Adaptive mesh refinement for numerical simulation of MHD instabilities in tokamaks: JOREK code

Hocine Sellama, Guido Huijsmans, Pierre Ramet

Adaptive mesh refinement for numerical simulation of MHD instabilities in tokamaks: JOREK code

Hocine Sellama Guido Huijsmans, Pierre Ramet

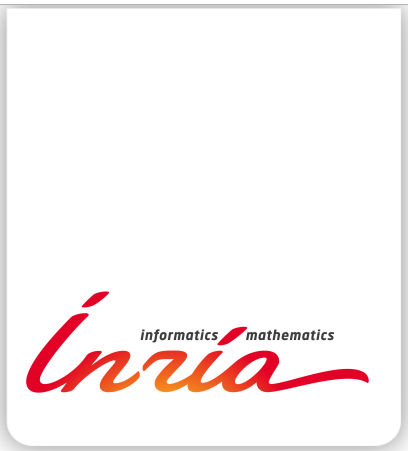
RESEARCH REPORT

N° 8635

November 2014

Project-Teams Hiepacs

ISSN 0249-6399 ISRN INRIA/RR--8635--FR+ENG



Adaptive mesh refinement for numerical simulation of MHD instabilities in tokamaks: JOREK code

Hocine Sellama*†‡§ Guido Huijsmans¶, Pierre Ramet*†‡§

Project-Teams Hiepacs

Research Report n° 8635 — November 2014 — 19 pages

Abstract: The purpose of this paper is to illustrate both validity and advantages of the implementation of the adaptive mesh raffinement strategy in the recent version of the $3D$ non-linear MHD code JOREK which uses a technique based on the bicubic Bézier surfaces developed in the paper of Czarny-Huijsmans[1]. We describe the physcal model and establish a refinement criteria. Then, we also present the numerical results of adaptive mesh raffinement simulation for the a tearing instability test case and to the test case of injection mechanism of a small pellet of frozen hydrogen into a tokamak.

Key-words: Adaptive mesh refinement, Bézier finite elements, MHD, plasma, instabilities, tearing mode, pellets injection.

* Bordeaux University, Talence, France
† Inria Bordeaux, Talence, France
‡ Institut Polytechnique de Bordeaux, Talence, France
§ CNRS - LaBRI UMR 5800, Talence, France
¶ ITER Organization, St Paul Lez Durance, France

RESEARCH CENTRE
BORDEAUX – SUD-OUEST

200 avenue de la Vieille Tour
33405 Talence Cedex

Raffinement de maillage adaptatif pour la simulation numérique des instabilités MHD dans les tokamaks : le code JOREK

Résumé : L'objectif de ce papier est de présenter à la fois la validité et les avantages de l'implémentation d'une stratégie de raffinement de maillage dans la dernière version du code JOERK pour la simulation MHD non linéaire qui utilise des surfaces de Bésier présenté dans Czarny-Huijsmans[1]. Nous décrivons tout d'abord le modèle physique ainsi que le critère de raffinement. Puis nous présentons des résultats numériques d'une simulation utilisant le raffinement de maillage adaptatif pour un cas test d'instabilité de mode "tearing" ainsi que pour un cas test d'injection d'un glaçon d'hydrogène dans un tokamak.

Mots-clés : Raffinement de maillage adaptatif, Eléments finis de Bézier, MHD, plasma, instabilitées, mode "tearing", injection de glaçons.

1 Introduction

In this paper, we describe our algorithm of adaptive refinement strategy for solving the MHD equations . Building on a technique used in Demkowicz et al[10] and the work presented in O. Czarny, G. Huijsmans[1], where an approach based on bicubic Bézier surfaces has been developed to the simulation of MHD instabilities(using 3D JOREK code) which provides geometric continuity C^1 and naturally allows to implement the refinement procedure. They had implemented a static(at equilibrium) refinement strategy for the 2D version of the JOREK code.

Our dynamic refinement process is intended to increase the accuracy of spatial discretization in regions where the spatial scales are insufficiently resolved and decrease computing time for the same resolution with a simulation without refinement, in particular, the surfaces which present a deformations due to the appearance of instabilities. This technique is developed and implemented in 3D JOREK code to improve the simulation of MHD instabilities that are needed to evaluate mechanisms to control the energy losses observed in the standard tokamak operating scenario(ITER), and numerical simulation of these phenomena becomes crucial for better understanding them. As a consequence, it is important to refine the grid as much as possible where instabilities are formed. This should be done adaptively, because the location of the instabilities can change, as in the case of the pellets injection.

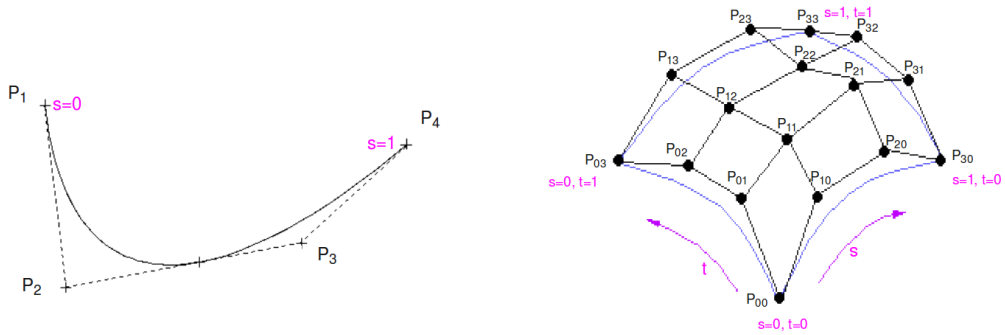
The applications we study are complex in that they require simulations at a very large scale; several tens or hundreds teraflops of computation using several terabyte of data are often needed. To carry out these more and more accurate full-scale simulations without increasing the number of unknowns in a uniform way, a technique consists in handling a finer grid where the solution varies abruptly and a coarser grid at other places accordingly to some regularity rules, and quite specific criteria. A mechanism which detects the appearance of instabilities is implemented. Mesh refinement is controlled by a specific criteria which are established according to the objective of the grid adaptive refinement and the simulated physical model.

This paper is divided into six sections and a conclusion. In the first section, we summarize and remind the definition of the Bézier surfaces and the parametric representation used in[1] for initial grid mesh. The second part of the paper concentrates to describes the refinement strategy, define the parametric representation of the new elements and establish the dof for the new nodes(active, constrained). In the third section, we describe the implementation and performance of a suitably preconditioned GMRES solver in JOREK. This scheme leads independent matrices that are factorized and solved in parallel using the PaStiX solver. This solver is suitable for any heterogeneous parallel distributed architecture such as clusters of multicore nodes. In particular, it offers a high performance version with a low memory overhead for multicore node architectures, which fully exploits the advantage of shared memory by using an hybrid MPI-thread implementation. The purpose of the fourth section is to explain and describe the physical models simulated in this paper to test the the efficiency of adaptive refinement method. The fifth is devoted to the refinement criteria used to determine the elements to be refined. We give criteria for refinement simulated physical models and motivate, to physical and numerical point of view , our choice.

in the sixth section, we present the numerical results for tow test cases(tearing mode, pellet injection) and compare them with those for a uniform grid without refinement, we discuss the behavior of growth rates of unstable turbulent mode and the effect of refinement.

The Bézier surface is formed as the cartesian product of the blending functions of two orthogonal Bézier curves. Bézier curves are a special case of cubic Hermite interpolation. the curves are constructed as a sequence of cubic segments, rather than linear ones. But whereas Hermite interpolating polynomials are constructed in terms of derivatives at endpoints, Bezier curves uses

a construction with Bernstein polynomials, in which the interpolating polynomials depend on certain control points. In what follows, we use the cubic and bicubic Bernstein polynomials.



