Horizon 2020
European Union funding
for Research & Innovation

E-Infrastructures

H2020-EINFRA-2015-1

EINFRA-5-2015: Centres of Excellence for computing applications

EoCoE

Energy oriented Center of Excellence for computing applications

Grant Agreement Number: EINFRA-676629

D5.1

Flux surface meshing software - Specification Report

Project and Deliverable Information Sheet

EoCoE	Project Ref:	EINFRA-676629
	Project Title:	Energy oriented Centre of Excellence
	Project Web Site:	http://www.eocoe.eu
	Deliverable ID:	D5.1
	Lead Beneficiary:	CEA (INRIA)
	Contact:	Hervé Guillard
	Contact's e-mail:	herve.guillard@inria.fr
	Deliverable Nature:	Report
	Dissemination Level:	PU*
	Contractual Date of Delivery:	M6 05/04/2016
	Actual Date of Delivery:	M13 30/11/2016
	EC Project Officer:	Carlos Morais-Pires

* - The dissemination level are indicated as follows: PU – Public, CO – Confidential, only for members of the consortium (including the Commission Services) CL – Classified, as referred to in Commission Decision 2991/844/EC.

Document Control Sheet

Document	Title :	Flux surface meshing software - Specification Report
	ID :	D5.1
	Available at:	http://www.eocoe.eu
	Software tool:	LaTeX
Authorship	Written by:	Hervé Guillard
	Contributors:	
	Reviewed by:	Yanick Sarazin

Table of Contents

1. Introduction

Small scale turbulence as well as large scale MHD instabilities in Tokamak plasmas are believed to be quasi bi-dimensional: the fluctuations scales in the parallel direction to the magnetic field are typically 3 to 4 orders of magnitude larger than those in the transverse plane. This property is used in many tokamak plasma simulation codes to have a different spatial resolution in the parallel and perpendicular direction. However the necessity to consider a sufficiently fine spatial resolution in the transverse direction (that can be of the order of the Larmor radius) make particularly attractive the use of grids-meshes which are adapted to the magnetic flux surfaces in each 2D poloidal plane while the third periodic toroidal direction can be discretized with great accuracy with Fourier series. The shape of the poloidal cross-section of the averaged flux surfaces in the poloidal planes is governed by magnetic equilibrium. These magnetic equilibria are themselves computed by dedicated equilibrium codes. The purpose of the WP5.1 task of the EoCoE project is to develop a generic software to construct 2D triangular/quadrangular meshes of the poloidal plane of a tokamak, which will be aligned to magnetic flux-surfaces for any prescribed magnetic equilibrium. This single software will gather some of the algorithms used in the tokamak fusion community and until now developed independently in several laboratories as well as some published algorithms developed in the meshing community.

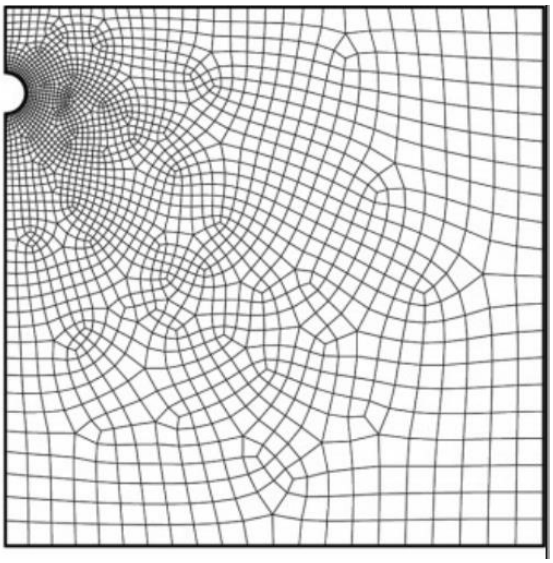
2. Mesh requirements of the simulation codes

This section is based on a questionnaire that has been send in November 2015 to the developers of several simulation codes used for tokamak plasma in the teams involved in the work package Fusion4Energy of EoCoE. From the answers to this questionnaire, the code to be considered are:

Code name	Purpose	Principal developer
SOLPS	plasma and neutral simulations of the edge and SOL	IPP Garching
FBGKI	Edge and SOL plasma simulations	LJAD and Inria, Nice
GYSELA	gyrokinetic code for turbulence studies in the core	IRFM Cadarache
TOKAM3X	fluid code for edge turbulence	IRFM Cadarache
JOREK	MHD stability studies	IRFM Cadarache
MISHKA	linear MHD stability studies	IRFM Cadarache
HELENA	Equilibrium code	IRFM Cadarache
SOLEDGE	plasma and neutral simulation of the edge and SOL	IRFM Cadarache
PlaTo	Edge and SOL plasma simulations	LJAD and Inria, Nice

These 9 codes use different set of governing equations, different models as well as different numerical methods and meshes. In the following, we adopt the following definitions:

Structured mesh – or grid: A structured mesh in 2D (we will use alternatively the term grid) is one in which the cells (quadrilaterals) can be arranged in an I x J array where I, J are the grid dimensions. The term "structured" refers to the structure provided to the organization of the cells within that array. In particular, in a grid the neighbors of a cell are known implicitly: an interior point (or an interior cells) at (i,j) has neighbors at (i+1,j), (i−1,j), (i,j−1), (i,j+1). Therefore, in a grid (except on the boundary) an interior point (or cells) has exactly 4 neighbors. This structure contrasts with an unstructured mesh in which a connectivity table has to be maintained to find the neighbors. A structured grid is made of quadrilaterals. However, a mesh made solely of quadrilaterals is not obligatory a structured one as shown by the figure on the right since the number of neighbors of a node (resp. cell) is not constant and a connectivity table has to be maintained to keep track of the neighbors of one node (resp. cell).



Block-Structured mesh: This type of mesh relies on the partition of the computational domain into (a small number of) sub-domains where each sub-domain is discretized by a structured grid. Block-structured meshes inherit from structured grids many advantages. In particular, within each sub-region there is no need to construct a connectivity table since the neighbor connectivity is implicit in the numbering of the cells. However, a connectivity table between the different sub-regions has to be constructed and stored. Block-structured meshes allow for more flexibility than structured meshes for the representation of complex geometries. The difficult part of this meshing technique lies in the appropriate definition of the sub-domains and many algorithms used for the construction of block-structured meshes require some manual inputs from the user to define these sub-regions. The problem of defining a structured grid inside the regions is considerably simpler and many algorithms can be used for this purpose once the boundaries of the sub-domains have been identified and discretized.

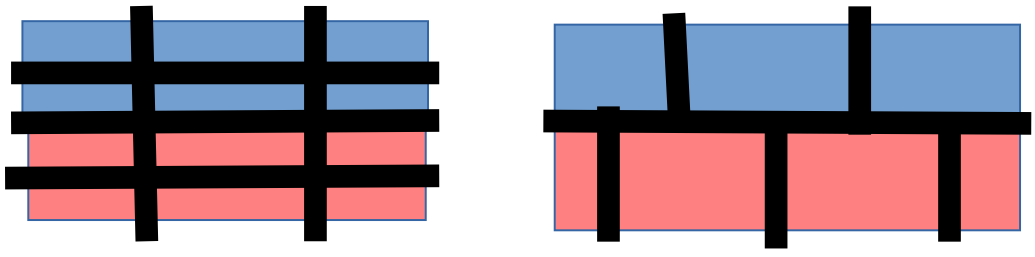
Unstructured meshes: An unstructured mesh is a tessellation[1] of a domain of the plane by simple polygonal shape in an irregular pattern. In practice, the polygons are very often triangles or quadrangles or a mixture of the two. In contrast to structured grids, unstructured meshes need a list storing neighboring relations between the mesh elements. Thanks to the Delaunay algorithm, the generation of unstructured meshes of triangles of any 2D polygonal domain is nowadays totally automatic and a very large number of software (including many open source codes) is available. Some recent algorithms have also been published for the generation of unstructured quadrangular meshes.

