Title
High-resolution magnetohydrodynamic equilibrium code for unity beta plasmas

Journal
Journal of Computational Physics, 216(1)

Author
Gourdain, P A

Publication Date
2006-07-20

DOI
10.1016/j.jcp.2005.12.005

Peer reviewed

High resolution magnetohydrodynamic equilibrium code for unity beta plasmas

P.-A. Gourdain[a], J.-N. Leboeuf, and R. Y. Neches
Department of Physics and Astronomy, UCLA, Los Angeles CA 90024, USA

Abstract

There is great interest in the properties of extremely high-beta magnetohydrodynamic equilibria in axisymmetric toroidal geometry and the stability of such equilibria. However, few equilibrium codes maintain solid numerical behavior as beta approaches unity. The free-boundary algorithm presented herein utilizes a numerically stabilized multigrid method, current density input, position control, magnetic axis search, and dynamically adjusted simulated annealing. This approach yields numerically robust behavior in the spectrum of cases ranging from low to very high beta configurations. As the convergence time depends linearly on the total number of grid points, the production of extremely fine, low-error equilibria becomes possible. Such a code facilitates a variety of intriguing applications which include the exploration of the stability of extreme Shafranov shift equilibria.

Keywords
MHD
Magnetohydrodynamics
Plasma
Tokamak
High
Beta
Unity
Equilibrium
Multigrid
Shafranov
Category (AMS)
65N55 Partial differential equations, boundary value problems Multigrid methods

[a] gourdain@ucla.edu

1. Introduction

The Grad-Shafranov (GSh) equation describes magnetohydrodynamic (MHD) equilibria in toroidal axisymmetric confinement devices, such as tokamaks [1]. Numerical solutions thereof are very important for interpretation of experimental data, in the design of experiments in existing devices, and the development of new magnetic confinement configurations. MHD equilibrium solutions are also used for the initial conditions of macro- and micro-stability analysis, as well as particle and heat transport properties of magnetically confined plasmas.

Sophisticated techniques have been devised to solve the Grad-Shafranov (GSh) equation. There exist two distinct classes of numerical techniques: Lagrangian schemes that use curvilinear flux coordinates to map plasma geometry and which involve adaptive grid [2], variational [3], perturbative [4] or inverse coordinates [5] methods. The second class of techniques is based on an Eulerian scheme, relying on a two-dimensional (2D) mesh without any direct link to plasma shape [6, 7]. We will use a direct approach, where the geometric space is meshed instead of the flux space [8], so plasma with diverted configurations, i.e. X-points or separatrices, can be handled easily. Figures of merit for magnetic confinement devices and their equilibria are the dimensionless ratios of plasma pressure (p) to magnetic field (B) pressure

$$(1) \qquad \beta = \frac{2\mu_0 p}{B^2},\ \beta_P = \frac{2\mu_0 p}{B_P^2} \text{ and } \beta_T = \frac{2\mu_0 p}{B_T^2},$$

where the subscripts P and T denote respectively the poloidal and toroidal directions. These local quantities are defined where the pressure is maximum and have global counterparts,

$$(2) \qquad <\beta> = \frac{2\mu_0 < p >}{< B^2 >},\ <\beta_P> = \frac{2\mu_0 < p >}{< B_P^2 >} \text{ and } <\beta_T> = \frac{2\mu_0 < p >}{< B_T^2 >},$$

where <.> denotes the standard volume averaged quantities over the total plasma volume V_0, i.e.

$$(3) \qquad < g > = \frac{1}{V_0}\int_{V_0} g dV\ .$$

The pressure reaches a maximum at the magnetic axis. The plasma β determines the overall efficiency of magnetic confinement, while the poloidal beta (β_P) indicates the paramagnetism (if smaller than unity) or diamagnetism (if greater than unity) of the plasma. This quantity is directly linked to the Shafranov shift, the outboard radial shift of the magnetic axis. For MHD stability reasons [9, 10], viable high β plasma equilibria only exist for high β_P, with commensurate large Shafranov shifts. Herein, these configurations will generically be referred to as high-β.

The calculation of equilibria at high-β poses a significant challenge. High-β configurations are characterized by atypical plasma current profiles with a large Shafranov shift, as illustrated in Figure 1. Technical difficulties due to resolution and convergence have limited existing equilibrium codes to medium β configurations [11]. Few codes have been able to converge above 50% peak β [2, 12] and at least one of them can handle β above 80% [8], by making use of the inherent adaptivity of the contour dynamics

method [13]. Unfortunately, the results from this code cannot be used with standard MHD stability codes [14,15] due to the mostly sparse non-uniform mesh on which it solves the GSh equation.

To handle the flux compression due to large Shafranov shift at high-β, numerical convergence on a uniform mesh without adaptivity requires a high resolution mesh. Thus, a high-β solver requires a large grid. For practical reasons, a solver must also retain good performance for a wide range of cases. The multigrid method is the fastest elliptic solver for large problems [16] and meets the requirements of our equilibrium computations. We present in this paper a novel method which handles high-β plasma equilibria and bridges the gap between existing codes and future experiments. We will show that the method can compute unity β equilibrium solutions by combining the multigrid method [17] (also used in modern MHD codes [12, 18]) with a set of numerically stabilizing elements. These elements include shape control, magnetic axis search, and dynamically adjusted numerical relaxation. We will demonstrate that the numerical convergence of the overall algorithm is proportional to the total number of grid points, yielding good performance at high resolution.

Following this introduction, the physics model is presented in Section 2, then a summary of the multigrid method is given in Section 3 for completeness. Section 4 regroups the standard techniques for solving those boundary value problems in electromagnetism which are an integral part of solving the GSh equation. The novel techniques yielding convergence at high β are detailed in Section 5. Section 6 presents the complete computational algorithm. Code validation and performance analysis are discussed in Section 7, while Section 8 is devoted to the application of the numerical algorithm to extreme β equilibria. Conclusions are given in Section 9.

2. General MHD equilibrium

The physics model of the equilibrium of a plasma in the axisymmetric magnetic geometry of a tokamak is examined. The equilibrium is given by the GSh equation [19],

$$(4) \qquad \Delta^* \psi = -\left(\mu_0 R^2 \frac{dp}{d\psi} + \frac{1}{2} \frac{dF^2}{d\psi} \right) \text{ where } \Delta^* \psi \equiv R \frac{\partial}{\partial R} \left(\frac{1}{R} \frac{\partial \psi}{\partial R} \right) + \frac{\partial^2 \psi}{\partial Z^2}.$$

in the coordinate system (R, Z, ϕ), as illustrated in Figure 2. We can also express Eqn. (4) as a function of the toroidal current density flowing along the ϕ direction,

$$(5) \qquad \mu_0 J_\phi(R,\psi) = -\left(\mu_0 R \frac{dp}{d\psi} + \frac{1}{2R} \frac{dF^2}{d\psi} \right)$$

which yields

$$(6) \qquad \Delta^* \psi = \mu_0 R J_\phi(R,\psi)$$

These equations describe the local equilibrium between the gradient of the fluid pressure p and the Lorentz force. Normally, p increases monotonically from the plasma-vacuum interface to the magnetic axis [20], where it reaches a maximum. The quantity ψ is the flux of the poloidal induction B_p in the (R, Z) plane,

$$\vec{B}_p = -\frac{1}{R}\frac{\partial \psi}{\partial Z}\vec{e}_R + \frac{1}{R}\frac{\partial \psi}{\partial R}\vec{e}_Z . \quad (7)$$

The function F represents the net poloidal current flowing through a disk centered on the Z-axis and located on the (R,ϕ) plane [20],

$$F = B_\phi R . \quad (8)$$

It can be demonstrated that p and F are functions of ψ only. Thus, the surfaces of constant pressure correspond to surfaces of constant flux. The flux ψ varies monotonically from the plasma edge to the magnetic axis. Because of the presence of ψ on both sides of the equality, the GSh equation is classified as a nonlinear elliptic partial differential equation. In general, this type of equation can only be solved via iterative methods, such as a Picard iteration scheme [6]. The flux ψ_{n-1} at iteration step n-1 is used to find the flux ψ_n at step n,

$$\Delta^* \psi_n = \mu_0 R J_\phi(R, \psi_{n-1}) . \quad (9)$$

When

$$\varepsilon_n = |\psi_n - \psi_{n-1}| \quad (10)$$

is small enough, convergence is achieved and an approximate solution to the GSh equation has been found.

3. The multigrid method

First introduced by Brandt [17], the multigrid method uses uniform grids of different densities to solve elliptic partial differential equations. If the finest grid holds N points, the total computational time of the method is proportional to N. On the coarsest grid the discretized differential equation is solved exactly using classical methods, such as the successive over relaxation (SOR) algorithm [21]. On finer grids, only the correction to the coarse solution is computed. This correction requires less computation time than the exact solution. After successive iterations on the finest level, the solution can be approximated to a given error threshold. To illustrate the components of the algorithm, this section presents a two-grid example, the principal operators of the method, and the full multigrid (FMG) algorithm. Formal descriptions of this method are offered by Wesseling [22] and Trottenberg [23].

3.1. The two grid process

We wish to solve the following equation

$$\Xi U = f \quad (11)$$

where Ξ is a linear operator. We introduce the following notations:

- N is the total number of points in the grid;,
- the subscript m denotes the fine mesh size,
- the subscript M denotes the coarse mesh size,
- $\wp$ is a prolongation operator which transfers information from coarse to fine mesh,
- $\Re$ is a restriction operator which transfers information from fine to coarse mesh,
- U denotes the solution of the discretized equation,

- u denotes an approximation of the solution,
- the error between exact and approximate solutions is $v = U-u$,
- the residue, or defect, is $d = -\Xi v$.

For a simple approach, one may solve the problem on the fine grid by solving for the error between the exact and approximate solutions on the coarser grid. Figure 3 shows the two-grid example. The "defect" d_m between the non-converged solution u_m and f is restricted to the coarser grid, i.e. d_M, where we solve for the error v_M. Once v_M is found, it is interpolated back onto the fine grid, i.e. v_m. Then v_m is used to correct u_m and get a better approximation of the solution. The process continues until the convergence criteria are met. In this minimal example, the solution is computed only on the coarse grid. Thus, this approach will fail to find corrections that cannot be resolved on the coarsest grid. Some improvements are required to obtain a realistic elliptic solver.

3.2. Restriction, prolongation and smoothing operators

We suppose that all grid elements of the coarse grid M are included in the finer grid m and that the distance between coarse grid vertices is twice the distance of the fine grid vertices. The prolongation and restriction operators apply only to the elements that are not common to both m and M grids. The common elements to both grids do not need such operators as the information can be transferred directly. The prolongation operator $\wp$ is usually a simple bilinear interpolation, but may be more complex.

$$\wp = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}. \tag{12}$$

$\wp$ must be chosen carefully to maintain accurate transfer of the information. To find the corresponding restriction we need to define a functional scalar product in vector space

$$\left\langle u_m \middle| v_m \right\rangle_m = m^2 \sum_{x,y} u_h(R,Z) v_h(R,Z). \tag{13}$$

This allows us to define the restriction operator as follow

$$\left\langle u_M \middle| \Re v_m \right\rangle_M = \left\langle \wp u_M \middle| v_m \right\rangle_m. \tag{14}$$

Thus, where $\wp$ has the form of Eqn. (12), $\Re$ has the following mathematical representation,

$$\Re = \begin{bmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{bmatrix}. \tag{15}$$

Furthermore $\wp$ and $\Re$ are adjoint with respect to $<.|.>_m$. This minimizes information distortion between $\wp$ and $\Re$ due to the transition between grids.

The last aspect of the procedure is the smoothing operation. As seen in Figure 3, the only numerically significant computation takes place when solving for the defect on the coarse grid. Because the applied correction retains the precision of the coarser grid, when the defect is re-introduced in the approximate solution on the finer grid it does not give the converged solution. The precision on the correction computed on the coarse grid is insufficient for the finer grid. Thus, an operator solving for corrections on the fine grid has to be devised. This operator does not need to obtain an exact solution to the system; it should only solve for the part which cannot be resolved on the coarser grid. The efficiency of the multigrid method depends on the fact that this calculation requires significantly less computation than the exact solution. One may imagine the fine corrections as "ripples" over coarser corrections, and observe that this operator smooths the correction applied to the approximate solution. For linear system inversion, the most widely used smoothing operator is the Gauss-Seidel method [21].

Smoothing is usually applied before solving on the finer grid ("pre-smoothing") and after solving ("post-smoothing"). The addition of restriction, prolongation, and smoothing operators lends the two-grid algorithm much-needed robustness.

3.3. Multigrid algorithm

Generalizing from the previous example, we can easily devise an algorithm which uses more than two grids (Figure 4). The cycle between the grids m-n and m-(n-1) takes place exactly as in the two-grid method. Adding finer grids to this cycle is trivial; the solution on grid m-(n-2) goes through the restriction and prolongation processes, transferring information to and from grid m-(n-1). Once the defect is known on the m-(n-1) grid, additional restrictions and prolongations enable full resolution on the coarsest level m-n. To resolve corrections on finer grids, a smoothing process takes place before transferring the information from one grid to another. This process can be repeated several times. The general multigrid algorithm can thus utilize an arbitrary number of grids. For a grid m-k, the number of cycles is γ_k.