(*a*) (*b*)

Figure 1: Left, cubic Bézier curve. Right, bicubic Bézier surface

A cubic Bézier curve is simply described by four ordered control points, P_1, P_2, P_3, and P_4. It's traced by leaving the point P_1, by heading to P_2 and by arriving at the point P_4 according to the direction $(P_3; P_4)$. Generally, the curve does not pass through P_2 nor P_3: these points are simply there to give an information of direction. The distance between P_1 and P_2 determines the length of displacement in the direction of P_2 before turning to P_4. A Bézier curve passes through its two end control points; P_1 and P_4, these two special control points are called nodes. The parametric representation for a Bézier curve is described by the scalar product expression

$$H(s) = \langle \vec{S}(s),\, \vec{C}_p \rangle \tag{1}$$

where $\vec{S}(s) = (B_0(s), B_1(s), B_2(s), B_3(s))^T$, $\vec{C}_p = [P_1, P_2, P_3, P_4]^T$, and $B_i(s) = C_3^i s^i (3-s)^{3-i}$, are the cubic Bernstein polynomials and P_i, $i = 1..4$, are the control points.

We can rewrite the parametric representation in nodal values and vectors in the form:

$$H(s) = \langle \vec{S}_v(s),\, \vec{C}_v \rangle \tag{2}$$

where

$$\begin{aligned}
\vec{C}_v &= \left[P_1, \vec{u}_1, \vec{u}_4, P_4\right]^T \\
\vec{S}_v(s) &= \left[\,(1-s)^2\,(1+2s)\,,\; s\,(1-s)^2\,,\; s^2\,(s-1)\,,\; s\,(3-2s)\,\right]^T \\
\vec{u}_1 &= P^{'}(s=0) = 3\,(P_2 - P_1) \\
\vec{u}_4 &= P^{'}(s=1) = 3\,(P_4 - P_3).
\end{aligned}$$

To extend the original parametric representation for a Bezier curve into Bezier surface, we evaluate the influence of all sixteen control points $P_{i,j} = \big(x, y, z\big)_{ij}$, for $i, j = 0..3$. In addition, more function values can be added to the $P_{i,j}$ vectors to describe the variation values along the curve $P_{i,j} = \big(x, y, z(x,y)\big)_{ij}$. The parametric representation for a Bezier surface is described by

$$H(s,t) = \langle \vec{S}(s),\, \mathcal{P} \cdot \vec{S}(t) \rangle \tag{3}$$

where $\mathcal{P}$ is a matrix whose coefficients are the control points $P_{ij}, i, j = 0, ..3$.

A Bézier surface necessarily passes through its four corner control points. These special control points are called nodes or vertices of Bézier patche.
The parametric representation in nodal values and vectors on one element, whose vertices are noted from 1 to 4, is given by

$$H(s,t) = \sum_{j=1}^{4} \langle \vec{S}_j(s,t),\, \vec{C}_j \rangle \tag{4}$$

where

$$\begin{aligned}
\vec{S}_1(s,t) &= (t-1)^2 (s-1)^2 \left[(1+2\,s)(1+2\,t),\, 3s(1+2\,t),\, 3t(1+2\,s),\, 9s\,t \right]^T \\
\vec{S}_2(s,t) &= s^2 (t-1)^2 \left[(1+2\,t)(3-2\,s),\, 3(1+2\,t)(s-1),\, 3t(3-2\,s),\, 9t(s-1) \right]^T \\
\vec{S}_3(s,t) &= s^2 t^2 \left[(3-2\,t)(3-2\,s),\, 3(2\,t-3)(1-s),\, 3(t-1)(3-2\,s),\, 9(1-t)(1-s) \right]^T \\
\vec{S}_4(s,t) &= t^2 (1-s)^2 \left[(3-2\,t)(1+2\,s),\, 3s(3-2\,t),\, 3(t-1)(1+2\,s),\, 9s(t-1) \right]^T,
\end{aligned}$$

$$\vec{C}_j = \left[P_j,\, h_u^j \vec{u}_j,\, h_v^j \vec{v}_j,\, h_w^j \vec{w}_j \right]^T \quad j = 1, ..4.$$

with

$$P_1 = P_{00}, P_2 = P_{30}, P_3 = P_{33}, P_4 = P_{03}$$

$$\begin{aligned}
\vec{u}_1 &= \frac{1}{3} H_s^{'}(0,0) = (P_{10} - P_{00}), & \vec{u}_2 &= \frac{1}{3} H_s^{'}(1,0) = (P_{30} - P_{20}), \\
\vec{v}_1 &= \frac{1}{3} H_t^{'}(0,0) = (P_{01} - P_{00}), & \vec{v}_2 &= \frac{1}{3} H_t^{'}(10) = (P_{31} - P_{30}) \\
\vec{w}_1 &= \frac{1}{9} H_{s,t}^{''}(0,0) = (P_{11} + P_{00} - P_{01} - P_{10}), & \vec{w}_2 &= \frac{1}{9} H_{s,t}^{''}(1,0) = 9(P_{20} + P_{31} - P_{21} - P_{30}) \\
\vec{u}_3 &= \frac{1}{3} H_s^{'}(1,1) = (P_{33} - P_{23}), & \vec{u}_4 &= \frac{1}{3} H_s^{'}(0,1) = (P_{13} - P_{03}) \\
\vec{v}_3 &= \frac{1}{3} H_t^{'}(1,1) = (P_{33} - P_{32}), & \vec{v}_4 &= \frac{1}{3} H_t^{'}(0,1) = (P_{03} - P_{02}) \\
\vec{w}_3 &= \frac{1}{9} H_{s,t}^{''}(1,1) = (P_{33} + P_{2,2} - P_{2,3} - P_{32}), & \vec{w}_4 &= \frac{1}{9} H_{s,t}^{''}(0,1) = (P_{13} + P_{02} - P_{0,3} - P_{12})
\end{aligned}$$

Consider a field $\phi\,(x(s,t), y(s,t))$ defined over a 2D geometrical domain. In Czarny-Huijsmans[1], it was explained that a corner where 4 Bézier patches meet is defined by 9 control points and if we assume that the 2D mesh (x_{ij}, y_{ij}) is given, the 9 values ϕ_{ij} are the degrees of freedom for ϕ at one node. Using the continuity G^1 between bezier patches, we reduce the degrees of freedom for ϕ to 4 (see [1]). The 4 variables per node are

$$\phi, \frac{\partial \phi}{\partial s}, \frac{\partial \phi}{\partial t}, \frac{\partial \phi}{\partial s \partial t},$$

which are the third compenents of $P, \vec{u}, \vec{v}, \vec{w}$ respectively. Therefore, the bicubic representation on one element is completely defined by the position P and the 3 directions $(\vec{u}, \vec{v}, \vec{w})$ at these

four nodes(vertices) P_1, P_2, P_3, P_4.
In the case of a several fields $U_k(s,t), k = 1,..N_{var}$, the degree of freedom per node for U is $4 \cdot N_{var}$ and the variables are

$$U_{kj}, \frac{\partial U_{kj}}{\partial s}, \frac{\partial U_{kj}}{\partial t}, \frac{\partial U_{kj}}{\partial s \partial t}, \quad j = 0,..N_{nodes}, \quad k = 1,..N_{var}.$$

The total number of degrees of freedom is therefore $4N_{nodes}N_{var}$.