1 A tesselation is the tiling of a subdomain of the plane using one or more geometric shapes called tiles, with no overlaps and no gaps

Given these definitions, the meshes used in the different simulation codes can be categorized as follows:

Code name	Discretization methods	Elements	Type of mesh
SOLPS	FD	Quadrangles + triangles	Block-structured
FBGKI	Spectral elements	Quadrangles	Unstructured
GYSELA	Semi-Lagrangien	Quadrangles	Structured
TOKAM3X	FD/FV	Quadrangles	Block-structured
JOREK	FE	Cubic Hermite Bezier quadrangles	Block-structured
HELENA	FE	Cubic Hermite Bezier quadrangles	Structured
MISHKA	FE	Cubic Hermite Bezier quadrangles	Structured
SOLEDGE	FD/FV	Quadrangles	Block-structured
PlaTo	FE	P1 or Powell-Sabin triangular elements	Unstructured

Examination of this table shows that there is a great variety in the meshes used in the simulation codes. The most important difference between the codes is between the ones that use block-structured meshes and the ones that use unstructured meshes. For the codes that use a block-structured strategy, a closer examination of the codes shows that all codes use node conforming meshes, that is every node of one block is also a node of the connected neighboring blocks.



Conforming Non-conforming

The meshing software must conform to this requirement and create block-structured node conforming meshes. Another important distinction between the meshes used by the simulation codes is between the ones that require only C^0 continuity between the elements and the ones that require more regularity. This is the case of the JOREK code that uses a cubic Hermite discretization that require C^1 continuity between the elements as well as the PlaTo software using Powell-Sabin elements where C^1 continuity is also required between the elements. For Powell-Sabin triangular elements, C^1 continuity is automatically ensured if the segment connecting the center of gravity of two triangles cuts the common edge of the two triangles. This is usually the case and the configuration displayed in the figure 3 is exceptional and is usually avoided by triangular meshing software. There is therefore no particular care for generating triangular Powell-Sabin meshes.

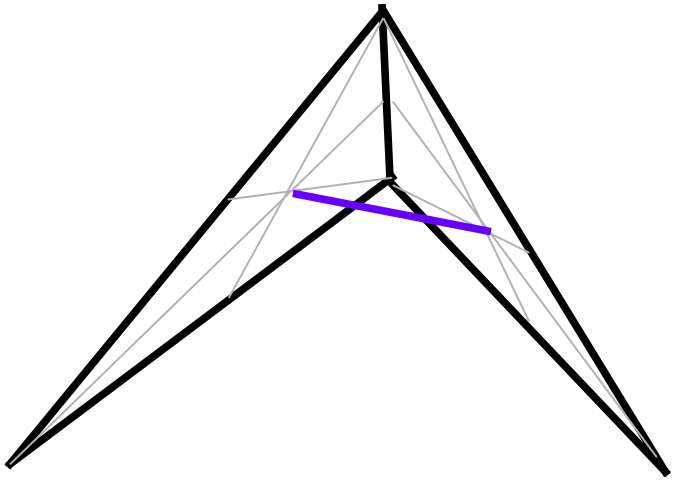


Figure 3: Non-continuity of the derivatives for Powell-Sabin triangular elements.

One the other hand, cubic Hermite Bezier discretization requires for ensuring the continuity of derivatives that each node of the quadrangulation has exactly 4 neighbors. Flux aligned meshes respect this constraint except on singular points of the magnetic flux field, namely on the O point where in a polar grid the number of neighbors is large and equal to the number of angles and on the X points where the number of neighbors is usually equal to 8. Provided the meshes respect this constraint, the JOREK code includes a specific treatment of these configurations to avoid numerical problems in the neighborhood of these singular points. The meshing software will have therefore to enforce these constraints.

3. Definition of the meshing software

3.1 Input

To construct flux surface aligned meshes of the poloidal plane, the minimal inputs that the meshing software must require are

- The boundaries of the computational domain
- The flux function inside this computational domain

Boundaries of computational domain: According to the answers to the questionnaire, the computational domains considered by all the simulation codes are restricted to the vacuum vessel of the Tokamaks or to a part of it. Specifically, edge plasma codes (SOLPS,FBGKI, TOKAM3X, SOLEDGE) exclude a central part of the core plasma around the O point whose boundary is a constant flux surface. For computational reasons this is also the case of GYSELA. Moreover, in some machines, (for instance MAST see the opposite figure) poloidal coils are located inside the vacuum vessel and the computational domain of the simulation codes must exclude these parts of the machine. Thus instead of being defined by the walls of the vacuum vessel, the boundaries of the computational domain is rather defined by a (part of) constant flux surface. The definition of these constant flux surface boundaries cannot be automated as it depends on the user's interest.

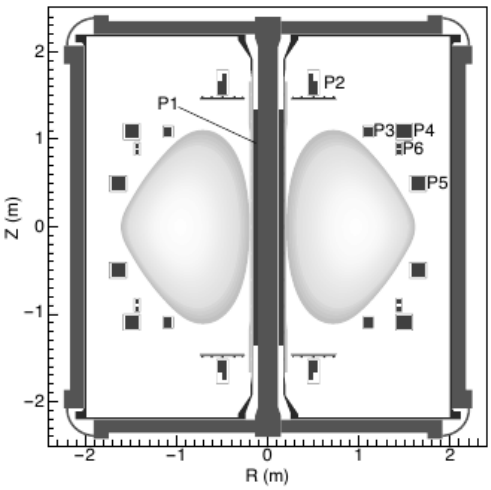


Figure 4: Plan view of MAST showing the PF coils (reproduced from [Sykes et al, 2001])

Flux function inside the computational domain: In today machines, the topology of the flux surfaces cannot be specified by an analytic function but results from the numerical solution of an equilibrium problem called the Grad-Shafranov equation [Grad and Rubin, 1958, Shafranov, 1966]. Thus to obtain the equilibrium flux field ψ inside the computational domain, one must first solve a non-linear elliptic problem:

$$\Delta^* \psi = G(\psi)\, \chi_p + \Sigma\, \chi_c I_c / S_c$$

where χ_p denote the characteristic function of the plasma region while χ_c denote the characteristic function of the region where the coil c is located. S_c is the surface of the coil. The function $G(\phi)$ depends on the machine and the discharge as well as the current I_c passing through the coil c. This problem is a free-boudary problem since the plasma region χ_p is an unknown and its boundary is defined as the first flux surface that passes either through the active X point (divertor case) or that crosses a material boundary (limiter case). This complex non-linear problem is itself solved on a computational mesh. There is a priori no reason for the computational grid where the flux function is computed to contain the boundary of the vacuum vessel or the in-vessel flux surfaces that the user wishes to be a boundary of its computational domain.

However in the definition of the present meshing software, we will consider that this is the case

