Physics AUC, vol. 27, 1-9 (2017) **PHYSICS AUC**

# Minimization algorithm in the simulation of the Wall Touching Kink Modes

Steinbrecher Gyorgy, Pometescu Nicolae
Department of Physics, University of Craiova,
Str. A. I. Cuza, No.13, 200585 - Craiova, Romania

**Abstract**

The discretized variational principle in the simulation of the Wall Touching Kink Modes (WTKM) is reformulated in terms of independent variables and a a corresponding constrained minimization algoritm is elaborated. In a frame of a general formalism an efficient algoritm for constrained linear minimization that is elaborated, that is addapted to this class of problems. The FORTRAN programme that realize the algorithm is described.

## 1 Introduction

The simulation of the currents in the tokamak wall was studied in [1], by using the boundary element method for solving the MHD equations in the thin wall approximation by using the triangular, linear, conforming finite element method. But a new problem arise when on the internal face of the tokamak wall a new conducting plate is welded (for instance, a limiter). In general, the position of the limiter is not related to the existing triangulation, and so, nonconforming finite elements appears by the triangulation of the limiter [2]: the edges and vertices of the triangles on the outer circumference of the limiter, in the generic case are in the interior of the triangles resulted from the finite element study of the tokamak before the limiter welding. Consequently the physical data attached to the finite elements of the limiter are related by linear constraints to the data attached to the finite elements from the tokamak wall.

A problem to be solved is how to include these new constraints such that the modifications in the existing code to be minimal. A specific problem of the boundary elements method is that at each iteration step, large scale quadratic optimization must be performed, where the Hessian matrix is not sparse.

Next, the article is organized as follows: In the first section will be introduced the notations in order to formulate the general formalism for the problem we intend to solve. For that, the (discretized) variational principle is reformulated in the terms of independent variables. The new objective function will be introduced in Section 2 and described an efficient algorithm for constrained linear minimization that is adapted to this class of problems. In Section 3 will be described the FORTRAN programme that realize the algorithm and in the final section we give some conclusions.

## 2 Notations

We denote the set of all variables, in the discretized version, the potentials and the currents, attached to the set of all vertices, by $\mathcal{V}$ . This is represented by a vector with $|\mathcal{V}|$ components $\mathbf{X} = \{X_1, ..., X_{|\mathcal{V}|}\}$ . Here in general $|\mathcal{A}|$ denote the cardinality of the set $\mathcal{A}$. In this stage there are no restriction (conformity, Neumann or Dirichlet boundary conditions) on the variables $\mathbf{X} = \{X_1, ..., X_{|\mathcal{V}|}\}$ The general form of the functional after discretization is of the form

$$Q(\mathbf{X}) = \frac{1}{2} \langle \mathbf{X}, \mathbf{H}\mathbf{X} \rangle + \langle \mathbf{L}, \mathbf{X} \rangle + C \tag{1}$$

$$= \frac{1}{2} \sum_{m,n} X_m H_{m,n} X_n + \sum_m X_m L_m + C \tag{2}$$

Here $\mathbf{H}$ is the positive definite Hessian matrix of the quadratic form $Q(\mathbf{X})$ , constructed from the mutual capacitances and inductances ( *see Eqs. (5.6 , 5.9)* from [1]) , $\langle \mathbf{L}, \mathbf{X} \rangle$ , $C$ are linear and constant terms included for the sake of generality required for the possibility to test the programme.

The subset of $\mathcal{V}$ that consists of all variables that are on the boundaries (the welding line where the limiter is fixed to the wall, the inner edge of the limiter and the variables associated on the boundary of the holes in the tokamak wall) are subjected to restrictions denoted by $\mathcal{B}$ . The rest, the set of independent variables, will be denoted by $\mathcal{I}$. So

$$\mathcal{V} = \mathcal{B} \cup \mathcal{I} \quad , \quad \mathcal{B} \cap \mathcal{I} = \varnothing$$

***Remark*** *In the set $\mathcal{B}$ of boundaries we include, as usual, the lines that define the holes in the tokamak wall, but we also include the line of welding where the new metal plate (possibly a limiter ) is attached to the tokamak wall. In this last case the triangulation on the tokamak wall remains the same as before the welding. A separate triangulation of the new plate is performed. The vertex points the triangle lying on of the inner edge (that are not in contact with the wall) are included in the set $\mathcal{B}$. The vertex points of the triangles of the plate triangulation, that are also on the tokamak wall are included in the set $\mathcal{B}$. For a general case, these new points, appeared after welding, are in the interior of some triangle constructed before welding, so they are no more independent variables, they are subjected to linear constraints resulted from linear interpolation. Consequently, the values of the potentials on these new set of points are expressed, by linear interpolation, by the values of the potentials on the triangles from the tokamak wall.*

