Fusion Engineering and Design 96–97 (2015) 664–667

Contents lists available at ScienceDirect

Fusion Engineering and Design

journal homepage: www.elsevier.com/locate/fusengdes

# CREATE-NL+: A robust control-oriented free boundary dynamic plasma equilibrium solver

R. Albanese [a], R. Ambrosino [b], M. Mattei [c,*]

[a] Ass. EURATOM/ENEA/CREATE, Universita' di Napoli "Federico II", Naples, Italy
[b] Ass. EURATOM/ENEA/CREATE, Universita' di Napoli "Parthenope", Naples, Italy
[c] Ass. EURATOM/ENEA/CREATE, Seconda Universita' di Napoli, Naples, Italy

ARTICLE INFO

*Article history:*
Received 5 October 2014
Received in revised form 30 June 2015
Accepted 30 June 2015
Available online 30 July 2015

*Keywords:*
Free-boundary equilibrium solver
Tokamak
Plasma shape control
Vertical stabilization control

ABSTRACT

CREATE-NL+ is a FEM (Finite Elements Method) solver of the free boundary dynamic plasma equilibrium problem, i.e. the MHD (Magneto Hydro Dynamics) time evolution of 2D axisymmetric plasmas in toroidal nuclear fusion devices, including eddy currents in the passive structures, and feedback control laws for current, position and shape control. This is an improved version of the CREATE-NL code developed in 2002 which was validated on JET and used for the design of the XSC (eXtreme Shape Controller), and for simulation studies on many existing and future tokamaks. A significant improvement was the use of a robust numerical scheme for the calculation of the Jacobian matrix within the Newton based scheme for the solution of the FEM nonlinear algebraic equations. The improved capability of interfacing with other codes, and a general decrease of the computational burden for the simulation of long pulses with small time steps makes this code a flexible tool for the design and testing of magnetic control in a tokamak.


## 1. Introduction

Simulation requirements from the plasma control engineering community involved in shape, position, and profile control problems call for the development of robust and efficient solvers for the so called Plasma Dynamic Free Boundary Equilibrium (PDFBE) problem. Such numerical solvers should be capable to interface with control design environments, with tools for pulse preparation, validation and testing, and with codes simulating other phenomena like transport, plasma–wall interaction, three dimensional dynamics, etc. During the last decades many research teams developed their own equilibrium simulation codes. Among others PROTEUS [1], MAXFEA [2], DINA [3], CORSICA [4], CEDRES++ [5] have been extensively used for nonlinear static and/or transient analyses in the presence of close loop controllers, some of them also implementing transport equations for the self-consistent evolution of plasma current profiles.

The available codes have their own peculiarities in terms of: (1) numerical schemes for the solution of the PDE (Partial Differential Equation) problem: finite differences, finite elements, boundary elements, etc.; (2) programming languages (FORTRAN, C++, Matlab); (3) possibility to simulate eddy currents, transport phenomena, iron components; (4) explicit or implicit integration of feedback controller equations; (5) possibility to run in direct simulation mode, i.e. assign inputs as PF currents or voltages, or inverse simulation mode, i.e. fix measurements output as constraints and find some input variable.

CREATE-NL+ code, implemented in Matlab, is a FEM code simulating the time evolution of 2D axisymmetric plasmas in toroidal nuclear fusion devices in the presence of current and/or voltage driven active circuits, currents induced in the passive conductors, and iron components.

The first version of CREATE-NL code was developed in 2002 with the objective of simulating JET plasmas in order to design and test innovative multivariable controllers like the XSC [6], although never systematically presented to the scientific community. Since that date, it was used for several activities including vertical stabilization studies on JET and ITER, shape controllability analyses on ITER, EAST, MAST, ASDEX-U, TCV, FTU, preliminary studies on FAST and DEMO.

The code requires as inputs a set of machine configuration data (geometry, active coils and passive structures configuration, first wall definition, etc.); a set of input signals. PF coils voltages can be then generated by a feedback control law, whereas plasma related quantities can be generated by a detailed transport simulation code including heating and current drive systems (see Fig. 1).

