Technical Article

Is it possible to solve a FEM static case without assembling and storing an *Ax* = *b* matrix system?

***Abstract* — In this paper we present a technique to solve magnetostatic problem FEM cases where the assembling, storing and solving the matrix system *Ax* = *b* is performed in a different way compared with the classical techniques used nowadays and for more than 50 years. It is a new scheme to calculate the node potential unknowns, much simpler than traditional techniques widely used in virtually all the FEM codes. With the proposed method, assembling and solving are considered in a single procedure and, if the computation time is not a concern, memory allocation is virtually unnecessary. We point out the difficulties related to this procedure, noticing that, in spite of some drawbacks, the new scheme is effective and provides reliable results.**

I. INTRODUCTION

The FEM applied to magnetostatic cases requires the calculation of the elemental contribution matrices (rigidity matrix and source vectors). The evaluation of such matrices is normally performed element by element and assembled in the global matrix system commonly designated *Ax* = *b*. Boundary conditions are inserted and the system is solved by a direct method (as Gauss Elimination) or by an iterative one (as ICCG) [1] [2]. When *Ax* = *b* is solved, the magnetic field **H** and the induction **B** are calculated. To do so, the steps of assembling, storing and solving are necessary.

On the other hand, when applying the FDTD method for, as example, wave propagation, it is possible to operate by “cells” around the nodes [3]. Inspired by this procedure, we investigate the way to act similarly for magnetostatic cases. The resulting procedure is iterative and relaxation techniques are applied on Gauss-Seidel solving method [1][16][22].

As our work progressed, we noticed that this technique is quite effective in terms of memory allocation and, moreover, its implementation is much simpler compared to the regular way of classical FEM codes. The main reason is the fact that it is not necessary to assemble, store and solve an *Ax* = *b* system as normally done, although we use the same final matrix terms and equations. As a matter of fact, such procedures are implicitly performed in the proposed technique and it is not necessary to proceed with the classical “calls” for the corresponding subroutines. At this stage of our work, the only drawback is the computation time which is somewhat higher than traditional implementations. It will be discussed in this paper.

For simplicity, in this work we call “*T scheme*” the *traditional* one (using assembling by elements) while calling “*N scheme*”, the *new* proposed one, which is based on *node* cells. In this paper, we will present some examples showing the effectiveness of the *N scheme* as well as the challenges that are inherent to it.

II. THE CLASSICAL FEM – *T SCHEME*

It is well established and known that quasi-static cases are normally solved with the main steps below:

```
- Do for elements:
        Calculate the elemental and source matrices;
        Assemble them in a matrix system $Ax = b$;
        Insert boundary conditions;
-Solve the $Ax = b$ system;
-Calculate fields.
```

Depending on the adopted implementation, the boundary conditions can be inserted during the assembling step. Calculating the elemental contribution and source matrices is normally simple and commonly described in the literature. The major difficulties for large cases (typically 3D problems) occur on the assembling, storing and solving the *Ax* = *b* system. Normally an iterative solver must be implemented and, even nowadays it can be found in the public domain, some steps must be accomplished as:

- renumbering the nodes to take into account only the unknowns;
- setting an addressing array to locate the non-zero terms;
- setting the pointers to the above array;
- allocating memory for the non-zero terms (taking care of the matrix symmetry);
- allocating memory for the pre-conditioner matrix;
- solving the *Ax* = *b* system;
- renumbering back the unknowns to match the original mesh numbering in order to calculate fields.

If, for any reason, one wishes to use another solver, the arrays above must be possibly readapted, which increases the burden. Of course, each implementation has its own characteristics and they may differ from each other, but anyone working on codes, knows very well that the FEM is considered as “complicated” and somewhat discouraging because of the points above.

III. THE PROPOSED TECHNIQUE – *N SCHEME*

The *N scheme* requires working by nodes and the “cells” of elements around nodes. Even though the major interest of this technique is for 3D cases, for didactical purposes let us consider the Fig. 1, where a very small and simple 2D mesh is presented.

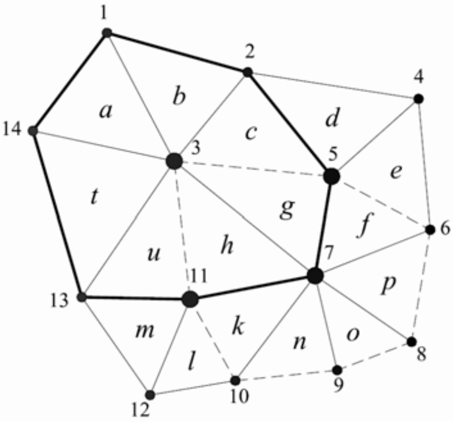


Fig. 1 – 2D mesh

In Fig. 1, the nodes are indicated by numbers while the elements by letters. Suppose that the nodes 3, 5, 7 and 11 are unknowns and the others (1, 2, 4, 6, 8, 9, 10, 12, 13 and 14) have imposed boundary condition values. Firstly, we work with the node 3. The corresponding elements are: *a, b, c, g, h, t* and *u* and this cell is indicated in thick lines (the cell of the node 7 is indicated with dashed lines). When calculating the elemental matrix for this element (a typical rigidity matrix for a linear 2D triangular element), we obtain an *a*(3,3) matrix; for the source we have a *s*(3) vector. Suppose that the potential unknown is called *v* and that the element *a* is the only one acting for the node 3. If the nodes numbering created by the mesher has the sequence 14, 3 and 1, the elemental matrix system given by this sole element is:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \begin{bmatrix} v_{14} \\ v_3 \\ v_1 \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} \qquad (1)$$