In the case of 3D general toroidal geometry the variables are

$$U_{kj}(\varphi), \frac{\partial U_{kj}}{\partial s}(\varphi), \frac{\partial U_{kj}}{\partial t}(\varphi), \frac{\partial U_{kj}}{\partial s \partial t}(\varphi), \quad j = 0,..N_{nodes}, \quad k = 1,..N_{var}$$

where φ is the toroidal angle. Using the Fourrier harmonics in the dirction of symmetry φ, we obatin the expansion coefficients

$$U_{kjm}, \frac{\partial U_{kjm}}{\partial s}, \frac{\partial U_{kjm}}{\partial t}, \frac{\partial U_{kjm}}{\partial s \partial t}, \quad m = 1,..N_{tor}. \quad j = 0,..N_{nodes}, \quad k = 1,..N_{var}$$

where N_{tor} is the number of toroidal harmonics. The total number of degrees of freedom for the field U is therefore $4N_{nodes}N_{var}N_{tor}$. A typical number (for 16 harmonics on a grid of 101 by 128 finite elements) is of the order of 5 million.

2 Adaptive mesh refinement

The small size of the elements of the grid allows a higher resolution and a flexible mesh adaptation, but this is at the expense of an increased amount of computational overhead and memory. The dynamic refinement process provides the ability to add new elements and nodes where and when needed during the simulation(in the time loop) to improve the resolution in certain areas of the simulation domain. This process is controlled by some appropriate refinement criteria and regularity rules.

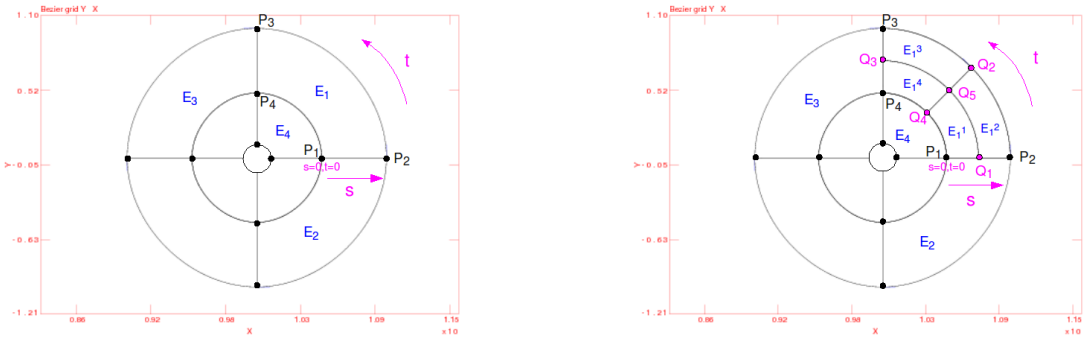
If we start with an initial grid of given resolution, at each time step, we determine the elements to be refined using the nodes where the refinement criteria are fulfilled(e.g. for the a tearing instability test case, the regions of strong gradients of current or pressure) described in detail later. The refinement consists of subdividing an element of the initial grid(parent) into two or four elements(sons) which involves the creation of new nodes and adds degrees of freedom($d.o.f$) to the problem.

More concretely, Let E_1 a Bézier patche parameterized by the surface $H(s,t)$ and E_2, E_3, E_4 are her neighbors (figure 2). The surfaces $H(s = \alpha, t), H(s, t = \beta)$, where $\alpha, \beta \in [0,1]$ are constants chosen initially (here, $\alpha = \beta = 0.5$), subdivide the patche E_1 into four Bézier patches, $E_1^1, E_1^2, E_1^3, E_1^4$. This operation creates five additional nodes $\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3, \mathcal{Q}_4, \mathcal{Q}_5$. The Bézier coefficients (position and vectors basis) of the new nodes can be determined from Bézier representation of the refined element using the local coordinates of these nodes. Indeed, the position and the vector basis at these nodes are defined by (see[1]):

$$P_{\mathcal{Q}_i} = H(\alpha_i, \beta_i), \ \vec{u}_{\mathcal{Q}_i} = \frac{1}{3} H_s^{'}(\alpha_i, \beta_i), \ \vec{v}_{\mathcal{Q}_i} = \frac{1}{3} H_t^{'}(\alpha_i, \beta_i), \ \vec{w}_{\mathcal{Q}_i} = \frac{1}{9} H_{s,t}^{''}(\alpha_i, \beta_i), \ i = 1,..5, \quad (5)$$

where (α_i, β_i) are the local coordinates of the new nodes $\mathcal{Q}_i$, $i = 1,..5$. The node $\mathcal{Q}_5$ is an active node and the $4\,N_{var}\,N_{tor}$ degrees of freedom are:

$$U_{\mathcal{Q}_5,km}, \frac{\partial U_{\mathcal{Q}_5,km}}{\partial s}, \frac{\partial U_{\mathcal{Q}_5,km}}{\partial t}, \frac{\partial U_{\mathcal{Q}_5,km}}{\partial s \partial t}, \ k = 1,..N_{var}, \ m = 1..N_{tor}$$



(*a*) (*b*)

Figure 2: (a)-Simple polar grid before refinement (b)- After refinement (element E_1)

which be represented in the stiffness matrices and RHS of the sons elements $E_1^1, E_1^2, E_1^3, E_1^4$. The node $\mathcal{Q}_2$ will be treated as a boundary node. $\mathcal{Q}_1, \mathcal{Q}_3, \mathcal{Q}_4$, are called constrained nodes whose the nodal values (dof) can be obtained according to the values at parent nodes $(P_1, P_2), (P_2, P_3)$, $(P_3, P_4), (P_4, P_1)$ respectively, without adding an additional degrees of freedom to the problem. Therefore, in the stiffness matrices and RHS of sons elements, the constrained nodes are replaced by their parent nodes. Indeed, by determining the position ip of new nodes in the new element which it belongs(e.g the node $\mathcal{Q}_4$ is at position $ip = 3$ in the element E_1^1 and first position($ip = 1$) in the element E_4^1), we can determine, the rows of the stiffness matrix of the parent element that will be modified to obtain the stiffness matrix of son element by the following relationship: for $i_{tor} = 1 \dots N_{tor}$, $i_{var} = 1 \dots N_{var}, i = 1 \dots 4$

$$pos + (i_{tor} - 1) \times N_{var} \times N_{tor} + N_{tor}(i_{var} - 1) + i - 1, \tag{6}$$

where

$$pos = (ip - 1) \times 4 \times N_{var} \times N_{tor} + 1$$

The degrees of freedom for the constrained nodes $\mathcal{Q}_1, \mathcal{Q}_3, \mathcal{Q}_4$ can be written linearly according to the degrees of freedom of their parent nodes.

Let $A_{E_1^1} U^m_{E_1^1} = rhs_{E_1^1}$ be the local linear system obtained using the Bézier finite elements method on the new element E_1^1, where $A_{E_1^1}$ is the local stiffnes matrix and

$$\vec{U}_{E_1^1} = \left(\vec{C}_{P_1}, \vec{C}_{\mathcal{Q}_1}, \vec{C}_{\mathcal{Q}_5}, \vec{C}_{\mathcal{Q}_4}\right)^T \tag{7}$$

with

$$\vec{C}_{P_1} = \left(U_{P_1,km}, \frac{\partial U_{P_1,km}}{\partial s}, \frac{\partial U_{P_1,km}}{\partial t}, \frac{\partial^2 U_{P_1,km}}{\partial s \partial t}, \quad k = 1 \dots N_{var}, , \ m = 1 \dots N_{tor}\right)^T$$

$$\vec{C}_{\mathcal{Q}_j} = \left(U_{\mathcal{Q}_j,km}, \frac{\partial U_{\mathcal{Q}_j,km}}{\partial s}, \frac{\partial U_{\mathcal{Q}_j,km}}{\partial t}, \frac{\partial^2 U_{\mathcal{Q}_j,km}}{\partial s \partial t}, \ k = \dots N_{var}, \ m = 1 \dots N_{tor}\right)^T, \ j = 1, \ 5, \ 4,$$