As described, the multigrid algorithm traverses between the finest and the coarsest grid. At the expense of speed, the quality of the solution can be improved by reversing the direction of traversal at an intermediate level ($\gamma_0, \ldots, \gamma_{n-1} \geq 2$). When the algorithm traverses all grids in order of rank, it is called the "V-cycle" (Figure 5-a); when the traversal is reversed at intermediate grid ranks, it is called the "W-cycle" (Figure 5-b). The reversing criterion, and thus the quantity γ_k, is controlled by the "schedule" of grid m-k. For a fixed error threshold, the schedule parameter may be adaptive; the number of cycles at a given grid rank will be adjusted to accommodate the error threshold.

In principle, the multigrid algorithm will converge when a null solution is given as first guess. However, using the exact solution on the coarsest mesh is a better initial guess to accelerate the convergence. The addition of restriction operators, prolongation operators, smoothing operators and schedule controls to the simple two-grid example results in the full multigrid algorithm (FMG), illustrated in Figure 5-c with 4 grids.

4. Resolution of the problem on a Cartesian grid

In this section, we describe a set of standard techniques which can solve efficiently the GSh equation on a Cartesian grid. To do so, we must establish a computational domain, invert the GSh equation on this domain, and consistently handle the coil and plasma contributions on the computational boundary. To match the magnetic topology of real machines, we will only consider free-boundary solutions. We allow the user to configure any external coil systems[b] constrained to the symmetry of a tokamak. The minimal set of coils is comprised of horizontal field coils to control the vertical position of the plasma (along the Z-axis), vertical field coils to control the horizontal position (along the R-axis), and elongation coils to control the plasma shape. We assume that the (R, Z, ϕ) space is magnetically homogeneous, and does not contain non-linear materials such as iron or μ-metals.

4.1. Domain of definition

The computational domain Ω is a simple Euclidian (rectangular) coordinate system encompassing the plasma-vacuum boundary (henceforth called the last closed flux surface, or LCFS). Particularly in free-boundary mode, it is highly advisable to allow a gap of several grid points between the LCFS and the mesh boundary $\partial\Omega$. Figure 6 summarizes the major properties of the domain of definition. We do not assume vertical symmetry, and thus the domain must circumscribe the whole plasma volume.

The computation domain is homogeneously discretized. The indexes i and j label the grid elements horizontally and vertically and increase with R and Z respectively. The mesh parameters ΔR and ΔZ define the distance between adjacent points in R and Z. The total number of mesh points vertically and horizontally must be a multiple of two of the coarsest grid, so it can be efficiently handled by the prolongation and restriction operators. It is convenient to define and subsequently generate the "stack" of grids from the coarsest one, using iterative formulas,

$$N^R_{m-k} = 2\times(N^R_{m-k-1}-1) \text{ and } N^Z_{m-k} = 2\times(N^Z_{m-k-1}-1) \text{ along the } R \text{ and } Z \text{ directions,} \quad (16)$$

where k is the coarseness level and k=0 labels the finest grid.

4.2. Interior equation and boundary conditions

Having established our definition of the computational domain, it is possible to invert Eqn. (4). This equation is discretized as follow

$$\frac{R_i}{\Delta R^2}\left\{\frac{1}{R_{i+1/2}}(\psi^n_{i+1,j}-\psi^n_{i,j})-\frac{1}{R_{i-1/2}}(\psi^n_{i,j}-\psi^n_{i-1,j})\right\}+\frac{1}{\Delta Z^2}\left\{\psi^n_{i,j+1}-2\psi^n_{i,j}+\psi^n_{i,j-1}\right\}=\mu_0 R_i J_{\phi_{i,j}}(R_i,\psi^{n-1}). \quad (17)$$

Because of the homogeneous mesh, ΔR and ΔZ are constant on the whole grid. If we define

[b] We define as "coil system" a set of electrically connected coils sharing a single function and feedback systems (vertical position, horizontal position or shape control). They are treated as if they were connected in series, and carry the same current.

$$(18) \qquad f_{i,j}^{n} = \mu_0 R_i J_{\phi_{i,j}}(R_i, \psi^{n-1})$$

$$(19) \qquad u_{i,j}^{n} = \psi_{i,j}^{n}$$

and

$$(20) \qquad \Xi u_{i,j}^{n} = \frac{R_i}{\Delta R^2 R_{i+1/2}} u_{i+1,j}^{n} + \frac{R_i}{\Delta R^2 R_{i-1/2}} u_{i-1,j}^{n} + \frac{1}{\Delta Z^2} u_{i,j+1}^{n} + \frac{1}{\Delta Z^2} u_{i,j-1}^{n} - \left\{ \frac{R_i}{\Delta R^2} \frac{1}{R_{i+1/2}} + \frac{R_i}{\Delta R^2} \frac{1}{R_{i-1/2}} + \frac{2}{\Delta Z^2} \right\} u_{i,j}^{n}$$

then we have

$$(21) \qquad \Xi u_{i,j}^{n} = f_{i,j}^{n}.$$

The multigrid method can be applied directly on the operator of Eqn. (21). Now that we have the interior equation we can examine the boundary conditions on $\delta\Omega$. The flux at the boundary of the domain comes from a set of external coils and the plasma itself. A tokamak requires at least three different coil systems, labeled *CS*, to operate properly. A current I_{CS} runs through each coil system *CS*. The flux created by such a set can be easily computed using

$$(22) \qquad \psi_{CS}(P) = I_{CS} \bar{\psi}_{CS}(P) \quad \forall P \in \Omega$$

where

$$(23) \qquad \bar{\psi}_{CS}(P) = \sum_{C \in CS} G(P, P_C) \quad \forall P \in \Omega.$$

G(P,P') is the Green's function of the Δ^* operator. It has the following analytic form [24]

$$(24) \qquad G(P,P') = -\frac{\sqrt{RR'}}{k}\left[\left(2-k^2\right)K(k) - 2E(k)\right] \quad \forall P, P' \in \Omega$$

where

$$(25) \qquad k = 2\sqrt{\frac{RR'}{\left(R+R'\right)^2 + \left(Z-Z'\right)^2}}.$$

K and *E* are the elliptic integrals of first and second kind. Hastings' polynomials [25] or infinite series [26] can be used to approximate such functions. The same method cannot be applied to the plasma itself as it is not computationally efficient. By using a standard technique involving the Von Hagenow theorem the plasma flux on the boundary, given by Eqn. (26),

$$(26) \qquad \psi(P) = \int_{plasma} G(P, P(R,Z)) J_\phi(R,Z) dR dZ \quad \forall P \in \partial\Omega.$$

can be computed using a function *V* such as

$$(27) \qquad \Delta^* V = \mu_0 R J_\phi \text{ on } \Omega/\partial\Omega$$

and

$$(28) \qquad V=0 \text{ on } \partial\Omega.$$

In this case, we obtain

$$(29) \qquad \psi(P) = \int_{\partial\Omega} G(P, P(R,Z)) \frac{\partial V}{\partial n} \frac{dl}{R} \quad \forall P \in \partial\Omega.$$

Because *V* satisfies Eqn. (27), the multigrid method can solve the problem in *O*(*N*) operations. The integral becomes singular for *P*(*R*,*Z*)=*P* and we have to compute the self-field contribution ψ_{SF} [27] using Eqn. (30),

$$(30) \qquad \psi_{SF}(P) = \frac{1}{2\pi}\left[\frac{\partial V}{\partial n}\right]_P \Delta l\left[\ln\frac{\Delta l}{16R_p}+1\right] \quad \forall P \in \partial\Omega \,,$$

where $\Delta l=\Delta R$ if P is on the horizontal parts (parallel to the R-axis) of $\partial\Omega$ or $\Delta l=\Delta Z$ if P is on the vertical parts (parallel to the Z-axis) of $\partial\Omega$. Away from the singular point, Eqn. (29) can be integrated numerically.

4.3. Plasma shape and position

Shape/position control [28] is a well known issue in free-boundary codes and the method presented here summarizes its most basic implementation. The required set of coils must generate a vertical field, for position control along the R-axis, a horizontal field, for position control along the Z-axis, and an elongation field, to prevent the collapse of the plasma shape at high β. Experimental operation is achieved by using a physical or a magnetic limiter, which determines the position of the LCFS Experiments can have a variety of limiter designs, but for numerical purposes, a point limiter P_{lim} will suffice, i.e.

$$(31) \qquad \psi_{edge}=\psi(P_{lim}).$$

The geometric position of this point may change at each computational step, but it must remain part of a self-consistent structure (i.e. -- the plasma may contact the limiter structure at different points, but the limiter itself must remain in a fixed position). The simplest limiter geometry is a vertical line, given by the equation $R=R_{lim}$. For this case, the geometric definition of the LCFS (and thus the global plasma position and shape), can be constrained by a set of geometric points

$$(32) \qquad S_{LCFS}=\{P_{lim}, P_{VF}, P_{EL}, P_{HF}\}.$$

The shape control algorithm must then solve for the current in each coil system such that the LCFS will pass through all points in the set S_{LCFS}. Given the plasma flux in each point of S_{LCFS} and the normalized flux created by each coil system, we may solve the following matrix equation,

$$(33) \qquad \begin{pmatrix} \psi_{plasma}(P_{VF})-\psi_{plasma}(P_{\lim}) \\ \psi_{plasma}(P_{EL})-\psi_{plasma}(P_{\lim}) \\ \psi_{plasma}(P_{HF})-\psi_{plasma}(P_{\lim}) \end{pmatrix} = -\begin{bmatrix} \overline{\psi}_{VF}(P_{VF})-\overline{\psi}_{VF}(P_{\lim}) & \overline{\psi}_{EL}(P_{VF})-\overline{\psi}_{EL}(P_{\lim}) & \overline{\psi}_{HF}(P_{VF})-\overline{\psi}_{HF}(P_{\lim}) \\ \overline{\psi}_{VF}(P_{EL})-\overline{\psi}_{VF}(P_{\lim}) & \overline{\psi}_{EL}(P_{EL})-\overline{\psi}_{EL}(P_{\lim}) & \overline{\psi}_{HF}(P_{EL})-\overline{\psi}_{HF}(P_{\lim}) \\ \overline{\psi}_{VF}(P_{HF})-\overline{\psi}_{VF}(P_{\lim}) & \overline{\psi}_{EL}(P_{HF})-\overline{\psi}_{EL}(P_{\lim}) & \overline{\psi}_{HF}(P_{HF})-\overline{\psi}_{HF}(P_{\lim}) \end{bmatrix} \begin{pmatrix} I_{VF} \\ I_{EL} \\ I_{HF} \end{pmatrix}.$$

The normalized coil fluxes were introduced in Eqn. (23). If the vertical, horizontal and elongation fields form a decoupled system, the matrix should be invertible, provided an appropriate choice is made for the points P_{lim}, P_{VF}, P_{EF} and P_{HF}. Ideally, they should be spread evenly around the LCFS. Because their geometric positions are typically unknown, it is usually necessary to define P_{VF}, P_{EF} and P_{HF} relative to P_{lim}. Using this procedure, a set of carefully chosen parameters can constrain the final plasma geometry to the desired shape and position (Figure 7). The defined constraints do not amount to a fixed boundary system. Rather, they provide a required measure of geometric stasis to avoid the numerical instability that otherwise plagues high β convergence. As an example, Figure 8 shows medium beta cases. Elongation is necessary to sustain a circular shape. Without elongation coils, the plasma has a “comet” shape, due to an X-point created on the high field side. The stronger vertical field necessary to compensate for the higher internal inductance of the plasma creates this X-point, which moves radially outward as beta increases. This

limits the Shafranov shift attainable in non-elongated cases. Thus extremely shifted configurations require elongation, even for circular shaped plasmas.

5. High beta equilibrium numerical procedure

High-β configurations have extreme Shafranov shifts, compressing the flux surfaces on the low field side of the plasma (Figure 1-b). This characteristic complicates high-β computations; steep gradients and strong ψ-J_ϕ coupling render calculations extremely sensitive to computational noise. The crucial factors for high-β equilibrium computations are:

- current density as a free function input,
- magnetic axis search,
- relaxation parameter,
- plasma position control,
- high resolution grid.

The following sections deal with the first four points. Position control is usually present in free boundary codes and only its necessity for high β convergence, even in the case of circular plasmas, will be discussed. The last point, high resolution, comes naturally from the multigrid method and will not be elaborated further. This is not really an issue for medium shift (< 70%) but it really becomes an essential parameter for higher magnetic axis shifts.

5.1. J_ϕ as a free function input

Solutions of the GSh equation are strongly dependent on the two input functions $p(\psi)$ and $F(\psi)$. If these two functions are known, the procedure presented in this work will find a unique flux distribution. However, many interesting plasma properties at high β are a direct consequence of the current density profile. To investigate these properties, it is desirable to specify the current profile directly, rather than the more traditional input functions $p(\psi)$ and $q(\psi)$ or $F(\psi)$.

