MNT23D: A Graphical User Interface for Formulation and Element Matrix Assembling in the Finite Element Method

K. Z. Nóbrega, *IEEE Student Member,* A. M. Frasson, H. E. Hernández-Figueroa, *IEEE Senior Member*
UNICAMP-FEEC-DMO , Albert Einstein Avenue 400 Caixa Postal: 6101 CEP: 13083-970

***Abstract* — This paper discusses an integrated Finite Element Modeling package to derive an to analyze two and three-dimensional formulations of Maxwell's equations governing classical electromagnetic propagation. In fact, the interactive developer here presented converts the point-wise partial differential operators to equivalent scalar integral operators, retrieves results, stores them in an external database, and finally calculates and exports the element matrices, necessaries to the Finite Element Method (FEM), in pre-established programming language formats, such as FORTRAN and C. The element matrices are calculated in an exact and efficient manner. Our developer takes into account linear and quadratic nodal elements, which can be triangles (2D) or tetrahedrons (3D). For the former, line and surface integrals are considered, while for the latter, surface and volume integrals are computed. This interactive developer is intended to alleviate the considerable programming effort normally demanded by the FEM, which is a powerful numerical tool widely used in engineering and applied sciences. Therefore, this developer may be beneficial for either professional and educational FEM activities. Here, the usefulness of the present developer is demonstrated through key electromagnetic examples.**

***Index Terms* — Finite Element Method, MATLAB, Symbolic Calculation, Education, Electromagnetics.**

I. Introduction

Technological advances in computing technology have made a variety of tools available for solving complicating numerical problems in engineering, applied and earth sciences. Certainly, two of the most significant numerical methods available for that purpose is the FEM and the Finite Differences.

Currently, the FEM has been widely used and its fundamentals explored and modified to develop new and powerful numerical techniques. In fact, there are number of finite element analysis (FEA) package available [1]-[3], they are either bundled with solid modeling tools or distributed as stand-alone packages. A first problem with FEA software is that it can be quite complicated to use and most of the time engineers who have some knowledge of finite element theory are needed to use them and interpret the results. As a second point, these software packages are generally too specialized and expensive, not being an interesting tool to that people who would like to develop their own programs, formulations and improve their capabilities on the FEM.

K. Z. Nóbrega, bzuza@dmo.fee.unicamp.br, A. M. Frasson, frasson@dmo.fee.unicamp.br, H. E. HernándezFigueroa, hugo@dmo.fee.unicamp.br , Tel +55-19-37883735. This work has the financial support of FAPESP Project 00/04593-6, and CNPq.

The MNT23D (Mounting Nodal formulations on 2 and 3 Dimensions) graphical user interface (GUI) provides a common interface to the user who would like to elaborate their own formulation (be scalar or vectorial) and after to perform an exact computer calculation of the element matrices in the FEM. To do this, we have made use of the mathematical and visualization capabilities of MATLAB® that are tightly integrated into our developer, simplifying the data analysis and the presentation of the results.

The programs available under the MNTNOD23D interface have been naturally divided into two categories: sources codes to manipulate and to become discrete vectorial and scalar operators (inner product, curl, gradient, divergent, and so on), making possible to the user create his own formulation and operators; the other code calculates by itself the so-called element matrices (EM's), once there are already defined the matricial operators desired. As a point of fact, both programs can run independent from each other, e.g., it is not necessary to create operators to calculate the EM's. It will become clear with the examples shown later.

In the remainder of this paper, we will briefly describe the basic theory of the FEM which makes possible to discuss, by using of some examples, the capabilities of the codes executed by MNTNOD23D. Some of the examples will be integrated, so that the output of one program will be the input for another program, making explicit the MNT23D's high level of integration. Finally, we will discuss the main characteristics of our developer, showing that it is possible to use it not only in electromagnetic's problems, but in other fields also.

II. Nodal FEM Formulation

Different kinds of elements are available in the FEM formulation, like rectangular, quadrilateral, rectangular bricks, hexahedral, tetrahedral and so on. However, in this paper we summarize the study only for the two main elements that are widely used in electromagnetics, optics and microwaves, i.e., triangular and tetrahedral elements, for 2D and 3D respectively.

To solve an equation with FEM it is necessary to define an equivalent integral formulation. It can be done using a variational method or the Galerkin Method. In this paper will focus our attention only on the latter one.