Supposing that the values of v_1 and v_{14} are known (as Dirichlet boundary conditions, for instance) we can write, for the potential v_3:

$$a_{2,2}v_3 = s_2 - a_{2,1}\,v_{14} - a_{2,3}v_1 \qquad (2a)$$

If the numbering sequence is 3, 14 and 1, the equivalent expression is:

$$a_{1,1}v_3 = s_1 - a_{1,2}\,v_{14} - a_{1,3}v_1 \qquad (2b)$$

and for the sequence 1, 14 and 3, we have

$$a_{3,3}v_3 = s_3 - a_{3,1}\,v_1 - a_{3,2}v_{14} \qquad (2c)$$

The above system ((2a), (2b) or (2c)) must interact with other elements surrounding the node 3 (which correspond to the regular assembling on the *T scheme*) and such equations are calculated for the all the elements of this cell. We proceed with the sum of all the diagonal terms related to the node 3. To generalize it, let us call "n" the unknown node for this cell (node 3) and "j" and "l" the other two nodes of the element (as 1 and 14 for the element *a*). The generic diagonal term is called $a_{\mathrm{n,n}}^{elem}$; the sum of the right hand side is defined as:

$$r_{\mathrm{j,l}}^{elem} = s_{\mathrm{n}} - a_{\mathrm{n,j}}\,v_{\mathrm{j}} - a_{\mathrm{n,l}}v_{\mathrm{l}}$$

The sum of the diagonal terms for the example above is

$$diag\_sum = a_{\mathrm{n,n}}^{a} + a_{\mathrm{n,n}}^{b} + a_{\mathrm{n,n}}^{c} + a_{\mathrm{n,n}}^{g} + a_{\mathrm{n,n}}^{h} + a_{\mathrm{n,n}}^{u} + a_{\mathrm{n,n}}^{t} \qquad (3)$$

For the right hand side we have

$$right\_sum = r_{\mathrm{j,l}}^{a} + r_{\mathrm{j,l}}^{b} + r_{\mathrm{j,l}}^{c} + r_{\mathrm{j,l}}^{g} + r_{\mathrm{j,l}}^{h} + r_{\mathrm{j,l}}^{t} + r_{\mathrm{j,l}}^{u} \qquad (4)$$

The node 3 has interaction with the unknown nodes 5, 7 and 11 through the common elements *c, g, h* and *u*. Since the procedure is iterative, the potential of these nodes will be placed on the right hand side with their current values (exactly as the boundary condition nodes). Once *diag_sum* and *right_sum* are calculated by (3) and (4), we can obtain the value of v_3 by:

$$v_3 = \frac{right\_sum}{diag\_sum} \qquad (5)$$

We will perform similar operations for the unknowns v_5, v_7 and v_{11}. Once it is done, we have a first approximation for these unknowns. Then the second iteration starts, with unknown values somewhat closer to the solution. As the iterative procedure progresses, the convergence is reached. The "influence" of boundary imposed conditions and the sources terms are slowly spread and, because we are dealing exactly with the same classical elemental matrices and sources, the solution is, in principle, exactly the same as we obtain with the *T scheme*. For starting the iterative procedure, the potential on the unknown nodes is imposed as zero.

IV. IMPLEMENTATION

The majority FE codes have all the necessary parts to implement the *N scheme*, except for a particular mesh table here presented. Generally, the pre-processors supply the array *nodes* by *elements* as shown in the Table I, for the example of mesh Fig. 1.

TABLE I. NODES BY ELEMENTS.

Element	Nodes
a	1,14,3
b	1,3,2
c	2,3,5
d	2,5,4
e	4,5,6
…	…..
u	11,3,13

We need, from the Table I, to set the arrays of the Table II. They indicate the number of elements surrounding a node (array NN) and the numbering of these elements (array EN), as presented below:

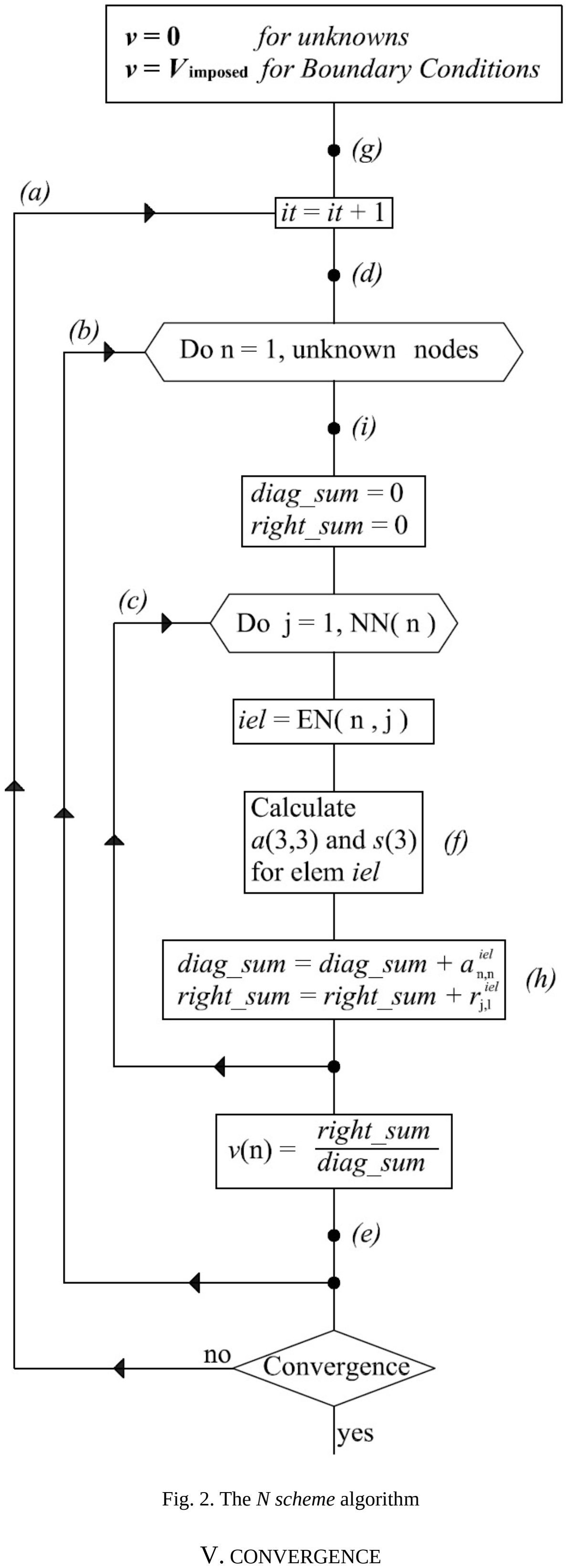
TABLE II. ELEMENTS BY NODES.

Node	Number of Elements (NN)	Elements (EN)
1	2	*a,b*
2	3	*b,c,d*
3	7	*a,b,c,g,h,t,u*
4	2	*d,e*
5	5	*d,e,f,g,c*
……	……	…..
14	2	*a,t*

Table II arrays are easily obtained from the Table I. In the Fig. 2 we present the main algorithm for working with the *N scheme*.

We remark some points in this algorithm:
- the loop *(a)* is set for the iterations necessary to reach the solution;
- the loop *(b)* "turns" only for the unknown nodes;
- the loop *(c)* takes into account the elements surrounding the node n.

From this simple algorithm, we observe the main contribution of this work: the potentials are calculated without the classical assembling, storing and solving steps. The solution is implicit in the proposed scheme.



v = 0 for unknowns
v = V imposed for Boundary Conditions
(g)
(a)
it = it + 1
(d)
(b)
Do n = 1, unknown nodes
(i)
diag_sum = 0
right_sum = 0
(c)
Do j = 1, NN(n)
iel = EN(n , j)
Calculate a(3,3) and s(3) for elem iel
(f)
diag_sum = diag_sum + a iel n,n
right_sum = right_sum + r iel j,l
(h)
v(n) = right_sum / diag_sum
(e)
no
Convergence
yes

Fig. 2. The *N scheme* algorithm

V. CONVERGENCE

The algebraic operations related to the proposed method are, strictly, identical to the Gauss-Seidel method ones [1]. The convergence with this method is slow and it is necessary to apply strong over-relaxation to reach the solution [1][16][17][18][19][20][21][22][27]. For applying it, let us define, for a node n, the difference between the potential v_{n}^{it} at the current iteration *it* and v_{n}^{it-1} at the previous one.

$$dif\ v_{\mathrm{n}} = v_{\mathrm{n}}^{it} - v_{\mathrm{n}}^{it-1} \qquad (6)$$

The relaxation procedure, in code language, is defined by

$$v_{\mathrm{n}} = v_{\mathrm{n}} + (w-1)\ dif\ v_{\mathrm{n}} \qquad (7)$$

where *w* is the relaxation factor. We prefer to work directly with the factor $R = w-1$ and the above expression becomes

$$v_{\mathrm{n}} = v_{\mathrm{n}} + R\ \ dif\ v_{\mathrm{n}} \qquad (8)$$

Here, if *R=0*, where is no relaxation. For $0 < R < 1$, we have over-relaxation and, for $-1 < R < 0$, under-relaxation. With the examples presented below, it is necessary to use $0.8 < R < 0.99$. As a matter of fact, it is difficult to determine the relaxation coefficient on FE, and, after tests, we noticed that the following expression for R is quite effective:

$$R = R_{final}(1-e^{-it/T}) \qquad (9)$$

where *it* is the iteration number. A good compromise is $R_{final} = 0.96$ and *T=14.11*. Therefore, the curve for *R* as function of the iterations is shown in the Fig. 3. Using these data, for the *20th* iteration, we have *R=0.75*. This equation is used because the potential variations are already quite large at the first iterations and become smaller as the iterative procedure progresses.

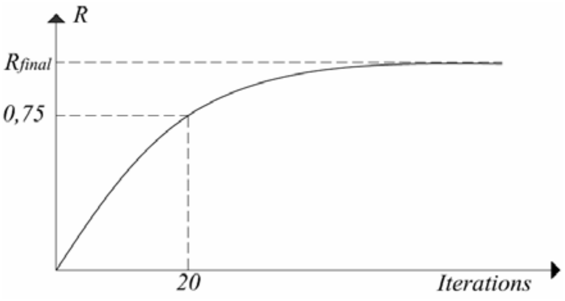


Fig.3 – Relaxation factor *R* as function of the iterations

In the algorithm of Fig. 2 at the point indicated by the letter *(d)*, *R* is calculated according to the equation (9), Fig. 3. *R* is then employed (equation (8)) on the point *(e)* indicated in the Fig. 2. Applying over-relaxation is very simple and the computational effort to perform it is minimal. Without it, the convergence is hardly reached. As criterion for stopping the iterations, a relative error smaller than 1.e-5 is required for the unknown nodes.