The general form of the constraints on the boundaries are of the form

$$X_b = \sum_{i \in \mathcal{I}} F_{b,i} \; X_i + S_b; \; b \in \mathcal{B} \tag{3}$$

where the matrix $F_{b,i}$ and the (possible) source term vector $S_b$ encodes the boundary condition. We denote the affine submanifold of $\mathbf{R}^{|\mathcal{V}|}$ given by the constraints Eqs.(3) with $\mathbf{Z}$ , its dimension is $|\mathcal{V}| - |\mathcal{B}| = |\mathcal{I}|$

Our first goal is to develop a formalism that despite is not optimal, from the point of view of memory management, it is sufficiently compact such that the corresponding Fortran program is easy to be verified with synthetic data. In this end **we expand** the arrays $\mathbf{F}, \mathbf{S}$ , that in Eq.(3) has low dimension, $|\mathcal{B}| \times |\mathcal{I}|$ , respectively $|\mathcal{B}|$ to larger,

the extended array $\widetilde{\mathbf{F}}$ with dimensions $|\mathcal{V}| \times |\mathcal{V}|$ , respectively the extended array of the sources $\widetilde{\mathbf{S}}$ of dimension $|\mathcal{V}|$ as follows.

$$\widetilde{F}_{b,b'} = 0;\ b \in \mathcal{B},\ b' \in \mathcal{B} \tag{4}$$
$$\widetilde{F}_{i,j} = \delta_{i,j};\ i \in \mathcal{I},\ j \in \mathcal{I} \tag{5}$$
$$\widetilde{F}_{i,b} = 0;\quad i \in \mathcal{I}, b \in \mathcal{B} \tag{6}$$
$$\widetilde{F}_{b,i} = F_{b,i} = unchanged;\quad i \in \mathcal{I}, b \in \mathcal{B}$$

The corresponding expansion of the vector $\mathbf{S}$ is similar

$$\widetilde{S}_i = 0;\ i \in \mathcal{I} \tag{7}$$
$$\widetilde{S}_b = S_b = unchanged;\ b \in \mathcal{B}$$

With these conventions we introduce the parametrization of the submanifold $\mathbf{Z}$ by the vector $\mathbf{Y} = \{Y_1, ...Y_{|\mathcal{V}|}\}$ of the restrictions Eq.(3) as follows

$$\mathbf{X} = \widetilde{\mathbf{F}}\mathbf{Y} + \widetilde{\mathbf{S}} \tag{8}$$
$$X_n = \sum_{m \in \mathcal{V}} \widetilde{F}_{n,m}\ Y_m + \widetilde{S}_n;\ n \in \mathcal{V} \tag{9}$$

where $\mathbf{Y}$ is an arbitrary vector with $|\mathcal{V}|$ components. By Eqs.(4-6, 9 ) results that in the case when

$$Y_i = 0, i \in \mathcal{I} \tag{10}$$

results that

$$\widetilde{\mathbf{F}}\mathbf{Y} = 0$$

Consequently without loss of generality in the parametrization from Eq.(9) we impose the restriction

$$Y_b = 0;\ b \in \mathcal{B} \tag{11}$$

This subspace of the variable $\mathbf{Y}$ , will be denoted by $\mathbf{U}$ , it has the dimension $|\mathcal{I}|$, like the subspace $\mathbf{Z}$ defined by Eqs.(3).

# 3 The new objective function

Now the minimization problem of the objective function $Q(\mathbf{X})$ from Eqs(1, 2) with restriction Eq.(3), or equivalently

$$\min_{\mathbf{X} \in \mathbf{Z}} Q(\mathbf{X})$$

by the representations Eqs.(9, 11) can be reformulated as

$$\min_{\mathbf{X} \in \mathbf{Z}} Q(\mathbf{X}) = \min_{Y \in \mathbf{U}'} Q^{(new)}(\mathbf{Y}) \tag{12}$$

where the new quadratic form $Q^{(new)}(\mathbf{Y})$ is given by the following set of equations

$$Q^{(new)}(\mathbf{Y}) = \frac{1}{2}\left\langle \mathbf{Y}, \mathbf{H}^{(\mathbf{new})}\mathbf{Y} \right\rangle + \left\langle \mathbf{L}^{(\mathbf{new})}, \mathbf{Y} \right\rangle + C^{(new)} =$$
$$\frac{1}{2}\sum_{m,n} Y_m H^{(new)}_{m,n} Y_n + \sum_m Y_m L^{(new)}_m + C^{(new)} \tag{13}$$