is the load vector which contain the $4 \times 4 \times N_{var} \times N_{tor}$ $d.o.f$ at the nodes $P_1, \mathcal{Q}_1, \mathcal{Q}_5, \mathcal{Q}_4$ ($4 \times N_{var} \times N_{tor}$ $d.o.f$ per node). Using (5) and the local coordinates $(s = \alpha, t = 0)$, $(s = 0, t = \beta)$

of the nodes $\mathcal{Q}_1$ and $\mathcal{Q}_4$ respectively, we show that the *d.o.f* of the constarined nodes $\mathcal{Q}_1, \mathcal{Q}_4$ depend linearly on the *d.o.f* of their parent nodes, (P_1, P_2) and (P_1, P_4) respectively, in the following way: If we define the matrices $M^1_{\mathcal{Q}_1}, M^2_{\mathcal{Q}_1}, M^1_{\mathcal{Q}_4}, M^2_{\mathcal{Q}_4}$ by

$$
\begin{aligned}
M^1_{\mathcal{Q}_1} &= \begin{pmatrix} (\alpha-1)^2(1+2\alpha) & 3\alpha(\alpha-1)^2 & 0 & 0 \\ 6\alpha(\alpha-1) & 3(3\alpha-1)(\alpha-1) & 0 & 0 \\ 0 & 0 & 3(1+2\alpha)(\alpha-1)^2 & 9\alpha(\alpha-1)^2 \\ 0 & 0 & 18\alpha(\alpha-1) & 9(3\alpha-1)(\alpha-1) \end{pmatrix} \\
M^2_{\mathcal{Q}_1} &= \begin{pmatrix} \alpha^2(3-2\alpha) & -3\alpha^2(1-\alpha) & 0 & 0 \\ -6\alpha(\alpha-1) & \alpha(3\alpha-2) & 0 & 0 \\ 0 & 0 & -3\alpha^2(-3+2\alpha) & 9\alpha^2(\alpha-1) \\ 0 & 0 & -18\alpha(\alpha-1) & 9\alpha(3\alpha-2) \end{pmatrix} \\
M^1_{\mathcal{Q}_4} &= \begin{pmatrix} (\beta-1)^2(1+2\beta) & 0 & 3\beta(\beta-1)^2 & 0 \\ 0 & 3(1+2\beta)(\beta-1)^2 & 0 & 9\beta(\beta-1)^2 \\ 6\beta(\beta-1) & 0 & 3(3\beta-1)(\beta-1) & 0 \\ 0 & 18\beta(\beta-1) & 0 & 9(3\beta-1)(\beta-1) \end{pmatrix} \\
M^2_{\mathcal{Q}_4} &= \begin{pmatrix} \beta^2(3-2\beta) & 0 & 3\beta^2(\beta-1) & 0 \\ 0 & 3\beta^2(3-2\beta) & 0 & 9\beta^2(\beta-1) \\ 6\beta(1-\beta) & 0 & 3\beta(3\beta-2) & 0 \\ 0 & 18\beta(1-\beta) & 0 & +9\beta(3\beta-2) \end{pmatrix}
\end{aligned}
$$

we have

$$
\begin{aligned}
\vec{C}_{\mathcal{Q}_1} &= \left[M^L_{\mathcal{Q}_1}, M^R_{\mathcal{Q}_1}\right] \begin{bmatrix} \vec{C}_{P_1} \\ \vec{C}_{P_2} \end{bmatrix} \\
\vec{C}_{\mathcal{Q}_4} &= \left[M^L_{\mathcal{Q}_4}, M^R_{\mathcal{Q}_4}\right] \begin{bmatrix} \vec{C}_{P_1} \\ \vec{C}_{P_4} \end{bmatrix}
\end{aligned} \tag{8}
$$

where $M^L_{\mathcal{Q}_p}, M^R_{\mathcal{Q}_p},\ p = 1, 4$, are the following block diagonal matrices of dimension $(4N_{var}N_{tor})^2$

$$
M^L_{\mathcal{Q}_p} = \begin{pmatrix} M^1_{\mathcal{Q}_p} & & & 0 \\ & M^1_{\mathcal{Q}_p} & & \\ 0 & & M^1_{\mathcal{Q}_p} & \\ & & & \ddots \end{pmatrix},\ M^R_{\mathcal{Q}_p} = \begin{pmatrix} M^2_{\mathcal{Q}_p} & & & 0 \\ & M^2_{\mathcal{Q}_p} & & \\ 0 & & M^2_{\mathcal{Q}_p} & \\ & & & \ddots \end{pmatrix}
$$

Therefore, we can rewrite the vector $\vec{U}_{E^1_1}$ by replacing the *dof* of the constrained nodes $\mathcal{Q}_1, \mathcal{Q}_4$ by the *dof* of the parents nodes, P_2, P_4 respectively, which do not belong to the element and obtain

$$
U_{E^1_1} = M_c \hat{U}_{E^1_1}, \text{ where } \quad \vec{\hat{U}}_{E^1_1} = \left(\vec{C}_{P_1}, \vec{C}_{P_2}, \vec{C}_{Q_5}, \vec{C}_{P_4}\right)^T, \tag{9}
$$

and

$$
M_c = \begin{pmatrix} I_d & 0 & 0 & 0 \\ M^L_{\mathcal{Q}_1} & M^R_{\mathcal{Q}_1} & 0 & 0 \\ 0 & 0 & I_d & 0 \\ M^L_{\mathcal{Q}_4} & 0 & 0 & M^R_{\mathcal{Q}_4} \end{pmatrix}
$$

The constrained nodes are treated using (8) without add an additional degrees of freedom for the problem and the local linear system of the element E_1^1 can be rewritten:

$$\hat{A}_{E_1^1}\hat{U}_{E_1^1} = \widehat{rhs}_{E_1^1}, \text{ with } \hat{A}_{E_1^1} = M_c^T A_{E_1^1} M_c, \ \widehat{rhs}_{E_1^1} = M_c^T rhs_{E_1^1}, \tag{10}$$

and the same method can be applied to the sons elements E_1^2, E_1^3, E_1^4.

To ensure that the refinement level differences of more than one between neighboring elements and the refined solution remains C^1-continuous, we impose the following regularity rule [10]: the father element has no constrained node i.e. the new node can not have a constrained parent node by requiring the systematic refinement of the neighboring element to remove the constrained node which becomes an active node(in Fig 3-*a* the nodes $\mathcal{Q}_1, \mathcal{Q}_4$ are constrained nodes which become active nodes in *b*).

In the case of refinement with several Levels, this regularity rule also allows not having to treat the problem the complexity of the dependance of the dof of the constrained nodes compared to the dof of their parent nodes(e.g without this rule, the dof of $q1$ would depend of dof of parent node $\mathcal{Q}_1$, and the degree of freedom of $\mathcal{Q}_1$, which would be a constrained node at the second level of refinement, would depend of dof of their parent nodes P_1, P_2).

Special precautions have to be taken during the refinement procedure to avoid that neighbors of the level of refinement smaller are refined again, as a consequence, it minimize the unnecessary increase of the number of elements and degrees of freedom.

To avoid interpolation along edges, the constrained nodes on the boundary become systematically a boundary nodes .

There is a risk of producing an irregular grid in the phase of the adaptive refinement i.e. the formation of nonuniform strictures(scattered holes throughout the mesh Fig 4). The formation of such structures may generate the occurrence of numerical noise and non-physical instabilities, in particular, the existence of constrained node in zone of strong gradients lead to the increase of the error. A solution is proposed and motived in section 5(refinement criteria).