The proposed technique has some similarities with the "frontal" methods, commonly used in the 1970s [24][25][26]. However, these methods require partial assembling using direct solvers (as Gauss Elimination) [23]. In that time, memory allocation was the main concern. Here, such a assembling is not necessary and the Gauss-Seidel iterative solver is used.

VI. STRATEGIES AND EXAMPLES

Many 2D and 3Ds magnetostatic cases were treated in our studies with success, even though, in some cases, the numbers of iterations were quite high. No failure was detected in any case and all the induction distribution plots presented below were obtained with the *N scheme*.

VI.1 An Electrical Machine (2D Vector Potential)

In the Fig. 4 we have a 2D mesh of an electrical machine. This machine has a thin airgap. The number of elements and nodes are 8305 and 4387, respectively. The classical magnetic vector potential formulation is used to solve it [6]. In the Fig. 5 we have the magnetic induction distribution obtained.

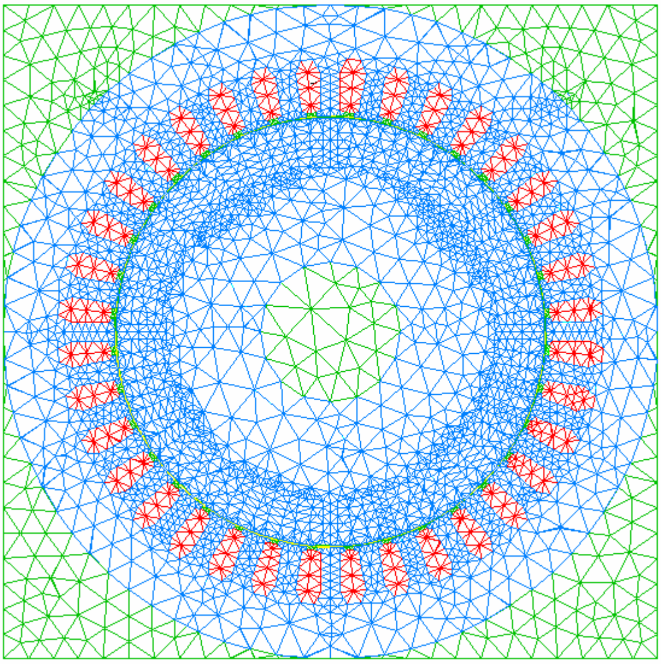


Fig.4 – Mesh of an electrical machine with thin airgap

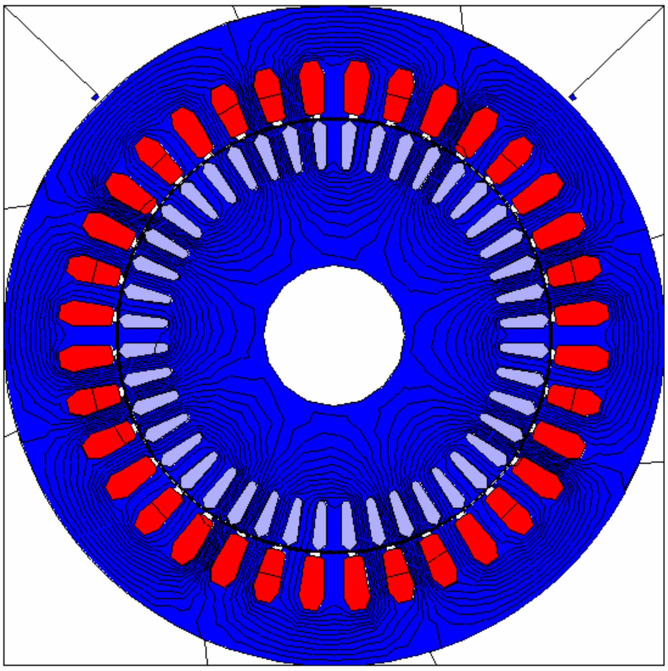


Fig.5 – Magnetic induction distribution

For all the examples, the curve of R related to the expression (9) is used and here the number of iterations is 876. More detailed analyses, comparisons and discussions will be presented in Tables III and IV at the end of this section.

VI.2 A 3D linear case (source-field method)

In the following examples, we use the "source-field" method. It is not our purpose to describe it here. Its basis are discussed in [7] and its full presentation is available in [8][9]. We just mention that, with this method, the coil current density $\mathbf{J}$ is replaced by $\mathbf{H}_\mathbf{s}$ using the equation $rot\,\mathbf{H}_\mathbf{s} = \mathbf{J}$ [15]. The magnetic field $\mathbf{H}$ is given by $\mathbf{H} = \mathbf{H}_\mathbf{s} - grad\,\Omega$. The main equation to be solved is $div\,\mathbf{B} = div\,\mu\,(\mathbf{H}_\mathbf{s} - grad\,\Omega) = 0$, where μ is the magnetic permeability and Ω is the reduced scalar potential [5]. Working with first order tetrahedra, the elemental matrices *a*(4,4) and *s*(4) are, with classical notations [2]:

$$a_{\mathrm{i,j}} = \int_{Vol_n} \mu\, grad\, N_i^t \cdot grad\, N_j \;\, dv \qquad (10)$$

$$s_{\mathrm{i}} = \int_{Vol_n} grad\, N_i^t \cdot \mu\; \mathbf{w}\, H_{sc} \;\, dv \qquad (11)$$

where N is here the nodal shape function and $\mathbf{w}$ is the edge shape functions of the tetrahedron n of volume Vol_n; H_{sc} is the circulation of the source-field on the edges of the element.