In a first step, a differential equation like (1) must be defined

$$\mathrm{L}\Phi=\mathrm{f} \tag{1}$$

where L is the differential operator, Φ is the field and f the source.

Applying the Galerkin formulation to (1) the following equivalent integral formulation is obtained,

$$\int_{\Omega}\mathrm{L}\Phi\,.\;w\,d\Omega=\int_{\Omega}\mathrm{f}\,.\;w\,d\Omega \tag{2}$$

where w is the weight or test function and Ω is the domain of the problem.

In the FEM, the problem's interior domain is partitioned into a number of logically regular and contiguous subdomains called *elements* (Ω_i), see Fig. 1, and the field is approximated by an expansion of the form

$$\Phi=\sum_{j=1}^{N}c_j\phi_j \tag{3}$$

ϕ_j being a set of basis functions, N the total number of nodes and c_j the unknown coefficients defined over the triangle or tetrahedral nodes. Notice, that a nodal formulation is adopted here, therefore, it does not apply for edge elements or vector formulation.

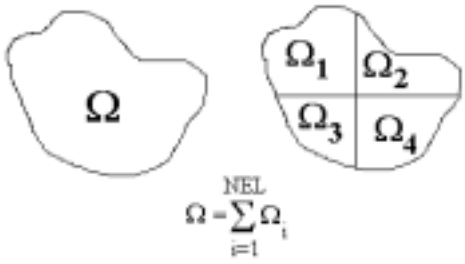

Fig.1 – Ω-Domain 's discretization.

Substituting (3) in (2) we find the expression

$$\sum_{j=1}^{N}c_j\int_{\Omega}\mathrm{L}\phi_j\,.\,w\,\mathrm{d}\Omega=\int_{\Omega}\mathrm{f}\,.\,w\,\mathrm{d}\Omega \tag{4}$$

which is one equation of N unknowns. To get a set of N independent equations one can take the same set basic functions for the weight function, i.e.,

$$\omega=\phi_i \tag{5}$$

Rewriting the left side of (4) as follows,

$$\int_{\Omega}\mathrm{L}\phi_j\,.\,\phi_i\,\mathrm{d}\Omega=\sum_{e=1}^{NEL}\int_{\Omega e}\mathrm{L}\phi_j^e\,.\,\phi_i^e\,\mathrm{d}\Omega_e \tag{6}$$

In (6) ϕ_j^e is the part of ϕ_j in the element e, in such a way that,

$$\phi_j=\sum_{e=1}^{NEL}\phi_j^e\quad .$$

The integrals over the domain Ω_e are called *element matrices*. These matrices are represented in (6) by

$$\int_{\Omega e}\mathrm{L}\phi_j^e\,.\,\phi_i^e\,\mathrm{d}\Omega_e\quad . \tag{7}$$

It is important to emphasize that such integrals can spend some time because it is common the presence of complex operators. Only with the purpose to facilitate the numerical integration, a transformation of a given integral defined on the domain Ω_e, to another on the domain Λ. The domain Λ is called the *master element*. As a matter of fact, no transformation of the physical domain is involved in the finite element analysis.

The transformation between Ω_e and Λ (or equivalently, between $(x,\ y,\ z)$ and $(\xi\ \eta,\ \theta)$) is accomplished by a coordinate transformation of the form

$$\begin{aligned}x&=\sum_{j=1}^{m}x_j^eL_j(\xi,\eta,\theta)\\ y&=\sum_{j=1}^{m}y_j^eL_j(\xi,\eta,\theta)\\ z&=\sum_{j=1}^{m}z_j^eL_j(\xi,\eta,\theta)\end{aligned}\quad , \tag{8}$$

where L_j denotes the well known area or volume coordinates [4] and m an index that can be 3 or 4, depending on the domain of discretization.

