MULTI-BLOCK DECOMPOSITION AND MESHING OF 2D DOMAIN USING GINZBURG-LANDAU PDE

Jovana Jezdimirović Alexandre Chemin Jean François Remacle

Université catholique de Louvain, Louvain la Neuve, Belgium jovana.jezdimirovic@uclouvain.be

ABSTRACT

An in-depth method to generate multi-block decomposition of the arbitrary $2D$ domain using 2D cross fields solution of Ginzburg-Landau partial differential equation (PDE) is presented. It is relied on parameterization of multi-block decomposition of the domain, obtained by using particular PDE for the purpose of generating 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 adequate distribution of singularities, advocating its usage. Multi-block graph was generated by the separatrices and extraordinary vertices of the domain (singularities, corners and separatrices intersections) and obtained blocks were parameterized/remeshed. As a result, a mechanism to obtain multi-block structured all-quad mesh in automatic manner is developed.

Keywords: Ginzburg-Landau, cross fields, multi-block decomposition, all-quad mesh

1. INTRODUCTION

Multi-block structured meshes offer numerous advantages for mesh generation in general, as reported by an increasing number of authors [1, 2, 3, 4]. 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 multi-block decomposition and all 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 $\mathcal{C}_\Omega$ [5]. The cross field is then used for computing a multi-block decomposition 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 block decomposition that is used to build a finite element quadrilateral mesh with the use of an elliptic smoother. Figure 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. Multi-block decomposition is generated to be aligned with the cross field. Control over adding/reducing the number of blocks for creation/discarding of boundary layers is presented. The numerical methodology used avoids the problem of limit cycles in the reported examples from [6] and [7]. Finally, the true outcome of this paper is a fully functional multi-block mesh generator that is available in Gmsh [8], 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 [9] and [10].

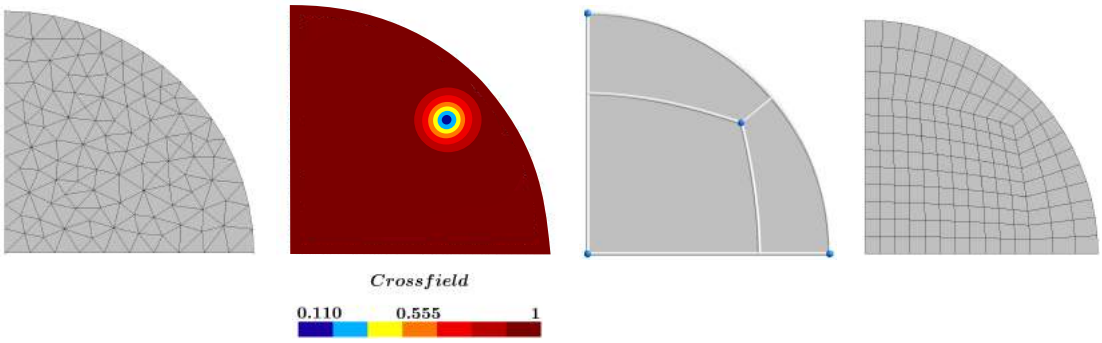

Figure 1: Simplicial mesh on Ω, computed cross field, generated separatrices and final quad mesh (from left hand side to right hand side respectively)

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 [11, 12]. When it comes to developments in correlation with our work, similarly to [3] and [6], we used a PDE based approach for multi-block decomposition purpose. In our case, more in depth developed procedures with extension to surfaces and specifities of the chosen PDE are shown. To avoid possible misalignments in the resulting parameterization, contrary to [13], multi-block decomposition remained aligned with the cross field. The techniques used for computing the cross field are from on [5] and they may seem to be similar to [14], 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 [14] and [15], 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 multiblock decompositions.

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, $\mathbf{c} = \{\mathbf{u}_k\}_{k\in[|1,4|]}$. 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 2.

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 $\mathcal{C}_\Omega = \{\mathbf{c}(\mathbf{x}), \mathbf{x} \in \Omega\}$.

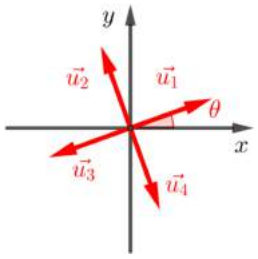

Figure 2: Cross definition

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 3).

Let's define $\mathcal{C}_\mathcal{U}$ as an 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 quadrilateral multi-block decomposition of Ω by tracing integral lines of $\tilde{\mathcal{C}}_\Omega$, represented in figure 4, which are identical to 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 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.

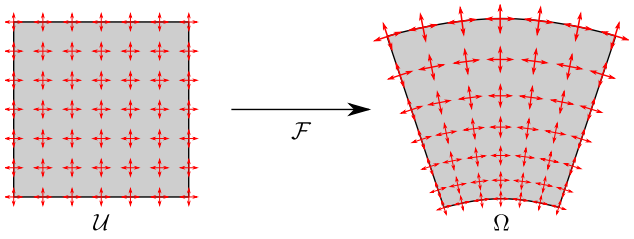

Figure 3: Conformal transformation between unitary square and physical domain with associated cross fields

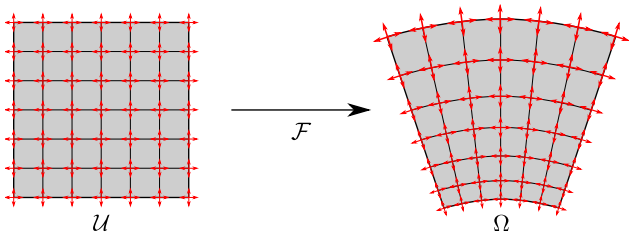

Figure 4: Quadrilateral multi-block decomposition of domain Ω obtained by propagating integral lines of $\mathcal{C}_\Omega$

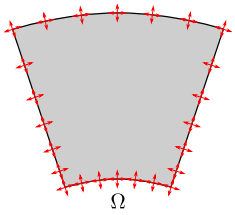

Figure 5: Reference problem for 2D cross field computation

The quadrilateral multi-block decomposition 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:

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

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 :

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

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 $\mathbf{0}$:

$$[\tilde{\mathbf{u}}_1, \tilde{\mathbf{u}}_2] = \nabla_{\tilde{\mathbf{u}}_2}\tilde{\mathbf{u}}_1 \nabla_{\tilde{\mathbf{u}}_1}\tilde{\mathbf{u}}_2 = \nabla\tilde{\mathbf{u}}_1 \cdot \tilde{\mathbf{u}}_2 - \nabla\tilde{\mathbf{u}}_2 \cdot \tilde{\mathbf{u}}_1 = \mathbf{0}. \tag{3}$$

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

$$\nabla(\log(l)) = \begin{bmatrix}\theta_{,y}\\ -\theta_{,x}\end{bmatrix} \text{ on } \Omega. \tag{4}$$

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 multi-block decomposition.