When dealing with large meshes, the evaluation of elemental matrices requires numerical integration and are, therefore, time consuming. The algorithm of Fig. 2 can be slightly modified to save computation time. The elemental matrices can be calculated outside the loop *(a)* and can be kept in memory by using the three-dimensional array *a*(elem, 4,4) and *s*(elem,4). Notice that, in the step *(f),* inside the internal loop *(c)*, the matrices *a*(4,4) and *s*(4) are, normally, calculated 4 times, since each tetrahedron has four unknown nodes (excepting few boundary conditions nodes). Then, we perform the calculations of the block *A* shown in the Fig. 6 below:

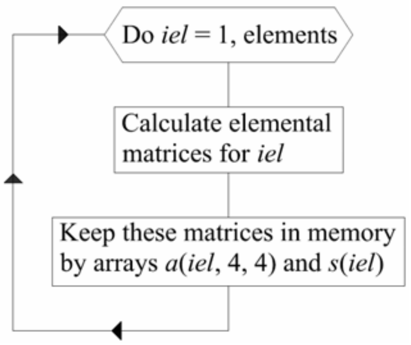

Do *iel* = 1, elements
Calculate elemental matrices for *iel*
Keep these matrices in memory by arrays *a*(*iel*, 4, 4) and *s*(*iel*)

Fig. 6 – Block *A* for elemental matrices kept in memory

In the main algorithm of Fig. 2 the step *(f)* is suppressed and the block *A* above is inserted in the point *(g)*, outside the iterations loop *(a)*. The expressions (3) and (4) can be applied directly in *(h)* using the three-dimensional array *a*(iel, 4,4) and *s*(iel,4). It saves much computation time. This procedure needs memory allocation but in a much simpler way (block *A*, Fig. 6) than the classical assembling, as commented in the beginning of this paper.

The 3D domain, shown in the Fig. 7, has 7895 nodes and 40615 elements. It corresponds to ¼ of a "U" magnetic circuit domain. It is necessary 140 iterations to reach convergence.

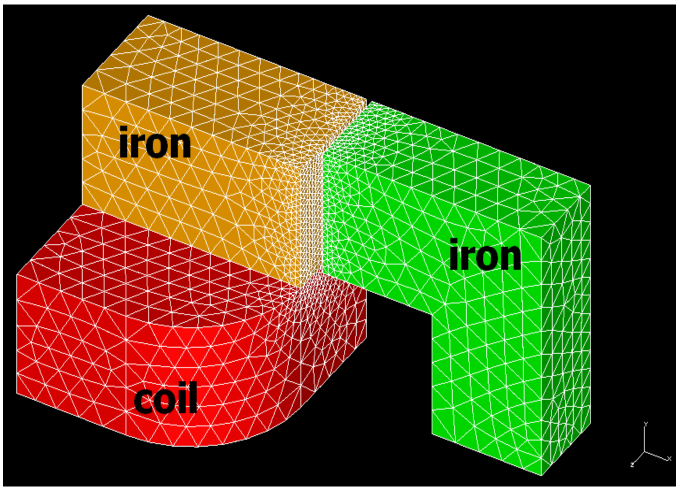


Fig.7 – Mesh of a linear 3D case

VI.3 A 3D non-linear case (source-field method)

The same domain of the previous example is used here to solve a non-linear case. The iron has a typical B(H) with saturation induction around 1.6 Tesla [6].

Using our proposed *N scheme*, we point out another relevant advantage compared with the classical *T scheme*. The value of the permeability can be modified along the iterations, in the loop *(a)*. When using an iterative method (as ICCG) in the regular *T scheme*, the whole matrix *A* can not be modified during the iterations procedure. In this case, the block *A* presented in the previous example (for *a*(iel,4,4) and *s*(iel,4)) "moves" from the point *(g)* to the point *(i)* in Fig. 2. Again, *(f)* is excluded. We obtained convergence in 175 loops, a good performance.

Observing the algorithm, there is one more possible simplification for non-linear problems, which will speed up the calculation: the elemental contributions (10) and (11) can be calculated without considering the permeability μ. In the loop *(b)* at the point *(i)*, the fields on all the elements are evaluated as well as their permeability. μ is then introduced on expressions (3) and (4) (step *(h)* of Fig. 2). It is faster than calculating the whole the elemental contributions (it needs numerical integrations). The Fortran code of this modified algorithm is indicated at the end of this paper.