In typical FEM problems, the integrand in (6) usually has differential operations in x, y or z , the so-called global derivations. In the way it is important to define and make explicit the relation

$$\begin{bmatrix}\partial\phi^e/\partial\xi\\ \partial\phi^e/\partial\eta\\ \partial\phi^e/\partial\theta\end{bmatrix}=[J]\begin{bmatrix}\partial\phi^e/\partial x\\ \partial\phi^e/\partial y\\ \partial\phi^e/\partial z\end{bmatrix}\quad , \tag{9}$$

where *[J]* is known as the Jacobian matrix and given by

$$[J]=\begin{bmatrix}\partial x/\partial\xi & \partial y/\partial\xi & \partial z/\partial\xi\\ \partial x/\partial\eta & \partial y/\partial\eta & \partial z/\partial\eta\\ \partial x/\partial\theta & \partial y/\partial\theta & \partial z/\partial\theta\end{bmatrix}^e\quad . \tag{10}$$

Once defined the global differentiation it is also as important as to define the integral relation

$$\iiint_{\Omega e}f(x,y,z)dxdydz=\iiint_{\Lambda}f(\xi,\eta,\theta)\left|J(\xi,\eta,\theta)\right|d\xi d\eta d\theta \tag{11}$$

where $|J(\xi,\eta,\theta)|$ is the determinant of the Jacobian's matrix, known as the Jacobian. Equation (11) has implicit and important characteristics:

Usually, $f(x,y,z)$ has an integrand that gets double differentiation. However, for the usual linear approximation of the basic functions, ϕ_j^e, this operation is not possible. In this way, it is necessary to convert the point-wise partial differential operator to an equivalent but "weaker" scalar integral operator admitting lower order derivatives. It is done using integration by parts, which rewrite the double differentiation in a product of two first differentiations applied simultaneously in both weight and basic functions. Because of it, it is introduced a new integrand defined along the boundary (Γ_e) of Ω_e. Thus, for the 2D domain, it is possible an integral of line along the edge of an element, and for the 3D domain, this is an integral along a surface.

As the second point, there is the value of $|J(\xi,\eta,\theta)|$. In fact, $|J|$ is twice the triangle area, i.e., $2A_e$, and six times the tetrahedral volume ($6V_e$) (both defined in Ω_e) for the 2D and 3D domain, respectively. However for the boundary Γ_e, $|J|$ is equal to the length of the edge of the triangle for the 2D domain, and two times the face's area, $2A_f$, for the 3D domain [1] .

III. MNT23D FUNDAMENTALS

Concerning the theoretical description of FEM, our developer is devoted to the calculus of (1) and (7). In fact, from (1) we expect to give to the user the mobility to create and/or to manipulate vectorial and scalar operators, in such a way that after that it is possible to evaluate the integral operators, represented by (6), for the simplified field basis, giving an algebraic system of equations on the nodal field vector and its first-order derivatives.

A customized FEM package intended for designing has to be developed to provide fast and applicable design in an user-friendly environment. Thinking about it, we have used the MATLAB, a well known interpreter used in engineering, physics, chemistry, etc., and its Symbolic Toolbox that became possible the high level of integration.

Using MATLAB's graphical functions that integration was facilitated. It included all information associated with program's presentation, the dataflow, designing, numerical calculations, etc. In this way, we have done a GUI which modularize the process. The two basic's layouts of our developer are illustrated on Fig. 2.

The Illustration of Fig. 2 leads someone to have a main idea about how our developer works. The develop is shared in two basic GUI's. Beginning with Fig. 2a, one has a series of vectorial operators, unit vectors, generic operations, etc. in such a way that it is possible to manipulate two and three dimensional vectorial operators (like curl, divergent, inner dot, transverse curl, transverse divergent) and/or scalar ones such as gradient, transverse gradient, multiplication, difference, sum, with these operation applied in unit vector, basis field, weight function, and multiplication by tensors or scalar constants.

Once it was previous done (or not), one is able to import (or not) such operators exported in a file, into the format .KF, and use these results to calculate the EM's in a very fast and exact way. The matrices are calculated with the second GUI (see Fig. 2b). Now, for the sake of brevity, we will not explain all the functions presented in those layout, but instead of it only the most important characteristic we will presented.