Note that $H = \log(l)$ is defined in [16] 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.

A multi-block decomposition of a domain Ω will be done 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 generate the multi-block decomposition.

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:

$$\vec{u} = (\cos 4\theta, \sin 4\theta), \quad \theta \in [0, \frac{\pi}{2}). \tag{5}$$

In this representation, $\vec{u}$ is invariant by a rotation of $\frac{\pi}{2}$, thus represents a cross with 4 orthogonal branches, as shown on figure 6 and figure 7.

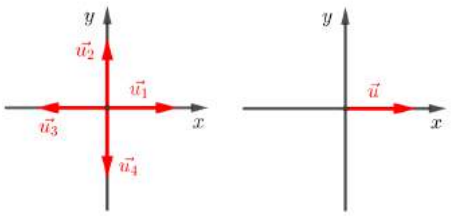

Figure 6: Reference cross (left) and its representation with $2D$ vector $\vec{u}$ (right)

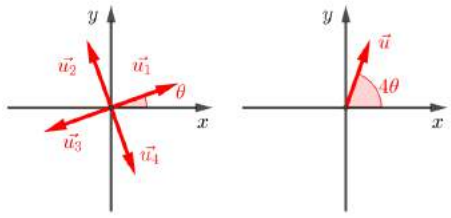

Figure 7: Rotation of reference cross by θ (left) and its representation with $2D$ vector $\vec{u}$ (right)

4.2 Computing 2D cross field through partial differential equations

The idea of computing a cross field $\vec{u} = (u_1, u_2)$ 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 $\vec{u} \in (H_1(\Omega))^2$ and simply minimize the Dirichlet energy:

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

with appropriate boundary conditions. Minimizing E^D is equivalent to solve $\nabla^2 \vec{u} = 0$. 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 block decomposition, 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 $\vec{u}$ actually leaves S^1 away from boundaries and is not a cross anymore. There are two possible options to force $\vec{u}$ to stay in S^1: (i) choosing $\vec{u} \in S^1$ *explicitly* or (ii) choosing $\vec{u} \in (H_1(\Omega))^2$ and penalize $\vec{u}$ away from S^1.

The first approach thus consists in choosing $\vec{u} \in S^1$ explicitly and writes $\vec{u}(\theta) = e^{i\theta}$. In this case, $\vec{u}$ 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,

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

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 [17]. 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-Landay energy functional:

$$E^{GL}(\vec{u}) = E^D + E^P = \frac{1}{2}\int_\Omega |\nabla \vec{u}|^2 + \frac{1}{4\epsilon^2}\int_\Omega \left(|\vec{u}|^2 - 1\right)^2, \tag{6}$$

where the parameter ϵ has a dimension of length and in literature is known as coherence length [16]. Energy (6) contains two terms: the standard Dirichlet energy and a penalization. The only minimizer of the Ginzburg-Landau functional is solution of:

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

The only vector field that satisfies both $\nabla^2\vec{u} = 0$ and $|\vec{u}| = 1$ is the constant vector field. Consequently, whenever $\epsilon \neq 0$, the Ginzburg-Landau formulation cannot force $\vec{u} \in S^1$ 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 $\vec{u} \in H_1(\Omega, S^1)$ i.e. choosing $\vec{u} \in S^1$ while keeping both its components. In this context, it can be shown [16] that the only minimizer of the Dirichlet energy E^D is solution of

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

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

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

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 crossfields will be computed by minimizing energy functional defined by the equation 6. Crouzeix-Raviart finite elements are used for the interpolation and a Newton-Raphson scheme for solving the nonlinear problem (computational details in [5]), obtained result is shown in the figure 8.

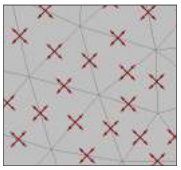

Figure 8: Generated crosses on triangulated surface Ω

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:

$$\chi = 2 - 2g - b. \tag{7}$$

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

$$\chi = n - n_e + n_f. \tag{8}$$

According to (7) and (8) the mesh on the surface with Euler characteristic $\chi \neq 0$ will have irregular vertices. As shown in [5], 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:

$$index(x_i) = \frac{1}{2\pi}\oint_{C_i} d\theta \tag{9}$$

where θ is an angular reference and C_i is a closed curve on the surface Ω containing 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 (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 four, 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:

$$\sum_i index(x_i) = \chi(\Omega). \tag{10}$$

According to [16], the result of the minimization of the Ginzburg-Landau energy (6) in $2D$ supports coexistence of $index(x_i) = \pm\frac{1}{4}$, as shown on figure 9.

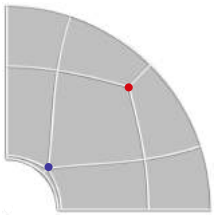

Figure 9: Coexistance of positive singularity (red) and negative singularity (blue) in $2D$

In a special case, following [18], the result of (6) for $\vec{u} \in S^1/C_N$, corresponds to the elliptic Fekete points on a sphere (figure 10).

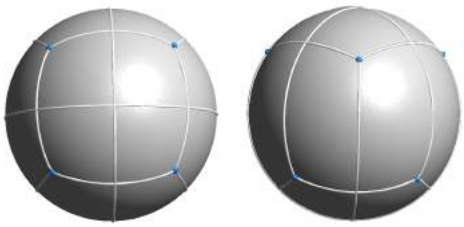

Figure 10: Singular points forming an anticube on the sphere

5. TRACING SEPARATRICES

As already reported [3, 6, 7, 13] the tracing of separatrices of computed cross field accomplishes the goal of domain partitioning. This section describes 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, [3, 18]. 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 $||\vec{u}||$, 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 11 or 12. Finding values of a cross field in these points is done by linear interpolation, similarly as in [3].

$$\theta_{P_i} = \theta_{P_i S_i} \pm \alpha \tag{11}$$

$$\theta_{Q_i} = \theta_{Q_i S_i} \pm \alpha \tag{12}$$

with α representing the tolerance criteria.

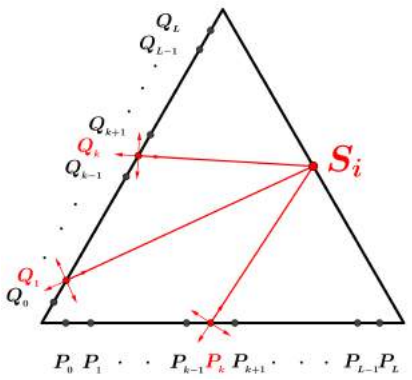

Figure 11: Obtaining separatrices on C_i

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 $T = T_i - \{C_1, ..., C_n\}$ following the adequate direction of the cross field until stopping criteria is fulfilled. We used the propagation scheme described in [3] relying on Heun's (a variation of Runge-Kutta 2) numerical scheme:

$$P'_{i+1} = P_i + h'_i \cdot \vec{u_i}(P_i) \tag{13}$$

$$\vec{d_i} = \frac{\vec{u_i}(P_i) + \vec{u_j}P'_{i+1}}{2} \tag{14}$$

$$P_{i+1} = P_i + h_i \cdot \vec{d_i} \tag{15}$$

where P_i is no-singular point on critical element C_i; $\vec{u_i}$ and $\vec{u_j}$ 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 on figure 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 $\vec{p_i} = \overrightarrow{S_i P_i}$. Further on, the information about the direction $\vec{u_j}$, at the cross at point P'_{i+1} is used to obtain direction $\vec{d_i}$ generating point P_{i+1} and allowing further propagation in the same manner.