The procedure of adaptive refinement can be summarize as follows:

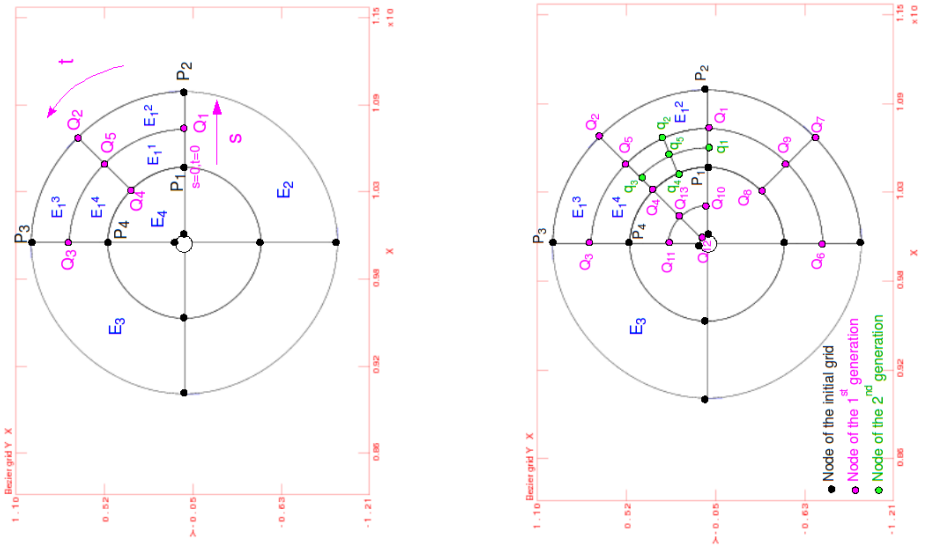
1. *We start with an initial 3D grid of given resolution, an initial list* $list_{elm}$ *of* N_{elm} *elements and list* $list_{node}$ *of* N_{node} *nodes . At the end of each time step, physical variables of simulated model are computed at each node on the initial grid.*

2. *At the end of each time step, we determine the list of elements to be refined using the nodes where the refinement criteria are fulfilled (e.g. the infinity norm of gradient* $\|\nabla\|_\infty$ *of current exceeds a given value for Tearing mode test case)*

3. *Return a list of elements that need to be refined:* $list(i)$, $i = 1..Nref$. *If the refinement is not needed(*$Nref = 0$*), by-pass the refinement activity steps and proceed to next time-step.*

4. *If* $Nref > 0$, *for each* $iref = 1..Nref$.

   - *Return a list of elements neighbors*
   - *Return a list of neighbors elements of* $iref$
   - *Checks whether the neighbors of* $iref$ *on sides need to be refined before the element* $iref$. *we store it on a "waiting list" and identify an element with a corresponding refinement kind that has to be refined first.*
   - *If the element* $iref$ *does not satisfy the regularity rule we store it on a waiting list and identify an element that has to be refined first*
   - *Return list with elements that need to be refined before* $iref$ *and refine them first*



(*a*) (*b*)

Figure 3: Illustration of the principle of refinement on a polar grid with several generations(Levels) of refined elements

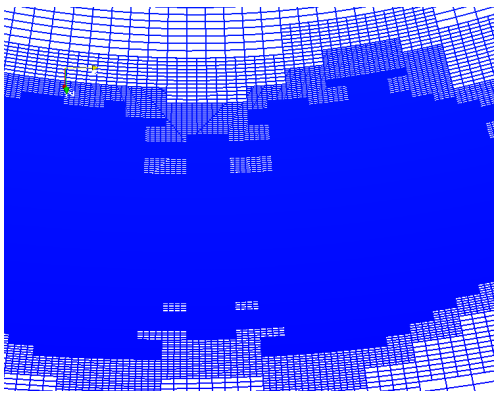


Figure 4: Unstructured grid induced by 3 levels of adaptive refinement

- *Create new nodes* $node_{iref}(n), n = 1, ..N_{new}$ *where* $N_{new} \leq 5$
  - *Compute local coordinates* $(P, \vec{u}, \vec{v}, \vec{w})_n,\ n = 1, ..N_{new}$ *of new nodes using(8).*
  - *Checks whether the new node already exist (the neighbor has sons meaning the new node already exists)*
  - *Determinate the nature of the new node(constrained , active, boundary). The constrained nodes on the boundary, become systematically a boundary nodes*

- *Add the new node to the node list* $list_{node}$
- *Initialize the index of new active nodes for global matrix containing all the degrees of freedom*
- *Create the new sons elements of* $iref$
  - *Determinate Bézier coefficients of sons element*
  - *Determinate the Generation rank of new element*
  - *Determinate neighbors of new elements(sons)*
- *Add the new element to the element list* $list_{elm}$*. The node numbers of the original mesh after the refinement are still kept the same*
- *Determinate the position of new nodes in the new element*
- *Construct the stiffness matrices and RHS of sons elements(the constrained nodes are replaced by their parent nodes) by modifying those the stiffness matrices and RHS of parent elements using(10)*

5. *Collect the element matrices into one large sparse matrix for unrefined elements(elements that do not have sons).*
6. *distribute nodes and elements over cpu's*
7. *The resulting matrix-vector system is solved using PaStiX solver*
8. *Update values of physical variables at constrained nodes using(8)*

This algorithm ensures a sufficient refinement which avoids refinement level differences of more than one between neighboring elements and ensures that the refined solution remains C^1 continuous. We fix the maximal number of elements to be refined according to the initial grid so that the refinement procedure has an interest, this precaution has to be taken before the refinement procedure.

Remark In our algorithm, it is possible to use the refinement in a single direction s or t (e.g. poloidal or radial direction for a polar grid) i.e. subdividing the parent element into two elements son, by setting ($\alpha = 0$ and $\beta = 0.5$) or ($\alpha = 0.5$ and $\beta = 0$), but this choice was shown to have limitations and proved ineffective for our application and choice of refinement criteria. For instance, if the direction of refinement is the same as direction where the gradient is strong, refinement will not be very effective.

3 JOREK code and solvers

In this section, we will sketch up the main steps of the JOREK environment (see [16] for more details on the JOREK code).

Spatial discretization and time integration scheme JOREK is a three dimensional MHD fluid code that takes into account realistic Tokamak geometry. High spatial resolution in the poloidal plane is needed to resolve the MHD instabilities at high Reynolds and/or Lundquist numbers. Bezier patches (2D cubic Bezier finite elements) are used to dicretize variables in this plane. Hence, several physical variables and their derivatives have a continuous representation over a poloidal cross-section. The periodic toroidal direction is treated via a sine/cosine expansion. The temporal discretization is performed by a fully implicit second-order linearized Crank-Nicholson scheme.

Set of equations Some of the variables modelled in JOREK code are: the poloidal flux Ψ, the electric potential u, the toroidal current density j, the toroidal vorticity ω, the density ρ, the temperature T, and the velocity $v_{parallel}$ along magnetic field lines. Depending on the model chosen, the number of variables and the number of equations on them are setup. At every time-step, this set of reduced MHD equations is solved in a weak form leading to a large sparse implicit system. The fully implicit method leads to very large sparse matrices. There are some benefits to this approach: there is no *a priori* limit on the time step, the numerical core adapts easily on the physics modelled (compared to semi-implicit methods that rely on additional hypothesis). There are also some disadvantages: high computational costs and high memory consumption for the parallel direct sparse solver such as PaStiX [15].