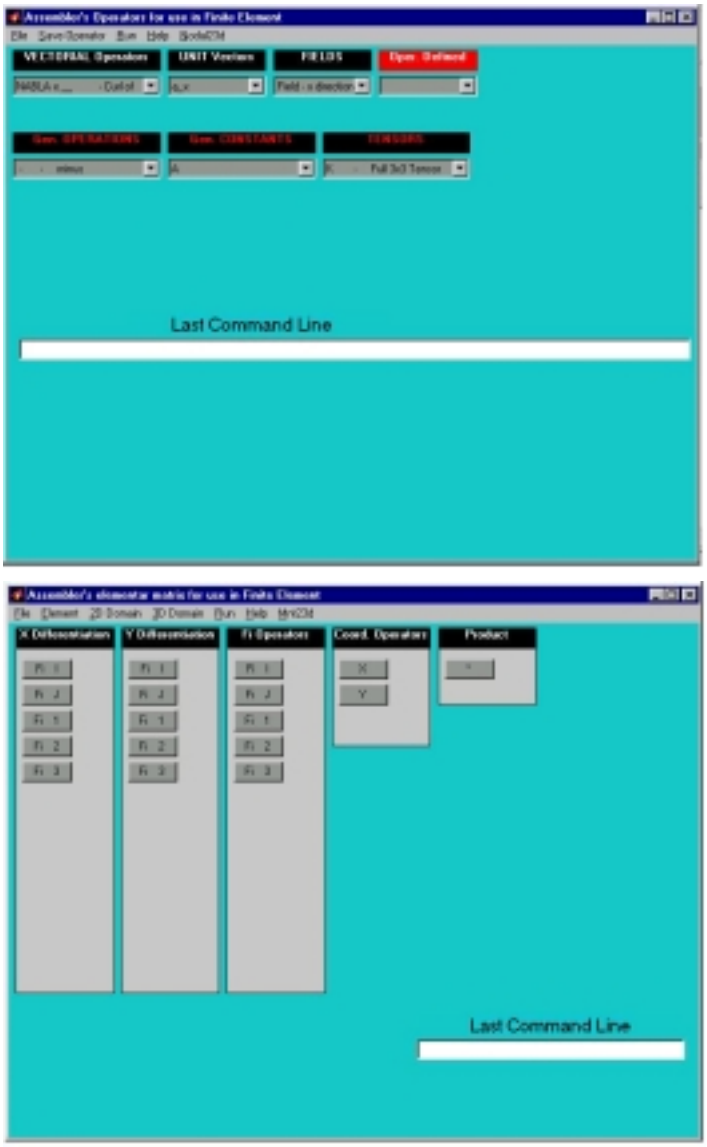

Fig. 2. The MNT23D's Layout. The top figure shows the program which manipulates the operators. The button figure is the element matrix's assembler.

Both interface are point-and-click and shows the last command line executed by the user, which warn a

possible mistake. Also, if some incompatible operation is done, the programs stop their execution printing or not an error message. After the execution is done, the programs print a message in the screen to the user.

Concerning the EM calculation, the developer works with triangular and tetrahedron nodal elements for 2D and 3D domains, respectively. The user also has the possibility to set an appropriate accuracy order for those elements, choosing between linear or quadratic basis functions. A third important characteristic is the capability to evaluate line and surface integrals for 2D and surface and volume integrals for 3D domains, which are specially helpful when the complexity of the mathematical model increases. The analytical by-hand computation of these integrals may be cumbersome and induce to erroneous results, so, the present developer helps the user in this task avoiding such undesired mistakes.

As a last facility, the element matrices output can easily be exported to FORTRAN, C and MATLAB, emphasizing that saving the EM in a file in any of these formats, this matrix keep its name into the language format. In other words, considering the possibility to have the necessity to generate many matrices, if someone exports the matrix to the file *A.m* , the elements of the matrix are *A(1,1)*, *A(1,2)*, ... and so on, saving time of the user to open each file generated and rename the matrices.

IV. EXAMPLES

In this section, it will be shown one single basic example on 2D discretization but it is a very complex representation for our purpose, becoming clear what the program provides to the user, and its functionalities. In fact, it makes possible to express: the mobility possible to operate and to create vectorial operators followed by generic operations; the calculation of element matrices of surface and line domains; the differences between linear or quadratic field approximations; the exportation's format in different languages and the matrice's name according to the saved file name.