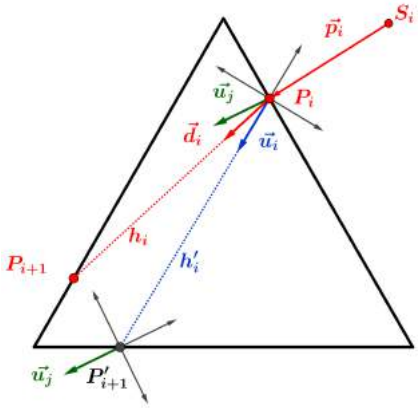

Figure 12: Separatrices propagation method

5.3 Stopping criteria for separatrices propagation

Separatrices generated in the manner described above are traced until they reached critical patch E_{pi} (figure 13) or 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.

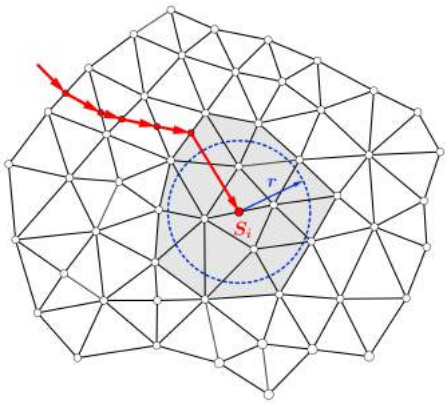

Figure 13: Critical patch

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 separatrice from two different singular points (as shown in figure 14).

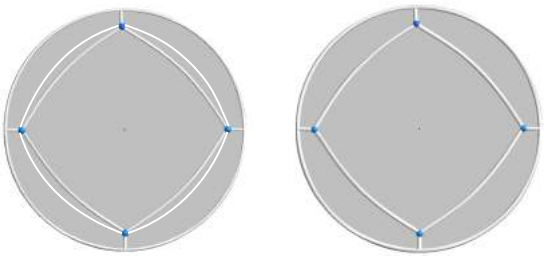

Figure 14: Multi-block decomposition of a circle: separatrices before (left) and after cleaning (right)

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 15-16. 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 15 is a case where the set of separatrices intersected is empty. The block 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).

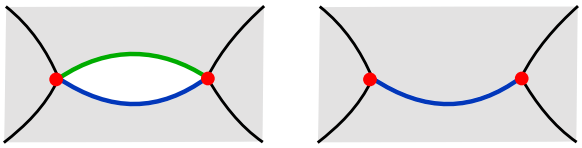

Figure 15: An empty set of separatrices' intersection (before and after)

Figure 16 is a case where the set of separatrices intersected 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).

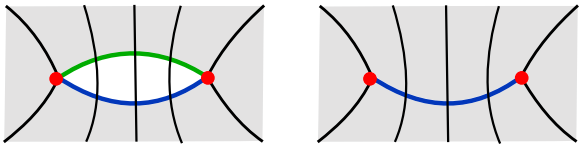

Figure 16: Not an empty set of separatrices' intersection (before and after)

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

The reason for taking into account separatrices intersections is demonstrated in figures 17-18: 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).

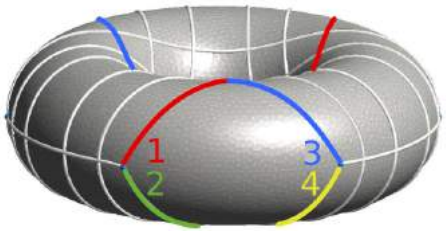

Figure 17: Multi-block decomposition of a torus

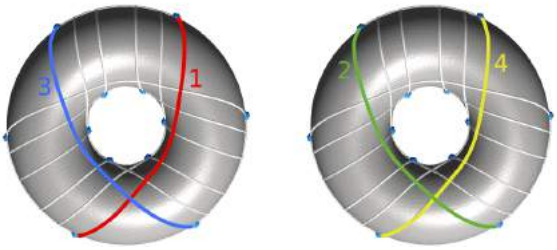

Figure 18: Overview of intersecting separatrices

6. EXAMPLES OF MULTI-BLOCK DECOMPOSITION

In this section we present the results obtained by applying Ginzburg-Landau PDE for the purpose of generating multi-block decomposition of the given domain Ω. By the definition of separatrices, each block 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 multi-block decomposition on:

- geometrical properties of the domain (figures 19/20)
- the type of boundary conditions imposed (figures 21/22)
- the value of coherence length parameter ϵ (figure 23).

Depending on requirements, we can add (remove) a boundary layer (as shown in figures 21/22) by imposing weak or strong boundary conditions. Using different values for global coherence length ϵ, we can generate different multi-block decompositions of the same domain. As shown on the figure 23, both of the decompositions are valid.

Figure 19: Eccentricity $e = 0.71$

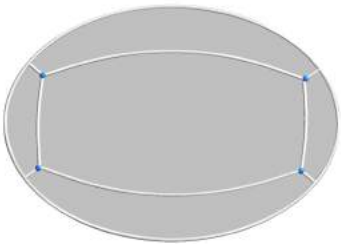

Figure 20: Eccentricity $e = 0.95$

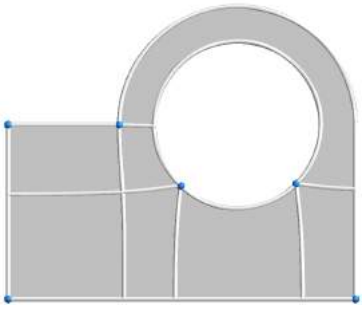

Figure 21: Imposed weak boundary conditions

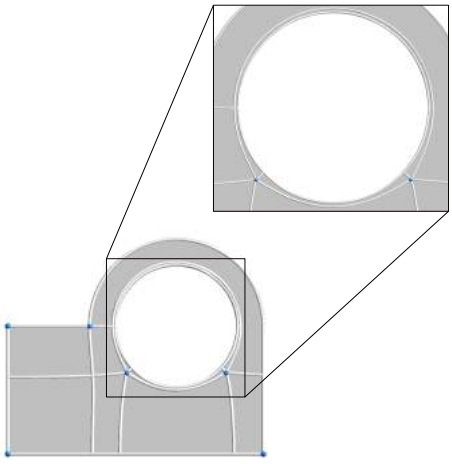

Figure 22: Imposed strong boundary conditions

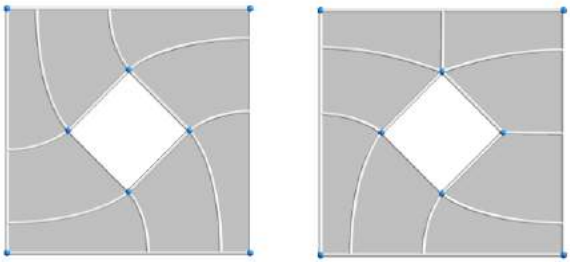

Figure 23: Coherence length $\epsilon = 0.001$ (left) and $\epsilon = 0.01$ (right)

We have chosen a few examples (figures 24-29) to demonstrate abilities of our algorithm for creating the multi-block decomposition of topologically and/or geometrically challenging domains.