The CREATE-NL numerical solver was formulated to deal with a different number of equations and variables since the core solver

* Corresponding author.
*E-mail address:* massimiliano.mattei@unina2.it (M. Mattei).

http://dx.doi.org/10.1016/j.fusengdes.2015.06.162
0920-3796/

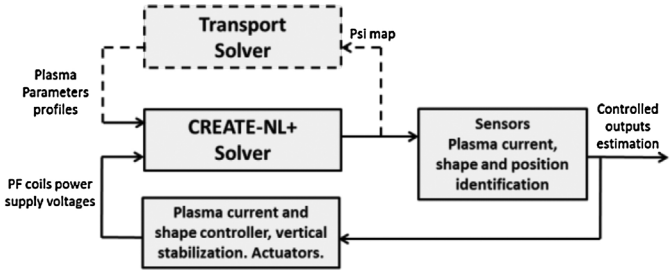


**Fig. 1.** The CREATE-NL+ solver in interaction with a feedback control block, a transport equation solver and a plasma shape, position and current identification block.

is based on a pseudo-inverse procedure. In facts CREATE-NL was used also to solve shape and profile identification problems.

From a numerical point of view, the increased robustness of CREATE-NL+ is guaranteed by a robust numerical procedure for the plasma boundary search, and by a reliable numerical solution of the nonlinear algebraic equations arising from the FEM formulation. These equations are solved with a Newton Method approach where the calculation and inversion of the Jacobian matrix plays an important role both for the computational time and the numerical stability. It was in fact observed that, in the presence of plasma related profiles provided in a numerical form, discontinuities and jumps due to FEM approximation, the calculation of the Jacobian matrix adopted in CREATE-NL determined a fragility of the solver. An ad hoc solution was found for a fast numerical Jacobian calculation which is one of the most important improvements implemented in CREATE-NL+.

## 2. The PDFBE problem

It is widely recognized that electromagnetic modeling is fundamental for plasma current, shape and position control design. Plasma, control circuits, and passive conductors have to be taken into account for this purpose. The system model is governed by Maxwell's equations in their quasi-stationary form with the constitutive relationships

$$\nabla \times E = -\frac{\partial \boldsymbol{B}}{\partial t}; \quad \nabla \times \boldsymbol{H} = \boldsymbol{J}; \quad \nabla \cdot \boldsymbol{B} = 0 \tag{1a}$$

$$\boldsymbol{B} = \mu \boldsymbol{H}; \quad \boldsymbol{J} = \sigma(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B} + \boldsymbol{E}_i) \tag{1b}$$

where $\boldsymbol{E}$ is the electric field, $\boldsymbol{B}$ the magnetic flux density, $t$ the time, $\boldsymbol{H}$ the magnetic field, $\boldsymbol{J}$ the current density, $\mu$ the magnetic permeability, $\sigma$ the electric conductivity, $\boldsymbol{v}$ the velocity, $\boldsymbol{E}_i$ the impressed field. In the plasma region, the Lorentz term $\boldsymbol{v} \times \boldsymbol{B}$ has an important effect and the plasma velocity is determined by the momentum balance

$$\rho \left( \frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right) = \mathbf{J} \times \mathbf{B} - \nabla p \tag{2}$$

where pressure, velocity and density being coupled by thermodynamic and fluid equations. In cylindrical coordinates $(r, \phi, z)$, $\boldsymbol{B}$ and $\boldsymbol{J}$ can be expressed in terms of two scalar functions, namely, the poloidal magnetic flux per radians $\psi(r,z)$ and the poloidal current function $f = \mu I_{pol}/2\pi = rB_\phi$, $B_\phi$ being the toroidal field, and $I_{pol}$ the poloidal current. Therefore magnetic flux and current densities can be expressed as

$$\boldsymbol{B} = \frac{1}{r} \nabla \psi \times \boldsymbol{i}_\phi + \frac{f}{r} \boldsymbol{i}_\phi, \tag{3a}$$

$$\boldsymbol{J} = \frac{1}{r} \nabla \left( \frac{f}{\mu} \right) \times \boldsymbol{i}_\phi - \frac{1}{\mu_0 r} \Delta^* \psi \boldsymbol{i}_\phi, \tag{3b}$$

where $i_\phi$ is the unit vector in the toroidal direction, $\mu_0$ is the magnetic permeability of the vacuum and the toroidal current component is related to $\psi$ via the second order differential operator $\Delta^*$

$$\Delta^* \psi \equiv r \frac{\partial}{\partial r} \left( \frac{1}{\mu_r r} \frac{\partial \psi}{\partial r} \right) + \frac{\partial}{\partial z} \left( \frac{1}{\mu_r} \frac{\partial \psi}{\partial z} \right) \tag{4}$$