Equilibrium Each JOREK simulation begins with the soluion of the static magnetic equilibrium equation (so-called Grad-Shafranov equation) in the 2 dimensions of the cross-section plane. A key-point is the ability to handle magnetic equilibria which include an X-point. High accuracy is needed to have a correct representation of this equilibrium and avoid spurious instabilities whenever the whole simulation 3D+t is launched.

JOREK is able to build a Bezier finite element grid aligned with the flux surfaces both inside and outside the separatrix (*i.e.* the flux surface containing the X-point). This strategy allows one to improve the accuracy of the equilibrium representation. The flux surfaces are represented by sets of 1D Bezier curves determined from the numerical solution of the equilibrium. The Grad-Shafranov solver is based on a Picard's iteration scheme.

After the Grad-Shafranov solution step, a supplementary phase is required: the time-evolution equations are solved only for the $n=0$ mode (the first toroidal harmonic, *i.e.* purely axisymmetric) over a short duration. First, very small time-steps are taken, then they are gradually increased. This process allows the plasma equilibrium flows to establish safely in simulations involving a X-point.

Sparse solver & preconditioning A direct parallel sparse matrix solver (PaStiX or others) is used to solve the large linear systems within JOREK. In order to minimise the memory requirements of the fully implicit solver and to access larger domain sizes, a preconditioner accompanied with a GMRES iterative solver have been included a few years ago. *[can be removed] Preconditioning transforms the original linear system $A\,x = b$ into an equivalent one which is easier to solve by an iterative technique. A good preconditioner P is an approximation for A which can be efficiently inverted, chosen in a way that using $P^{-1}\,A$ or $A\,P^{-1}$ instead of A leads to a better convergence behaviour. Usually, GMRES iterative solver is applied in conjunction with a preconditioner. The preconditioner typically incorporates information about the specific problem under consideration.*

The JOREK *physics-based* preconditioner has been constructed by using the diagonal block for each of the *n_tor* Fourier modes in the toroidal direction of the matrix A. The preconditioner represents the linear part of each harmonic but neglects the interaction between harmonics (similar to a block-Jacobi preconditioning on a reordered matrix). So, we set many coefficients of the original matrix A to zero, in order to get a block-diagonal matrix with m independent submatrices on the diagonal. The preconditioner P consists in the composition of m independent linear systems $(P_i^\star)_{i\in[1,m]}$, with $m = \frac{n\_tor+1}{2}$. Practically, the set of processors are split in m independent MPI communicators, each of them treats only one single linear system $P_i^\star$ with a sparse direct solver.

Each submatrices can then be solved using an efficient version of PaStiX [a] for multicore node

[a] http://pastix.gforge.inria.fr

architectures, which uses a hybrid MPI-thread programming to fully exploit the advantage of shared memory. This approach will also be suitable for upcoming heterogeneous systems with GPU accelerators.

The preconditioned parallel approach avoids large costs in terms of memory consumption compared to the first approach that considers the whole linear system to solve (it saves the memory needed by the sparse solver to store the decomposition of whole A - *e.g.* L, U factors). Nevertheless, the whole linear system A has to be built (in parallel) in order to perform the matrix-vector multiplication needed by the GMRES. But, the cost in memory and in computation is far less than invoking the parallel sparse solver on the large matrix A.

This strategy improves the scalability (m independent systems), and the parallelisation performance of the code. The drawback is that in some specific circumstances, the iterative scheme may not converge.

4 Physical model

The present version of the JOREK code is implemented to solve the reduced MHD model in toroidal geometry. It uses a fully implicit scheme for the time evolution of the MHD equations(Crank-Nicholson scheme). The variables, poloidal flux ψ, potential u, current J, velocity v or (vorticity w), ρ and temperature T are discretized using 2D Bézier finite elements in the poloidal plane direction and Fourier harmonics in the toroidal direction.

Fixed large toroidal magnetic field written in term of the poloidal flux function ψ

$$\vec{B} = \frac{F_0}{R}\vec{e}_\varphi + \frac{R_0}{R}\vec{\nabla}\psi(t) \times \vec{e}_\varphi, \quad F_0 = B_0 R_0$$

the reduced MHD equations are given [3], [2] by

$$\frac{\partial \psi}{\partial t} = R^2 \eta(T) \nabla \cdot \left(\frac{1}{R^2} \nabla_\perp \psi \right) - R^2 \vec{B} \cdot \nabla u \tag{11}$$

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \vec{v}) + \nabla \cdot (D_\perp \nabla_\perp \rho) + S_\rho \tag{12}$$

$$\rho \frac{\partial T}{\partial t} = -\rho \vec{v} \cdot \nabla T - (\gamma - 1)\rho T \nabla \cdot \vec{v} + \nabla \cdot \left(\mathcal{K}_\perp \nabla_\perp T + \mathcal{K}_\parallel \nabla_\parallel T \right) + S_T \tag{13}$$

$$\rho \vec{B} \cdot \frac{\partial \vec{v}}{\partial t} = \vec{B} \cdot \left(-\rho \left(\vec{v} \cdot \vec{\nabla} \right) \vec{v} - \vec{\nabla}(\rho T) + \vec{J} \times \vec{B} + \mu \Delta \vec{v} \right) \tag{14}$$

$$J = \Delta^* \psi, \quad w = \nabla \cdot \nabla_\perp u, \quad \vec{v} = -\frac{R}{F_0}\vec{\nabla} u(t) \times \vec{e}_\varphi + v_\parallel(t)\vec{B} \tag{15}$$

4.1 The Tearing mode model

4.2 The Pellet injection model

5 Refinement criteria

In this section, the refinement criteria are established according to the objective of the grid adaptive refinement and the simulated physical model. The objectives of adaptive refinement process is to help understanding of plasma instabilities, especially the Edge Localized Modes (ELMS), and better estimate the energy loss due to this phenomenon. It is intended to increase the accuracy of spatial discretization in regions where the spatial scales become insufficiently resolved during the time simulation, in particular, the surfaces which present a deformations due

to the appearance of instabilities. This leads us to define a mechanism which detect the trigger of these instabilities and a refinement criteria. As a consequence, Our choice of refinement criteria is mainly motivated by physics arguments specific for our application i.e. MHD instabilities in tokamak plasmas and play important roles in simulation of these instabilities.

The finite plasma resistivity, by permitting a change of topology of magnetic configuration, may give rise to instabilities called tearing mode. The tearing modes, driven by current gradients, lead to the formation of magnetic islands (Fig. 5) on rational q surfaces and to the deformation of current and vorticity profile(Fig. 6:local flattening $q = 2, 3$).

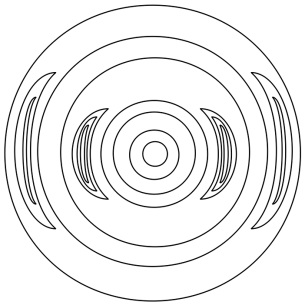


Figure 5: Formation of magnetic islands

(a) (b)

Figure 6: Local perturbation of current (a) and vortecity (b) for $\eta = 10^{-7}$