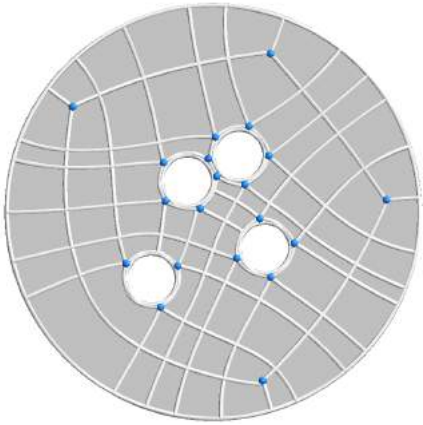

Figure 24: Euler characteristic $\chi = -3$

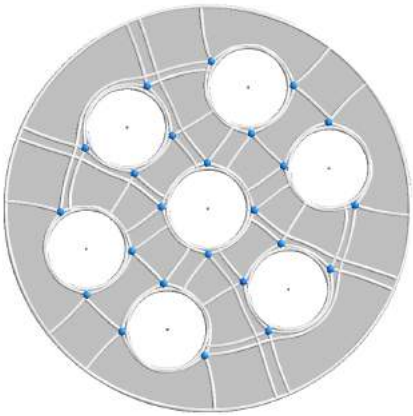

Figure 25: Euler characteristic $\chi = -6$

Following [5], generating cross field using Ginzburg-Landau functional has many desirable properties for meshing purposes. Recent reports on cross field generation ([6, 7]), 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 multi-block decomposition of the domain. The decompositions we obtained, on reported domains by [6] and [7], are shown in figures 30 and 31.

Figure 26: Closed manifold 1

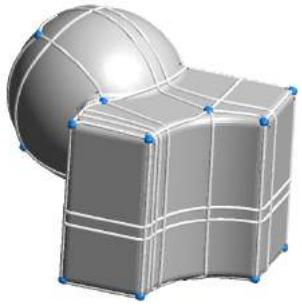

Figure 27: Closed manifold 2

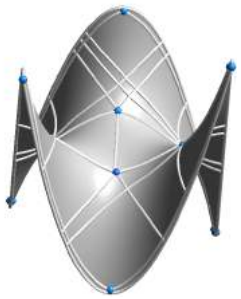

Figure 28: Open manifold

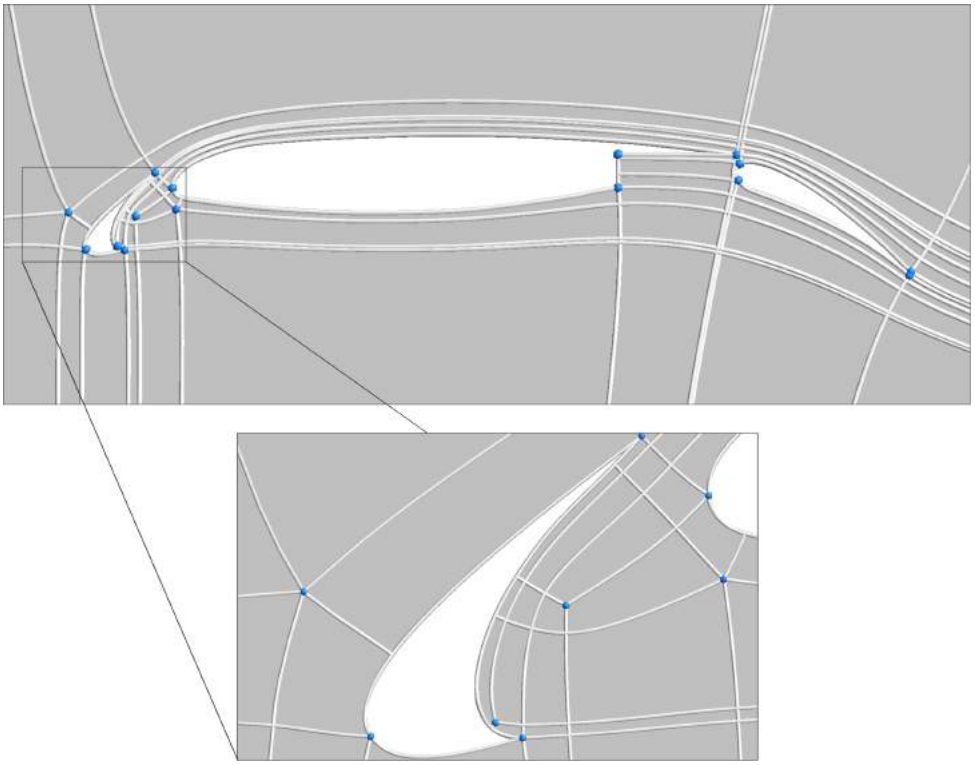

Figure 29: Multi-block decomposition of the wing

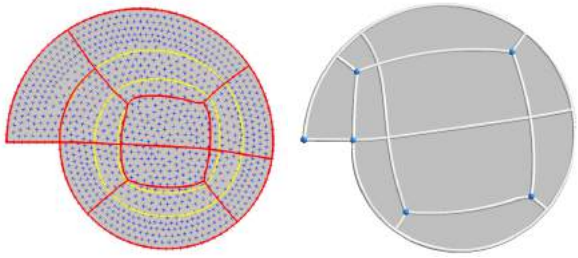

Figure 30: Multi-block decomposition by [6] (left) and by our approach (right)

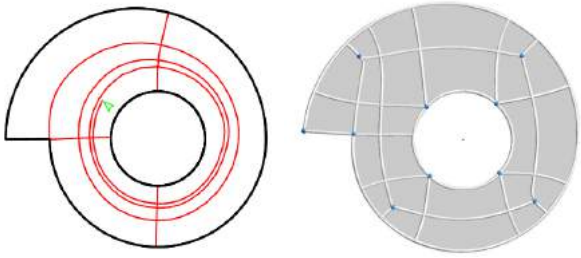

Figure 31: Multi-block decomposition by [7] (left) and by our approach (right)

7. PLANAR GRAPH EMBEDDING OF THE MULTI-BLOCK DECOMPOSITION

For the purpose of the efficient and practical representation of complex domains Ω, according to [10], we will generate the planar graph embedding based on its multi-block decomposition (figure 32). To do so, we will use extraordinary vertices T_i, $i \in \{1, \cdots, n\}$, 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 $V = \{T_1, T_2, \ldots, T_n\}$ and corresponding edges $E = \{\{T_1, T_2\}, \{T_2, T_3\}, \ldots, \{T_j, T_n\}\}$. For the purpose of meeting well-defined connectivity (figure 33) 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, where from graph edges E are derived. Information on orientation and data structure ensured that each edge will be traversed only once and all quads $Q_i = \{T_i, T_j, T_k, T_t\}$ will be extracted. The used algorithm 2, showed below, is the adaptation of work described by [19].

