer Graphics Forum*. Vol. 37. 6. Wiley Online Library. 2018, pp. 149–173.
- [91] Olga Diamanti, Amir Vaxman, Daniele Panozzo, and Olga Sorkine-Hornung. “Designing N-PolyVector fields with complex polynomials”. In: *Computer Graphics Forum*. Vol. 33. 5. Wiley Online Library. 2014, pp. 1–11.
- [92] Murray Eisenberg and Robert Guy. “A proof of the hairy ball theorem”. In: *The American Mathematical Monthly* 86.7 (1979), pp. 571–574.
- [93] Ashish Myles, Nico Pietroni, and Denis Zorin. “Robust Field-aligned Global Parametrization: Supplement 1, Proofs and Algorithmic Details”. In: *Visual Computing Lab* (2014).
- [94] D Barnette, E Jucovič, and M Trenkler. “Toroidal maps with prescribed types of vertices and faces”. In: *Mathematika* 18.1 (1971), pp. 82–90.
- [95] Ernest Jucovič and Marián Trenkler. “A theorem on the structure of cell-decompositions of orientable 2-manifolds”. In: *Mathematika* 20.1 (1973), pp. 63–82.
- [96] Ivan Izvestiev, Robert B Kusner, Günter Rote, Boris Springborn, and John M Sullivan. “There is no triangulation of the torus with vertex degrees 5, 6,..., 6, 7 and related results: Geometric proofs for combinatorial theorems”. In: *Geometriae Dedicata* 166.1 (2013), pp. 15–29.
- [97] Arthur J Schwartz. “A generalization of a Poincaré-Bendixson theorem to closed two-dimensional manifolds”. In: *American Journal of Mathematics* (1963), pp. 453–458.
- [98] Ashish Myles, Nico Pietroni, and Denis Zorin. “Robust field-aligned global parametrization”. In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), pp. 1–14.
- [99] Nicolas Ray and Dmitry Sokolov. “Robust polylines tracing for n- symmetry direction field on triangulated surfaces”. In: *ACM Transactions on Graphics (TOG)* 33.3 (2014), pp. 1–11.
- [100] Ryan Viertel, Braxton Osting, and Matthew Staten. “Coarse Quad Layouts Through Robust Simplification of Cross Field Separatrix Partitions”. In: *arXiv preprint arXiv: 1905.09097* (2019).

- [101] David Eppstein, Michael T Goodrich, Ethan Kim, and Rasmus Tamstorf. “Motorcycle graphs: Canonical quad mesh partitioning”. In: *Computer Graphics Forum*. Vol. 27. 5. Wiley Online Library. 2008, pp. 1477–1486.
- [102] Pierre-Gilles De Gennes. *Superconductivity of metals and alloys*. CRC Press, 2018.
- [103] Russell J Donnelly. *Quantized vortices in helium II*. Vol. 2. Cambridge University Press, 1991.
- [104] JM Kosterlitz and David J Thouless. “Two-dimensional physics”. In: *Progress in low temperature physics*. Vol. 7. Elsevier, 1978, pp. 371–433.
- [105] Michael Tinkham. *Introduction to superconductivity*. Courier Corporation, 2004.
- [106] Olga Diamanti, Amir Vaxman, Daniele Panozzo, and Olga Sorkine-Hornung. “Integrable polyvector fields”. In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), p. 38.
- [107] J Jezdimirovic, A Chemin, PA Beaufort, and JF Remacle. “Elliptic Fekete points obtained by Ginzburg-Landau PDE”. In: *Proceedings, 26th international meshing roundtable. Sandia National Laboratories Albuquerque, NM* (2017).
- [108] Nilanjan Mukherjee. “CSALF-Q: A bricolage algorithm for anisotropic quad mesh generation”. In: *Proceedings of the 20th International Meshing Roundtable*. Springer, 2011, pp. 489–509.
- [109] Patrick M Knupp. “Winslow smoothing on two-dimensional unstructured meshes”. In: *Engineering with Computers* 15.3 (1999), pp. 263–268.