According to Eqs. (1, 2 , 9) the new Hessian matrix is given by

$$\mathbf{H}^{(new)} = \widetilde{\mathbf{F}}^T \mathbf{H} \widetilde{\mathbf{F}}$$
$$H_{m,n}^{(new)} = \sum_{p.q} \widetilde{F}_{p,m} H_{p,q} \widetilde{F}_{qn} \tag{14}$$

Similarly the new linear term is

$$\mathbf{L}^{(\mathbf{new})} = \widetilde{\mathbf{F}}^T \mathbf{L} + \widetilde{\mathbf{S}}^T \mathbf{H} \widetilde{\mathbf{F}}$$
$$L_n^{(new)} = \sum_{p.} \widetilde{F}_{p,n} L_p + \sum_{p.q} \widetilde{S}_p H_{p,q} \widetilde{F}_{qn} \tag{15}$$

By the same reasoning, the new constant term is

$$C^{(new)} = C + \left\langle \mathbf{L}, \widetilde{\mathbf{S}} \right\rangle + \frac{1}{2} \left\langle \widetilde{\mathbf{S}}, \mathbf{H} \widetilde{\mathbf{S}} \right\rangle \tag{16}$$
$$C^{(new)} = C + \sum_{p.} \widetilde{S}_p L_p + \frac{1}{2} \sum_{m,n} \widetilde{S}_m H_{m,n} \widetilde{S}_n \tag{17}$$

# 4 The structure of the Fortran90 programmes.

The programmes are written such that they can be used for a large class of quadratic minimization problems.

## 4.1 Generation of the initial data, without boundary conditions

### 4.1.1 The Hessian matrices used in test

The synthetic data for test must be chosen such that the quadratic form associated to the Hessian matrix is positive definite, and the asymptotic behavior for large indices must be similar to that of mutual capacities and mutual inductance matrices from Ref. [1] . We used two forms

$$H^{(1)}(m,n) := d\delta_{m,n} + \frac{1}{(m+n+a)^p}; d > 0; \quad a > 0; \; p \in \mathbb{N} \tag{18}$$

respectively

$$H^{(1)}(m,n) := d\delta_{m,n} + \left[ \frac{\sin[a(m-n)]}{(m-n)} \right]^p ; d > 0; \quad a > 0; \; p \in \mathbb{N} \tag{19}$$

It can be verified that these Hessian matrices are positive definite by using the identities

$$\frac{1}{(m+n+a)^p} = \int_0^\infty \ldots \int_0^\infty dx_1..dx_p \exp\left[ -(m+n+a) \sum_{k=1}^p x_k \right]$$
$$\left[ \frac{\sin[a(m-n)]}{(m-n)} \right]^p 2^p = \int_{-a}^a \ldots \int_a^a dx_1..dx_p \exp\left[ i\,(m-n) \sum_{k=1}^p x_k \right]$$

### 4.1.2 Programming details

The initial data are generated such that the result of the constrained optimization are already known. **The generation of the matrix $H_{m,n}$ and the array $L_m$ and constant** $C$ from Eq. (2) is performed in the module quadraticformdatamod. It has the following entries:

*module quadraticformdatamod*

*implicit none* ! contains all of the constant scalars, arrays, matrices and their generating subroutines

*integer, parameter:: nvariables=10* ! Number of free variables in the objective function.

*real(8), parameter::hessa=0.0d0* ! parameter in the test hessian function, shift , only for test runs

*real(8), parameter::hessdiag=1.0000d-4* ! diagonal term of hessian , only for test runs

*integer, parameter::hessn=2* ! parameter in the test hessian function, exponent , only for test runs

*real(8), dimension(:,:), ALLOCATABLE:: Hessian* ! Used in "objective function module", give the quadratic term of objective function

*real(8), dimension(:), ALLOCATABLE:: Linearterm* ! Used in "objective function module", give the linear term of objective function

*real(8):: constantterm* ! Used in "objective function module", give the constant term of objective function .

The initialization is controlled by the subroutine *subroutine initializQuadrform(errorflag)*

When called from the main program, activates the following subroutines:

*subroutine allocatearrays(nvariables, succesfullallocated)*

This subroutine allocate the Hessian matrix and the array of linear terms. Their numerical values, as well as of the constant $C$ are fixed in the subroutines

*subroutine generateHessianmatrix (nvariables, errorflagout)*

*subroutine generateLinearterm(nvariables, errorflagout)*

*subroutine generateconstantterm(nvariables)*

For test runs the matrix elements of the Hessian matrix are provided by the function *function hessianfunct(nvariables, i, j, errorflaghfunct) result( hess ) , having the heading:*

*integer, intent(in):: nvariables, i, j*

*integer, intent(out)::errorflaghfunct*

Its algebraic form is selected such that the resulting Hessian matrix is pozitive. It contains free parameters defined in the front of this module: *parameter::hessa*, and *parameter::hessdiag*.