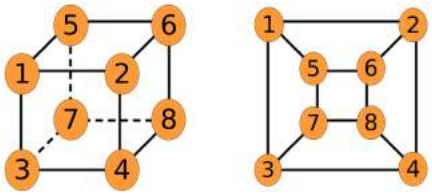

Figure 32: Illustration of a multi-block decomposition (left) and its planar graph embedding (right)

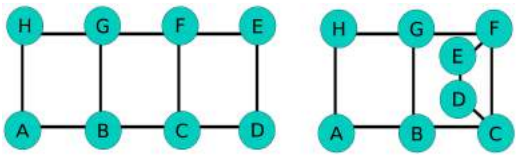

Figure 33: Importance of orientation data: connected (left) and disconnected graph (right)

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 $Q = Q_1 \bigcup Q_2 \bigcup \ldots \bigcup Q_n$, 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, blocks defined by a planar graph embedding, are directly used for applying algebraic (1) and afterwards (2) elliptic grid generation (figure 34). The result obtained is all quad mesh. Theoretical and computational details on these methods are given in the following sections.

8.1 Transfinite bilinear interpolation

For the aim of refinement of each block, transfinite interpolation (TFI) is used as a, according to [3, 20, 21], 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 $\vec{c}_1(u)$, $\vec{c}_3(u)$, $\vec{c}_2(v)$ and $\vec{c}_4(v)$ (shown in figure 35), the position of point $\vec{X}_i(u,v)$ in the given domain is defined by the equation 16:

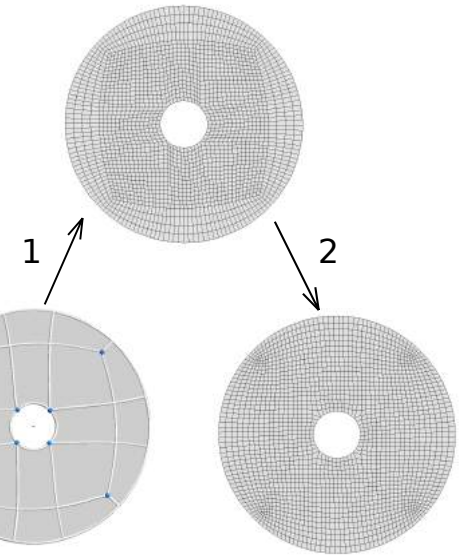

Figure 34: Illustration of the pipleine for the planar cases

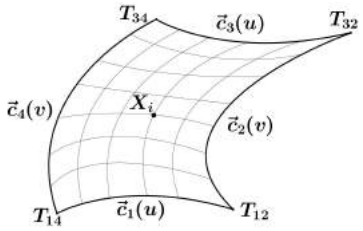

Figure 35: TFI grid points on physical domain Q_i

$$\begin{aligned}\vec{X}_i(u,v) &= (1-v)\cdot\vec{c}_1(u)\\ &+ v\cdot\vec{c}_3(u)+(1-u)\cdot\vec{c}_2(v)+u\cdot\vec{c}_4(v)\\ &- \big[(1-u)\cdot(1-v)\cdot\vec{T}_{12}\\ &\quad +u\cdot v\cdot\vec{T}_{34}\\ &\quad +u\cdot(1-v)\cdot\vec{T}_{14}\\ &\quad +(1-u)\cdot v\cdot\vec{T}_{32}\big].\end{aligned} \tag{16}$$

As a result, a structured quad mesh $M_t = (V, E, Q)$, with vertices V, edges E and corresponding quadrilaterals Q is generated (figure 36 left).

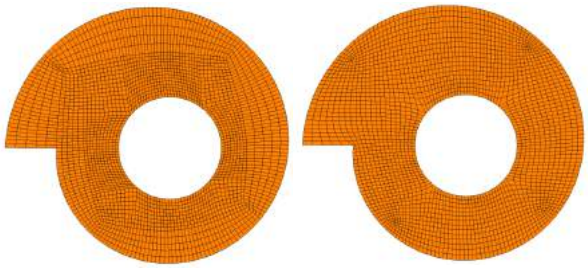

Figure 36: TFI mesh (left) and mesh smoothing (right)

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 36 right).

8.2 Grid smoothing

To insure robustness and computational efficiency of a PDE based algorithm, which can be adopted for an unstructured mesh, our approach followed the work described in [22]. This work represents the Winslow smoothing [23] on 2D unstructured mesh and its based on solving the second-order nonlinear elliptic partial differential equations:

$$\begin{cases} g_{22}x_{\xi\xi} - 2g_{12}x_{\xi\eta} + g_{11}x_{\eta\eta} = 0 \\ g_{22}y_{\xi\xi} - 2g_{12}y_{\xi\eta} + g_{11}y_{\eta\eta} = 0 \end{cases} \tag{17}$$

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

$$\begin{cases} g_{11} = x_\xi x_\xi + y_\xi y_\xi \\ g_{12} = x_\xi x_\eta + y_\xi y_\eta \\ g_{22} = x_\eta x_\eta + y_\eta y_\eta. \end{cases} \tag{18}$$

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

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 [24]:

$$\eta(q) = max\Big(1 - \frac{2}{\pi} max\Big(|\frac{\pi}{2} - \alpha_i|\Big), 0\Big). \tag{19}$$

Improvements of the mesh quality using Winslow smoother are shown on a few examples in the table below. Notations used for values of the minimum and average quality of elements are respectively η_ω and $\overline{\eta}$, 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.