The interesting point in choosing J_ϕ instead of p and F is that the shift of the magnetic axis can be controlled directly by the current density profile. To illustrate this point we look at the slab solution of the GSh equation

$$\frac{d}{dR}\left(\frac{1}{R}\frac{d\psi}{dR}\right) = \mu_0 J_\phi(R)\,. \tag{34}$$

Because J_ϕ is given as the input function, we can integrate directly Eqn. (34) and we get

$$\psi(R) = H(R) + \frac{A}{2}R^2 + B \text{ where } H(R) = \int R\left(\int \mu_0 J_\phi(u)du\right)dR\,. \tag{35}$$

The H function is completely defined by J_ϕ and the A and B constants can be found when ψ is given at the boundary of the domain, i.e. the LCFS. Thus we obtain

$$(36)\qquad A = 2\frac{H(R_{edge}^{HFS}) - H(R_{edge}^{LFS})}{\left(R_{edge}^{LFS}\right)^2 - \left(R_{edge}^{HFS}\right)^2} \text{ and } B = \left[\frac{H(R_{edge}^{HFS})}{\left(R_{edge}^{HFS}\right)^2} - \frac{H(R_{edge}^{LFS})}{\left(R_{edge}^{LFS}\right)^2}\right]\left[\frac{1}{\left(R_{edge}^{LFS}\right)^2} - \frac{1}{\left(R_{edge}^{HFS}\right)^2}\right]^{-1}.$$

The superscripts LFS (Low Field Side) and HFS (High Field Side) denote the values of R on the right and left side of the magnetic axis, as shown in Figure 2. Now the magnetic axis is located where $\mathrm{d}\psi/\mathrm{d}R=0$. Moreover R_{axis} comes from solving the following equation

$$(37)\qquad H'(R_{axis}) + AR_{axis} = 0\,.$$

The Shafranov shift is therefore directly defined by J_ϕ and the 1-D geometry,

$$(38)\qquad ShS = \frac{R_{edge}^{HFS} + R_{edge}^{LFS} - 2R_{axis}}{R_{edge}^{LFS} - R_{edge}^{HFS}}.$$

Although this 1-D property extends to the 2-D domain, analytic solutions do not exist. Highly shifted solutions can be easily defined using J_ϕ as input. Current density input guarantees tight control over the high β convergence.

As Eqn. (5) shows, a valid input function should span across all R values, i.e. from one edge of the plasma to the other while crossing the magnetic axis [20]. Without loss of generality, a convenient input function could be defined along a straight line going from one edge of the plasma to the other, parallel to the R-axis and crossing the magnetic axis. Because the exact physical positions of the plasma edges and magnetic axis are not known in advance, one may supply the evenly distributed current data that are denoted by the set

$$(39)\qquad \{J\}=\{J_1, \ldots, J_n\}.$$

The algorithm redistributes the current data set (39) from edge to edge, along the R-axis passing through the magnetic axis, as shown in Figure 9. Then spline approximation is applied so that data interpolation is as accurate as possible. From the "splined" current density distribution $J_{\phi\ spl}$, $dp/d\psi$ and $dF^2/d\psi$ can be computed. If we use the following convention to index the flux

$$(40)\qquad \psi_k = \psi_{edge} + (\psi_{axis} - \psi_{edge})\frac{k}{n} \text{ for } k\in\{0, .., n\},$$

we have

$$(41)\qquad \left.\frac{dp}{d\psi}\right|_k = -\frac{R_k^{HFS} J_{\phi_k}^{\ HFS} - R_k^{LFS} J_{\phi_k}^{\ LFS}}{R_k^{LFS^2} - R_k^{HFS^2}} \ \forall k\in\{0, .., n\text{-}1\}$$

and

$$(42)\qquad \left.\frac{dF^2}{d\psi}\right|_k = 2\mu_0 R_k^{LFS} R_k^{HFS} \frac{R_k^{LFS} J_{\phi_k}^{\ HFS} - R_k^{HFS} J_{\phi_k}^{\ LFS}}{R_k^{LFS^2} - R_k^{HFS^2}} \ \forall k\in\{0, .., n\text{-}1\}.$$

Here k indexes the values of the different functions in ψ_k. The 2D current distribution can then be found using Eqn. (5). The number n is the total number of samples taken from the edge of the plasma to the magnetic axis. This number can be the total number of points between the edge and the axis, or it can be smaller. Experience shows that using one third of the mesh resolution is excellent, providing that the distributions $dp/d\psi$ and $dF^2/d\psi$ are also approximated with splines. Figure 10 shows all the functions derived from the current density spline of Figure 9.

Equations (41) and (42) are not defined on the magnetic axis, i.e. $k=n$, and $dp/d\psi|_n$ and $dF^2/d\psi|_n$ cannot be computed by using this method. Because the position of the magnetic axis is always available, hence $R^{LFS}|_n$ and $R^{HFS}|_n$, it is possible to approximate geometrically the values of the derivatives there. The distributions

$$\{dp\}=\{dp/d\psi|_k,\ k=0,\ldots,n-1\} \tag{43}$$

and

$$\{dF^2\}=\{dF^2/d\psi|_k,\ k=0,\ldots,n-1\} \tag{44}$$

can be remapped onto the geometric space by using the distributions $\{R^{LFS}\}$ and $\{R^{HFS}\}$. The spline approximation of these distributions gives the values of $dp/d\psi|_n$ and $dF^2/d\psi|_n$ for $R_{axis}=R^{LFS}|_n=R^{HFS}|_n$. The information of the distributions R^{HFS} (Figure 10-a) and R^{LFS} (Figure 10-b) as functions of the flux is combined with the distributions of the derivatives of p (Figure 10-c) and F (Figure 10-d) to construct the spatial distribution of the derivatives. As an example, Figure 11 shows the interpolation of the computed $dp/d\psi|_n$. Then the splines of $dp/d\psi$ and $dF^2/d\psi$ can be generated from the edge to the magnetic axis. Once $(dp/d\psi)_{spl}$ and $(dF^2/d\psi)_{spl}$ are known the algorithm can compute the current density distribution in the whole plasma by using the flux distribution from the previous step.

5.2. Plasma position control

Plasma shape and position are manipulated using a set of coil systems as discussed earlier. To prevent plasma drift, the code must anchor the plasma inside the computational grid, not supposing any up-down symmetry. Position drifts can cause numerical oscillations in high β cases, and thus must be minimized. We have found that convergence at high β cannot occur without minimal shape feedback. This is a necessary stabilizing element in high β convergence, the actual shape control of the plasma just being a "side effect" of the numerical procedure. If the boundary is not controlled properly, the flux oscillations inside the plasma propagate to the LCFS. The mode structure of these oscillations is an up-down motion of the plasma core. On the other hand, if the LCFS is properly constrained, the internal oscillations do not deform the LCFS. This is a necessary condition for high beta convergence. Even for circular plasmas, where up-down numerical stability is not an issue at low beta, plasma position control is always mandatory for convergence at high β. The technique presented in the preceding section works well for any Shafranov shift.

5.3. Magnetic axis search

To match $dp/d\psi$ and $dF/d\psi$ with the input distributions, it is necessary to calculate the position of the magnetic axis. Finding the maximum value of the flux distribution cannot yield axis position during high β convergence. Numerical noise will cause the maximum to wander from one iteration to the next, as Figure 12-a shows. This generates a mismatch between actual and input function profiles. We have observed that this problem is occasional during the convergence, but creates a bottleneck that can prevent convergence. A method less vulnerable to numerical noise must be applied. Instead of finding the global extremum of ψ, we would rather calculate a set of local extrema and average over them. To find the extrema, the usual

approach is to find the zeros of the gradient of the function under scrutiny (i.e. $|\nabla\psi|$). Of course, $|B_P|$ can be used instead. To decrease the computational cost, we restrict the search to an area A_x

$$(45) \qquad A_x = \left\{ P \in (R,Z) / \frac{\psi(P) - \psi_{edge}}{\psi_{axis} - \psi_{edge}} > x \right\} \text{ with } x \in]0,1[\,.$$

If $x \sim 0$, the search is across the whole plasma, which would not be efficient. If $x \sim 1$, the search is restricted to a region very close to the maximum flux, and the averaging strategy holds no benefit. Experience suggests that $x = 0.9$ gives very good results.

While standard minimizing methods calculate a single minimum, we wish to find several local minima, so a more stable average axis position can be computed. For this purpose, we have devised an original method, called the "evaporating lake" model (Figure 12-b). To explain the method by way of metaphor, one may imagine the two-dimensional $|B_P|$ distribution as the bottom of a lake at a certain level h_0. The evaporation of the water will uncover the highest regions of the lake bed, leaving the deepest parts submerged. As the level falls to h_k, the location of the local minima will be revealed by pools that form in the deepest structures of the bed. If the procedure is halted at step n, while a small amount of water still remains, small pools can be observed around each minimum. By taking the center of mass of the total water area A_n, we obtain a weighted average of the minima. Practically, a minimum area A_{final} is defined. At each iteration step k, the level is decreased to

$$(46) \qquad h_k = \frac{A_{k-1}}{A_{k-2}} h_{k-1} \,.$$

and the area of the 2D distribution is computed

$$(47) \qquad A_k = \left\{ P \in A_{k-1} \subset A_{k-2} \subset ..... \subset A_x / |\, B_P(P) | < h_k \right\}.$$

The algorithm will stop when $A_k < A_{final}$. In general, A_{final} is defined relative to A_x, such as

$$(48) \qquad A_{final} = \varepsilon . A_x \,,\ \varepsilon \in]0,1[.$$

The size of A_{final} depends on the high β flux distribution. In general a value around 5% of A_x gives good results. The effectiveness of this method results from two major points. First, the search is restricted to the neighborhood of the actual magnetic axis. Second, the global minimum of the function is known (i.e. 0). The evaporating lake method may seem unnecessarily complex for the calculation of a value that is not used directly in the calculation of the solution. However, poor results in the magnetic axis search lead to poor matching of the input function. If the input function is not strictly respected, the resulting non-physical J_ϕ distribution will prevent the convergence of high-β cases.

5.4. Improved simulated annealing

Usually algorithms of the Picard or Marder-Weitzner types [28] converge extremely slowly due to the lack of an adaptive relaxation scheme. The relaxation parameter proposed in this paper is an essential control for high β convergence. The principle derives from the simulated annealing technique [29, 30], where the convergence is rapid during the initial computational steps and slows down as we approach the final

solution. The first step is to define the relaxation parameter η_k at each iteration k. All quantities follow from the flux, so it seems reasonable to use

$$\psi_k = (1-\eta_k)\psi_{k-1} + \eta_k \psi_{kk}\ ,\ \eta_k \in]0,1], \tag{49}$$

where ψ_{kk} is the flux computed at iteration step k and ψ_k is the flux used to compute $J^{k+1}{}_{\phi}$. We also define the computational error

$$\varepsilon_k = \max_{P\in\Omega} |\psi_{kk}(P) - \psi_{k-1}(P)|\ , \tag{50}$$

similar to Eqn. (10), independent of the relaxation parameter. The convergence is obtained when $\varepsilon_k < \varepsilon_{convergence}$ and an approximate solution to the GSh equation has been found. To obtain a robust and rapid algorithm that converges at high β, we define the following simple rules for the relaxation parameter η_k:

$$\frac{\varepsilon_k}{\varepsilon_{k-1}} > \alpha_t \Rightarrow \begin{cases} \eta_k = \alpha_d \eta_{k-1} \\ \psi_k \equiv \psi_{k-2} \end{cases}, \alpha_d \in]0,1[\ , \tag{51}$$

$$k \bmod(n_{relax}) = 0 \Rightarrow \eta_k = \alpha_r \eta_{k-1}, \alpha_r \in [1,+\infty[\ , \tag{52}$$

$$0 < \eta_{\min} < \eta_k < \eta_{\max} \le 1\ . \tag{53}$$

where:

- α_t is the error threshold ratio when the convergence slows down;
- α_d is the damping factor of the relaxation parameter;
- n_{relax} is the number of iterations completed before adjusting the relaxation parameter;
- α_r is the relaxation parameter gain.

When the ratio $\varepsilon_k/\varepsilon_{k-1}$ is above a certain level $\alpha_t>1$, the error increases. At this point, the algorithm diverges. To remedy this problem, we have to slow down the convergence, following Eqn. (51). Furthermore the newly computed flux ψ_{kk} is replaced by the flux computed two iterations before, ψ_{k-2}. By using this technique, we prevent corruption of the convergence by introducing "divergent solutions". This is a major difference with the simulated annealing technique which does not differentiate between satisfactory and unacceptable solutions.

Furthermore it is also preferable to dynamically adjust the relaxation parameter so an optimum value can be found. This compromise between speed and stability makes the algorithm more efficient. Eqn. (52) modifies the relaxation parameter every n_{relax} iterations, increasing it by a factor α_r. Finally Eqn. (53) brackets the relaxation parameter, preventing extreme numerical shifts. A practical application regarding these parameters is presented in Section 7.3.

After defining an adequate input function, adjusting the plasma shape, searching for the magnetic axis and relaxing the convergence rate, the overall computational algorithm can now be described.