$\mu_r$ being the relative magnetic permeability.

At the time scale of interest for current, position, and shape control, because of the low plasma mass density $\rho$, inertial effects can be neglected. Hence at equilibrium, the plasma momentum balance (2) can be rewritten as the well-known Grad–Shafranov equation, i.e. the following PDE (Partial Differential Equation) problem

$$\Delta^* \psi = -f \frac{df}{d\psi} - \mu_0 r^2 \frac{dp}{d\psi} \quad \text{in plasma region} \tag{5a}$$

$$\Delta^* \psi = -\mu_0 r j_{ext}(r, z, t) \quad \text{in} \quad \text{conductors} \tag{5b}$$

$$\Delta^* \psi = 0 \quad \text{elsewhere} \tag{5c}$$

$$\psi(r, z, t) = \psi_0(r, z), \psi(0, z, t) = 0, \quad \lim_{r^2+z^2 \to \infty} \psi(r, z, t) = 0 \tag{5d}$$

$j_{ext}$ is the toroidal current density in the external conductors and coils. The above equations are used to calculate the poloidal flux $\psi$ function at time $t$ provided that the plasma boundary can be determined, the toroidal current density in the PF coils is known, and the functions $p(\psi)$ and $f(\psi)$ are assigned within the plasma. Under simplifying assumptions, functions $p(\psi)$ and $f(\psi)$ can be expressed in terms of few plasma parameters, for example poloidal beta $\beta_p$ and internal inductance $li$. As for $j_{ext}$, this can be expressed as a linear combination of the circuit currents. Therefore, the magnetic flux and the plasma configuration can be determined when prescribing the vector of currents $\boldsymbol{I}$ (including poloidal field coils and plasma currents) along with functions $p(\psi)$ and $f(\psi)$. The time evolution of these currents is given by a circuit equation.

$$\dot{\boldsymbol{\Psi}} + \boldsymbol{RI} = \boldsymbol{V} \tag{6}$$

where $\dot{\boldsymbol{\Psi}}$ is the vector of magnetic fluxes linked with the circuits, $\boldsymbol{R}$ is the resistance matrix, and $\boldsymbol{V}$ is the vector of applied voltages. The flux vector is defined as the integral of the flux function $\psi$ over the conductor regions (see (7) for the relationship among continuous variables).

The relationship between the toroidal current density in the control circuits and the poloidal flux can be obtained from Faraday's and Ohm's laws. In principle, the active powered coils and the passive conductors can be treated in the same way. The only difference is in the applied voltage, which is zero in the passive conductors. It can be shown that

$$j_{ext} = -\frac{\sigma}{r} \frac{\partial}{\partial t} \psi + \frac{\sigma}{2\pi r} u \tag{7}$$

where $u$ is the voltage applied to the coil. Eq. (7) must then be integrated over the conductor regions.

In order to recast the PDE equilibrium problem to a finite dimensional problem a first order FEM is adopted. Plasma current density can be assigned in terms of $p(\psi)$ and $f(\psi)$ functions or described by means of a finite number of parameters using the following relationships:

$$J_{pl}(r, \psi) = r \sum_{k=1}^{M_a} a_k \chi_{ak}(\bar{\psi}, \alpha) + \frac{1}{r} \sum_{k=1}^{M_b} b_k \chi_{bk}(\bar{\psi}, \alpha) \tag{8a}$$