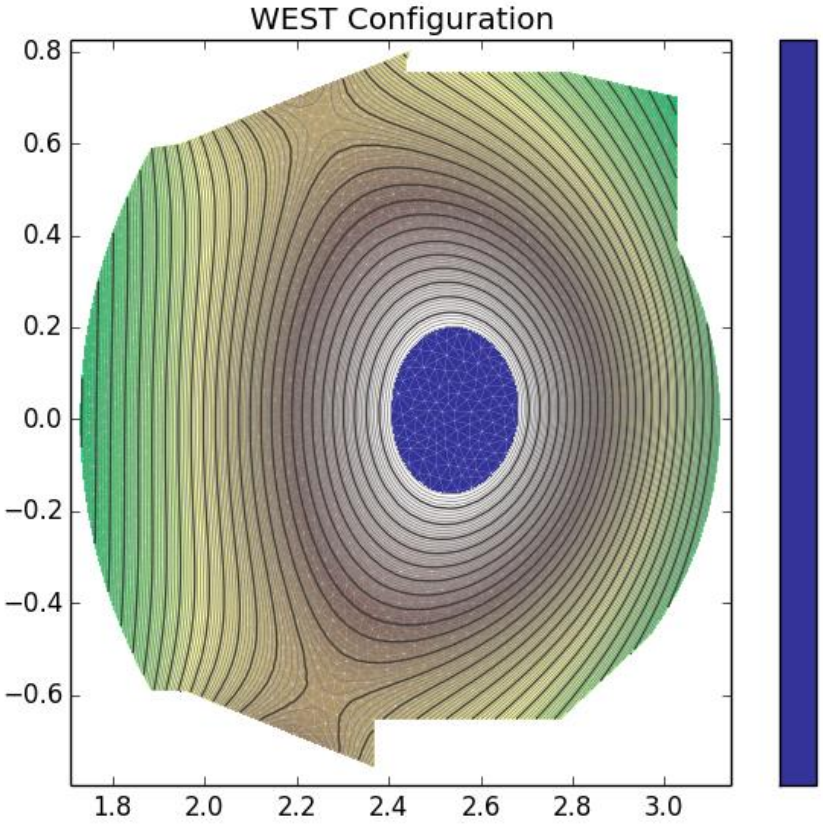


Figure 5: Example of an input mesh and flux function defined on this mesh

and that the mesh where the flux function is defined includes the boundaries of the computational domain that the user want to consider. Moreover, in order to allow for the maximum flexibility in the definition of the computational domain, the software will have as input a triangular mesh and the flux function will be defined on this mesh. This corresponds to the output of the equilibrium code Cedres++ that is described in [Heumann et al, 2015]. A light open source MATLAB implementation of this software is available and can be downloaded from: http://www-sop.inria.fr/members/Holger.Heumann/Software.html. Figure 5 shows an example of mesh (in

grey) on which is displayed the iso-contours of the flux ψ.
In annex A, we give the precise format of the input files that the software will require.

3.2 Pre-processing of the input

The construction of flux aligned meshes depends on the smoothness of the iso-contours computed by the equilibrium codes. In particular, the inputs (mesh and flux solution) must have a sufficient quality and resolution to generate "good" flux aligned meshes. The problem is particularly important near the X point since in many cases, the iso-contours of the flux function in this region are not very regular (see figure 6).

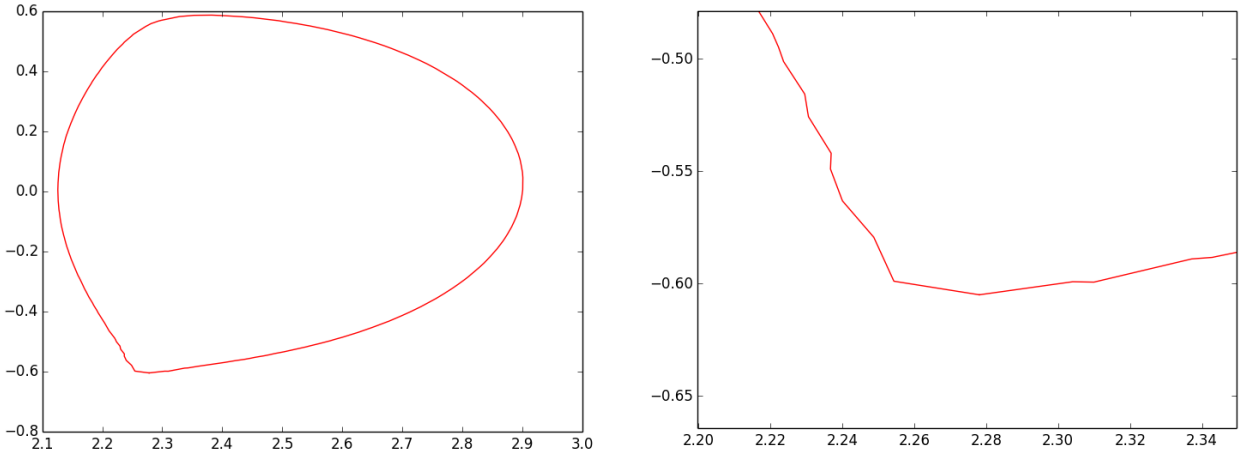


Figure 6: Iso-contour of the flux passing through the X-point (left), zoom around the X-point (right).

It can be considered that it will be the responsibility of the user to provide input of sufficient quality for this purpose, however, since it is not ensured that the users of the simulation codes can have sufficient control and knowledge of the equilibrium codes, we believe that it can be interesting to provide in the software some tools to improve the quality of the input. These tools will be different depending on the information that the user has from the equilibrium code.

3.2.1 Case 1: Additional information on the equilibrium problem.

If the computational domain does not include poloidal coils, the solution of the non-linear Grad-Shafranov problem depends only on the (non-homogeneous) boundary conditions of the computational domain and the right-hand side $G(\psi)$

$\Delta^* \psi = G(\psi)\ \chi_p$

therefore if the user has access to the function $G(\phi)$, it will be possible to recompute the solution of the Grad-Shafranov problem. The function $G(\phi)$ is usually the sum of two different terms, one representing the pressure contribution while the other one comes from the toroidal field. Assuming that these functions are known to the user, one can recompute the equilibrium problem (with fixed boundary) in order to have more regularity. This can be done by combining three different techniques:

Mesh-refinement (h-refinement): The software will provide a tool allowing uniform as well as non-uniform mesh refinement in user specified regions. Given this new mesh and the function $G(\phi)$, the solution of the equilibrium problem can be recomputed resulting in new inputs.

Higher-order scheme (p-refinement): The software will provide a tool allowing the use of high-order method to compute the solution of the equilibrium problem. Specifically, a Grad-Shafranov solver with fixed-boundary using Powell-Sabin quadratic elements will be implemented in the software. The Powell-Sabin finite element can use the same mesh that the original one. It is expected that the use of this family of C^1 elements will provide solutions that will be smoother than the original input. Note however that it is not guaranteed that the use of high-order elements alone can yield an improved regularity as remarked in the following quotation from [Heumann et al, 2015]: "*We are solving here a nonlinear elliptic problem with discontinuous coefficients(in the case of iron-transformer tokamaks) and discontinuous right-hand side. The standard convergence theory for finite elements and elliptic regularity theory does not yield improved approximation results for polynomials of degree higher than 1*". For this reason, [Heumann et al, 2015] recommends the use of h-p refinement.

Smoothing of the right-and side: The right-hand side $G(\phi)\ \chi_p$ of the equilibrium solver is discontinuous. It is possible that this induces some lack of regularity of the level sets of the solution in the vicinity of the X point (for the authors of this report, this is unclear from a mathematical point of view). A possible solution is therefore to smooth this right-hand side by replacing the characteristic function by a C^1 function using hyperbolic tangent function. This is the strategy that is used for the grid construction in the Jorek software [jorek wiki] and we will use here the same strategy. Of course this technique can be combined with the previous two ones using h-p refinement.

3.2.2 Case 2: No information except the mesh and solution files

The only information that the user possesses is contained in the .mesh and .sol files. In this case, if the flux solution is not regular enough to create a smooth grid, it will be necessary to smooth the input data. The software will thus provide a procedure allowing one to smooth the

data from the solution file. This can be easily done for instance by solving an elliptic problem.

3.3 Meshing algorithms for unstructured triangular and quadrangular meshes

3.3.1 Triangular meshes.

For unstructured triangular meshes, the generation of flux aligned meshes is a specific example of anisotropic mesh adaptation. The term "anisotropic" refers here to the fact that for any point of the computational domain, the optimal mesh steps can be different according to the direction. For the construction of anisotropic adaptive meshes, this problem is usually approached through the construction of a riemanian metric specifying the mesh size in the different directions. In other words, the construction of an anisotropic adaptive mesh is considered to be equivalent to the generation of an isotropic uniform mesh with a modified computation of the distances. A metric is specified by a positive definite symmetric matrix M and the distance between two points $\mathbf{a}$ and $\mathbf{b}$ is defined by:

$$d(\mathbf{a},\mathbf{b}) = \sqrt{({}^t\mathbf{ab}\ M\ \mathbf{ab})}$$

In two dimensions, a positive definite symmetric matrix can be written as the product

$$M = R\ \Lambda\ {}^tR$$

where R is an orthogonal matrix that contains the eigenvectors $\mathbf{n}_1$ and $\mathbf{n}_2$ while Λ is the diagonal matrix of the positive eigenvalues. Therefore a useful geometrical interpretation of the riemanian structure defined on the domain, is to consider that on each point of the domain, the unit ball is transformed into an ellipsis whose axis are given by the two directions $\mathbf{n}_1$ and $\mathbf{n}_2$ and whose length h_1 and h_2 on the axis $\mathbf{n}_1$ and $\mathbf{n}_2$ are given by the inverse of the square root of the eigenvalues of Λ. More detailed information on this topic can be found in the text-book [Frey and Georges, 2008], as well as in the reports and articles [Mavriplis, 1995], [Hecht, 1998], [Coupez, 2000], [Frey, 2001], [Laug and Bourouchaki, 2003] [Huang, 2005] or the theses [Leservoisier, 2001], [Alauzet, 2003], [Dobrzynski, 2005], [Mesri, 2007].