6. Computational algorithm

The numerical algorithm which computes the flux regroups many of the items discussed earlier. Figure 13 illustrates the several steps followed by the procedure to solve for the flux distribution using the FMG method. After computing the plasma boundary flux from Eqn. (29), the multigrid method gives the whole

plasma flux distribution, solving Eqn. (17). The shape feedback is then applied by inverting Eqn. (33), preventing spatial drifts of the plasma in high β cases.

The complete algorithm presented in Figure 14 combines the flux computation procedure including the high β convergence core: i.e. the magnetic axis search, the relaxation parameter computation and the relaxation parameter adjustment. The search for the magnetic axis uses Eqn. (47) and allows input function matching. The relaxation parameter keeps the algorithm stable by reducing flux changes in Eqn. (49). It relies on dynamical corrections governed by Eqn. (51), (52) and (53) to optimize convergence speed. Finally, the input function introduced in Section 5.1 defines the current density distribution, by matching $\mathrm{d}p/\mathrm{d}\psi$ and $\mathrm{d}F^2/\mathrm{d}\psi$.

An obvious improvement can accelerate further the convergence. It is directly inspired by the FMG algorithm. At the beginning of the computation, the solution does not require a high degree of precision. Hence a coarser grid can be used. As the solution improves, the resolution of the grid has to increase and a finer grid should replace the coarser one. Figure 14 demonstrates the grid switch where m becomes m+1 when

$$\varepsilon_k < \varepsilon_{grid\ switch}\text{, with } \varepsilon_{grid\ switch} > \varepsilon_{convergence}. \qquad (54)$$

The grid change should adhere to Eqn. (18) and (19). Usually $\varepsilon_{grid\ switch}=10.\varepsilon_{convergence}$ is quite successful.

Despite the integration of all these elements around the multigrid method, the time to compute one step stays proportional to the total number of points N. Figure 15 shows the time dependence of grid density d, proportional to the total number of grid points N for a fixed domain Ω. Figure 15-a illustrates the results for a 6-grid mesh on a 1 GHz Pentium IV with a density ranging from 2 to 110 points/cm^2. The linear behavior is striking, with a computational time step $k \rightarrow k$+1 around 2d seconds. Consequently the resolution of the finest grid corresponds to one computational point per millimeter of geometrical space. Figure 15-b shows more modest densities, from 2 to 30 points/cm^2, with 5, 6, 7 or 8 grids on a 2004 2 GHz PC. The dependence is still linear with the number of points. In this particular case, a computational iteration fortuitously lasts d seconds.

After presenting the problem, the different equations and the general algorithm based on the full multigrid method, the rest of the paper now focuses on validating the code, especially for high β configurations.

7. Code validation

To validate the numerical procedure, a code has been written. CUBE (Code for Unity Beta Equilibria) is a simple computer program that takes into account many factors directly linked to experimental results. The code can only accept geometries without magnetic materials. In this section, we will use the particular geometry of the high aspect ratio Electric Tokamak (ET) [31] at UCLA (R = 5 m, a = 1 m, κ < 1.5, B_ϕ = 0.25 T). The computational domain spans from R = 4 m to R = 6 m and Z = -1 m to Z = 1 m. To evaluate code results, several errors have to be defined and a systematic scan of the parameter space has to be performed.

7.1. Error definitions

To properly evaluate code results we look at three relative errors. The first one is the convergence error motivated by Eqn. (50) which monitors directly the convergence of the flux distribution;

$$\varepsilon_k = \frac{\max_{P\in plasma} \| \psi_{kk}(P) - \psi_{k-1}(P) \|}{2\left| \max_{P\in plasma} \psi_{kk}(P) + \min_{P\in plasma} \psi_{kk}(P) \right|}. \tag{55}$$

This is a numerical error. The magnetic error ξ_k,

$$\xi_k = \frac{\max_{P\in plasma} \| \vec{\nabla} \times \vec{B}(P) - \vec{J}(P) \|}{2\left| \max_{P\in plasma} \| \vec{J}(P) \| + \min_{P\in plasma} \| \vec{J}(P) \| \right|}, \tag{56}$$

evaluates the geometrical deviation between the curl of the magnetic field and the current density, directly from the steady-state Maxwell's equations. Finally, the force balance or $J\times B$ error ζ_k monitors the accuracy of the solution of the GSh equation by evaluating the difference between Lorentz and pressure gradient forces,

$$\zeta_k = \frac{\max_{P\in plasma} \| \vec{\nabla} p(P) - \vec{J} \times \vec{B}(P) \|}{2\left| \max_{P\in plasma} \| \vec{\nabla} p(P) \| + \min_{P\in plasma} \| \vec{\nabla} p(P) \| \right|}. \tag{57}$$

7.2. Error evolution

The computational error is the dominant factor influencing the quality of the final solution. To assess code results we have to consider the numerical, geometrical and equilibrium errors discussed previously, performing such an evaluation in the worst computational conditions. In a spirit of fairness we look only at the maximum value of each error. Stability codes are globally sensitive to local inaccuracies; thus only local errors should be considered. No averages or integrations are performed to artificially reduce these values. In the majority of cases, the maxima of the errors are found only at a few grid points.

This is true at low β where the resolution is not an issue. As the Shafranov shift increases other factors have to be taken into account. If we examine a 75% shift configuration with a grid density of 5 points/cm^2, Figure 16-a demonstrates the slowdown of the convergence as we approach the solution. While the magnetic and force balance errors decrease quickly when ε_k>1%, a saturation effect clearly appears below this threshold. Even if the magnetic error improves quite rapidly as ε_k diminishes, the force balance error stays constant. Hence the quality of the solution is resolution-limited. High resolution is needed to compute high β equilibria.

After the influence of the computational error, we now scrutinize the impact of the grid density on the quality of the solution. Figure 16-b shows the improvement of the errors when finer grids are used. Except at low density where, due to extra iterations, the 75% shift cannot be properly resolved, the density does not influence the computational error. Therefore the solution requires the same number of iterations to converge. The total computational time stays proportional to N for a fixed computational error.

Figure 16-c shows the influence of the Shafranov shift on the errors and the number of iterations. Above a shift of 65%, the test grid density of 5 points/cm^2 is not sufficient to properly resolve the squeeze of the flux surfaces and the magnetic and force balance errors climb. Despite this increase, the dependence between the computational error $\varepsilon_{convergence}$ and the number of iterations is very strong. Above a shift of 77%, the resolution is quite poor and the number of iterations increases due to the feedback on the relaxation parameter. To prevent numerical oscillations due to the low-resolution grid, the relaxation parameter has to be drastically reduced and the number of iterations soars. It is now the computational error which limits the quality of the convergence.

Finally Figure 16-d illustrates the typical convergence for a highly Shafranov-shifted equilibrium. The error diminishes regularly and there is no numerical oscillation. As a concluding remark, the grid density scan in Figure 16-b goes up to 110 points/cm^2. This translates into a resolution of 1 point per mm. This is an exceptional result that is especially useful for inputs to gyro-kinetic codes [32, 33] which require a precision comparable to the Larmor radius of the ions, i.e. 2-3mm. This level of accuracy is practical due to the $O(N)$ time dependence of the code.

7.3. Relaxation parameter dependence

The relaxation parameter plays a great part in regulating the convergence rate of the algorithm at high β. The adjustment of the different parameters presented in Eqn. (51) to (53) is of paramount importance. The algorithm can slow down tremendously if they are not carefully picked. Unfortunately they depend upon geometry and Shafranov shift and no general rule can apply to find the best choice. While only trial and error can educate the savvy user in finding the best values for the problems at hand, we present here guiding rules that should be observed to maintain rapid convergence while keeping numerical oscillations in check.

The initial value η_{init} introduced previously is critical because the feedback on the relaxation parameter should be slow. Thus if η_{init} is too high the algorithm will spend time reducing its value until stable operation is achieved. The computational time will be used to find an optimum to the relaxation parameter η_k instead of calculating the solution. A good starting point should be

$$\eta_{init} = \frac{1 - ShS}{1 + 3ShS}, \; ShS \in [0,1[\,, \tag{58}$$

The quantity α_t is the threshold factor where the computational error deviation triggers a reduction of the relaxation parameter and a set back in the flux distribution. An optimum error increase of 1 to 5% from one computational step to the next should be observed. If set too low, the relaxation parameter will rapidly reach its minimum value, due to numerical noise in the error. If set too high, numerical oscillations may go undetected, preventing the convergence.

The quantity α_d is the damping factor of the relaxation parameter and slows down the speed of the algorithm as it gets closer to the solution, preventing numerical oscillations. Once again its value depends upon the equilibrium geometry but a fast reduction of the relaxation parameter can freeze the algorithm. An

acceptable value should stay around 0.9. To this damping factor, we associate a relaxation gain α_r. Its role is to increase the relaxation parameter when computational oscillations disappear. This way the algorithm adjusts its own convergence rate in an optimal manner. To moderate this behavior, the relaxation parameter increase takes place only every n_{relax} computational steps. Figure 17 demonstrates the actual behavior of the relaxation parameter for a Shafranov shift of 83%, a threshold factor α_t=1.01 and a grid density of 22 points/cm^2.

Figure 17-a shows the proper behavior of the feedback control on the relaxation parameter. The solution is found after 202 iterations. The convergence rate stays constant overall despite the swings of the relaxation parameter. The search for the optimum spans the first 100 iterations. After this initial phase, the control procedure keeps the oscillations of the relaxation parameter around the optimum found. Perhaps incidentally, the optimum seems to approximately coincide with the initial value of the relaxation parameter.

Figure 17-b demonstrates the behavior of the code when the oscillations in the relaxation parameter damp too fast from a value that is slightly higher than its optimum. The convergence speed slows down in the second phase of iterations, finding the solution after 220 steps.

In the case of a slow relaxation process, the optimum search lasts too long and the algorithm converges before the optimum is found. The case presented in Figure 17-c may lead to an erroneous conclusion. Despite the low number of iterations before the solution is found, the feedback system control is completely disabled. This flaw is compelling if the initial value of the relaxation parameter is too low, violating Eqn. (58). Figure 17-d illustrates this behavior where the convergence is reached after 375 steps.

Therefore it is essential to keep the feedback system flexible, i.e. with a good dynamical range. Despite the influence of the geometry on the relaxation parameter feedback loop, the values presented here should always be used as a starting point to the optimization of the problem at hand.

After studying error evaluation under the most unfavorable conditions, satisfactory answers have been obtained regarding code robustness and precision. With Shafranov shifts above 80% and a precision ranging from 1 point per cm to 1 point per mm, the average error level can stay below 0.01% and the maximum error value below 0.1%. This validation enables us to now examine a physical case, namely unity β configurations.

8. Applications to highly shifted and unity beta equilibria

If we focus on the circular equilibria suggested by Cowley [34], we find highly shifted configurations with a strong diamagnetic behavior. Figure 18 shows the principal results for a total plasma current of 168 kA and a β peak of 100%, with a global β near 40%. The most important result is the magnetic well dug in the toroidal field by the diamagnetic poloidal currents circulating in the plasma. Figure 19-a illustrates the high β "squeeze" with a 3D view of the current density completely pushed outward. This perspective view emphasizes the difficulty of obtaining such currents in real devices, compared to the usual "bell shape" current of Figure 1-a. Figure 19-b shows the force balance error for the same computation. The maximum

force balance error is close to 0.2 %. This suggests a higher grid density should be used. Nevertheless this level of error is quite acceptable for stability codes. Figure 20 shows an extreme magnetic axis shift of more than 90% of the minor plasma radius, with a β peak of 70% and a global β close to 30%. A finer grid was used and the maximum force balance error is below 0.05% for this case. The Shafranov shift clearly compresses the surfaces on the low field side, in a region called the boundary layer. It is interesting to note that the flux depends only upon R in the "core" region of the plasma, as theory predicts [34]. To the authors' knowledge such results at extreme plasma β have not been published before except in their previous work [8]. Unfortunately, the contour dynamics method used in the prior publication was not appropriate for interfacing with stability codes due to the sparse mesh. CUBE fully accomplishes this goal, associating a high resolution interface with fast computational steps.

9. Conclusion

High β configurations have remained quite elusive when it comes to actual experiments. For many years it was also the case for numerical codes. The algorithm presented here is a significant step toward fast and reliable codes which can handle indifferently low or high β free boundary equilibria. After introducing the physical problem, we quickly presented the multigrid method as a rather efficient tool for solving elliptic equations on fine grids. We offer a detailed description of the algorithm actually used to solve the GSh equation, taking into account plasma and external conductors contributions and input functions. The numerical procedure was specifically tailored to handle extreme flux and current gradients, which tend to create numerical oscillations in standard MHD equilibrium codes. By using tight shape control, an accurate search of the magnetic axis, and a dynamically adjusted relaxation parameter, the computational algorithm successfully obtained high β configurations with a large outward radial shift of the magnetic axis. The numerical validation demonstrated the accuracy of the code and the influence of the different parameters on the convergence rate and precision. Then a direct application to a unity β case was presented as an example of code capabilities. To the authors' knowledge, other codes have yet to reach such extreme Shafranov shifts as obtained here, even when the multigrid method is used [35]. Furthermore, due to highly resolved pressure, current and flux distributions, CUBE makes possible precise studies of these equilibria with magnetohydrodynamic stability codes, which evaluate the macroscopic viability of these configurations, or gyrokinetic codes, which follow particle orbits and help to understand microscopic plasma behavior. The new opportunities that this algorithm delivers should be quite valuable for the development of viable approaches to high β plasmas.