$$\bar{\psi} = \frac{(\psi - \psi_a)}{(\psi_b - \psi_a)} \tag{8b}$$

where $\chi$ are basis function of the normalized flux and of a parameter vector $\alpha$. In this case additional equations are needed to close the problem, e.g. $\beta_p$, $li$ and $I_p$ fixed to a prescribed value. FEM approach

finally requires the solution of a nonlinear set of equations in the form

$$\boldsymbol{F}(\boldsymbol{\psi}, \boldsymbol{\pi}) = \boldsymbol{F}(\boldsymbol{x}_1, \boldsymbol{x}_2) = 0 \tag{9}$$

in the $n_1$ unknowns $\boldsymbol{\psi}$, which is the vector of fluxes in the spatial discretization nodes, and $n_2$ unknowns $\pi = (\boldsymbol{I}^{\mathrm{T}}\ \boldsymbol{\alpha}^{\mathrm{T}})^{\mathrm{T}}$, which is a vector of variables including coil currents and profile parameters. It is worth to notice that currents become unknowns if circuits are voltage driven.

Problem (9) is solved with an iterative Newton based method where boundary conditions in (5d) are treated via a suitable coupling with boundary integral equations [7]. The calculation and inversion of the $\boldsymbol{F}$ Jacobian matrix is the core of the solver. The candidate solution update in the iterative algorithm is

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^{k} - \left(\frac{\partial \boldsymbol{F}(\boldsymbol{x}^k)}{\partial \boldsymbol{x}}\right)^{-1} \boldsymbol{F}(\boldsymbol{x}^k) \tag{10}$$

where $\boldsymbol{x} = (\boldsymbol{\psi}^{\mathrm{T}}\ \boldsymbol{\pi}^{\mathrm{T}})^{\mathrm{T}}$, and $k$ denotes the iteration step.

## 3. Numerical calculation of the Jacobian matrix

If a first order mesh of $n_1$ elements is assumed for the solution of the FEM nonlinear algebraic problem (9), the first $n_1$ equations are obtained applying the Galerkin method to the PDE problem (5). The remaining $n_2$ come from circuit and other equations used, for example, to fit $\beta_p$, $li$ and $I_p$. If the number of variables is equal to the number of equations we have

$$\boldsymbol{F}_1(\boldsymbol{x}_1, \boldsymbol{x}_2) = 0 \quad n_1 \text{ symmetric equations} \tag{11a}$$

$$\boldsymbol{F}_2(\boldsymbol{x}_1, \boldsymbol{x}_2) = 0 \quad n_2 \text{ non-symmetric equations} \tag{11b}$$

The solution of these equations via Newton method is based on the following local linearization

$$\boldsymbol{J}_{1,\boldsymbol{x}_1} d\boldsymbol{x}_1 + \boldsymbol{J}_{1,\boldsymbol{x}_2} d\boldsymbol{x}_2 = -\boldsymbol{F}_1 \tag{12a}$$

$$\boldsymbol{J}_{2,\boldsymbol{x}_1} d\boldsymbol{x}_1 + \boldsymbol{J}_{2,\boldsymbol{x}_2} d\boldsymbol{x}_2 = -\boldsymbol{F}_2 \tag{12b}$$

with $\boldsymbol{J}_{i,x_j} = \partial \boldsymbol{F}_i / \partial \boldsymbol{x}_j$, requiring the calculation of four partial Jacobian matrices. If $n_1 \gg n_2$ the numerical calculation of $\boldsymbol{J}_{1:2,\boldsymbol{x}_2}$ can be obtained by means of $n_2 + 1$ evaluations of the $\boldsymbol{F}$ function using finite differences (FD). As for the $\boldsymbol{J}_{1;\boldsymbol{x}_1}$ term, this can be made numerically tractable with the procedure described in the following. Finally $\boldsymbol{J}_{2,\boldsymbol{x}_1}$ would necessarily require $n_1$ evaluations of the $\boldsymbol{F}$ function including the computation of the plasma boundary which is cumbersome. In order to avoid the explicit calculation of $\boldsymbol{J}_{2,\boldsymbol{x}_1}$ we can consider that