The linear term and constant term are generated such that the exact value of the minimization is the result returned by the special choice of the following real valued function:

*function lfunct(k) .*

## 4.2 Imposing boundary conditions.

The generation of the matrix $F_{b,i}$ , the source term array is $S_b$ , from Eq.(3), the generation of the new Hessian matrix $H_{m,n}^{(new)}$, the new linear term $L_m^{(new)}$, the new constant term $C^{(new)}$, that defines the new quadratic form from Eq.(13) resulting from the restriction of the quadratic form Eq.(2) on the submanifold **Z** imposed by the boundary conditions

Eq.(3), is realized according to the equations (14), (15) and (17). The explicit realization is in the following module:

*module boundarydatamod*

It has the following entries

*use quadraticformdatamod*

*implicit none*

*integer, parameter:: nboundaryelements= 40 ;* ! Number of variables to be eliminated by boundary conditions

*integer, dimension(:), ALLOCATABLE::boundarylist* ! boundarylist(k)=1 => variable $k$ is from set $\mathcal{B}$, the boundary set, else is = 0

*real(8), dimension(:), ALLOCATABLE:: Sbound* ! encode boundary sources, term $S(i)$ , zero for $i$ is in $\mathcal{I}$

*real(8), dimension(:,:), ALLOCATABLE:: Fboundarymatrix* ! here sparse matrix, encode boundary condition

*real(8), dimension(:,:), ALLOCATABLE:: newHessian* ! Used in "objective function module", give the quadratic term of objective function

*real(8), dimension(:), ALLOCATABLE:: newLinearterm* ! Used in "objective function module", give the linear term of objective function

*real(8):: newconstantterm* ! Used in "objective function module", give the constant term of objective function

The allocation and generation of the new arrays is controlled by

*subroutine initboundaryArrays(errorflag)*

It is activated from the main programme. By its call, finally the matrix $F_{b,i}$ , the source term array is $S_b$ , from Eq.(3) are allocated and computed. To this end, this subroutine activate the following subroutines

A.

*subroutine initboundarydata(errorflag)*

By its call, finally the matrix $F_{b,i}$ , the source term array is $S_b$ , from Eq.(3) are allocated and computed. It controls the following subroutines

*A1 .*

*subroutine allocateboundarydata(nvariables, errorflag1)*

*A2*

*subroutine generateboundarycond(nvariables, errorflag2)*

*B.*

*subroutine initializNewQuadrform(errorflag2)*

By calling this subroutine are allocated and computed the new Hessian matrix $H^{(new)}_{m,n}$, the new linear term $L^{(new)}_m$, the new constant term $C^{(new)}$. To this end the following subroutines are controlled:

*B1.*

*subroutine allocateNewarrays(nvariables, succesfullallocated)*

*B2.*

*subroutine generateNewHessianmatrix (nvariables, errorflhessiangen)*

*B3.*

*subroutine generateNewLinearterm(nvariables, errorflgenlin)*

*B4.*

*subroutine generateNewconstantterm(nvariables)*

By calling these previous subroutines the initialization phase of the programme is finished.

The constrained optimization is encoded in the optimization programme that used a slightly modified version of the Fletcher-Reeves conjugate gradient method. Our new version is particulariation of the general nonlinear optimization method to the case when the objective function is quadratic polinomial. It returns exact result after a single optimization cycle, for an ideal computer, or at most 2-3 iterations, due to rounding errors.

For the constrained optimization of the quadratic form defined in the Equation(13), the algorithm uses the objective function, gradient and Hessian of the objective function Equation(13). The Hessian is constant and was already computed. The gradient and the objective functions are contained in the module

*module Newobjective_functionMod*

It has the entries

*use quadraticformdatamod*
*use boundarydatamod*

It contains the following realization of the objective function

*function NewobjectivefunctionFunc(nvariables, variables, errorflag) result(f)*
*integer, intent(in):: nvariables* ! # of parameters
*real(8), intent(in)::variables(nvariables)* **! variables**
*integer,intent(out)::errorflag*
*real(8)::f* ! returned function value

The gradient is computed by the following subroutine

*subroutine newgradSubr( nvariables, variables, gradient , errorflag)*
*integer, intent(in)::nvariables*
*real(8), intent(in)::variables(nvariables)*
*real(8), intent(out)::gradient(nvariables)* ! The gradient
*integer, intent(out)::errorflag*

In this module we have the subroutine, that Projects to the subspace denoted by " U" defined by formula (13).