In particular, the thesis [Guégan, 2007] studies the construction of adapted triangular and tetraedral meshes for interface problems between two different fluids where the mesh is adapted to the level set of the distance function to the interface and is therefore extremely close to the

problem that we consider here.

For flux aligned meshes, the definition of the principal axes $\mathbf{n}_1$ and $\mathbf{n}_2$ of the metric matrix is directly related to the flux function ϕ with $\mathbf{n}_1$ for instance the normalized unit gradient of ϕ and $\mathbf{n}_2$ its orthogonal. In computer graphics, such a field of directions (invariant by rotations of multiples of $\pi/2$) is known as a cross field and many works are devoted to the automatic construction and interpolation of such fields minimizing the number of singularities in the field, see for instance [Ray et al, 2008] and [Kowalski, 2013]. In the present case, the cross field as well as its singularities are directly given by the flux function and its critical points. It remains however to specify the mesh size in the parallel and perpendicular direction of the iso-contours to define totally the metric field. Then the standard algorithms for triangular anisotropic mesh adaptation can be used. In the present software, we will use an algorithm similar to the one described in [Frey, 2001] and that relies on local operations as point insertion and removal, edge swapping and local displacements of the mesh points. The definition of the mesh size in the two orthogonal directions $\mathbf{n}_1$ and $\mathbf{n}_2$ is equivalent to specify:

1) the number of mesh points along a iso-contour (direction $\mathbf{n}_2$)

2) the number of nodes in the orthogonal direction (direction $\mathbf{n}_1$). This last quantity is equal to the number of iso-contours in the considered region.

These two quantities are highly dependent on the user's wishes and cannot be specified a priori. They are also non-uniform and depend on the interests of the user on the different regions of the domain. For instance, some users working on edge plasma will want to have an increased resolution near the strike points while some problems will require on the opposite way to have a very fine mesh in the core plasma. The software will therefore propose two ways to define the mesh resolution:

1) either as in "standard" anisotropic mesh generators [George, 1999], [George, 2001],[George, 2003], [Gruau and Coupez, 2005] by a metric map given by the user.

2) either by the input provided by the user through a graphical user interface (GUI) that will rely on a segmentation of the domain relying on the analysis of the critical points of the flux function (see section 3.4.1)

3.3.2 Unstructured quadrangular meshes

At the present time, while the generation of anisotropic triangular meshes can be considered as a well-mastered domain, the generation of unstructured quadrangular meshes is far from being at the same level of maturity. Essentially two kinds of methods have been proposed in the literature:

- Direct methods where the quadrilaterals are constructed at once, either by grid-based method (quad-tree) or by some kind of advancing front technique.
- Indirect methods that uses an initial triangular mesh. These methods use the triangles of the initial mesh and recombine them to form quadrangles, [Lee and Lo, 1994], [Bourouchaki et al, 1998]. The basic operation to transform a triangular mesh into a quadrangular one is just to suppress the common edge between two neighboring triangles. Note that some strategies use a mix of advancing front and triangle merge methods [Owen et al, 1999].

In the flux meshing software, we will implement an indirect method based on the use of the adapted triangular meshes that have been constructed by the algorithm detailed in section 3.3.1. This choice is made based on the fact that for triangular flux aligned meshes, the vertices of the initial triangular mesh are naturally aligned. In other words, flux aligned triangular meshes have naturally the property that indirect quadrangular mesh generation algorithms try to enforce in the choice of the position of the vertices of the initial triangulation, [Remacle et al, 2012], [Baudouin et al, 2014]. Therefore the choice of the edges of the triangles that have to be removed is obvious. To describe in a few words this algorithm, as in [Remacle et al, 2012], we will consider that generating a quadrangular mesh from a triangular one can be formalized as a problem of optimal perfect matching. More specifically, considering the graph G=(V,E) where V is the set of triangles and E the set of edges induced by the neighboring relations of the triangles (two triangles are neighboors if they share an edge) and a cost function c(E) that will penalize the edges of the triangulation that are not neither aligned or perpendicular to the iso-contours of the flux function, we will look for an optimal perfect matching of G(V,E). We recall that a matching is a subset $E' \subseteq E$ such that each node of V has at most one incident edge in E. This matching is called perfect if each node of V has exactly one incident edge in E and a perfect matching is optimal if c (E) is minimum among all possible perfect matchings.

Algorithms to find an optimal perfect matching have been published in the literature [Edmonds, 1969], [Edmonds et al 1969] and a computer implementation of these algorithms for the problem of quadrangle mesh generation is available in the open-source software gmsh [Geuzaine, 2009].

3.4 Meshing algorithms for block-structured meshes

Several simulation codes codes used for tokamak plasma studies in the teams involved in the work package Fusion4Energy of EoCoE use block-structured grids. The generation of block structured grids can be divided into two different tasks:

- First, one have to identify a partition of the computational domain into sub-domains such that the number of singular connection between the sub-domains is minimal and such that each sub-domain can be mapped to the square [0,1]x[0,1]. We will call this part, the domain segmentation problem.

- Second, one has to mesh in a structured way, each sub-domain ensuring some compatibility between the sub-domain (continuity of the finite element)

In the general case, the first part is by far, the most difficult part of the problem. It usually requires some manual input from the users relying on the knowledge that they have from the physical problem at hand. In the tokamak plasma community this is also the case for the simulation codes and meshing tools in use. The meshing tool CARRE [Marchand and Dumberry, 1996] relies on a a priori segmentation of the domain into several sub-domain corresponding to the known expected magnetic configurations (single null geometry, double null geometry, disconnected double null). This is also the case in the Jorek code [Czarny and Huysmans, 2008], [jorek wiki] where the meshing tool contains different subroutines corresponding to known configurations: grid_xpoint.f90, grid_double_xpoint.f90, etc or for the SOLEDGE code [Bufferand, 2012] where a graphical user interface has been designed to help the user to define the sub-domains.

While in the general case, the segmentation problem appears as a very difficult one, the construction of flux surface aligned block-structured grids can benefit from some peculiarities of the problem. Actually, it appears that all the partition strategies of the computational domain used for the generation of 2D meshes in tokamaks plasma modeling rely on some assumptions on the flux function and that it is possible to unify these approaches by a preliminary analysis of the equilibrium fields without having to use some pre-defined configurations. This analysis uses some notion of topology that we recall below.

3.4.1 Domain segmentation

3.4.1.1 Morse functions

Morse theory [Morse, 1934], [Milnor, 1963] relates the study of the critical points of a function f defined on a smooth manifold Ω to the topology of this space.
Here, we will restrict ourselves to 2D smooth manifolds and consider that Ω is a 2-D compact manifold diffeomorphic to a submanifold of the unit sphere S^2. In order to avoid technical difficulties, we will assume that the boundary of Ω is an iso-contour of f and considering the

embedding of S^2 into R^3 we introduce a virtual point located at the infinity and connected to this iso-contour to close the domain. By convention we assume that the value of this virtual point is the smallest value of the function in the domain.

We begin to state some definitions and classical results.

Critical points: Let C^r be the space of r differentiable scalar field defined on Ω. For r> 2, a point $\mathbf{p} \in \Omega$ is a critical or singular point of the function f if $\nabla f=0$

Regular critical points: A critical point is regular (or non-degenerate) if the Hessian H of f at $\mathbf{p}$ is invertible.

Morse function: A function f is a Morse function if all its critical points are regular.

The index $\lambda(\mathbf{p})$ of a regular critical point $\mathbf{p}$ is the number of negative eigenvalues of the Hessian H.