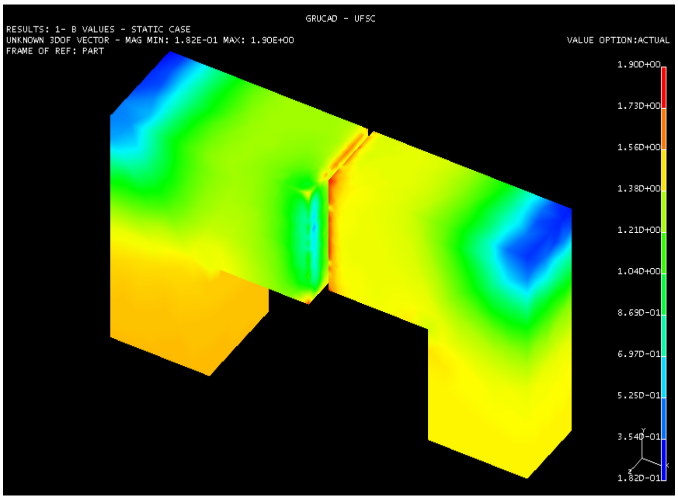


Fig.8 – Mesh and magnetic induction distribution in a non-linear case

As for this non-linear case, the magnetic induction distribution is plotted in Fig. 8.

VI.4 A 3D linear case with permanent magnets (source-field method)

Finally, the aim of this example is to demonstrate that the proposed scheme can be applied in a relatively large mesh. This case is related to an axial flux electrical motor having coils and permanent magnets [28]. The same source-field method is employed. As for the permanent magnet source, the source term is given by [8]:

$$sp_i = \int_{Vol_n} gradN_i^t \cdot \mathbf{B_r} \; dv \qquad (12)$$

where $\mathbf{B}_r$ is the permanent magnet remnant induction. The corresponding source term *sp*(iel,4) must be added to the coil source term *s*(iel,4). The mesh of this case has 63053 nodes and 361007 elements; it is shown in the Fig. 9, where the superior iron piece is removed to show the coils and the permanent magnets. The magnetic induction distribution is shown in Fig. 10a (for the permanent magnet only) and Fig 10b (permanent magnets and active coils).

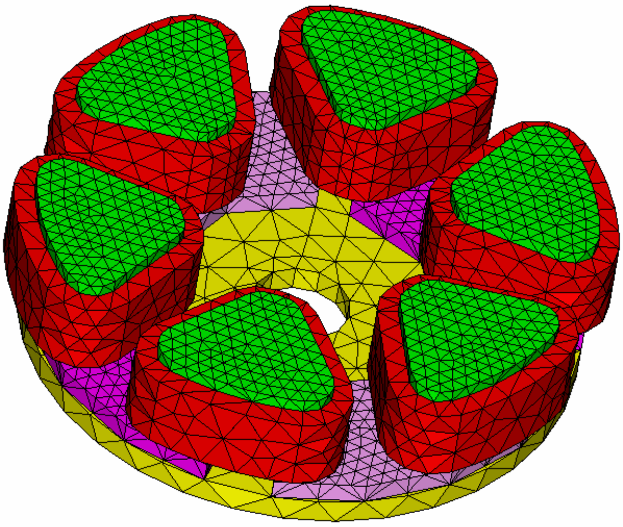


Fig.9 – mesh of a 3D case having coils and permanent magnets (permanent magnets in purple and coils in red).

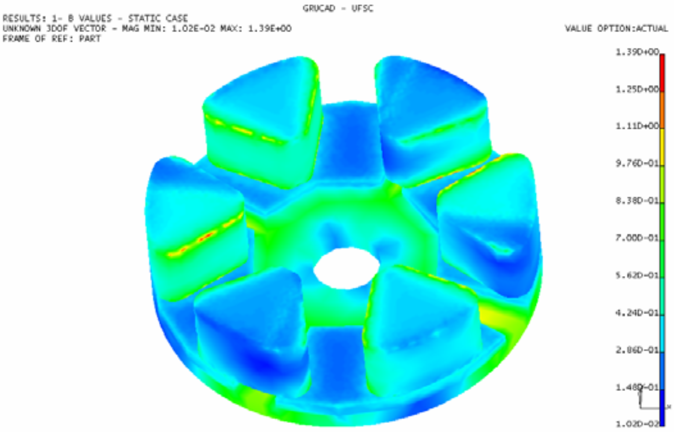


Fig 10a. Magnetic induction distribution due to the permanent magnets.

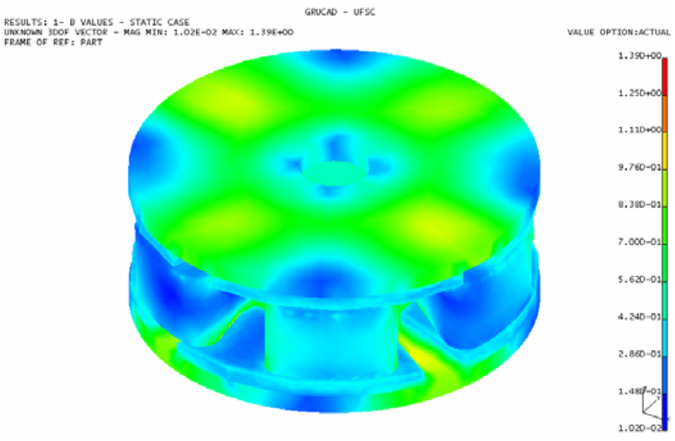


Fig 10b. Magnetic induction distribution due to coils and permanent magnets (iron top piece also shown).

VI.5 Results and discussion

In this section we will present some comparisons between the *N* and the *T schemes*. We point out that the *T scheme* solvers, belonging to the FEECAD 3D system, have been regularly used and tested [10][11][12]. Therefore, for this comparison, we consider its results as correct. The ICCG method is used on the *T scheme* solvers [29].