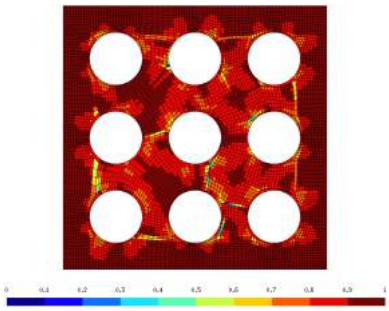

Figure 37: Quality of a mesh generated using TFI

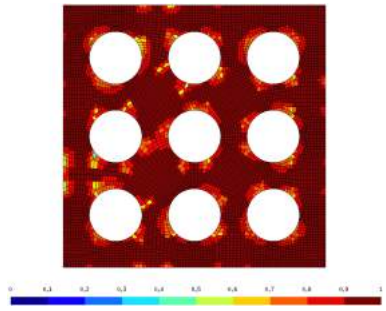

Figure 38: Quality of a mesh generated using Winslow smoother

Figure	h	Algorithm	Mesh quality		
			η_ω	$\overline{\eta}$	$\eta_{0.9}$
Fig 37/38	0.01	TFI	0.01	0.90	0.59
		Winslow	0.47	0.94	0.80
Fig 24	0.05	TFI	0.00	0.94	0.82
		Winslow	0.52	0.93	0.79
Fig 25	0.01	TFI	0.14	0.92	0.73
		Winslow	0.48	0.93	0.80

Table 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 multi-block decomposition can be done 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 39 points out the main steps of the workflow.

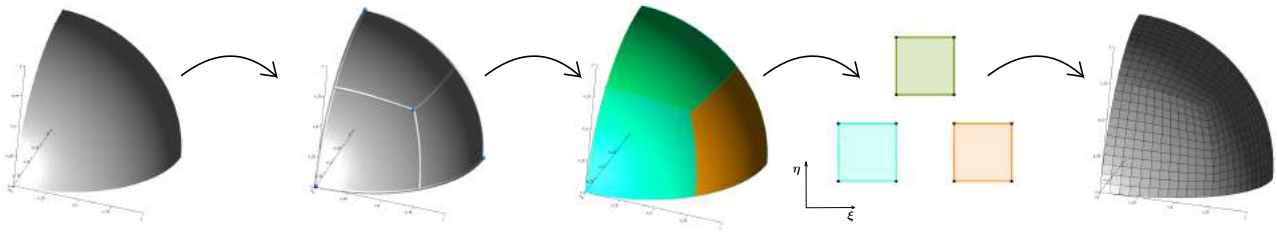

Figure 39: Illustration of the pipeline for the manifold cases

9.1 Parametrization of the domain

There are many ways to parameterize arbitrary 2D manifolds Ω [25, 26, 27] 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 multi-block decomposition. The choice made here is to define each quad of the multi-block decomposition as a subdomain Ω_i and parameterize it independently following the method proposed in [27] (figure 40). The parameterization relies on mean value coordinates [28] which guaranties a one to one parameterization of each subdomain. For most of the multi-block decompositions obtained, each block can be parameterized with only one atlas, but it can happen that it is necessary to split a quad block in subdomains in order to get a proper parameterization. This kind of cases is handled by the methodology proposed in [27].

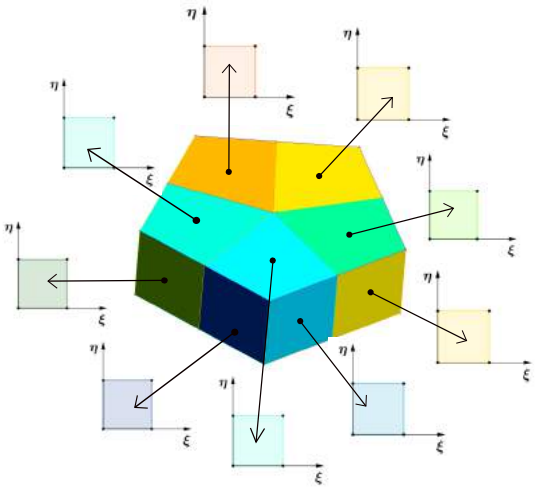

Figure 40: Illustration of the proposed parameterization

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 good quality all quad mesh.

It is important to note that, due to the choice of independent parameterization of each quad block, 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 39. First, the mutli-block decomposition is generated. Then each block is parameterized independently. On each one of them, an initial grid is created in the parametric space, smoothed with Winslow smoother [22] and then mapped back to Ω in the physical space.

10. DISCUSSION

Due to the behaviour 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 41 - 42). This issue is resolved by further propagation (figure 41), or, in cases when neither one of separatrices from the triangular block can not be further propagated, removal of the separatrice (figure 42).

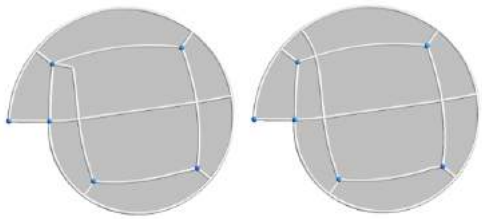

Figure 41: Before and after applying the algorithm for triangular block cleaning (repropagating)

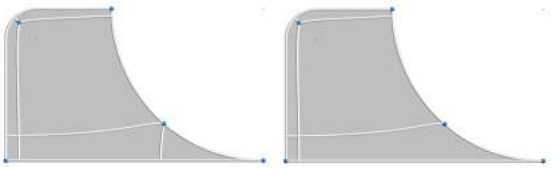

Figure 42: Before and after applying the algorithm for triangular block cleaning (removing)

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 [5] is not competitive with existing cross fields generators. Addressing this issue is the topic of the current work.

For the future directions of the work, the authors will address the thorough examinations of: optimization of performance time, robustness with large-scale numerical examples, reported problem of limit cycles existence and meshing with varying elements' sizes (figure 43).

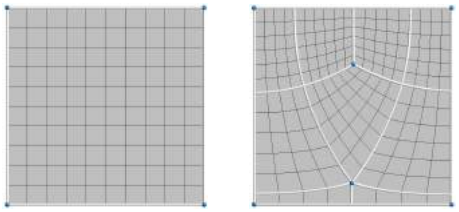

Figure 43: Uniform (left) and non-uniform quad mesh (right)

11. CONCLUSION

This paper presents a contribution in developing an algorithm for generating multi-block decomposition and all 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 [29, 30]. 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 the least, we demonstrated how to use our method to create all quad meshes in an automatic manner and made it available in Gmsh, the open source finite element mesh generator.

12. ACKNOWLEDGMENTS

The present study was carried out in the framework of the research project "Hextreme", funded by the European Research Council (ERC-2015-AdG-694020) and hosted at the Université catholique de Louvain.