A Morse function can only have isolated critical points, this is a consequence of the Morse lemma (see [Milnor, 1963] p 6)

In 2 dimensions, if we note μ_1, μ_2 the two eigenvalues of H, the only possible critical points of a Morse function are therefore

Maximum	index=2	$\mu_1 < 0$	$\mu_2 < 0$
Saddle	index=1	$\mu_1 < 0$	$\mu_2 > 0$
Minimum	index=0	$\mu_1 > 0$	$\mu_2 > 0$

Number of critical points: (The montaineer equation)
Let f be a Morse function defined on Ω, a region defined by a closed iso-contour, then the number of critical points (counting the virtual minimum) verify the relation:
$C_M - C_S + C_m = 2$

where C_M C_S and C_m are respectively the number of maxima, saddles and minima.

Now let $\mathbf{v}$ be a vector field defined on Ω, and $\mathbf{x_0} \in \Omega$, to this vector field can be associated the ordinary differential equation

$d\mathbf{x}/dt = \mathbf{v}(\mathbf{x})$ $\quad\quad$ $\mathbf{x}(0) = \mathbf{x_0}$

If we associate to the scalar field f the vector field $\mathbf{v} = \nabla f$ or the rotated vector field $\mathbf{v}^{\perp} = \mathbf{k} \times \nabla f$ where $\mathbf{k}$ is a unit vector field of R^3 orthogonal to Ω, we see that the iso-contours $\gamma(s)$ of f defined by $f(\gamma(s))=C$ where C is a constant are the orbits of the vector field $\mathbf{v}^{\perp}$. Conversely, the

orbits of ∇f are orthogonal to the
iso-contour of f.

The critical points of f are also the critical points of the vector fields $\mathbf{v}$ and the rotated vector field $\mathbf{v}^{\perp} = \mathbf{k} \times \nabla f$. The vector field $\mathbf{v}$ is curl free while $\mathbf{v}^{\perp}$ is divergence free and we have the following correspondence between the critical points

f	Maximun	Saddle	minimim
$\mathbf{v}$	Source	Saddle	Sink
$\mathbf{v}^{\perp}$	Center	Saddle	Center

Since the orbits of the divergence free vector field $\mathbf{v}^{\perp}$ are the iso-contours of f, we can use the results given in [Ma and Wang, 2002] to give a structural classification of the iso-contours of f:

Let f be a Morse function defined on Ω. Then the topological set of the iso-contours of f consists of finite connected components that are either

- Circle cells which are homeomorphic to open disks
- Circle bands which are homeomorphic to open annulus
- Saddle connections

The following result is also proven in [Ma and Wang, 2002] and gives a refined result on the possible saddle connections.

Let f be a Morse function defined on Ω. This field is structurally stable if and only if all saddle points are self-connected.

This last result establishes that the situation where two saddles are connected by an iso-contour passing through these saddles is not structurally stable. This is intuitively easy to understand: An arbitrary small perturbation near any of the two connected saddles will change the value of f at this point and break the saddle connection. For the plasma physics application considered here, where the function f is the magnetic flux, the situation where two saddles are on the same level set is known as a connected double null (CDN). As remarked for instance in [Marchand and Dumberry, 1996] the CDN is only an idealization and in real experiments, the two saddles are never on exactly the same iso-contour. For the meshing algorithm, we are considering in this work, there is no necessity to consider the CDN pattern and it will be sufficient to consider the disconnected double null DDN pattern as the generic one: if the values of the magnetic flux in the two saddles points are too close, we will simply fuse the two iso-values and there will be no need to mesh the inter-region between the saddle iso-contours since its surface will be zero.

To sum up the results of this section in a concrete way, we have established that Ω consists of connected regions that contain only closed orbits. These regions are either circle cells containing an extremum critical point or circle bands separated from the other connected regions by self connected saddles. In the next section, we will give a concrete way to construct these regions.

3.4.1.2 Reeb graph

In the previous section, we have seen that it is possible to split the domain Ω into connected component that are either homeomorphic to a disk or to an annulus. The Reeb graph [Reeb1946], gives a concrete way to construct and store this decomposition of the domain. To define formally the Reeb graph, we first define an equivalence relation between points of Ω.

Equivalence relation: Given a topological space Ω and a continuous function f: $\Omega \rightarrow R$, two points $\mathbf{p}$ and $\mathbf{q}$ are equivalent $p \sim q$ if they belong to the same connected component of a single iso-contour $f^{-1}(c)$ for some $c \in R$. The formal definition of the Reeb graph is then

Reeb Graph: The Reeb graph is the quotient space $\Omega/\sim$ endowed with the quotient topology.

Loosely speaking, the Reeb graph concatenates all the points belonging to the same connected component of a level set into a single representative.

In the case of a Morse function that have only isolated critical points, the construction of the Reeb graph can be visualized in the following way: Let us consider the graph $\mathbf{F}$ of the function f defined by $\mathbf{x} = (x_1, x_2, x_3) = (\mathbf{p}, f(\mathbf{p})) \in R^3$ for $\mathbf{p} \in \Omega$. We ``slice'' $\mathbf{F}$ by the plane $x_3 = C$ and evolve C in the positive direction starting from from $-\infty$. Each intersection of $\mathbf{F}$ with the plane $x_3 = C$ corresponds to a set of connected iso-contour $f^{-1} = C$. The change in topology of this set occurs at critical points of f. As C goes from $-\infty$ (the virtual pit) to the global maximum of f, a new connected level set will appear when C passes through a minimum. Conversely a connected iso-contour will disappear when C goes beyond a maximum. When C passes through a saddle, two components of the iso-contour $f^{-1} = C$ will merge giving birth to two new connected components.

The nodes of the Reeb graph then correspond to critical points of f and each edge corresponds to a change in the number of connected components that occur when the slicing plane passes through this node. Consequently an edge originates (resp. terminates) at the nodes corresponding to a minimum (resp. maximum) and these nodes have degree 1. If the node corresponds to a saddle, two components of the level set merge and two new closed components appear. The corresponding node of the Reeb graph is therefore of degree 3 and looks like the letter "Y".

For Morse functions defined on a orientable surface of genus 0, the Reeb Graph contains no loop and on a flat Ω, it is a tree. Reeb Graph are therefore also called sometimes contour tree. Due to their numerous applications in computational geometry and computer graphics, the efficient

construction of Reeb graphs has been well studied. A simple algorithm is given in [Takahashi et al, 1995]. As an example, Figure 7 below (left) shows the iso-contours of the flux function, figure 7 (right) shows the segmentation of the domain given by its critical points while figure 8 displays the associated Reeb graph.