It is therefore natural to concentrate computational effort on this surfaces by refining it at the beginning of the simulation for static(predefined) refinement. In the case of adaptive refinement, the infinity norm of gradient $\|\nabla \cdot\|_\infty$, which can be used for a variety of problems, is chosen as the refinement criterion. Indeed, the gradients of a solution may be an indication for large errors or perturbations, in particular, the grid elements are refined according to the infinity norm of the gradient of current J or velocity w or electric potential u:

$$\|\nabla J\|_\infty \geq \varepsilon_{var} \quad var = J, u, w$$

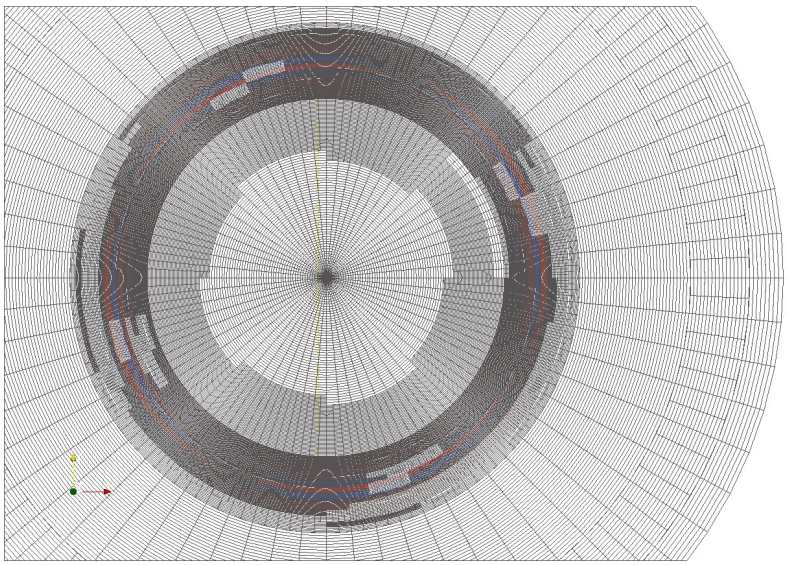
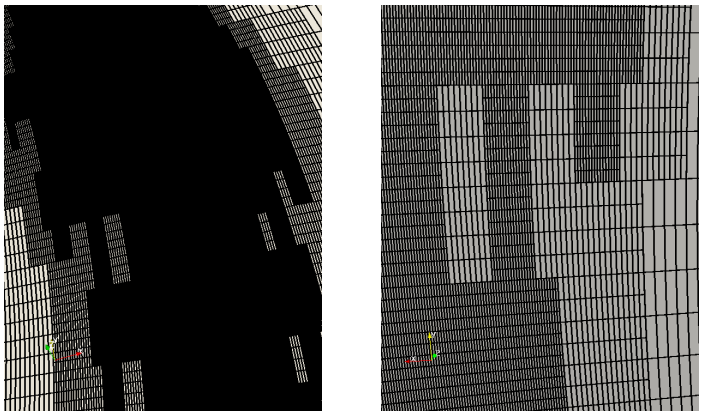


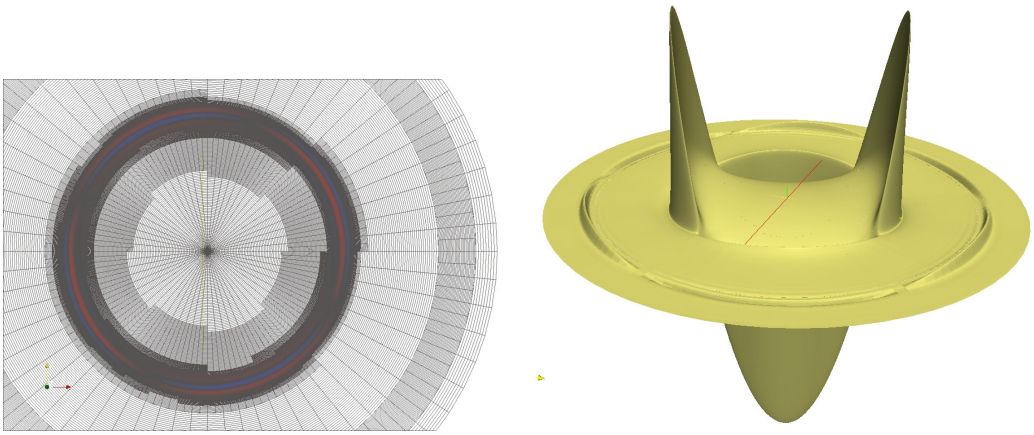
Figure 7: Irregular grid induced by the refinement without regularity control for tearing mode test case(with 3 levels)



(*a*) (*b*)

Figure 8: Zoom of irregular grid induced by the refinement without regularity control for pellets injection test case(with 3 levels)

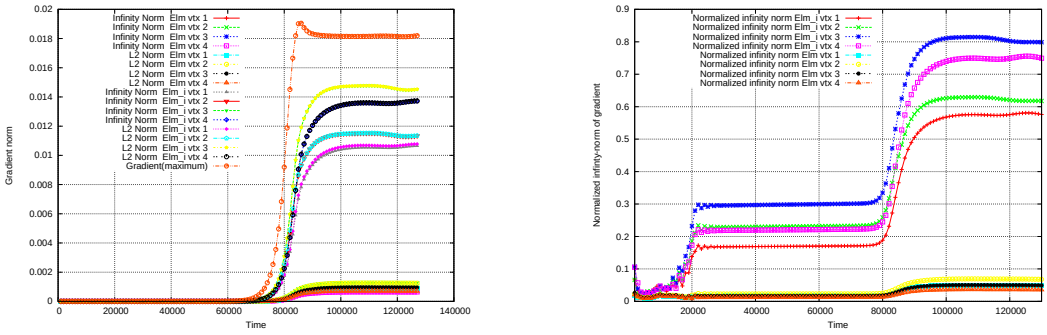
where ε_{var}, is threshold of gradient, fixed before the refinement procedure. The proper choices of ε_{var}, is very important for the sequence of refinement process. If we choose too small a value for ε_{var}, the refinement mechanism detects numerical noise rather than physical structures(instabilities), and which can result in too much refinement(too many elements). Otherwise, choosing ε_{var}, too large can result poor resolution. In figures 10, 11-b, we can observe that: in the middle part of the run, which corresponds to the phase between the trigger of tearing mode instability and the saturation, the current gradient for element of instability zone grows



(a) (b)

Figure 9: (a)- Refined grid obtained using adaptive refinement strategy (3 levels) with regularity control for tearing mode test case (b)- Electric potential

exponentially keeping pace with the maximum current Fig. 11, whereas, the current gradient for element away from the zone of instability grow very moderately(remains below of 2×10^{-3}). As consequence, ε_J can be chosen easily(here: $2 \times 10^{-3} < \varepsilon_J < 1.8 \times 10^{-2}$).



(a) (b)

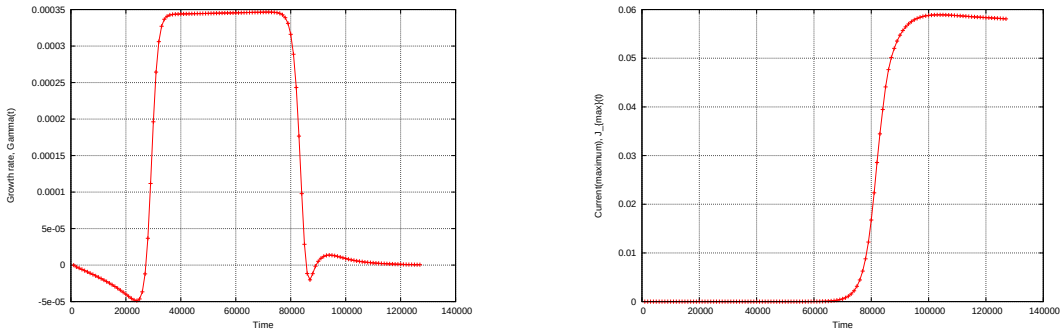
Figure 10: (a)- Current gradient norms as a function of time for four vertices of an element of instabilities zone(Elm_i) and four vertices of an other element(Elm) away from the zone of instability. The curve Gradient(maximum) denotes the maximum of current gradient on all elements(for $\eta = 10^{-6}$ without refinement, with polar grid 41×16). (b)- Normalized infinity norm of gradient $\frac{\|\nabla J\|_\infty}{|\nabla J|_{max}}$ as a function of time

We also note that other norm for refinement criteria can also be used. For example, for this application(tearing mode test case), the behavior L^2-norm remain similar to that of the Infinity-norm Fig 10. ε_J can be adaptively determined by calculating the growth rate of the current gradient at each time step.