Appendix A WINSLOW SMOOTHER

Using the finite difference discretization, equations 17 can be written, for each node n of the mesh, as:

$$D_n(\mathbf{x}) = G_{22}D_{\xi\xi}(\mathbf{x}_n) - 2G_{12}D_{\xi\eta}(\mathbf{x}_n) + G_{11}D_{\eta\eta}(\mathbf{x}_n) = 0, \tag{20}$$

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

$$\begin{cases} G_{11} = D_\xi(\mathbf{x}_n) \cdot D_\xi(\mathbf{x}_n) \\ G_{12} = D_\xi(\mathbf{x}_n) \cdot D_\eta(\mathbf{x}_n) \\ G_{22} = D_\eta(\mathbf{x}_n) \cdot D_\eta(\mathbf{x}_n), \end{cases} \tag{21}$$

with values $D_\xi, D_\eta, D_{\xi\xi}, D_{\xi\eta}$ 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 (22 - 23) are derived:
Approximations for $v_n = 4$:

$$\begin{cases} D_\xi(\mathbf{x}_n) = \dfrac{2}{v_n}\displaystyle\sum_{m=0}^{v_n-1}(\mathbf{x}_m - \mathbf{x}_n)\cos\theta_m \\ D_\eta(\mathbf{x}_n) = \dfrac{2}{v_n}\displaystyle\sum_{m=0}^{v_n-1}(\mathbf{x}_m - \mathbf{x}_n)\sin\theta_m \\ D_{\xi\xi}(\mathbf{x}_n) = \dfrac{4}{v_n}\displaystyle\sum_{m=0}^{v_n-1}(\mathbf{x}_m - \mathbf{x}_n)\cos^2\theta_m \\ D_{\xi\eta}(\mathbf{x}_n) = \dfrac{2}{v_n}\displaystyle\sum_{m=0}^{v_n-1}(\widehat{\mathbf{x}_m} - \mathbf{x}_n)\cos\widehat{\theta_m}\sin\widehat{\theta_m} \\ D_{\eta\eta}(\mathbf{x}_n) = \dfrac{4}{v_n}\displaystyle\sum_{m=0}^{v_n-1}(\mathbf{x}_m - \mathbf{x}_n)\sin^2\theta_m \end{cases} \tag{22}$$

where:

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

and $\widehat{\mathbf{x}_m}$ are associated to the diagonal nodes, as shown in figures 44-45.

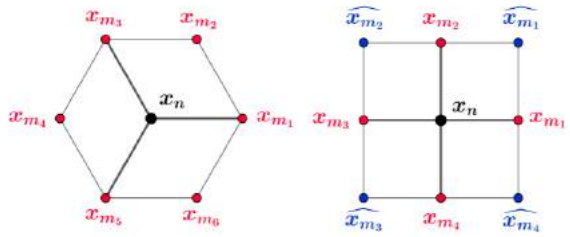

Figure 44: Dependency on v_n of $\mathbf{x}_n$ used for approximation of 3 and 4-valent nodes (left to right)

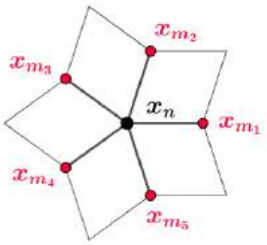

Figure 45: Dependency on v_n of $\mathbf{x}_n$ used for approximation of 5-valent nodes

Considering 3-valant nodes, equations demonstrated below are considering 6 neighbouring nodes (figure 44) and therefore the valant degree $v_n = 3$ rises up to $v_n = 6$, more detailed in [22].

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

$$\begin{cases} D_\xi(\mathbf{x}_n) = \dfrac{2}{v_n}\displaystyle\sum_{m=0}^{v_n-1}(\mathbf{x}_m - \mathbf{x}_n)\cos\theta_m \\ D_\eta(\mathbf{x}_n) = \dfrac{2}{v_n}\displaystyle\sum_{m=0}^{v_n-1}(\mathbf{x}_m - \mathbf{x}_n)\sin\theta_m \\ D_{\xi\xi}(\mathbf{x}_n) = \dfrac{2}{v_n}\displaystyle\sum_{m=0}^{v_n-1}(\mathbf{x}_m - \mathbf{x}_n)(4\cos^2\theta_m - 1) \\ D_{\xi\eta}(\mathbf{x}_n) = \dfrac{8}{v_n}\displaystyle\sum_{m=0}^{v_n-1}(\mathbf{x}_m - \mathbf{x}_n)\cos\theta_m\sin\theta_m \\ D_{\eta\eta}(\mathbf{x}_n) = \dfrac{2}{v_n}\displaystyle\sum_{m=0}^{v_n-1}(\mathbf{x}_m - \mathbf{x}_n)(4\sin^2\theta_m - 1) \end{cases} \tag{23}$$