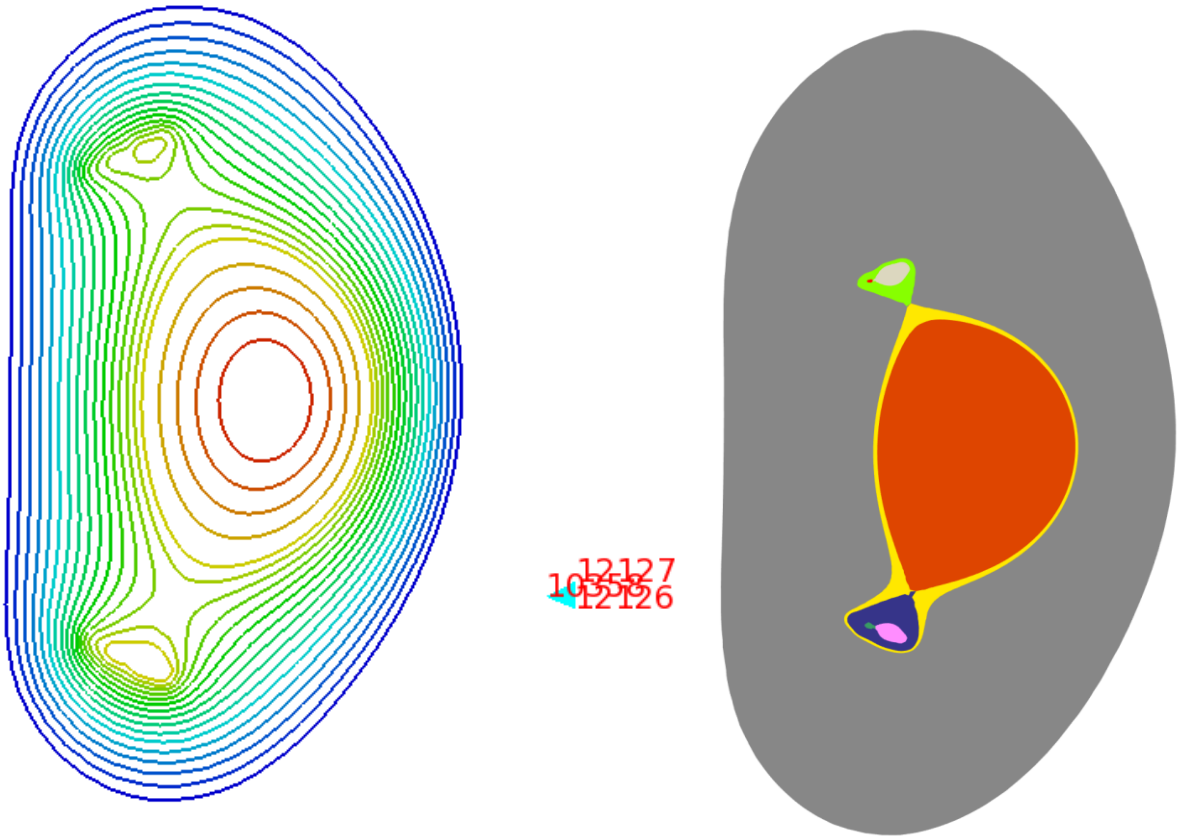


Figure 7: left (domain and isovalues), right: domain segmentation.

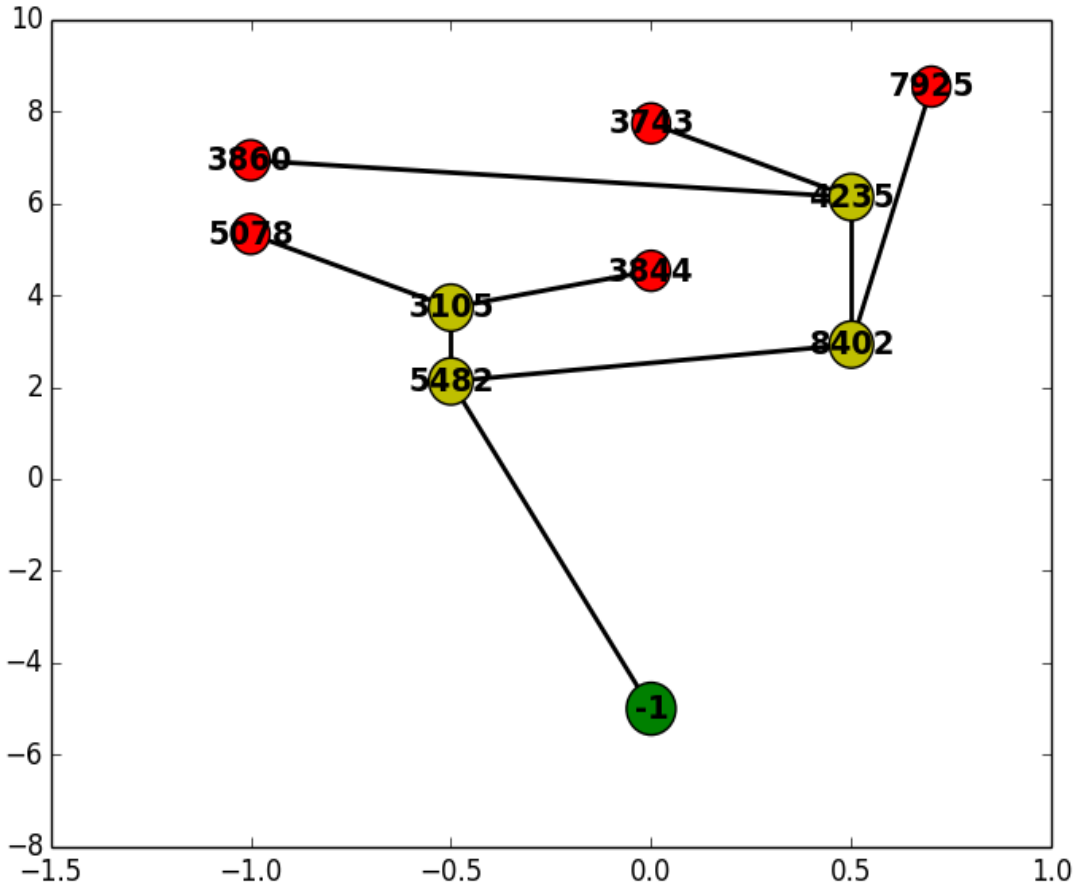


Figure 8: Reeb graph associated to the partition into sub-domains. Each edge of the graph corresponds to a single sub-domain displayed in figure 7.Each sub-domain can be mapped into the square [0,1]x[0,1]. This mapping can be singular on a single point if the sub-domain contains an extremum.

3.4.2 Meshing algorithm for block-structured grids

We are now in position to describe our meshing algorithm. Given a domain Ω and a Morse function f: $\Omega \rightarrow R$, we first identify the critical points of f and construct its Reeb Graph. Each edge of the Reeb Graph corresponds to
a connected subdomain of Ω that contains only closed level sets of f. Thus let e be an edge of the Reeb graph and e_1 , e_2 be its two extremities. Let Ω_e be the subdomain defined by e. In Ω_e the value of f evolves monotonically from $f_m=\min(f(e_1),f(e_2))$ to $f_M=\max((f(e_1),f(e_2))$.

Moreover, the domains Ω_e are of only of two different types: either the vertices e_1 , e_2 are two saddle points or one and only one of these two vertices, say e_1 is an extremum of f. We therefore have the following result:

Let e be an edge of the Reeb graph and Ω_e be the subdomain of Ω associated to e. Let f_m , f_M with $f_m < f_M$ be the value of f associated to the extremities of e then there exists a one to one mapping G^e: $(f,s) \in [f_m , f_M] \times [0,1] \rightarrow p \in \Omega_e$.

In other words, for each edge of the Reeb graph, there exists a mapping between $[f_m , f_M] \times [0,1]$ and Ω_e. Note that Ω_e can only be of two different types

- e is an extremal edge of the graph and one of its extremities, say e_1 is an extremum of f then Ω_e is homeomorphic to a disk
- e is internal edge of the graph and its two extremities are saddle points of f then Ω_e is a circle band

Remark: In the first case, the mapping G is singular since on the extremum the contour line is reduced to a single point. On Ω_e, the generated mesh is a polar mesh that contains a singular point at its center. In practical applications we assume that the numerical methods used in the simulation codes are able to handle the degenerate case of the center of a polar grid (this is for instance the case of the Jorek code) or that this center does not belong to the computational domain (edge plasma code).

The previous result shows that it is possible to construct a block structured grid, aligned with the iso-contours of f into each of the subdomain Ω_e

We now show that it is possible to glue these individual grids to obtain a single conforming mesh covering all the domain.

Then let us consider the Reeb graph, and assume that this graph contains K extremal edge or alternatively K vertices of degree 1 (the leaves). The main point is that the discretization of the boundary of the K Reeb sub-domain corresponding to the leaves totally determine the

conforming mesh covering all the domain.

To see that, we now construct the K paths P_k in the Reeb graph connecting the K extremal vertices to the virtual pit. These K paths are composed of a unique extremal edge and a sequence of edges whose vertices are saddle points. These K path merge on some saddle
We will call $[S_0^k,S_1^k]$,$[S_1^k,S_2^k]$,..., $[S_q^k,\mathbf{p}]$ the sequence of edges composing the k^{th} path with S_0^k the extremal vertex and **p** the virtual pit. Note that this implies that the edges are ordered and that in this way we have an order relation between edges of the same path. We adopt the notation $\Omega_{p,q}^k$ to designate the subdomain corresponding to the edge $[S_p^k,S_q^k]$ of the Reeb graph. To initialize the grid generation algorithm, we begin to construct the K grids corresponding to the leaves of the Reeb graph. In this construction, the only requirement imposed to these K grids will be that the saddle S_1^k belongs to the grids.