$$d\boldsymbol{x}_1 = -(\boldsymbol{J}_{1,\boldsymbol{x}_1})^{-1}\boldsymbol{F}_1 - (\boldsymbol{J}_{1,\boldsymbol{x}_1})^{-1}\boldsymbol{J}_{2,\boldsymbol{x}_2} d\boldsymbol{x}_2 = d\boldsymbol{x}_{10} + \boldsymbol{M} d\boldsymbol{x}_2 \tag{13}$$

with $\boldsymbol{M} = -(\boldsymbol{J}_{1,\boldsymbol{x}_1})^{-1}\boldsymbol{J}_{1,\boldsymbol{x}_2} \in \mathbb{R}^{n_1 \times n_2}$. From (13) we have

$$\boldsymbol{J}_{2,\boldsymbol{x}_1} d\boldsymbol{x}_{10} + \boldsymbol{J}_{2,\boldsymbol{x}_1} \boldsymbol{M} d\boldsymbol{x}_2 + \boldsymbol{J}_{2,\boldsymbol{x}_2} d\boldsymbol{x}_2 = -\boldsymbol{F}_2. \tag{14}$$

This allows to avoid the direct calculation of $\boldsymbol{J}_{2,\boldsymbol{x}_1}$ but of the two terms

$$(\boldsymbol{J}_{2,\boldsymbol{x}_1}\boldsymbol{M})(:,k) = \frac{(\boldsymbol{F}_2(\boldsymbol{x}_1 + \alpha \boldsymbol{M}(:,k), \boldsymbol{x}_2) - \boldsymbol{F}_2(\boldsymbol{x}_1, \boldsymbol{x}_2))}{\alpha}$$

$$\boldsymbol{J}_{2,\boldsymbol{x}_1} d\boldsymbol{x}_{10} = \frac{(\boldsymbol{F}_2(\boldsymbol{x}_1 + \alpha d\boldsymbol{x}_{10}, \boldsymbol{x}_2) - \boldsymbol{F}_2(\boldsymbol{x}_1, \boldsymbol{x}_2))}{\alpha}$$

by means of $n_2$ finite differences, $(:,k)$ denoting the extraction of the $k$-th column from a matrix. $\alpha$ has to be carefully selected to have a relative error consistent with the solver and the calculation of the plasma boundary. The use of a sufficiently small $\alpha$ also reduces the impact of the FD calculation on the convergence properties of the Newton method which becomes comparable with that one obtained with an analytic Jacobian calculation.

As for $\boldsymbol{J}_{1,\boldsymbol{x}_1}$, the simplest method for numerically approximating the derivatives in a Jacobian matrix is to use FD. Given a nonlinear system of $n_1$ equations in $n_1$ unknowns, the FD method allows a column wise calculation of the Jacobian based on $n_1$ evaluations of the nonlinear function to be nullified for $n_1$ independent sufficiently small variations of the unknowns. Another usual method is the Automatic Differentiation [8] implemented in various programming languages.

In equation (11a) each component of the vector valued function to be nullified depends only on a small number of unknowns within vector $\boldsymbol{x}_1$, this leads to a sparse $\boldsymbol{J}_{1,\boldsymbol{x}_1}$ Jacobian matrix. The sparsity pattern is also a priori known and the relationships among the unknowns are determined by the proximity of nodes and elements. For sparse Jacobian matrices it has been proved [9] that, when two or more columns are structurally orthogonal, i.e. there is no row in which more than one column has a nonzero, they can be computed simultaneously using finite differences by perturbing the corresponding independent variables. In this way, the number of evaluations of the nonlinear system of equations needed to compute the Jacobian decreases. A Matlab implementation of a graph coloring technique, to exploit the sparsity of the Jacobian matrix using AD techniques was also discussed in [10].

CREATE-NL+ implements a novel ad hoc method for the evaluation of sparse Jacobian matrices assuming that the sparsity pattern is known. This method is particularly advantageous in view of parallel computing implementations and is an alternative to AD techniques which do not assume any a priori knowledge on the sparsity structure of the Jacobian matrix.