Acknowledgements

One of the authors (P.-A. G) would like to thank L. Guazzotto for running the FLOW code to compare the results with CUBE in a high β mode. The authors are also grateful to S. C. Cowley for his help on high β. This work is partially supported by USDOE at UCLA through grants Nos. DE-FG02-04ER 54737 and DE-FG02-04ER54740.

1 F. Chen, Introduction to plasma physics and controlled fusion, (2nd Ed., Plenum Press, New York, 1984).
2 J. DeLucia, S. C. Jardin, A. M. M. Todd, Journ. Comp. Phys. **37**, p. 183 (1980).
3 L. L. Lao, S. P. Hirshman, R. M. Wieland, Phys. Fluids **24**, p. 1431 (1981).
4 L. E. Zakharov, A. Pletzer, Phys. Plasmas **6**, p. 4693 (1999).
5 R. Gruber, R. Iacono, F. Troyon, Journ. Comp. Phys. **73**, p. 168 (1987).
6 J. Blum, J. L. Foll, B. Thooris, Comp. Phys. Commun. **24**, p. 235 (1981).
7 J. D. Callen, R. A. Dory, Phys. Fluids **15**, 1523 (1972).
8 P.-A. Gourdain, J.-N. Leboeuf, Phys. Plasmas **11**, p. 4372 (2004).
9 M. D. Kruskal, M. Schwarzchild, Some instabilities of a completely ionized plasma, Proc. R. Soc. London, Ser. A, 223, 348 (1954).
10 V. D. Shafranov, Techn. Phys. **15** (2), 175 (1970)
11 M. W. Kissick, J.-N. Leboeuf, S. E. Kruger, Phys. Plasmas **10**, p. 1060 (2003).
12 L. Guazzotto *et al*., Phys. Plasmas **11**, p. 604 (2004).
13 N. J. Zabusky, M. H. Hughes, K.V. Roberts, Journ. Comp. Phys. **135**, p. 220 (1997).
14 R. C. Grimm, J. M. Greene, and J. L. Johnson, Methods in Computational Physics (Academic Press, New York, 1975), Vol. 16, p. 273.
15 A.H. Glasser, The direct criterion of Newcomb (DCON) for the stability of an axisymmetric toroidal plasma, Los Alamos Report LA-UR-95-528 (1997).
16 P. Wesseling, An Introduction to Multigrid Methods, (Wiley & Sons, New York, 1992).
17 A. Brandt, Mathematics of Computation **31,** 333 (1977).
18 M. W. Phillips *et al*., Phys. Plasmas **3**, 1673 (1996).
19 V. D. Shafranov, Reviews of Plasma Physics, (Consultants Bureau, New York, 1966), Vol. **2**, pp. 103-151.
20 J. P. Freidberg, Ideal Magnetohydrodynamics, (Plenum Press, New York, 1987)
21 R. W. Hockney, J. W. Eastwood, Computer Simulation Using Particles (Hilger, Bristol, 1988), pp. 175-181.
22 P. Wesseling, An Introduction to Multigrid Methods, (Wiley & Sons, New York, 1992).
23 U. Trottenberg, C. W. Oosterlee, A. Schüller, Multigrid (Academic Press, New York, 2001).
24 J. D. Jackson, Classical Electrodynamics (1rst Ed., Wiley & Sons, New York, 1962) pp. 141-143.
25 C Hastings Jr., Approximation for Digital Computers (Princeton Univ. Press, Princeton, N. J. 1955).
26 M. Abramowitz, I. A. Stegun, Handbook of Mathematical Functions, (Dover Publications, New York 1972), p 591.
27 K. M. Ling, S. C. Jardin, Journ. of Comp. Phys. **58**, p. 300 (1985).
28 J. Blum, Numerical Simulation and Optimal Control in Plasma Physics (Wiley & Sons, New York, 1989).
29 S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Science **220**, p. 671 (1983).
30 S. Kirkpatrick, Journ. Stat. Phys. **34**, p. 975 (1984).
31 R. J. Taylor, J.-L. Gauvreau, M. Gilmore, P.-A. Gourdain, D. J. LaFonteese, L. W. Schmitz, Nucl. Fus. **42**, p. 46 (2002).
32 A. M. Dimits, G. Bateman, M. A. Beer, B. I. Cohen, W. Dorland, G. W. Hammett, C. Kim, J. E. Kinsey, M. Kotschenreuther, A. H. Kritz, L. L. Lao, J. Mandrekas, W. M. Nevins, S . E. Parker, A. J. Redd, D. E. Shumaker, R. Sydora, and J. Weiland, Phys. Plasmas **7**, p. 969 (2000).
33 J. Candy and R. E. Waltz, Journ. Comp. Phys. **186**, Issue 2, p. 545 (2003).
34 S. C. Cowley, P. K. Kaw, R. S. Kelly, R. M. Kulsrud, Phys. Fluid B **3**, p. 2066 (1991).
35 Configurations with a Shafranov shift of 60% have been reached, L. Guazzotto, private communication (2005).

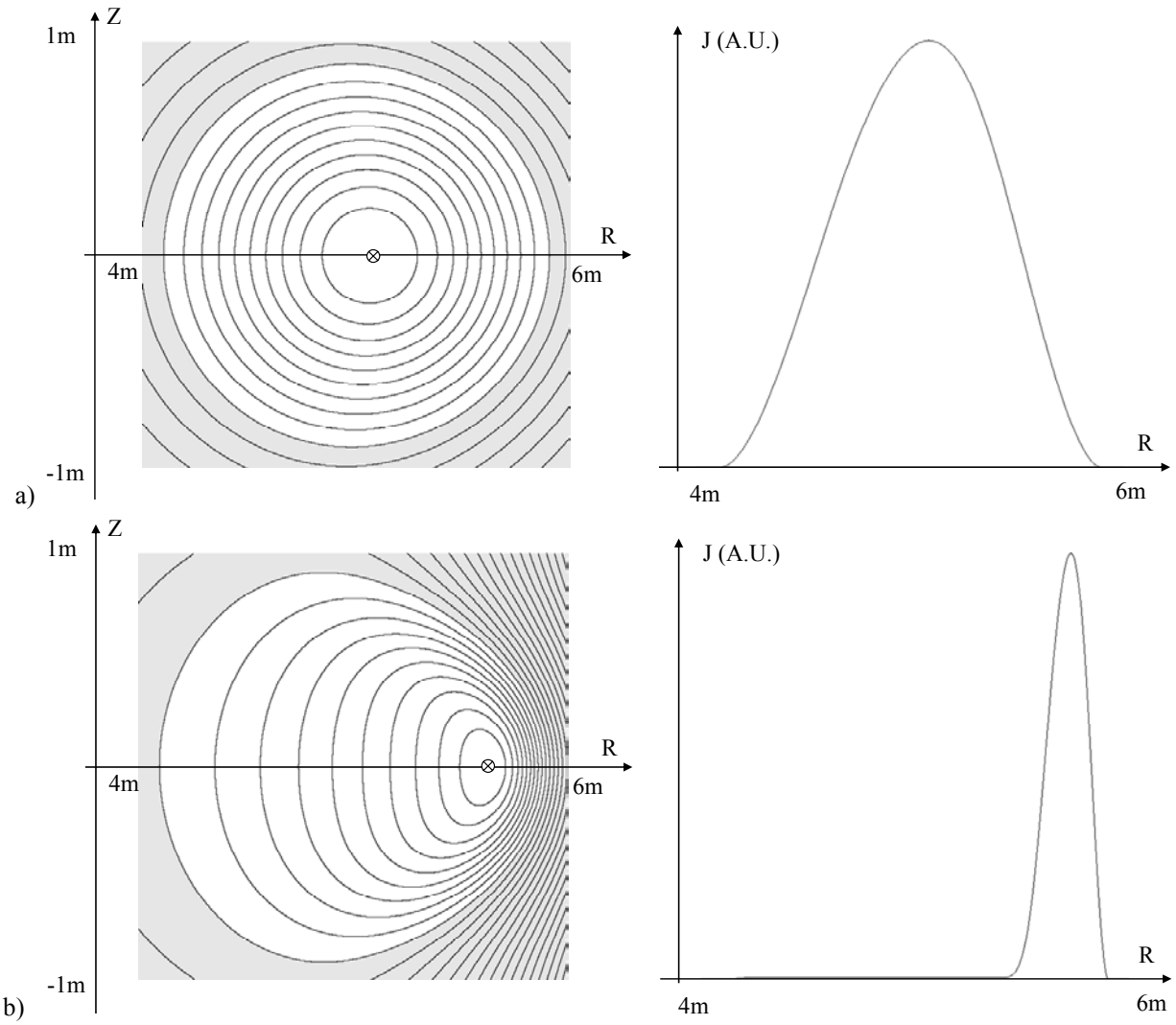


Figure 1 : a) Typical plasma configuration (radially outward Shafranov shift of 7% of the minor radius) and b) high β (shift of 70%) configuration. The flux surfaces are on the left and the toroidal plasma current density on the right. The extreme shift of the magnetic axis (⊗) compresses the flux surfaces and renders high β computations difficult and numerically unstable.

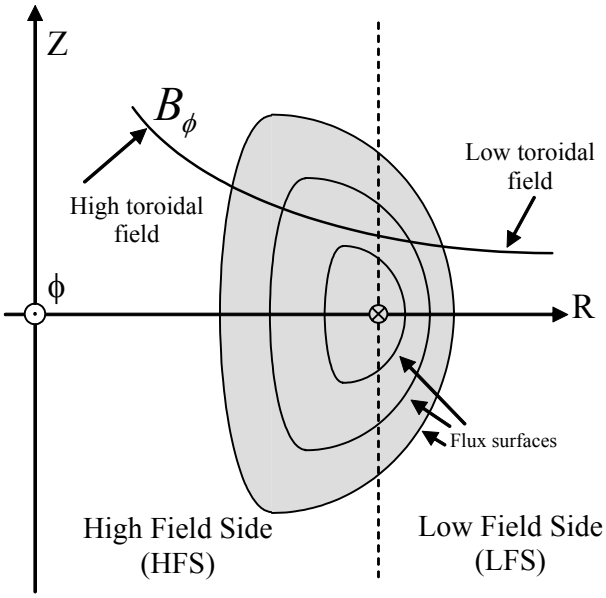


Figure 2 : Cylindrical coordinate system (R, Z, ϕ), toroidal field and plasma flux surfaces with magnetic axis (⊗).