Then the construction of the grids $\Omega_{p,q}^k$ will be done according to the following rules:

- if two extremal paths P_k and $P_{k'}$ meet at the saddle S_j, let $[S_{j-1}^k,S_j]$ and $[S_{j-1}^{k'},S_j]$ be the two edges of the Reeb graph distinct of $[S_j,S_{j+1}]$ that contain S_j, then construct first the grids for $\Omega_{j-1,j}^k$ and $\Omega_{j-1,j}^{k'}$
- the saddles S_j and S_{j+1} belong to the grid $\Omega_{j,j+1}^k$
- the number of nodes $n_{j,j+1}$ used to discretize the boundary of $\Omega_{j,j+1}^k$ verifies: $n_{j,j+1} = n_{j-1,j}^k + n_{j-1,j}^{k'}$

In other words, when traversing the Reeb graph, the subdomains corresponding to extremal critical points must be meshed first. Then for the subdomains enclosed between the level sets passing through two saddles, say the interior and exterior saddles, the subdomains corresponding to the edges whose common vertex is the interior saddle must have been meshed first and the discretization of the interior boundary (the boundary passig trough the interior saddle) is the union of the discretization of the two subdomains connected by this saddle.

To conclude this section, we need to define how to discretize a subdomain $\Omega_{p,q}^k$

The two case that we have to consider are

I) $\Omega_{p,q}^k$ is a subdomain with an extremal critical point. The resulting mesh will be a polar mesh. An arbitrary number as well as an arbitrary distribution of isocontours can be used. Given this distribution of iso-contours. The discretization of this domain is entirely given by the discretisation of its boundary. Algorithms present in the Jorek code can be used for that purpose.

II) $\Omega_{p,q}^{k}$ is a subdomain enclosed between the isovalues passing through 2 saddles. Again, since in a subdomain the value of f evolves monotonically, we can construct an arbitrary number of iso-contours between the interior boundary and the exterior boundary. The discretization of this subdomain then follows once the discretization of its interior boundary has been done (the term interior refers here to the order relation between the edges of the Reeb Graph). The task to be performed now is to connect in an optimal way two iso-contours. Algorithms present in the CARRE [Marchand and Dumberry, 1996] software for C^0 meshes can be used for this purpose or in the Jorek software for C^1 meshes.

4. Outputs

4.1 Unstructured meshes

4.1.1 Triangular meshes

The same format used for the input file will be used for the output

4.1.2 Quadrangular meshes

The format used for the input file conforms to the GMF data format. This is basically the same format than for triangular meshes except for the key-word quadrangle instead of triangle and the definition of the elements that now requires 4 integer instead of 3

4.2 Block-structured meshes

Block structured grids can be stored into two different format: either a connectivity table defines the neighboring relations between the blocks and inside each block, a regular (I,J) indexing of the unknows is performed or as for non-structured mesh, a global connectivity table is constructed. Since the simulation codes use the two ways of storing the meshes, the meshing software will provide the possibility to output the meshes in the two formats.

Annexes

Annex A: format of input files.

The format used for the input file conforms to the GMF data format, a general format designed to handle meshes of different types. The GMF is a keyword based file format, meaning that a mesh file consists of a list of keywords, each followed by its data. No keyword is mandatory and a file may contain any combination of them. The input will require a mesh files (.mesh) and a physical solution files (.sol) that contains the flux function.

A description of this format follows:

file.mesh format

MeshVersionFormatted 1 # A line containing a key word specifying the type of format

Dimension # A line containing the key-word dimension

2 # the number of dimension, here 2

Vertices # A line containing the key-word Vertices

17549 # A line displaying the number of vertices of the mesh

0.00000e+00 -5.80000e+00 0 # 2 reals for the coordinate R, Z and an integer flag (can be used for the boundary conditions for instance)

1.00665e-01 -5.79913e+00 0

2.01299e-01 -5.79651e+00 0

3.01873e-01 -5.79214e+00 0

…

Edges # A line containing the key-word Edges, Edges here refers to the boundary edges of the mesh

326 # the number of boundary segments, a boundary edge is defined by the index of its two extremities

1 2 1 # 2 integers for the index of the edge extremities and an integer flag (can be used for the boundary conditions for instance)

1 11742 1

2 3 1

…

Triangles# A line containing the key-word Triangles

34770 # the number of triangles

11587 11441 11586 32 # 3 integers for the indices of the vertices of the triangles and an integer flag (can be anything useful for the user)

994 1702 1716 32

…

End # A line containing the key-word End

Note that the numbering of the vertices, edges and triangles conforms to the FORTRAN usage and begins by 1 (one). Note also that this numbering is implicit and given by their order in the input file: In this example, the vertex whose coordinates are 1.00665e-01 -5.79913e+00 is the vertex 1 and so-on.

file.sol format

```
MeshVersionFormatted 1 #as in .mesh

Dimension #as in .mesh

2 #as in .mesh

SolAtVertices # A line containing the key-word SolAtVertices

17549 # A line displaying the number of vertices of the mesh

0.00000e+00 # the value of the solution on the vertices

5.40183e-06

3.05273e-05

7.20232e-05

1.30195e-04

…

End # A line containing the key-word End
```

Annex B: list of algorithms to implement.

1. Pre-processing of the equilibrium computations

- Local or uniform triangular mesh refinement: based on the vizir software [vizir, 2013]
- Equilibrium (Grad-Shafranov) computations with non-homogeneous fixed Dirichlet boundary condition with P1 element: to be integrated in the software, based on the Plato Inria library
- Equilibrium (Grad-Shafranov) computations with non-homogeneous fixed Dirichlet boundary condition with Powell-Sabin finite element: to be integrated in the software, based on the Plato Inria library
- Smoothing of scalar field by solving an elliptic equation: to be integrated in the software based on the Plato Inria library.

2. Unstructured meshes

- Flux aligned triangular mesh: to be coded based on [Frey, 2001]
- Indirect construction of quadrilateral meshes from triangular meshes based on [Remacle et al, 2012]

3. Block-structured grids

- Analysis of the topology of the flux function and domain segmentation (section 3.4.1): to be coded.
- Graphical interface allowing the user to specify the mesh parameters: to be coded using the vizir software [vizir, 2013]
- Meshing algorithm for the subdomains: to be coded based on the connecting algorithm in [Baudouin et al, 2014] for C0 meshes and on part of the meshing code in the Jorek software for C1 Hermite Bezier meshes.

References

[Alauzet, 2003] F. Alauzet. *Adaptation de maillage anisotrope en trois dimensions. Application aux simulations instationnaires en Mécanique des Fluides*. PhD thesis, Université Montpellier II, Montpellier, France,(2003).

[Baudouin et al, 2014] Baudouin, T.C., Remacle, J., Marchandise, E. et al, *Lloyd's energy minimization in the L_p norm for quadrilateral surface mesh generation*, Engineering with Computers pp 30: 97, doi:10.1007/s00366-012-0290-x, (2014).

[Bourouchaki et al, 1998] H. Bourouchaki and P. Frey. *Adaptive triangular-quadrilateral mesh generation*. Int. J. Numer. Meth. Engng., 41:915-934, (1998).

[Bufferand, 2012] H. Bufferand, *Development of a fluid code for tokamak edge plasma simulation. Investigation on non-local transport*, Thesis University Aix-Marseille, (2012).

[Coupez, 2000] T. Coupez. *Génération de maillages et adaptation de maillage par optimisation locale*. Revue Européenne des Éléments Finis, 9:403-423, (2000).

[Czarny and Huysmans, 2008] Czarny O and Huysmans G. *Bézier surfaces and finite elements for MHD simulations,* J. Comput. Phys. 227, 7423 (2008)

[Dobrzynski, 2005] C. Dobrzynski, *Adaptation de maillages anisotrope 3D et application à l'aérothermique des batiments*, Thèse, Université Pierre et Marie Curie, Paris, France (2005).