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 · x^{k+1}||_∞ > 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 x^{k+1} = 0
    Generate M_q^{k+1} from x^{k+1}
    k=k+1
end while
Write new smoothed mesh M_w = M_q^k
```

Appendix B LOCAL INTEGRABILITY OF THE CROSS FIELD

Starting from the definition of 2D cross field $\tilde{\mathcal{C}}_\Omega$ with the non uniform norms (equation 2) and replacing the corresponding values in equation 3, we obtain the result computed in 24, which shows that for $l \neq 0$, $\tilde{\mathcal{C}}_\Omega$ is integrable if l verifies the condition 4.

$$
\begin{aligned}
[\tilde{\mathbf{u}}_1, \tilde{\mathbf{u}}_2] &= \begin{bmatrix} l_{,x} \cdot \cos\theta - l \cdot \sin\theta \cdot \theta_{,x} & l'_y \cdot \cos\theta - l \cdot \sin\theta \cdot \theta_{,y} \\ l_{,x} \cdot \sin\theta + l \cdot \cos\theta \cdot \theta_{,x} & l'_y \cdot \sin\theta + l \cdot \cos\theta \cdot \theta_{,y} \end{bmatrix} \cdot \begin{bmatrix} -l \cdot \sin\theta \\ l \cdot \cos\theta \end{bmatrix} \\
&- \begin{bmatrix} -l_{,x} \cdot \sin\theta - l \cdot \cos\theta \cdot \theta_{,x} & -l'_y \cdot \sin\theta - l \cdot \cos\theta \cdot \theta_{,y} \\ l_{,x} \cdot \cos\theta - l \cdot \sin\theta \cdot \theta_{,x} & l'_y \cdot \cos\theta - l \cdot \sin\theta \cdot \theta_{,y} \end{bmatrix} \cdot \begin{bmatrix} l \cdot \cos\theta \\ l \cdot \sin\theta \end{bmatrix} \\
&= \begin{bmatrix} -l \cdot l_{,x} \cdot \sin\theta \cdot \cos\theta + l^2 \cdot \sin^2\theta \cdot \theta_{,x} + l \cdot l_{,y} \cdot \cos^2\theta - l^2 \cdot \sin\theta \cdot \cos\theta \cdot \theta_{,y} \\ -l \cdot l_{,x} \cdot \sin^2\theta - l^2 \cdot \sin\theta \cdot \cos\theta \cdot \theta_{,x} + l \cdot l_{,y} \cdot \sin\theta \cdot \cos\theta + l^2 \cdot \cos^2\theta \cdot \theta_{,y} \end{bmatrix} \\
&- \begin{bmatrix} -l \cdot l_{,x} \cdot \sin\theta \cdot \cos\theta - l^2 \cdot \cos^2\theta \cdot \theta_{,x} - l \cdot l_{,y} \cdot \sin^2\theta - l^2 \cdot \sin\theta \cdot \cos\theta \cdot \theta_{,y} \\ l \cdot l_{,x} \cdot \cos^2\theta - l^2 \cdot \sin\theta \cdot \cos\theta \cdot \theta_{,x} + l \cdot l_{,y} \cdot \sin\theta \cdot \cos\theta - l^2 \cdot \sin^2\theta \cdot \theta_{,y} \end{bmatrix} \\
&= \begin{bmatrix} l^2 \cdot \theta_{,x} + l \cdot l_{,y} \\ l^2 \cdot \theta_{,y} - l \cdot l_{,x} \end{bmatrix} = l^2 \begin{bmatrix} \theta_{,x} + \dfrac{l_{,y}}{l} \\ \theta_{,y} - \dfrac{l_{,x}}{l} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\end{aligned}
\tag{24}
$$

References

[1] Shepherd J.F., Johnson C.R. "Hexahedral mesh generation constraints." *Engineering with Computers*, vol. 24, no. 3, 195–213, 2008

[2] Benzley S.E., Perry E., Merkley K., Clark B., Sjaardama G. "A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis." *Proceedings, 4th international meshing roundtable*, vol. 17, pp. 179–191. Sandia National Laboratories Albuquerque, NM, 1995

[3] Kowalski N., Ledoux F., Frey P. "A PDE based approach to multidomain partitioning and quadrilateral meshing." *Proceedings of the 21st international meshing roundtable*, pp. 137–154. Springer, 2013

[4] Chan J., Wang Z., Modave A., Remacle J.F., Warburton T. "GPU-accelerated discontinuous Galerkin methods on hybrid meshes." *Journal of Computational Physics*, vol. 318, 142–168, 2016

[5] Beaufort P.A., Lambrechts J., Henrotte F., Geuzaine C., Remacle J.F. "Computing cross fields A PDE approach based on the Ginzburg-Landau theory." *Procedia engineering*, vol. 203, 219–231, 2017

[6] Viertel R., Osting B. "An Approach to Quad Meshing Based on Harmonic Cross-Valued Maps and the Ginzburg–Landau Theory." *SIAM Journal on Scientific Computing*, vol. 41, no. 1, A452–A479, 2019

[7] Fogg H.J., Armstrong C.G., Robinson T.T. "Automatic generation of multiblock decompositions of surfaces." *International Journal for Numerical Methods in Engineering*, vol. 101, no. 13, 965–991, 2015

[8] Geuzaine C., Remacle J.F. "Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities." *International journal for numerical methods in engineering*, vol. 79, no. 11, 1309–1331, 2009

[9] Bommes D., Lévy B., Pietroni N., Puppo E., Silva C.T., Tarini M., Zorin D. "Quad Meshing." *Eurographics (STARs)*, pp. 159–182. 2012

[10] Campen M. "Partitioning surfaces into quadrilateral patches: a survey." *Computer Graphics Forum*, vol. 36, pp. 567–588. Wiley Online Library, 2017

[11] Campen M., Bommes D., Kobbelt L. "Quantized global parametrization." *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, 192, 2015

[12] Bommes D., Campen M., Ebke H.C., Alliez P., Kobbelt L. "Integer-grid maps for reliable quad meshing." *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, 98, 2013

[13] Myles A., Pietroni N., Zorin D. "Robust field-aligned global parametrization." *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, 135, 2014

[14] Ray N., Li W.C., Lévy B., Sheffer A., Alliez P. "Periodic global parameterization." *ACM Transactions on Graphics (TOG)*, vol. 25, no. 4, 1460–1485, 2006

[15] Diamanti O., Vaxman A., Panozzo D., Sorkine-Hornung O. "Integrable polyvector fields." *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, 38, 2015

[16] Bethuel F., Brezis H., Hélein F., et al. *Ginzburg-Landau Vortices*, vol. 13. Springer, 1994

[17] Bommes D., Zimmer H., Kobbelt L. "Mixed-integer quadrangulation." *ACM Transactions On Graphics (TOG)*, vol. 28, no. 3, 77, 2009

[18] Jezdimirović J., Chemin A., Beaufort P.A., Remacle J.F. "Elliptic Fekete points obtained by Ginzburg-Landau PDE." *Proceedings, 26th international meshing roundtable*. Sandia National Laboratories Albuquerque, NM, 2017

[19] Brinkmann G., McKay B.D., et al. "Fast generation of planar graphs." *MATCH Commun. Math. Comput. Chem*, vol. 58, no. 2, 323–357, 2007

[20] Thompson J.F., Soni B.K., Weatherill N.P. *Handbook of grid generation*. CRC press, 1998

[21] Mukherjee N. "CSALF-Q: A bricolage algorithm for anisotropic quad mesh generation." *Proceedings of the 20th International Meshing Roundtable*, pp. 489–509. Springer, 2011

[22] Knupp P.M. "Winslow smoothing on two-dimensional unstructured meshes." *Engineering with Computers*, vol. 15, no. 3, 263–268, 1999

[23] Winslow A.M. "Numerical solution of the quasi-linear Poisson equation in a nonuniform triangle mesh." *Journal of computational physics*, vol. 1, no. 2, 149–172, 1966

[24] Remacle J.F., Lambrechts J., Seny B., Marchandise E., Johnen A., Geuzainet C. "Blossom-Quad: A non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm." *International journal for numerical methods in engineering*, vol. 89, no. 9, 1102–1119, 2012

[25] Remacle J.F., Geuzaine C., Compere G., Marchandise E. "High-quality surface remeshing using harmonic maps." *International Journal for Numerical Methods in Engineering*, vol. 83, no. 4, 403–425, 2010

[26] Marchandise E., de Wiart C.C., Vos W., Geuzaine C., Remacle J.F. "High-quality surface remeshing using harmonic mapsPart II: Surfaces with high genus and of large aspect ratio." *International Journal for Numerical Methods in Engineering*, vol. 86, no. 11, 1303–1321, 2011

[27] Beaufort P.A., Geuzaine C., Remacle J.F. "Automatic surface mesh generation for discrete models. A complete and automatic pipeline." Submitted

[28] Floater M.S. "Mean value coordinates." *Computer aided geometric design*, vol. 20, no. 1, 19–27, 2003

[29] Vaxman A., Campen M., Diamanti O., Panozzo D., Bommes D., Hildebrandt K., Ben-Chen M. "Directional field synthesis, design, and processing." *Computer Graphics Forum*, vol. 35, pp. 545–572. Wiley Online Library, 2016

[30] Nieser M., Polthier K. "Parameterizing singularities of positive integral index." *IMA International Conference on Mathematics of Surfaces*, pp. 265–277. Springer, 2009