TABLE III. *T SCHEME* DATA

Case	Mesh	ICCG iter.	Magnetic energy (J)	Time (sec)
Ex. VI.1: 2D motor	4387 nodes 8305 elem.	522	0.9951	0.4
Ex. VI.2: 3D (linear)	7895 nodes 40165 elem.	46	0.040825	1.3
Ex. VI.3: 3D (non-linear)	7895 nodes 40165 elem.	46 (x 10 iter. non-linearity)	0.41258 0.51914 (co-energy)	10.9
Ex. VI.4: 3D PM motor	63053 nodes 361007 elem.	91	.034537	10.2

TABLE IV. *N SCHEME* DATA

Case	iterations	Magnetic Energy	Time (sec)	Energy Error %	|B| Error%
Ex. VI.1	876	0.9957	9.3	0.06	0.08
Ex. VI.2	140	.040826	4.6	0.002	0.37
Ex. VI.3	175	0.41258 0.51925	28.7	0.00 0.02	0.27
Ex. VI.4	1317	0.033891	380	1.8	0.34

We compared the effectiveness of the *N scheme* based in two quantities: the magnetic energy and the |**B**| average relative errors. The latter is obtained by calculating, element by element, the errors on **B** magnitude and averaging them. It is clear that the *N scheme* conducts to correct solutions, a relevant result of this work. One interesting point: the non-linearity is very well solved with the *N scheme* and it provides very accurate fields.

Initially, the case presenting major difficulties, among all our tests (not only the cases here presented) is the Example VI.1. It is the 2D electrical motor, whose thin air-gap leads to these difficulties. Even for the *T scheme*, ICCG needs the largest *ratio=(number iterations)/(number nodes)* of our tests. It is also observed with the *N scheme*. In this example, the algorithm presented in the Fig. 2 was used, without considering the elemental contributions storage (the use of the Block *A* of Fig. 6). The use of this storage would speed up the calculation.

The drawback of the *N scheme* is the fact that the number of iterations is larger compared to the *T scheme*. Also a single iteration on the proposed method, in spite of the scheme simplicity, is longer than an ICCG iteration. Therefore, the computational time is larger, although, in absolute terms, they are quite reasonable, excepted for the last example, whose main goal is to show that the *N scheme* can handle large meshes. Notice also that such calculation times were obtained with an average performance notebook (AMD Turion 64 Mobile, 2.00 GHz, 1GB RAM).

This particular timing comparison must be taken with the precautions commented now. We are not comparing similar techniques. The complex tasks of the *T scheme* (commented in the beginning of this work) are not present in the *N scheme*. Secondly, we must observe some historical aspects. In the 1970s and 1980s, when 3D calculations become recurrent in our area, an immense effort was devoted to improve the performance of $Ax = b$ solvers. Impressive papers were presented and proposed outstanding algorithms to solve $Ax = b$ systems with very high precision, fast calculation and low number of iterations. We mention in the bibliography section two landmark papers, but certainly dozens of papers could be listed [13][14]. This area is still an important trend of research and more work has been accomplished on it for about 30 years. Our proposed *N scheme* is, at our best knowledge, a new way to solve FEM problems and, as explained in this paper, it conducts to numerical operations similar to the Gauss-Seidel procedure whose performance is quite equivalent to the Conjugate Gradient method without preconditioning [1]. Clearly, it is an open research area whose challenge is to improve the convergence of Gauss-Seidel method (or another possible alternative) keeping the simplicity of the *N scheme*. We estimate that improvements on this particular problem would be beyond the scope of this single work. Nevertheless, with the relaxation technique here presented the *N scheme* is already totally applicable. It is also robust: different initial values for the unknown nodes were tried and the final result was virtually the same, as well as the number of iterations. And, certainly, the community working in numerical analysis and particularly on solvers may provide improvements on this direction. In our research group, we will keep working on this particular topic and hopefully, some progress will be accomplished by us or colleagues.

VII. FINAL CONSIDERATIONS

We presented in this work a totally different way to handle FEM static cases. With the *N scheme* it is possible to handle FEM cases with a technique immensely simpler compared with the traditional FEM method.

In the discussion above presented, it was pointed out that some improvements, upon effective research, can be proposed in the future and, as said, this matter is an open research subject.

Some other aspects can be also noticed:

- the *N scheme* handles, without any distinction, symmetrical and non-symmetrical *A* matrices;
- a new and effective way to handle non-linearity is possible; using it, the permeability can be changed during the iterations;
- parallelization techniques may be also considered in a different way.

It can certainly be also applied on quasi-static cases and formulations based on vector potential. It will be investigated in a near future.

We believe that *N scheme* is an elegant way to handle FEM cases compared to the classical method. This statement is based on its overwhelming simplicity on implementing codes and on the fact that handling the $Ax = b$ system is much easier. We would say that with the traditional method, once the elemental matrices are calculated, the “most difficult part has to be done”. With the proposed *N scheme*, once such matrices are obtained, “it is already done”. The results are encouraging and, at this stage of our work, already satisfying and fully applicable.

Finally, for colleagues interested on trying this method, we provide the Fortran code of the algorithm of Fig. 2 (with the modifications presented in the section VI) by the Internet access:

www.grucad.ufsc.br/download/nscheme.txt

Its reading is certainly helpful for clarifying the method implementation.