*subroutine projection(vector)*
*real(8), intent(inout)::vector(nvariables)*

The constrained conjugate gradient optimization programme is contained in the module

*module ProjFletcherReevesMod*

It has the first entries:

*use quadraticformdataMod*
*use boundarydataMod*
*use newobjective_functionMod*

It contains the constrained minimization programme

*subroutine ProjFletcherReevesSubr1(nvariable,nitmax, gradientbound, metric, x0, xf, pnit, minvalue, gradientfinal, errorflag)*

The programme uses the following arguments

*integer, intent(in)::nvariable* ! nr of variables
*real(8), intent(in):: nitmax* ! max allowed nr of iteration, stop criteria
*real(8), intent(in)::gradientbound* ! stop criteria: if gradient module < gradientbound then stops
*real(8), intent(in):: metric(nvariable)* !! for rescalling the variables
*real(8), intent(in)::x0(nvariable)* ! initial point Side effect, it is modified

*real(8), intent(out)::xf(nvariable)* ! final point
*real(8), intent(out):: pnit* ! number of actual iterations
*real(8), intent(out)::minvalue* ! The final minimal value
*real(8), intent(out)::gradientfinal* ! module of gradient value after optimization, if close to zero
*integer, intent(out)::errorflag* ! = 0 optimization is succesfull, else = 1

**The mainprogamme has the following entries**

*program FRoptimizationmain*
*use ProjFletcherReevesMod*
*use quadraticformdatamod*
*use boundarydatamod*

In this test programme we have succesively the following subroutine calls for initialization:

*call initializQuadrform(errorflag)*
*call initboundaryArrays(errorflag)*

The follwing call is for final test

*call ProjFletcherReevesSubr1(nvariables,nitmax, gradientbound, metric, variables1, variablesfin, pnit, minvalue, gradientfinal, errorflag)*

If the programme is correct, the minimal value of the objective function must be close to zero and the returned values of "variablesfin" must be close to the selected already known exact solution.

# 5 Conclusions

In order to solve the problems that apear in the simulation of the WTKM, we propose a new general algorithm for the construction of the new objective function (that appears in the new optimization problem after attaching the limiter on the tokamak wall). The construction of the new objective function starts from the quadratic objective function, see [1], that appears in the simulation without limiter. The coupling of the currents and electric potentials in the limiter and tokamak wall are described by a set of linear constraints. By a suitable change of variables the initial constraints are transformed, and the constrained optimization is greatly simplified (compared to the general constrained conjugated gradient optimization method [3]). The FORTRAN90 test programme consists of main programme and four modules, that

- allocate and generate the initial data, the Hessian matrix and linear part of the objective function, as well as constant term, that is used for verification. The synthetic data for second order term were chosen such that the resulting matrix of relative capacitances of the triangulation be strictly positive definite
- allocate and generate the data related to attaching the limiter, in form of set of arrays, that defines the constraints related to boundary conditions on the contact line between limiter and tokamak wall
- constructs the new Hessian and new linear term, that, generate the objective function that describe the limiter-tokamak wall system.
- perform the constrained minimization.

The subroutines returns together to variables an error message, that in the case of errors stop the execution.

An advantage of the conjugate gradient methods, in the Fletcher-Reeves version [4], is that (at least when it is used for linear optimization) it can be efficiently run on parallel computers, by computing the gradients and conjugated directions of separate groups of variables on different processors. This advantage persists also in the our version of the constrained optimization.

**Acknowledgement**

This work has been carried out within the framework of the EUROfusion Consortium as a complementary project and has been received funding from the Romanian National Education Minister / Institute of Atomic Physics under contract _1EU-2/2//01.07.2016. The collaboration with Calin Vlad Atanasiu is acknowledged.

# References

[1] L. E. Zakharov, C. V. Atanasiu, K. Lackner, M. Hoelzl, E. Strumberger, *Electromagnetic thin wall model for simulation of plasma wall touching kink and vertical modes*, J. Plasma Phys. **81**, 515810610 .

[2] Zhong-Ci Shi, *Nonconforming finite element methods*, Journal of Computational and Applied Mathematics **149** (2002) 221–225

[3] M. H. B. M. Shariff, *A constrained conjugate gradient method and the solution of linear equations*, Computers Math. Applic. Vol. 30, No. **11**, pp. 25-37, 1995

[4] Fletcher R. , Reeves C. M. , *Function minimization by conjugate gradient*, Comput. J. vol.**7**, pag 149-154 (1964)