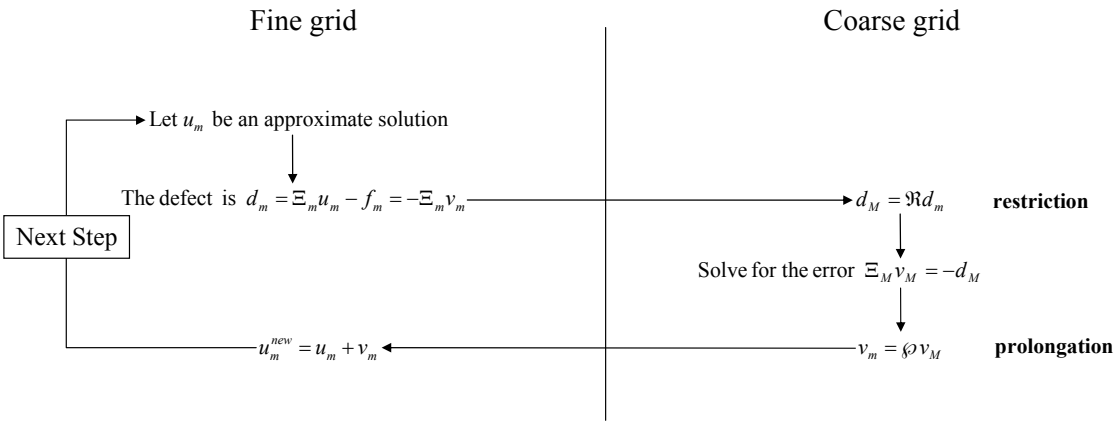


Figure 3 : The two-grid principle

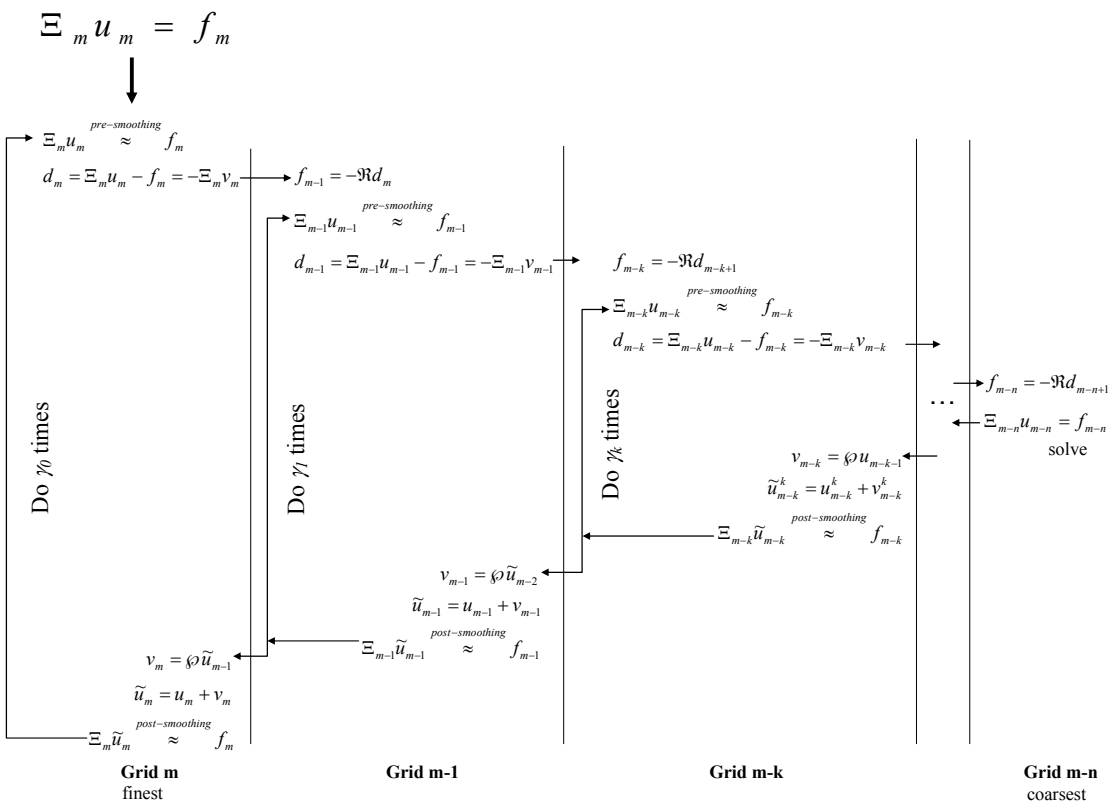


Figure 4 : Full multigrid algorithm

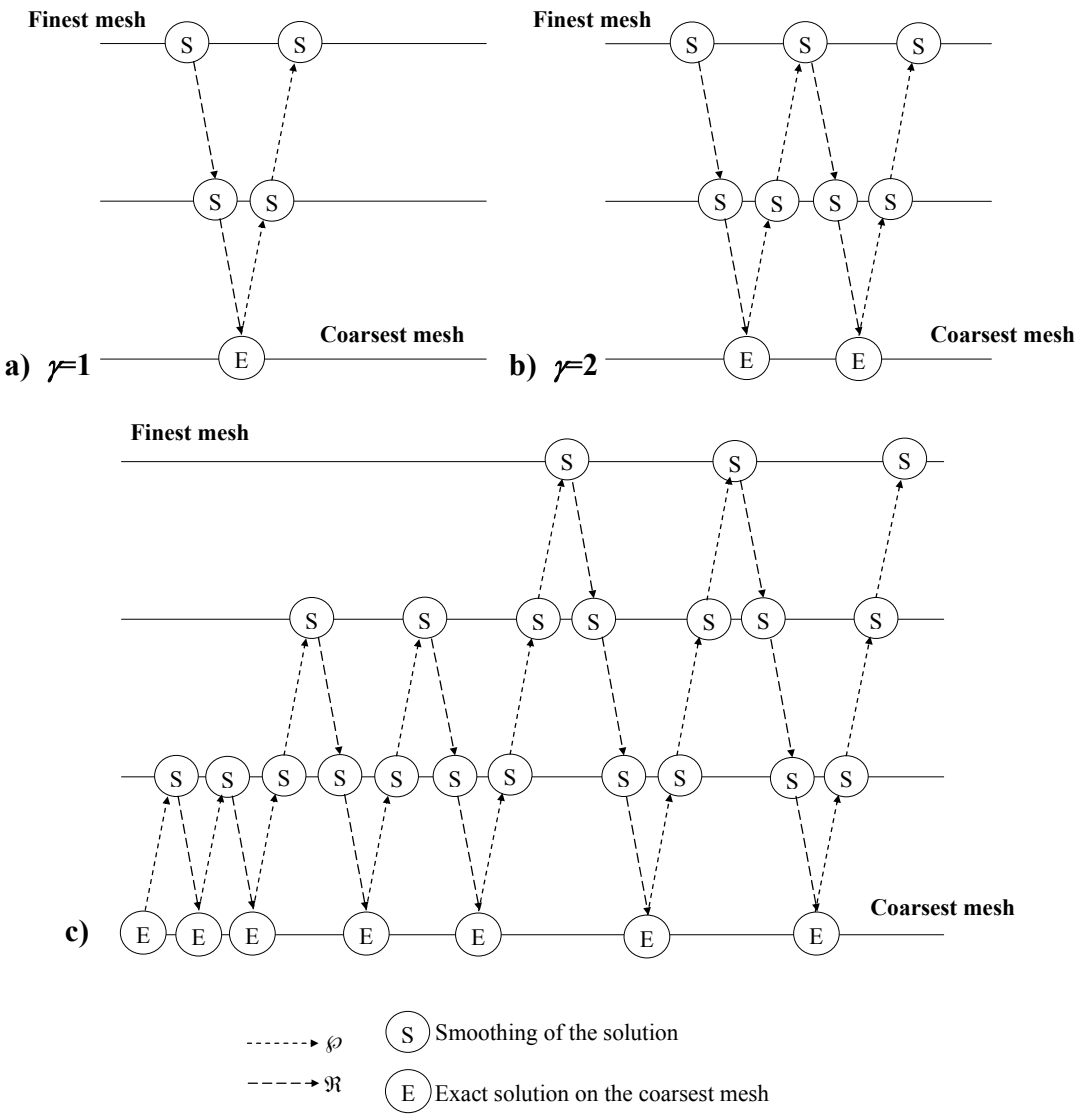


Figure 5 : Three-grid method with a) V-cycle (γ=1) or b) W-cycle (γ=2) and c) the full multigrid method with 4 grids and a schedule of 2 (W-cycles)

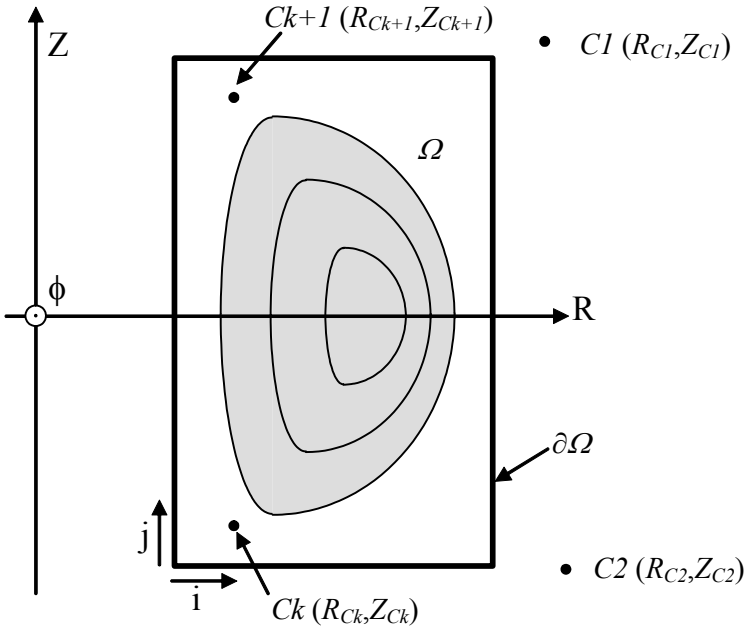


Figure 6 : Typical mesh showing the meshed domain Ω and its boundary ∂Ω, the plasma, and the set of equilibrium and shaping coils (inside or outside ∂Ω).

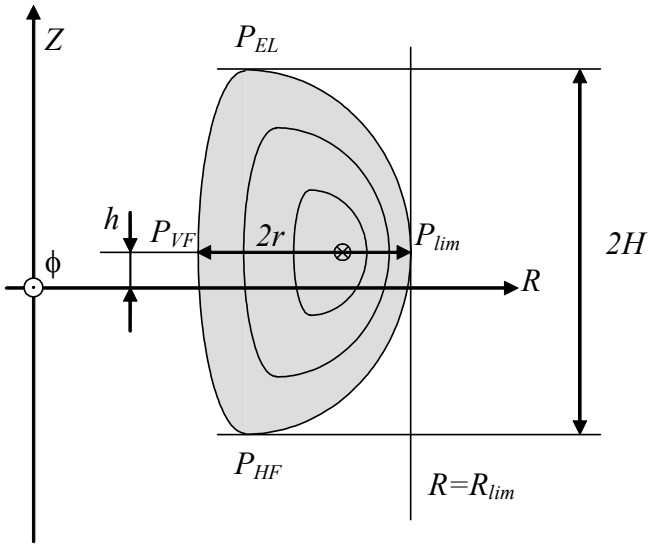


Figure 7 : Geometrical parameters that define P_{lim}, P_{VF}, P_{EF} and P_{HF} to compute I_{VF}, I_{EL} and I_{HF}

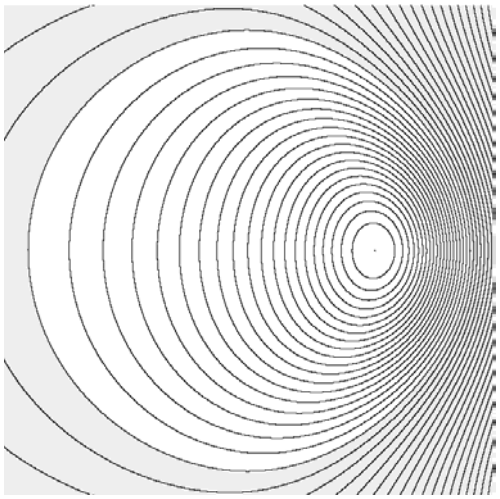
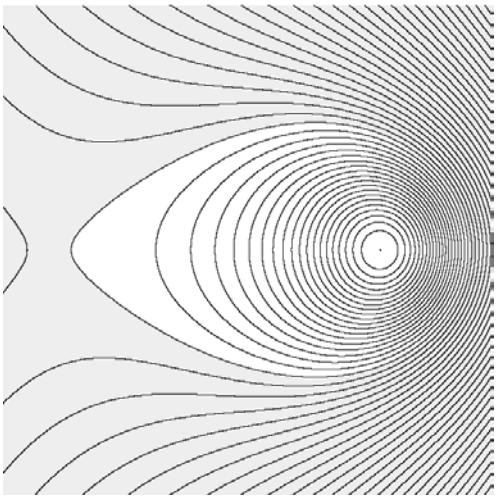
a)  b) 

Figure 8 : Medium beta plasma a) with and b) without elongation using the geometry of Figure 1. The existence of an X-point on the high field side due to the stronger vertical field limits the plasma beta for non elongated shapes. Elongation is mandatory to obtain circular shapes.

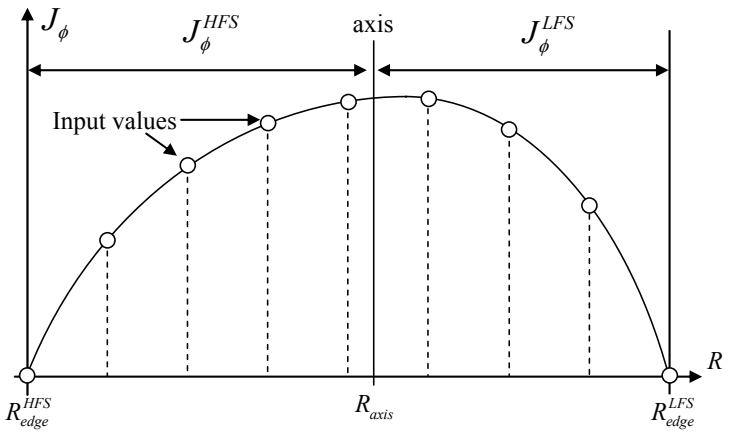


Figure 9 : Current density spline with its evenly distributed input values (o); the total span on the R-axis and the axis position are computed automatically by the algorithm.