VIII. References

[1] Bastos, J.P.A., *“Calculation of magnetic fields by 2D and 3D FEM – Contribution to the determination of characteristics of a variable reluctance motor”*, Docteur d’Etat thesis, Paris VI University, 1984 (*in French*)

[2] Ida, N., Bastos, J.P.A, *“Electromagnetics and Calculation of Fields”*, edited by Springer-Verlag, 1997

[3] Taflove, A., Brodwin, M.E., “Numerical solution of steady-state electromagnetic scattering problems using the Time-Dependent Maxwell’s equations”, *IEEE Trans. On Microwave Theory and Techniques*, 1975, Vol MTT-23, pp 623-630

[5] Binns, K.J., Lawrenson, P.J., Trowbridge, C.W., *“The analytical and numerical solution of electric and magnetic fields”*, John Wiley and Sons, 1992

[6] Bastos, J.P.A., Sadowski, N., *“Electromagnetic modeling by Finite Element Methods”*, edited by Marcel Dekker, 2004

[7] A. Bossavit: "Whitney forms: a class of finite elements for three-dimensional computations in electromagnetism", *IEE Proc., 135, Pt. A*, 8 (1988), pp. 493-500.

[8] Le Menach, Y., Clénet, S., Piriou, F., “Determination and utilization of the source-field in 3D magnetostatic problems”, *IEEE Trans. on Magnetics*, 1998, Vol. 34 (5)

[9] Dular, P., Robert, F., Remacle, J.F., Umé, M., Legros, W., “Computation of the source current density in inductors of any shape using a mixed formulation”, *3rd International Workshop on Electrical and Magnetic Fields*, 1996, pp 107-112

[10] Bastos, J.P.A., Ida, N., Mesquita, R.C., "Problem 10: a Solution using Personal Computers", *TEAM workshop, Proceedings*, 1994, pp 63-64

[11] Bastos, J.P.A., Ida, N., Mesquita, R.C., "Problem 13: a Solution using Personal Computers", *TEAM workshop, Proceedings*, 1994, pp 65-66

[12] Bastos, J.P.A., Ida, N., Mesquita, R.C., "Problem 20: a Solution using Personal Computers", *TEAM workshop, Proceedings*, 1994, pp 71-72

[13] Mejerink, J.A., va der Vorst, H.A., “An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix”, *Mathematics of Computation*, 1977, pp 148-162

[14] Saad, Y., Schultz, M.H., “GMRES: a generalized minimal residual algorithm for solving non symmetrical linear systems”, *SIAM J., Sci.Stat. Comput.*, 1986, pp 856-869

[15] Z. Ren, 'Auto-gauging of vector potential by iterative solver - numerical evidence', *3rd International Workshop on Electric and Magnetic Fields*, Liege-Beligium, may, 1996

[16] Southwell, “Relaxation methods in theoretical Physics”, *Oxford U.P.*, 1946, Vol I and II

[17] Stoer, J., Burlisch, R., *“Introduction to numerical analysis”*, Springer-Verlag, 1980

[18] Hasting, D.A., “Computer methods for electric potential and other field problems”, *American Journal of Physics*, 1975, 43-6, pp 18-24

[19] Carpenter, C.J., “Numerical solution of magnetic fields in the vicinity of current-carrying conductors”, *Proc. IEE.*, 1967, 114-11, pp 1793-1800

[20] Erdely, E.A., Fuchs, E.F., “Fields in electrical devices containing soft nonlinear materials”, *IEEE Trans on Magnetics*, 1974, 10-4, pp 1103-1108

[21] Lees, M.J., Tindall C.E., “ Field-theory analysis of saturation harmonics in induction machines”, *Proc. IEE*, 1974, 121, pp 276-279

[22] Varga, R.S., “*Matrix iterative analysis*”, Prentice-Hall (Englewood Cliffs, N.J.), 1962

[23] Krizek, M., Neittaanmaki, P., Stenberg, R., “Finite Element methods: fifty years of the courant element”, CFC, 1994

[24] Courant, R., “Variational methods fpr problems of equilibrium and vibration”, *Bull. AMS*, 1943, 49-1, pp 1-23

[25] Irons, B.M., “A frontal solution program for finite elements analysis”, *International Journal for Numerical Methods in Engineering*, 1970, 2-1, pp 5-32

[26] Ida, N., Lord, W., “Solution of linear equations for small computer systems”, *International Journal for Numerical Methods in Engineering*, 1984, 20, pp 625-641

[27] Gortler, S., Cohen, M.F., Slusallek, P., “Radiosity and relaxation methods”, *IEEE Computer Graphics and Applications*, November 1994

[28] Gieras, J.F., Wing, M., “*Permanent magnet motors technology; design and applications”*, edited by Marcel Dekker, 2002

[29] Mesquita, R. C., “Conjugate gradients method with preconditioning on the solution of equations system created by finite elements”, *Proceedings of the Congresso Brasileiro de Eletromagnetismo Aplicado*, 1992, pp. 165-174 (*in Portuguese*)

Acknowledgment

We are sincerely grateful to Alain Bossavit, Zhuoxiang Ren and Patrick Dular who kindly accepted to discuss this subject. Several remarks and fruitful suggestions coming from these colleagues are reflected in this text.

Authors name and affiliation

J.P. A. Bastos,
GRUCAD, Universidade Federal de Santa Catarina, Florianópolis, SC, 88040, Brazil, phone: 55-48-3721 9649,
email: jpab@grucad.ufsc.br