Let us assume a nonlinear algebraic differentiable vector valued function

$$\boldsymbol{F}(\cdot): \quad \boldsymbol{x} \in \mathbb{R}^n \rightarrow \boldsymbol{F}(\boldsymbol{x}) \in \mathbb{R}^m \tag{15}$$

where each scalar component of $\boldsymbol{F}$, say $\boldsymbol{F}_i$, depends only on a "small" number of components of the $\boldsymbol{x}$ vector. Let us denote with $K_i$ the set of indices corresponding to the elements of $\boldsymbol{x}$ causing a non-null variation of $\boldsymbol{F}$

$$K_i = \{j \in (1, \ldots, n) : \partial \boldsymbol{F}_i / \partial \boldsymbol{x}_j \neq 0\}. \tag{16}$$

Define also the maximum number of nonzero elements in each row of $\partial \boldsymbol{F}/\partial \boldsymbol{x}$, that is $N_{\max} = \max_i(card(K_i))$, where $card(\bullet)$ indicates the cardinality of its argument.

The classical method to compute $\boldsymbol{J}$ by means of finite differences, is to compute each element as

$$\boldsymbol{J}_{i,j} = \frac{(\boldsymbol{F}_i(\boldsymbol{x} + \Delta^j \boldsymbol{x}) - \boldsymbol{F}_i(\boldsymbol{x}))}{\varepsilon_x} \tag{17}$$

where $\Delta^j \boldsymbol{x}_k = \varepsilon_x$ if $k = j$, and equal to zero otherwise, $\boldsymbol{\varepsilon}_x$ being sufficiently small. This method requires a number of evaluations of the nonlinear function linearly increasing with the number $n$ of variables.

An efficient low cost numerical algorithm for the Jacobian matrix $\boldsymbol{J}$ can be applied under the hypothesis that $N_{\max} \ll n$. In fact, in the linear approximation:

$$\boldsymbol{J}(\boldsymbol{x})[d\boldsymbol{x}_1, \ldots, d\boldsymbol{x}_s] = [\boldsymbol{F}(\boldsymbol{x} + d\boldsymbol{x}_1) - \boldsymbol{F}(\boldsymbol{x}), \ldots, \boldsymbol{F}(\boldsymbol{x} + d\boldsymbol{x}_s) - \boldsymbol{F}(\boldsymbol{x})] \tag{18}$$

Hence one can consider the elements of the Jacobian matrix as the unknowns of linear systems in which the number of evaluations $s + 1$ of the nonlinear function $\boldsymbol{F}$ depends on the number of nonzero elements in $\boldsymbol{J}$.

With $N_{\max}$ independent and sufficiently small variations $d\boldsymbol{x}^{\{k\}}$ $k = 1, \ldots, N_{\max}$ of $\boldsymbol{x}$ we have

$$\boldsymbol{J}(\boldsymbol{x}) d\boldsymbol{x}^{\{k\}} = \boldsymbol{F}(\boldsymbol{x} + d\boldsymbol{x}^{\{k\}}) - \boldsymbol{F}(\boldsymbol{x}) \tag{19}$$

Since each row of $\boldsymbol{J}$ has no more than $N_{\max}$ nonzero terms, a number $card(K_i) \leq N_{\max}$ of equations is sufficient to evaluate the non-zero terms of the row. In particular, for $i = 1, \ldots, m$, we have

$$\boldsymbol{A}_i \boldsymbol{J}_i = \begin{bmatrix} \boldsymbol{F}_i(\boldsymbol{x} + d\boldsymbol{x}^{\{1\}}) - \boldsymbol{F}_i(\boldsymbol{x}) \\ \boldsymbol{F}_i(\boldsymbol{x} + d\boldsymbol{x}^{\{2\}}) - \boldsymbol{F}_i(\boldsymbol{x}) \\ \vdots \\ \boldsymbol{F}_i(\boldsymbol{x} + d\boldsymbol{x}^{\{card(K_i)\}}) - \boldsymbol{F}_i(\boldsymbol{x}) \end{bmatrix} \tag{20}$$

where $\boldsymbol{A}_i \in \mathbb{R}^{card(K_i) \times m}$ is computed on the basis of $\boldsymbol{x}$ variations as

$$\boldsymbol{A}_i = \begin{bmatrix} d\boldsymbol{x}^{\{1\}} & d\boldsymbol{x}^{\{2\}} & \ldots & d\boldsymbol{x}^{\{card(K_i)\}} \end{bmatrix}^T \tag{21}$$