[Edmonds, 1969] Edmonds J. *Maximum matching and a polyhedron with 0-1 vertices.* J. of Research at the National Bureau of Standards 1965; 69B(125-130).

[Edmonds et al 1969], Edmonds J, Johnson EL, Lockhart SC. *Blossom I: A computer code for the matching problem.* IBM T. J. Watson Research Center, Yorktown Heights, New York (1969).

[Frey, 2001] Frey P. J. *Yams: A fully automatic adaptative isotropic suface remeshing procedure*, Inria technical report 0252, INRIA (2001).

[Frey and Georges, 2008] P. Frey and P-L. George, *Mesh Generation*, 2nd Edition, 848 pages, Wiley-ISTE, (2008).

[George, 1999] George P.-L. *Tet meshing: Construction, optimization and adaptation*, in Proceedings of the 8th International Meshing Roundtable (South Lake Tahoe, CA, USA), p. 133-141 (1999).

[George, 2001] George P.-L. *Gamhic3d. A fully automatic adaptative mesh generation method in three dimensions*, Inria technical report (2001).

[George, 2003] George P.-L. Gamanic3d. *Adaptative anisotropic tetrahedral mesh generator,* Inria technical report (2003).

[Geuzaine, 2009] C. Geuzaine and J.-F. Remacle. *Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities*. International Journal for Numerical Methods in Engineering 79(11), pp. 1309-1331, (2009).

[Grad and Rubin, 1958] Grad, H., and Rubin, H. *Hydromagnetic Equilibria and Force-Free Fields* Proceedings of the 2nd UN Conf. on the Peaceful Uses of Atomic Energy, Vol. 31, Geneva: IAEA p. 190, (1958)

[Gruau and Coupez, 2005] Cyril Gruau, Thierry Coupez, *3D tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric*, Computer Methods in Applied Mechanics and Engineering, Volume 194, Issues 48-49, Pages 4951-4976, http://dx.doi.org/10.1016/j.cma.2004.11.020, (2005).

[Guégan, 2007] D. Guégan, *Modélisation numérique d'écoulements bifluides 3 D instationnaires avec adaptation de maillage*, Thèse Université de Nice, 2007

[Hecht, 1998] F. Hecht. BAMG: *Bidimensional anisotropic mesh generator.* Available from: http://wwwrocq.inria.fr/gamma/cdrom/www/bamg/eng.htm, INRIA-Rocquencourt, France, (1998).

- [Heumann et al, 2015] H. Heumann, J. Blum, C. Boulbe, B. Faugeras, G. Selig, J.-M. Ané, S. Brémond, V. Grandgirard, P. Hertout and E. Nardon (2015). *Quasi-static free-boundary equilibrium of toroidal plasma with CEDRES++: Computational methods and applications*. Journal of Plasma Physics, 81, p 905810301 (35 pages) (2015).

- [Huang , 2005] W. Huang. *Metric tensors for anisotropic mesh generation*. J. Comp. Phys., 204:633-665, 2005.

- [jorek wiki] JOREK: Some informations about the code, Revision 911, available on http://jorek.eu/wiki/doku.php

- [Kowalski, 2013] N. Kowalski, *Domain partitioning using frame fields: application to quadrilateral and hexahedral meshing*, Thèse, Université Pierre et Marie Curie, Paris, France (2013).

- [Laug and Bourouchaki, 2003] P. Laug and H. Bourouchaki. *BL2D-V2, mailleur bidimensionnel adaptatif.* RR-0275, INRIA, (2003)

- [Leservoisier, 2001] D. Leservoisier. *Stratégies d'adaptation et de raffinement de maillage en mécanique des fluides numérique*. PhD thesis, Université Pierre et Marie Curie, Paris VI, Paris, France, (2001).

- [Lee and Lo, 1994] Lee CK and Lo SH. *A new scheme for the generation of a graded quadrilateral mesh*. Computers and Structures, 52 pp 847-857, (1994).

[Lévy and Liu, 2010] Lévy B, Liu Y. *Lp centroidal voronoi tesselation and its applications*. ACM Transactions on Graphics(SIGGRAPH conference proceedings), 2010.

[Owen et al, 1999] Owen SJ, Staten ML, Canann SA, Saigal S. *Q-morph: an indirect approach to advancing front quad meshing*. International Journal for Numerical Methods in Engineering, 9 pp 1317-1340, (1999).

[Marchand and Dumberry, 1996] R. Marchand and M. Dumberry, *"CARRE: a quasi-orthogonal mesh generator for 2D edge plasma modelling"* , Computer Physics Communications, vol. 96, no. 2-3, pp. 232 - 246, (1996).

[Ma and Wang, 2002] Tian Ma and Shouhong Wang, *Structural classification and stability of divergence-free vector fields* ,"Physica D: Nonlinear Phenomena ",171,1-2, pp 107 - 126, (2002).

[Mavriplis, 1995] Dimitri J. Mavriplis. *Unstructured mesh generation and adaptivity*. Technical Report ICASE 95-26, NASA Langley, Hampton VA, Apr. 1995. Abstract at http://techreports.larc.nasa.gov/cgi-bin/NTRS.

[Mesri, 2007] Y. Mesri. *Gestion et contrôle des maillages non structurés anisotropes, applications en aérodynamique*. PhD thesis, Université de Nice, Nice, France, 2007.

[Milnor, 1963] Milnor John, *Morse Theory. Princeton University Press.* ISBN 0-691-08008-9, *(1963).*

[Morse, 1934] Morse, Marston, *The Calculus of Variations in the Large,* American Mathematical Society Colloquium Publication 18; New York, (1934).

[Ray et al, 2008] Nicolas Ray, Bruno Vallet, Wan-Chiu Li, Bruno Lévy. *N-Symmetry Direction Field Design*. *ACM Transactions on Graphics*, Association for Computing

Machinery, 27 (2), pp. 10. <10.1145/1356682.1356683>. <inria-00331900>, (2008)

[Reeb1946] Reeb, Georges, *Sur les points singuliers d'une forme de Pfaff complétement intégrable ou d'une fonction numérique*, C. R. Acad. Sci. Paris, 222, pp 847 - 849, (1946).

[Remacle et al, 2012] J.-F. Remacle, J. Lambrechts, B. Seny, E. Marchandise, A. Johnen and C. Geuzaine. *Blossom-Quad: a non-uniform quadrilateral mesh generator using a minimum cost perfect matching algorithm*. International Journal for Numerical Methods in Engineering 89, pp. 1102-1119, (2012).

[Remacle et al, 2013] J.-F. Remacle, F. Henrotte, T. Carrier-Baudouin, E. Béchet, E. Marchandise, C. Geuzaine and T. Mouton. *A frontal Delaunay quad mesh generator using the L∞ norm*. International Journal for Numerical Methods in Engineering, 94(5), pp. 494-512, 2013.

[Shafranov , 1966] Shafranov, V.D. *Plasma equilibrium in a magnetic field*, *Reviews of Plasma Physics*, Vol. 2, New York: Consultants Bureau, p. 103, (1966).

[Sykes et al, 2001] A. Sykes, R.J. Akers, L.C. Appel, E.R. Arendsa , P.G. Carolan, N.J. Conway,G.F. Counsell, G. Cunningham, A. Dnestrovskij b , Yu.N. Dnestrovskij b ,A.R. Field, S.J. Fielding, M.P. Gryaznevich, S. Korsholmc , E. Laird d , R. Martin, M.P.S. Nightingale, C.M. Roach, M.R. Tournianski, M.J. Walshe, C.D. Warrick,H.R. Wilson, S. You, MAST Team, NBI Team, *First results from MAST*, Nuclear Fusion, Vol. 41, No. 10, (2001).

[Takahashi et al, 1995] Takahashi, Shigeo and Ikeda, Tetsuya and Shinagawa, Yoshihisa and Kunii, Tosiyasu L. and Ueda, Minoru, *Algorithms for Extracting Correct Critical Points and Constructing Topological Graphs from Discrete Geographical Elevation Data*,Computer Graphics Forum},14:3, pp 181—192, (1995).

[vizir, 2013] https://www.rocq.inria.fr/gamma/gamma/vizir/

http://www.robertschneiders.de/meshgeneration/software.html