- [110] Alan M Winslow. “Numerical solution of the quasilinear Poisson equation in a nonuniform triangle mesh”. In: *Journal of computational physics* 1.2 (1966), pp. 149–172.
- [111] J-F Remacle, Jonathan Lambrechts, Bruno Seny, Emilie Marchandise, Amaury Johnen, and C Geuzainet. “Blossom-Quad: A non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm”. In: *International journal for numerical methods in engineering* 89.9 (2012), pp. 1102–1119.
- [112] J-F Remacle, Christophe Geuzaine, Gaëtan Compere, and Emilie Marchandise. “High-quality surface remeshing using harmonic maps”. In: *International Journal for Numerical Methods in Engineering* 83.4 (2010), pp. 403–425.

- [113] Emilie Marchandise, C Carton de Wiart, WG Vos, Christophe Geuzaine, and J-F Remacle. “High-quality surface remeshing using harmonic maps-Part II: Surfaces with high genus and of large aspect ratio”. In: *International Journal for Numerical Methods in Engineering* 86.11 (2011), pp. 1303–1321.
- [114] Pierre-Alexandre Beaufort, Christophe Geuzaine, and Jean-François Remacle. “Automatic surface mesh generation for discrete models-A complete and automatic pipeline based on reparametrization”. In: *Journal of Computational Physics* 417 (2020), p. 109575.
- [115] Michael S Floater. “Mean value coordinates”. In: *Computer aided geometric design* 20.1 (2003), pp. 19–27.
- [116] Matthias Nieser and Konrad Polthier. “Parameterizing singularities of positive integral index”. In: *IMA International Conference on Mathematics of Surfaces*. Springer. 2009, pp. 265–277.
- [117] Steve Smale. “Mathematical problems for the next century”. In: *The mathematical intelligencer* 20.2 (1998), pp. 7–15.
- [118] Theodor William Melnyk, Osvald Knop, and William Robert Smith. “Extremal arrangements of points and unit charges on a sphere: equilibrium configurations revisited”. In: *Canadian Journal of Chemistry* 55.10 (1977), pp. 1745–1761.
- [119] PD Dragnev. “On the separation of logarithmic points on the sphere”. In: *Approximation Theory X: Abstract and Classical Analysis*. Vanderbilt University Press, Nashville, TN. 2002, pp. 137–144.
- [120] DP Hardin and EB Saff. “Discretizing manifolds via minimum energy points”. In: *Notices of the AMS* 51.10 (2004), pp. 1186–1194.
- [121] Luigi Ambrosio and Xavier Cabré. “Entire solutions of semilinear elliptic equations in $\mathbb{R}^3$ and a conjecture of De Giorgi”. In: *Journal of the American Mathematical Society* 13.4 (2000), pp. 725–739.
- [122] Evguenii A Rakhmanov, Edward B Saff, and YM Zhou. “Minimal discrete energy on the sphere”. In: *Mathematical Research Letters* 1.6 (1994), pp. 647–662.
- [123] Edward B Saff and Amo BJ Kuijlaars. “Distributing many points on a sphere”. In: *The mathematical intelligencer* 19.1 (1997), pp. 5–11.
- [124] Enrique Bendito, A Carmona, Andrés M Encinas, José Manuel Gesto, A Gómez, C Mourino, and MT Sánchez. “Computational cost of the Fekete problem I: The Forces Method on the 2-sphere”. In: *Journal of Computational Physics* 228.9 (2009), pp. 3288–3306.
- [125] Keenan Crane, Mathieu Desbrun, and Peter Schröder. “Trivial connections on discrete surfaces”. In: *Computer Graphics Forum*. Vol. 29. 5. Wiley Online Library. 2010, pp. 1525–1533.

- [126] Nicolas Ray, Bruno Vallet, Laurent Alonso, and Bruno Levy. “Geometry-aware direction field processing”. In: *ACM Transactions on Graphics (TOG)* 29.1 (2009), pp. 1–11.
- [127] Gerik Scheuermann, Hans Hagen, Heinz Kruger, Martin Menzel, and Alyn Rockwood. “Visualization of higher order singularities in vector fields”. In: *Proceedings. Visualization’97 (Cat. No. 97CB36155)*. IEEE. 1997, pp. 67–74.