The proposed method can be expressed in a compact form by means of the definition of the following system:

$$\begin{bmatrix} A_1 & 0 & \ldots & 0 \\ 0 & A_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \ldots & 0 & A_m \end{bmatrix} \begin{bmatrix} \boldsymbol{J}_1 \\ \boldsymbol{J}_2 \\ \vdots \\ \boldsymbol{J}_m \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_m \end{bmatrix} \tag{22}$$

where

$$\boldsymbol{b}_i = \begin{bmatrix} \boldsymbol{F}_i(\boldsymbol{x} + d\boldsymbol{x}^{\{1\}}) - \boldsymbol{F}_i(\boldsymbol{x}) \\ \boldsymbol{F}_i(\boldsymbol{x} + d\boldsymbol{x}^{\{2\}}) - \boldsymbol{F}_i(\boldsymbol{x}) \\ \vdots \\ \boldsymbol{F}_i(\boldsymbol{x} + d\boldsymbol{x}^{\{card(K_i)\}}) - \boldsymbol{F}_i(\boldsymbol{x}) \end{bmatrix} \tag{23}$$

for $i = 1,\ldots,m$. The main advantage of the proposed method is that it relates the number of evaluations of the nonlinear function to be nullified to the sparsity pattern of the Jacobian matrix and, in particular, to the maximum number $N_{\max}$ of nonzero elements in the rows of $\boldsymbol{J}$.

## 4. Conclusions

CREATE-NL+ is a free boundary dynamic plasma equilibrium problem solver, derived from the pre-existent CREATE-NL (2002) which was validated on several tokamaks and used for many studies. A significant improvement is the use of a robust numerical scheme for the calculation of the Jacobian matrix within the Newton based scheme for the solution of the FEM nonlinear algebraic equations. This improvement allowed to robustify the solver in the presence of nonsmooth profiles coming from interacting transport simulation modules or feedback controllers, but also to fasten the calculation by about one order of magnitude. The proposed numerical scheme has a straightforward implementation which is oriented to an implementation on parallel computing devices. CREATE-NL+ is integrated with a suite of tools allowing the optimization of nominal preprogrammed currents to drive the plasma through a desired scenario and the design of closed loop control laws for shape control and vertical stabilization.

## References

[1] R. Albanese, et al., Numerical studies of the next European Torus via the PROTEUS code, in: 12th Conf. on Numerical Simulation of Plasmas, San Francisco, CA, 1987.
[2] P. Barabaschi, The Maxfea code, in: Plasma Control Technical Meeting, Naka, Japan, 1993.
[3] R.R. Khayrutdinov, V.E. Lukash, J. Comput. Phys. (1993) 109–193.
[4] J.A. Crotinger, et al., LLNL ReportUCRL-ID-126284 NTIS #PB2005-102154, 1997.
[5] H. Heumann, et al., Quasi-static free-boundary equilibrium of toroidal plasma with CEDRES++: computational methods and applications, J. Plasma Phys. (2015) 1–35.
[6] R. Albanese, et al., Plasma response models for current, shape and position control in jet, Fusion Eng. Des. 66–68 (2003) 715–718.
[7] R. Albanese, J. Blum, O. Barbieri, On the solution of the magnetic flux equation in an infinite domain, in: EPS. 8th Europhysics Conference on Computing in Plasma Physics, 1986.
[8] C.H. Bischof, et al., ADIFOR 2.0: system for the automatic differentiation of Fortran 77 programs, IEEE Comput. Sci. Eng. 3 (3) (1996) 18–32.
[9] A.R. Curtis, et al., On the estimation of sparse Jacobian matrices, J. Inst. Math. Appl. 12 (1974) 117–119.
[10] T.F. Coleman, A. Verma, ADMAT: an automatic differentiation toolbox for MATLAB, in: Proceedings of the SIAM Workshop on Object Oriented Methods for Inter-Operable Scientific and Engineering Computing, SIAM, Philadelphia, PA, 1998.