# Analysis of RWM in ITER including 3D volumetric blanket modules

M. Furno Palumbo[1], Y.Q. Liu[2], G. Rubinacci[3], S. Ventre[1], F. Villone[1]

[1]*Ass. EURATOM/ENEA/CREATE, DAEIMI, Università di Cassino, Italy*

[2]*Euratom/UKAEA Fusion Association, Culham Science Centre, OX14 3DB, UK*

[3]*Ass. EURATOM/ENEA/CREATE, DIEL, Università di Napoli Federico II, Italy*

The stabilization of Resistive Wall Modes (RWM) is an important concern for ITER. Indeed, such MHD instabilities will probably set performance limits to ITER advanced scenarios, unless some specific active feedback control schemes are envisaged.

In this respect, one issue that deserves attention is the effect of three-dimensional details of conducting structures surrounding the plasma. Usually, deviations from axisymmetry result in a detrimental effect on RWMs, in the sense that computations on a fully 3D mesh, including cuts, holes, ports and other details, often provide growth rates of the instability which are faster than purely axisymmetric estimates [1, 2].

However, there are some 3D features that can give a beneficial effect. One significant example is the presence of blanket modules (BM) inside the ITER vacuum vessel. These BM have a very complex three dimensional structure, with various different conducting materials (beryllium, copper, stainless steel), and several fine geometrical details, like slits, holes, channels etc. In addition, the BM are volumetric, in the sense that no straightforward thin-shell approximation is available. Here, we use the CarMa code [3] to analyse RWM in ITER with a detailed description of volumetric 3D BM. Due to the high number of BM (over 400), the resulting computational model is huge in terms of memory requirements and computational burden. To tackle this problem, a recently developed fast technique [4] will be used, together with suitable parallel computing methods.

## The CarMa code: model, fast technique, parallelization

The CarMa model [3] allows the analysis of RWM with 3D structures, including a volumetric description of the conductors in terms of a finite elements mesh. The resulting mathematical model is of the form:

$$\underline{\underline{L}}^{*}\frac{d\underline{I}}{dt}+\underline{\underline{R}}\,\underline{I}=\underline{V} \qquad (1)$$

where $\underline{I}$ is a vector of discrete 3D currents, $\underline{\underline{L}}^{*}$ is a modified inductance matrix, which takes into account the presence of plasma, $\underline{\underline{R}}$ is a 3D resistance matrix, $\underline{V}$ is related to external voltages. In order to study the RWM unstable eigenvalues, we must find the eigenvalues of the dynamical matrix resulting from (1). In [4] we have applied, to this purpose, the inverse iteration technique starting from an initial guess $\gamma_0$, which requires repetitive solutions of $\left(\underline{\underline{R}} + \gamma_0 \underline{\underline{L}}^*\right)\underline{I} = \underline{N}$ , where $\underline{N}$ is a suitable forcing term. This is done using an iterative method (GMRES), taking advance of the sparsification of the dense matrix $\underline{\underline{A}} = \left(\underline{\underline{R}} + \gamma_0 \underline{\underline{L}}^*\right)$, so that memory requirements and computation effort grows only almost linearly with the number of the unknowns. This allows the treatment of problems of huge dimensions, otherwise unaffordable.

To go further, we have parallelized this technique in order to tackle larger physical problems and to speed up the computation. To this purpose, two requirements must be respected: a) the computational cost to assembly the matrix $\underline{\underline{A}}$ must be equally distributed between the processors (***assembly balancing***); b) the computational cost and the memory occupation to implement the product $\underline{\underline{A}}\ \underline{I}$ must be equally distributed between the processors (***computation balancing***).

In more details, the vacuum part $\underline{\underline{L}}$ of $\underline{\underline{L}}^*$ is decomposed in the "near interaction" part (exactly computed) plus the "far interaction" part $\underline{\underline{L}}^{far}$ (approximated). Then, an efficient QR factorization is carried out to reduce the far interaction matrix between two boxes of a multilevel grid (see [4] for details). Let $ib_1$ and $ib_2$ two of such boxes; we call m (n) and $m_e$ ($n_e$) respectively the number of unknowns and the number of elements in $ib_1$ ($ib_2$). The local interaction matrix $\underline{\underline{B}}$ (of order $m\times n$) is factorized as the product of a matrix $\underline{\underline{Q}}$ (of order $m\times r$) times $\underline{\underline{R}}$ (of order $r\times n$), as $\underline{\underline{B}} \approx \underline{\underline{Q}}\ \underline{\underline{R}}$, where $r << m, n$.

The computational cost to assembly the interaction matrix is proportional to $m_e\times n_e$, and the reduced matrix dimension is equal $(m+n)\times r$, where r depend on the input required precision. Note that $(m+n)\times r$ is both the computational effort and memory resource needed to compute the local product $\underline{\underline{Q}}\ \underline{\underline{R}}\ \underline{x}$.

Assembly balancing and computational balancing is applied to $\underline{\underline{L}}^{far}$ and its product $\underline{\underline{L}}^{far}\underline{I}$. In particular, we follow three steps: 1) each processor assemblies and reduces all its local interactions; 2) each processor computes all local products $\underline{\underline{Q}}\ \underline{\underline{R}}\ \underline{x}$; 3) only at the end do the processors communicate each with the other to sum local results (we use MPI [5] as parallel library for communication between processors).

The local interactions are distributed in such a way that the total assembly cost per processor (proportional to the sum of $m_e\times n_e$ belonging to the processor) is balanced. Numerical experiments confirm that, if the size of the problem is large enough, the total reduced matrix per processor is automatically balanced among the processors. We notice that the computations of the box-box interactions are completely independent and the reduced memory is confined to the local processor: no communication overhead is necessary during this phase. This guarantees that the computational load decreases linearly with the number of processors.