- For $ne = 1 \ldots N_{elm}$

  - For $iv = 1 \dots N_{vertex}$
    - If $\|\nabla J\|_{\infty,iv} \geq \varepsilon_{var}$ or $\|\nabla J\|_{L^2,iv} \geq \varepsilon_{var}$ then, ne is added to the list of elements to be refined
    - Endif
  - Enddo

- Enddo



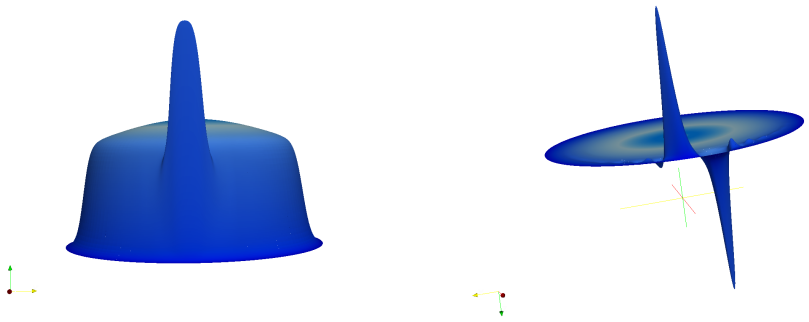
(a) (b)

Figure 11: (a)-Growth rate γ as a function of time (b)-the maximum current as a function of time(for $\eta = 10^{-6}$ without refinement, with polar grid 41×16)

The regularity rule introduced in [10] is not sufficient to prevent the emergence of nonuniform stricture(scattered holes throughout the mesh) of the refined grid (irregular grid Fig. 7 and 8). To avoid this problem induced by the refinement, we impose an additional control of grid regularity which consist of the systematic refinement of the neighbours of an element satisfying the refinement criterion, and any element having at least three refined neighbors (Fig. 9 the grid was refined around the zone of strong gradients using the control of regularity).

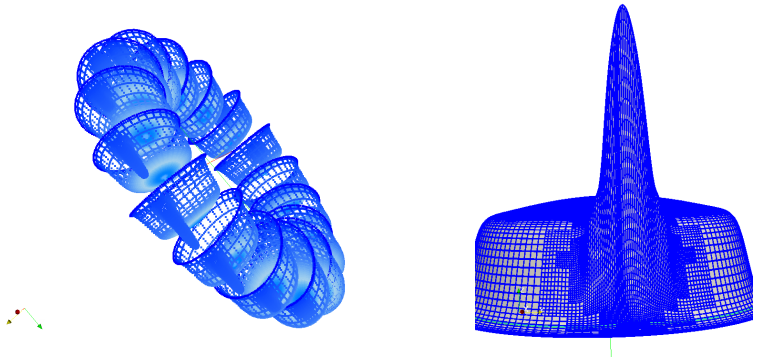
The numerical modeling of the pellet injection in an H-mode pedestal and the excitation of ELMs (by perturbing the initial density) was performed using the nonlinear 3D JOREK code. The injection of a pellet can trigger an instabilities and leads to the toroidal density perturbation (see Figures 12 and 13) which are characterized by a large pressure and density gradients in the vicinity of pellets.

Therefore, the choice of the gradient of density $\|\nabla\rho\|_\infty$ or parallel velocity $\|\nabla v_\parallel\|_\infty$ appears logical as a refinement criterion. The refined mesh obtained by the refinement algorithm for pellets injection test case using our refinement criterion is given in Figure 13.



(*a*) (*b*)

Figure 12: (a) Toroidal density perturbation (b) Toroidal perturbation of parallel velocity



(*a*) (*b*)

Figure 13: (a) perturbation of density profile with 11 harmonics in 3D (b) grid was refined around the zone of strong gradients (3 levels)

References

[1] O. Czarny, G. Huysmans, *Bézier surfaces and finite elements for MHD simulations. Journal of Computational Physics, 2008, vol 227, n 16, p 7423-7445*

[2] Briguglio, Zonca and Vlad, Phys. Plasmas Vol.5 No.9 (1998).

[3] G. Huysmans, MHD stability in X-point geometry: simulation of ELMs

[4] B. Philip et al. Implicit adaptive mesh refinement for 2D reduced resistive magnetohydrodynamics. Journal of Computational Physics, vol 227,N 20 (2008).

[5] R. Samtaney et al. 3D adaptive mesh refinement simulations of pellet injection in tokamaks. Comp Phys Comm 164 (2004) 220-228.

[6] Lang P. T. et al 2003 Nucl. Fusion 43 1110:20

[7] Lang P. T. et al 2007 Nucl. Fusion 47 754:61

[8] Lang P. T. et al 2008 Nucl. Fusion 48 095007

[9] G. T. A. Huysmans, S. Pamela, E. van der Plas and P. Ramet, Non-linear MHD simulations of edge localized modes (ELMs)

[10] L. Demkowicz, K. Gerdes, C. Schwab, A. Bajer, T. Walsh, HP90: A general and flexible Fortran 90 hp-FE code, Computing and Vizualization in Science 1 (1998) 145:163.

[11] H. R. Strauss, D. W. Longcopey, An Adaptive Finite Element Method for Magnetohydrodynamics, Journal of Computational Physics 147, 318:336 (1998)

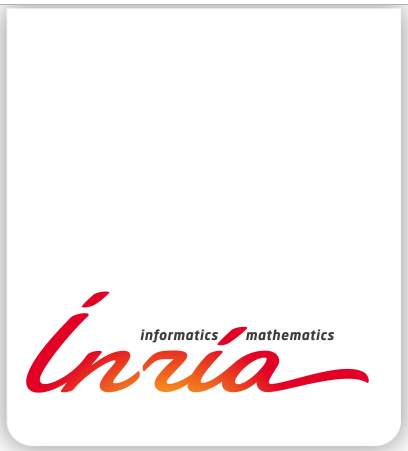
[12] H. Friedel, R. Grauer, C Marliani, Adaptive Mesh Refinement for Singular Current Sheets in, JOURNAL OF COMPUTATIONAL PHYSICS 134, 190:198 (1997).

[13] B. Fryxell, K. Olson, P. Ricker, F. X. Timmes, M. Zingale, D. Q. Lamb, P. Macneice, R. Rosner , J. W. Truran, H. Tufo FLASH: An Adaptive mesh Hydrodynamics code for Modeling Astrophysical Thermonuclear Flashes, THE ASTROPHYSICAL JOURNAL SUPPLEMENT SERIES, 131:273:334, 2000 November.

[14] U. Ziegler The NIRVANA code: Parallel computational MHD with adaptive mesh refinement

[15] Hénon, P. and Ramet, P. and Roman, J. PaStiX: A High-Performance Parallel Direct Solver for Sparse Symmetric Definite Systems, JOURNAL OF PARALLEL COMPUTING. 28, 301–321(2002).

[16] G. Latu, M. Becoulet, G. Dif-Pradalier, V. Grandgirard, M. Hoelzl, G. Huysmans, X. Lacoste, E. Nardon, F. Orain, C. Passeron, P. Ramet, A. Ratnani, Non regression testing for the JOREK code. INRIA Research Report http://hal.inria.fr/hal-00752270.



RESEARCH CENTRE
BORDEAUX – SUD-OUEST

200 avenue de la Vieille Tour
33405 Talence Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399