- [128] Gerik Scheuermann, Heinz Kruger, Martin Menzel, and Alyn P Rockwood. “Visualizing nonlinear vector field topology”. In: *IEEE Transactions on Visualization and Computer Graphics* 4.2 (1998), pp. 109–116.
- [129] Xavier Tricoche, Gerik Scheuermann, and Hans Hagen. “Higher order singularities in piecewise linear vector fields”. In: *The Mathematics of Surfaces IX*. Springer, 2000, pp. 99–113.
- [130] Tino Weinkauff, Holger Theisel, Kuangyu Shi, H-C Hege, and H-P Seidel. “Extracting higher order critical points and topological simplification of 3D vector fields”. In: *VIS 05. IEEE Visualization, 2005*. IEEE. 2005, pp. 559–566.
- [131] Shreeraj Jadhav, Harsh Bhatia, Peer-Timo Bremer, J Levine, Luis Gustavo Nonato, and Valerio Pascucci. *Consistent approximation of local flow behavior for 2d vector fields using edge maps*. Tech. rep. Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2011.
- [132] Michael J Borden, Steven E Benzley, and Jason F Shepherd. “Hexahedral Sheet Extraction.” In: *IMR*. 2002, pp. 147–152.
- [133] David Palmer, David Bommes, and Justin Solomon. “Algebraic representations for volumetric frame fields”. In: *ACM Transactions on Graphics (TOG)* 39.2 (2020), pp. 1–17.
- [134] Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. “Globally optimal direction fields”. In: *ACM Transactions on Graphics (ToG)* 32.4 (2013), pp. 1–10.
- [135] L Kronecker. “Über Systeme von Funktionen mehrerer Variablen, Monatsber. d”. In: *Kgl. Pr. Akad. Berlin* (1869), p. 695.
- [136] Michael S Floater and Kai Hormann. “Surface parameterization: a tutorial and survey”. In: *Advances in multiresolution for geometric modelling*. Springer, 2005, pp. 157–186.
- [137] Tadeusz Iwaniec and Jani Onninen. “Radó-Kneser-Choquet Theorem for simply connected domains (p -harmonic setting)”. In: *Transactions of the American Mathematical Society* 371.4 (2019), pp. 2307–2341.
- [138] Isadore Manuel Singer and John A Thorpe. *Lecture notes on elementary topology and geometry*. Springer, 2015.

- [139] Giorgio Marcias, Kenshi Takayama, Nico Pietroni, Daniele Panozzo, Olga Sorkine-Hornung, Enrico Puppo, and Paolo Cignoni. “Data-driven interactive quadrangulation”. In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), pp. 1–10.
- [140] Chi-Han Peng, Eugene Zhang, Yoshihiro Kobayashi, and Peter Wonka. “Connectivity editing for quadrilateral meshes”. In: *Proceedings of the 2011 SIGGRAPH Asia conference*. 2011, pp. 1–12.
- [141] Kenshi Takayama, Daniele Panozzo, Alexander Sorkine-Hornung, and Olga Sorkine-Hornung. “Robust and controllable quadrangulation of triangular and rectangular regions”. In: *Technical Report/ETH Zurich, Department of Computer Science 784* (2013).
- [142] Kenshi Takayama, Daniele Panozzo, and Olga Sorkine-Hornung. “Pattern-Based Quadrangulation for N-Sided Patches”. In: *Computer Graphics Forum*. Vol. 33. 5. Wiley Online Library. 2014, pp. 177–184.
- [143] Nico Pietroni, Stefano Nuvoli, Thomas Alderighi, Paolo Cignoni, and Marco Tarini. “Reliable Feature-Line Driven Quad-Remeshing”. In: *ACM Transactions on Graphics (TOG)* 40.4 (2021).
- [144] Marco Tarini. “Closed-form Quadrangulation of N-Sided Patches”. In: *arXiv preprint arXiv:2101.11569* (2021).