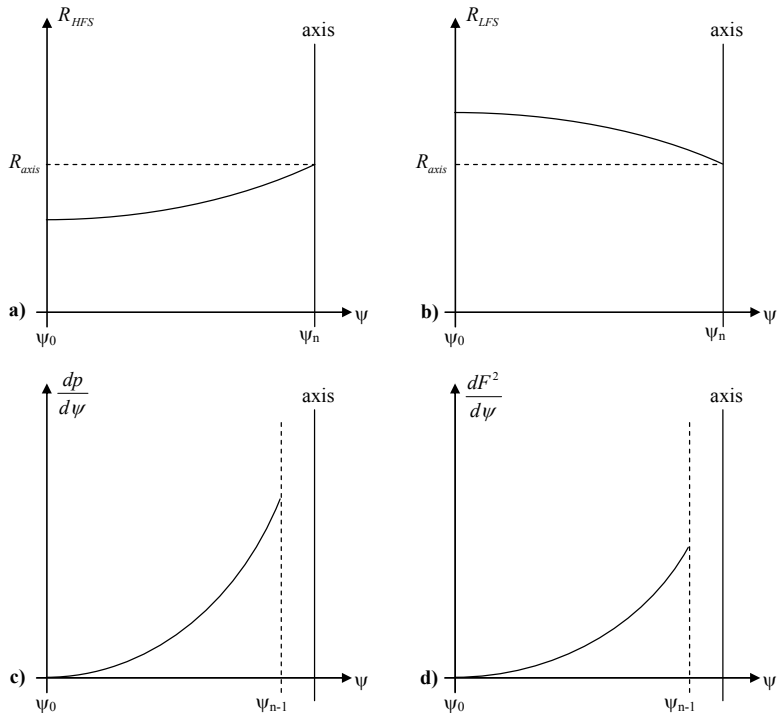


Figure 10 : R_{HFS}, R_{LFS}, $dp/d\psi$ and $dF^2/d\psi$ splines as functions of ψ. The derivatives cannot be computed on the axis

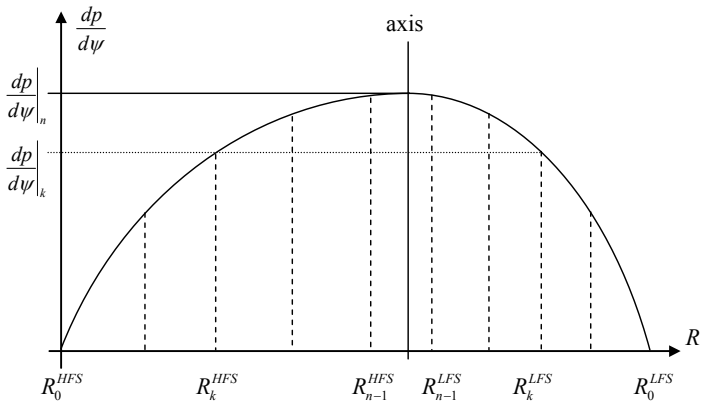


Figure 11 : Pressure derivative spline approximation for $dp/d\psi|_n$ computation

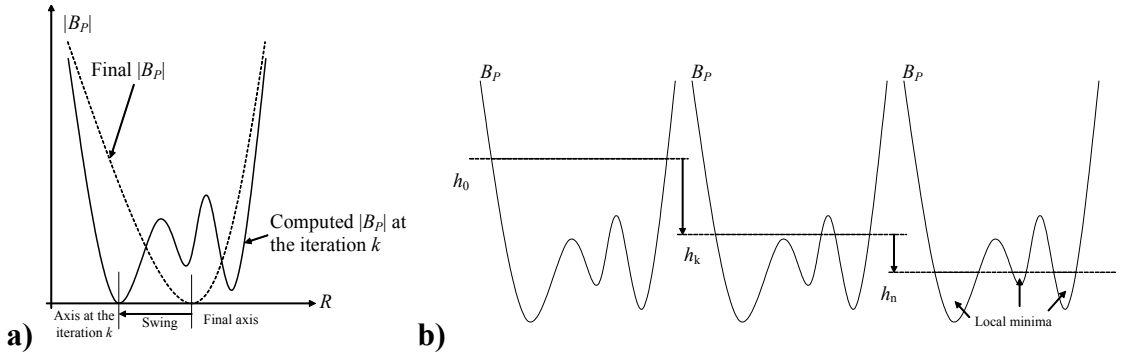


Figure 12 : a) axis swing due to the numerical noise of the flux at high β based on a simple minimum $|B_P|$ search and b) "evaporating lake" model for local minima search.

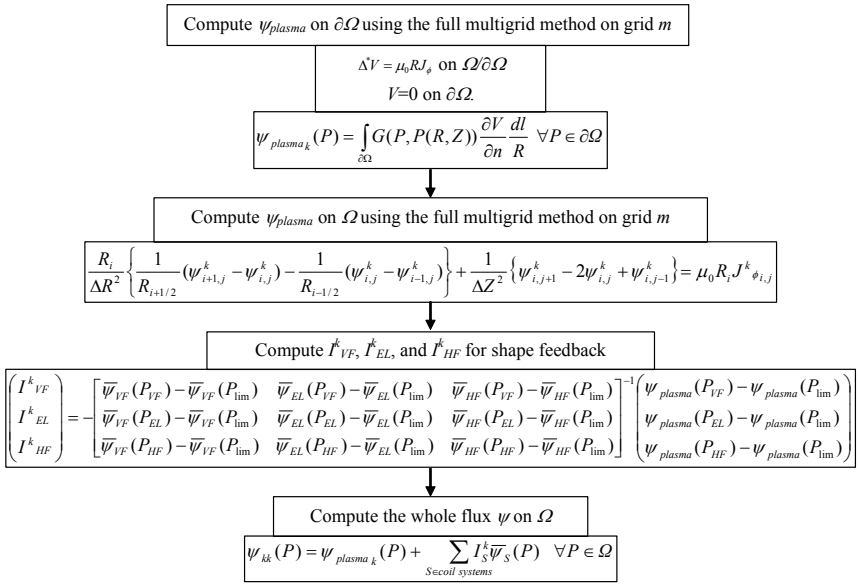


Figure 13 : Flux computation algorithm on grid *m*

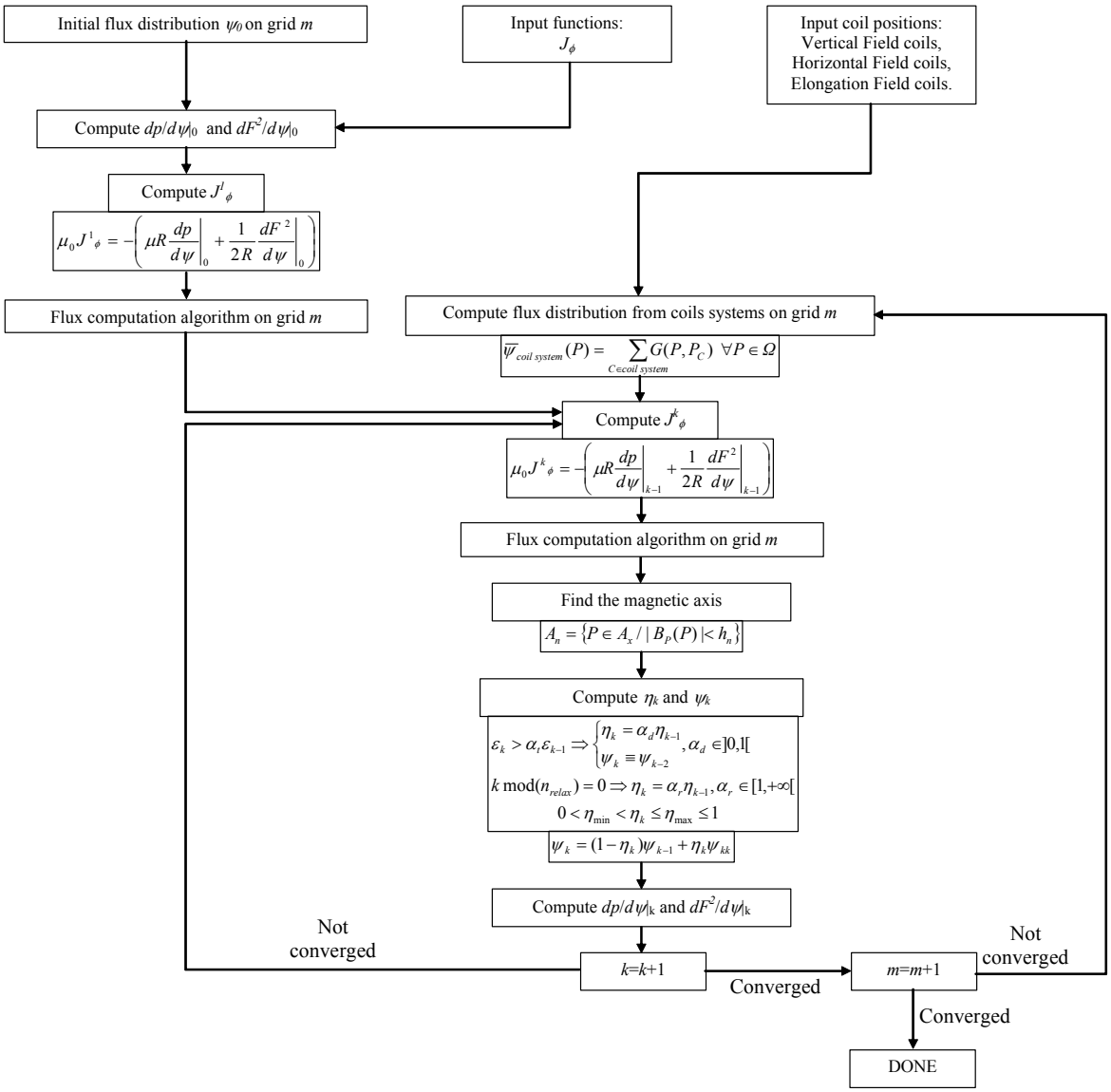


Figure 14 : Computational algorithm

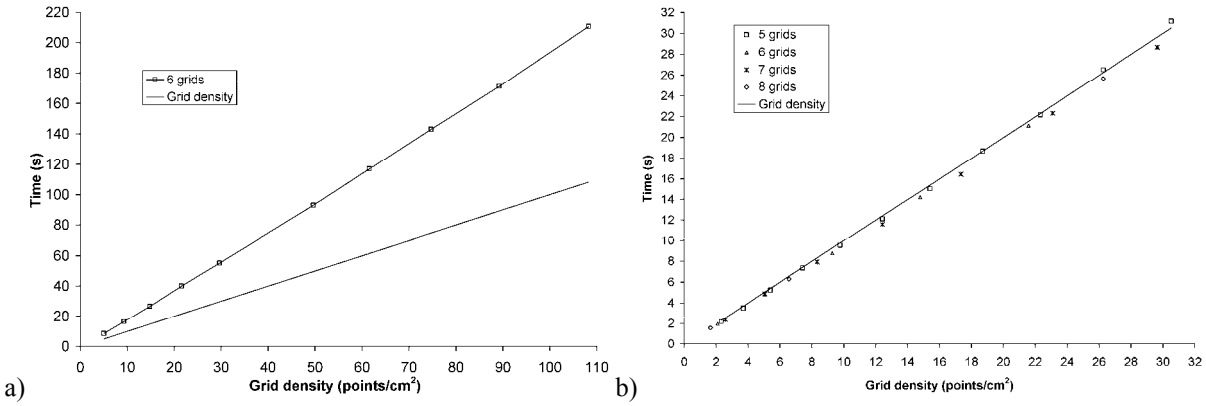


Figure 15 : Computational time versus grid density for k → k+1 step with a) 6 grids on a 1GHz PC with 2GB of memory; b) 5 to 8 grids on a 2GHz PC with 500MB of memory. The solid line corresponds to time equal to the grid density and helps to emphasize the *O(N)* time dependence of the computational algorithm.

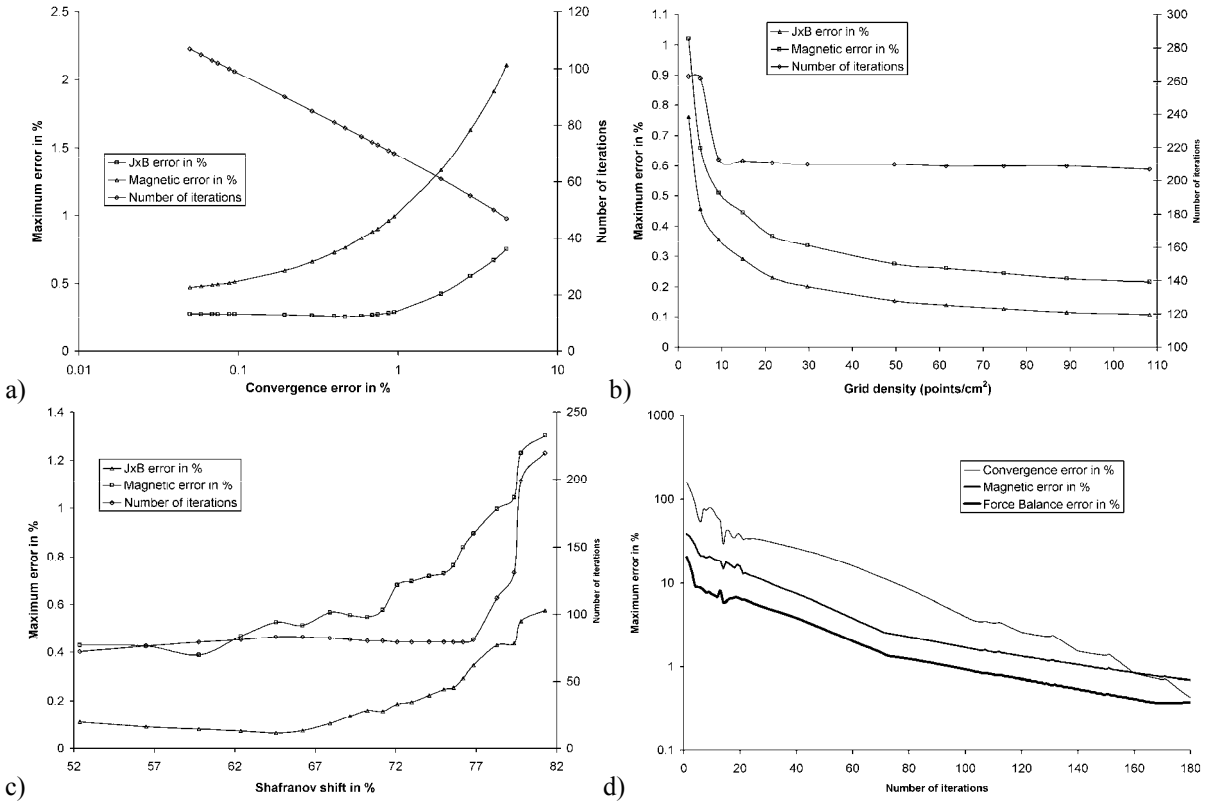


Figure 16 : a) Influence of the convergence error on the number of iterations, the magnetic error and the force balance error for a grid density of 5 points/cm^2 with a Shafranov shift of 75%; b) influence of the grid density on the number of iterations, the magnetic error and the force balance error for a Shafranov shift of 80% with a convergence error $\varepsilon_{convergence}$ of 0.05%; c) Influence of the Shafranov shift on the number of iterations, the magnetic error and the force balance error for a grid density of 5 points/cm^2 with a convergence error $\varepsilon_{convergence}$ of 0.5%; d) Typical convergence for a Shafranov shift of 83%, a grid density of 19 points/cm^2 and a convergence error $\varepsilon_{convergence}$ of 1%.