Finally it is important to recall that an iterative solver method properly works if a preconditioner is used. The preconditioner used is the matrix $\underline{\underline{P}} = \underline{\underline{R}} + \gamma_0 \underline{\underline{L}}^*\big|_R$, where $\underline{\underline{L}}^*\big|_R$ is the matrix $\underline{\underline{L}}^*$ computed on the sparsity pattern of matrix $\underline{\underline{R}}$. As preconditioner solver we use MUMPS, a free parallel solver for large sparse matrices [6].

## Study of ITER volumetric blanket modules

The above described technique has been applied to the study of ITER volumetric blanket modules (BM), for an ITER Scenario-4 configuration, with $\beta_N \approx 2.9$. Such BM are arranged on 18 poloidal rows; the number of BM in the toroidal direction can be 18 or 36, depending on the position, the total number being 432. Each of them is a rather complex volumetric object, in which a lot of fine geometrical details are present (cooling channels, voids, slits etc.). We gave a rather realistic volumetric description of the object (Fig. 1), with an equivalent anisotropic resistivity, weighting the void fractions in the reference directions [7]. On a single insulated blanket module, the slowest 10 electromagnetic time constants agree within 15-20% with a finer mesh taking into account all the geometrical details.

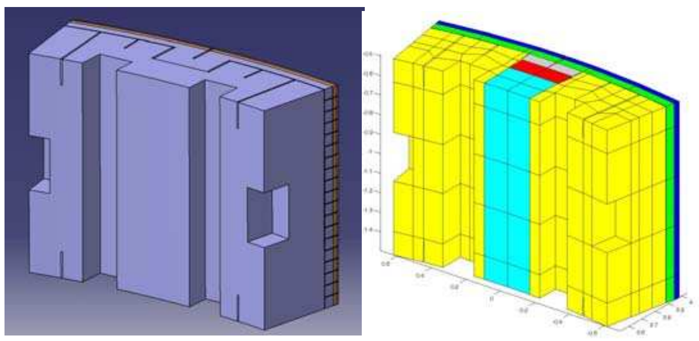
Fig. 1. Actual geometry of one BM and corresponding mesh.

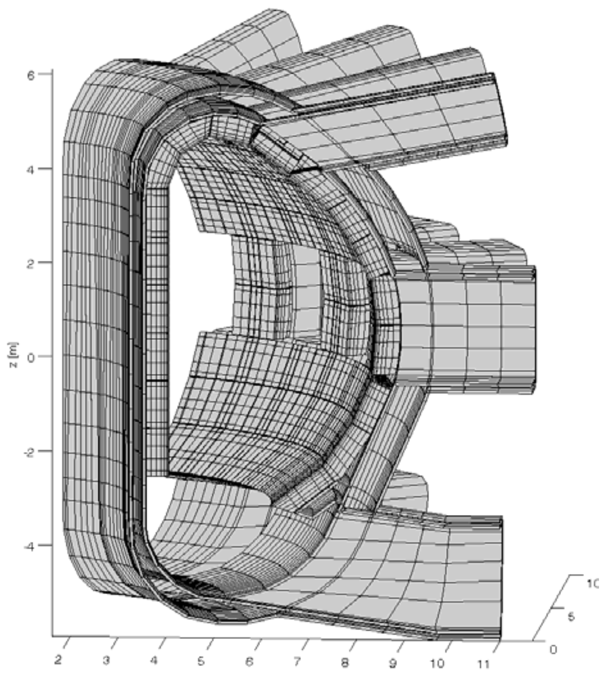

Fig. 2. Cutaway of ITER geometry.

We constructed several meshes, with an increasing number of BM in the various poloidal rows, as described in Table 1, where we also report the corresponding growth rates. We notice that the inner blanket modules (spanning rows from 1 to 6) have very little effect on the growth rate, while the inclusion of all BM provides a stabilizing effect of the order of 20%. Fig. 2 reports a cutaway of the mesh with all the BM – the actual one spans 360° toroidally. The computation time for this mesh has been around 14 hours, on a 32-processor computer. The authors acknowledge the computer resources, technical expertise and assistance provided by Barcelona Supercomputing Center - Centro Nacional de Supercomputación (Spain), and the help of Mathieu Gontier, Free Field Technologies, Belgium.

| BM rows included | Number of discrete unknowns | Estimated $\gamma$ [$s^{-1}$] |
|---|---|---|
| None (only vessel) | 24637 | 8.2 |
| 1-6 (inboard) | 68485 | 8.2 |
| 14-18 (outboard) | 89005 | 6.7 |
| 7-18 (top and outboard) | 162553 | 6.3 |
| 1-18 (all BM) | 206401 | 6.3 |

Table 1. Cases analysed and corresponding unstable RWM eigenvalues.

[1] F. Villone et al., *Phys. Rev. Lett.* **100** (2008) 255005

[2] F. Villone et al., *35th EPS Conf. on Plasma Phys.*, ECA Vol.32, P-2.080 (2008)

[3] A A. Portone, F. Villone et al., *Plasma Phys. Contr. Fusion* **50** (2008) 085004

[4] G. Rubinacci et al., *Journal of Computational Physics* **228** (2009) 1562–1572

[5] http://www.open-mpi.org/

[6] http://mumps.enseeiht.fr

[7] R. Albanese et al., “Electromagnetic disruption loads on ITER blanket modules”, submitted to COMPUMAG conference, November 2009