- [145] Edwin Catmull and James Clark. “Recursively generated B-spline surfaces on arbitrary topological meshes”. In: *Computer-aided design* 10.6 (1978), pp. 350–355.
- [146] Xiaopeng Zheng, Yiming Zhu, Na Lei, Zhongxuan Luo, and Xianfeng Gu. “Quadrilateral Mesh Generation III: Optimizing Singularity Configuration Based on Abel-Jacobi Theory”. In: *arXiv preprint arXiv:2007.07334* (2020).
- [147] Vladimir D Liseikin. *Grid generation methods*. Springer, 2017.
- [148] Christos Georgiadis. “Indirect generation of quadrilateral and hexahedral-dominant meshes: a hybrid approach based on cross-field guidance, topological labelling and patterns”. PhD thesis. UCL-Université Catholique de Louvain, 2021.
- [149] Jin Huang, Yiyong Tong, Hongyu Wei, and Hujun Bao. “Boundary aligned smooth 3D cross-frame field”. In: *ACM transactions on graphics (TOG)* 30.6 (2011), pp. 1–8.
- [150] N Kowalski, F Ledoux, and P Frey. “Block-structured hexahedral meshes for CAD models using 3d frame fields”. In: *Procedia Engineering* 82 (2014), pp. 59–71.
- [151] Yufei Li, Yang Liu, Weiwei Xu, Wenping Wang, and Baining Guo. “All-hex meshing using singularity-restricted field”. In: *ACM Transactions on Graphics (TOG)* 31.6 (2012), pp. 1–11.

- [152] Nicolas Ray, Dmitry Sokolov, and Bruno Lévy. “Practical 3D frame field generation”. In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), pp. 1–9.
- [153] Tengfei Jiang, Jin Huang, Yuanzhen Wang, Yiyong Tong, and Hujun Bao. “Frame field singularity correction for automatic hexahedralization”. In: *IEEE Transactions on Visualization and Computer Graphics* 20.8 (2013), pp. 1189–1199.
- [154] Timothy J Tautges. “The generation of hexahedral meshes for assembly geometry: survey and progress”. In: *International Journal for Numerical Methods in Engineering* 50.12 (2001), pp. 2617–2642.
- [155] Matthias Müller-Hannemann. “Shelling hexahedral complexes for mesh generation”. In: *Graph Algorithms And Applications 2*. World Scientific, 2004, pp. 475–507.
- [156] Matthew L Staten, Scott A Canann, and Steven J Owen. “BMSweep: locating interior nodes during sweeping”. In: *Engineering with Computers* 15.3 (1999), pp. 212–218.
- [157] Michael Kremer, David Bommers, Isaak Lim, and Leif Kobbelt. “Advanced automatic hexahedral mesh generation from surface quad meshes”. In: *Proceedings of the 22nd International Meshing Roundtable*. Springer, 2014, pp. 147–164.
- [158] Denis Kovacs, Ashish Myles, and Denis Zorin. “Anisotropic quadrangulation”. In: *Computer Aided Geometric Design* 28.8 (2011), pp. 449–462.
- [159] David Marcum and Frédéric Alauzet. “Aligned metric-based anisotropic solution adaptive mesh generation”. In: *Procedia Engineering* 82 (2014), pp. 428–444.
- [160] Muyang Zhang, Jin Huang, Xinguo Liu, and Hujun Bao. “A wave-based anisotropic quadrangulation method”. In: *ACM SIGGRAPH 2010 papers*. 2010, pp. 1–8.
- [161] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. “Anisotropic polygonal remeshing”. In: *ACM SIGGRAPH 2003 Papers*. 2003, pp. 485–493.
- [162] Daniele Panozzo, Enrico Puppo, Marco Tarini, and Olga Sorkine-Hornung. “Frame fields: Anisotropic and non-orthogonal cross fields”. In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), pp. 1–11.
- [163] Hans-Christian Ebke, Marcel Campen, David Bommers, and Leif Kobbelt. “Level-of-detail quad meshing”. In: *ACM Transactions on Graphics (TOG)* 33.6 (2014), pp. 1–11.