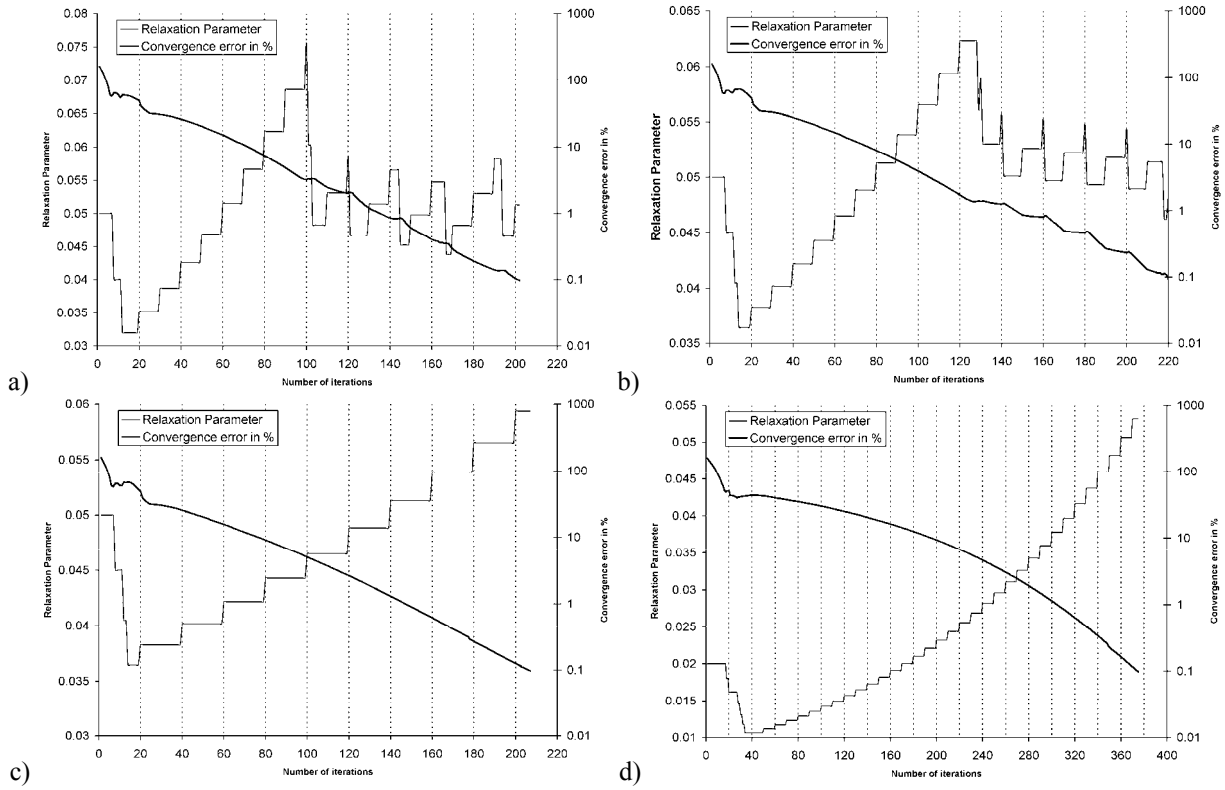


Figure 17 : Automatic adjustment of the relaxation parameter and the corresponding result on the convergence error for a Shafranov shift of 83%, α_f=1.01, $\varepsilon_{convergence}$=0.5% and a grid density of 22 points/cm^2. The following parameters were used: a) η_{init}=0.05, α_d=0.8, n_{relax}=10 and α_r=1.1; b) η_{init}=0.05, α_d=0.9, n_{relax}=10 and α_r=1.05; c) η_{init}=0.05, α_d=0.9, n_{relax}=20 and α_r=1.05; d) η_{init}=0.02, α_d=0.9, n_{relax}=10 and α_r=1.05.

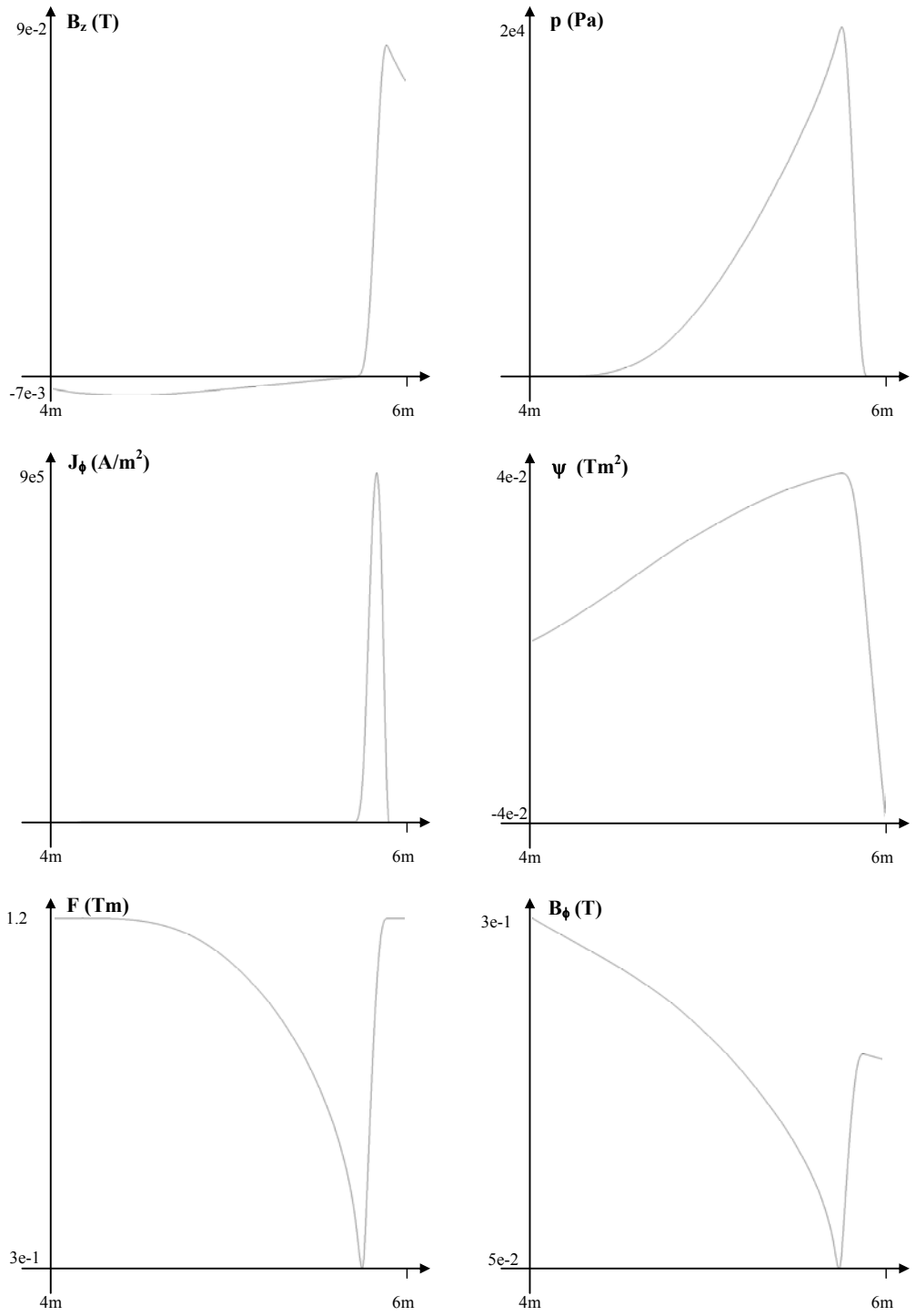


Figure 18 : Code results for a Shafranov shift of 83%, a grid density of 39 points/cm^2 and a β_{peak} of 100%.

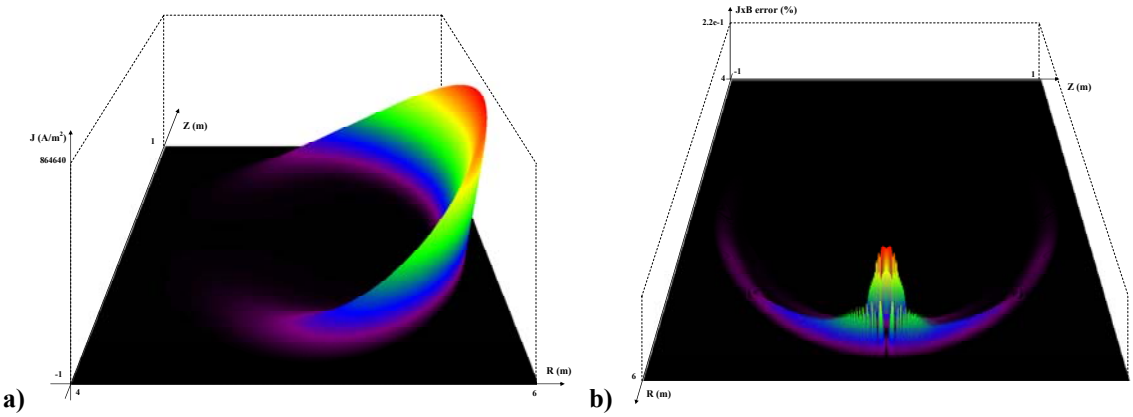


Figure 19 : a) 3D view of the current density and b) the force balance error for a Shafranov shift of 83%, a grid density of 39 points/cm^2 and a β_{peak} of 100% for a total current of 168kA.

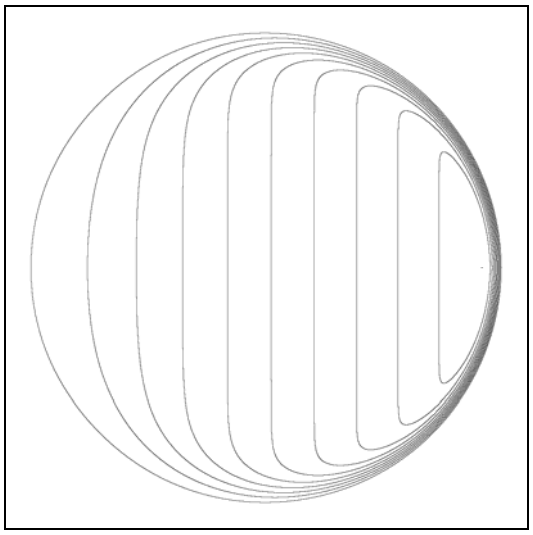
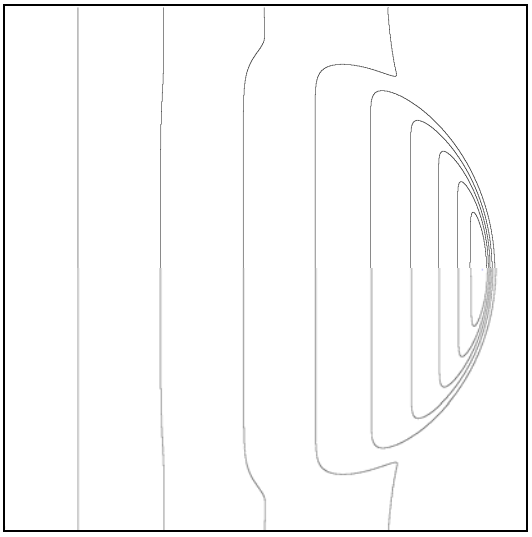
a)  b) 

Figure 20 : a) Flux surface distribution and b) |B| for a Shafranov shift of 91%, a grid density of 80 points/cm^2 and a β_{peak} of 70%. The magnetic well due to strong plasma diamagnetism is clearly visible on the |B| picture.