Example 1: To illustrate the use of our program, it will be considered the formulation proposed in [5]. That formulation is described for analysis of leaky optical waveguides with arbitrary cross-section. The formulation adopted solves the following equation involving the transverse magnetic field components, H_t :

$$\nabla_t \times (\varepsilon_{zz}^{-1} \nabla_t \times H_t) - \vec{z} \times [\varepsilon_t^{-1} \cdot \nabla_t \times (\vec{z} \nabla_t \cdot H_t)] - k_0^{\,2} H_t - k_0^{\,2} \gamma^2 \vec{z} \times [\varepsilon_t^{-1} \cdot (\vec{z} \times H_t)] = 0 \quad (12)$$

, where γ is the (complex) propagation constant and k_0 the wavenumber. In addition, it was considered a (complex) permivittivity tensor ε of the form, $\varepsilon = \varepsilon_t + \varepsilon_{zz} \vec{z}\vec{z}$, being ε_t an arbitrary 2x2 tensor given by $\varepsilon = \varepsilon_{xx} \vec{x}\vec{x} + \varepsilon_{xy} \vec{x}\vec{y} + \varepsilon_{yx} \vec{y}\vec{x} + \varepsilon_{yy} \vec{y}\vec{y}$.

Applying the Galerkin's Method, one can easily find a set of integrals similar to that shown in (13)

$$\begin{cases} S_1 = \iint (K_{zz} \nabla_t \times H_{tt}) \cdot (\nabla_t \times W_{tt}) & (13.a) \\ S_2 = \iint (\nabla_t \cdot H_{tt})_z \cdot [(\nabla_t \times K_{tt}^T) \cdot (\vec{z} \times W_{tt})] & (13.b) \\ S_3 = -k_0^2 \iint (H_{tt} \cdot W_{tt}) & (13.c) \\ S_4 = -k_0^2 \iint \{\vec{z} \times [K_{tt} \cdot (\vec{z} \times H_{tt})]\} \cdot W_{tt} & (13.d) \\ L_1 = \oint (K_{zz} \nabla_t \times H_{tt}) \cdot (W_{tt} \times \vec{n}) & (13.e) \\ L_2 = \oint \{(\nabla_t \cdot H_{tt})_z \times [K_{tt}^T \cdot (\vec{z} \times W_{tt})]\} \cdot \vec{n} & (13.f) \end{cases}$$

where $K_{tt} = \varepsilon_{tt}^{-1}$, $K_{zz} = \varepsilon_{zz}^{-1}$, $\vec{z}$ is the normal vector in the *z* direction, and $\vec{n}$ the normal vector associated to the boundary. The integral's denomination was given because they are surface and line integrals, respectively. Also from (13), H_{tt} and W_{tt} refers to the magnetic field, *H*, and the transverse weight function. In [2] the Galerkin's Method was applied only to the transverse fields. The *tt* index was used to represent the transverse coordinates which could be oriented in the *x* or *y* direction. We will make the assumption that the material characteristics do not change inside an element.

To illustrate the examples concerning the manipulation and creation of operators, we will take (13.a) and (13.b). For the former, there are shown in (14) only the response that our program returns to the user in the MATLAB's prompt

```
H_X_W_X = +kzz*dif(Fi(j),Y)*dif(Fi(i),Y)
H_X_W_Y = -kzz*dif(Fi(j),Y)*dif(Fi(i),X)
H_Y_W_X = -kzz*dif(Fi(j),X)*dif(Fi(i),Y)
H_Y_W_Y = +kzz*dif(Fi(j),X)*dif(Fi(i),X)
```
(14)

It was done manipulating the layout of Fig. 2a, using the pop-up buttons. From (14) it must be noticed the indexes following *H* and *W*. The first line of (14) is the result considering H_x and W_x , and so on. In this part, we have seen that the program has given to us the four possible combinations (*H* and *W* are transverse functions). The word *dif* means differentiation and *Fi(i)*, *Fi(j)* are the numerical representation of *W* and *H* (according to (3) and (6)), respectively, that could be used or not later.

Considering (13.b) with the same procedure commented before but, now, saving the results on the file *S2.kf*, and looking inside of it, someone will see somewhat similar to:

```
%       OP 1=
%    K_tt
%    Transposed
%    multiplied by
%    VECTORIAL PRODUCT of
```

```
%    unit vector in z direction (a_z)
%    AND
%    weight_t

%        OP 2=
%    TRANSVERSE CURL OF
%    OP1

%        OP 3=
%    unit vector in z direction (a_z)
%    multiplied by
%    TRANSVERSE DIVERGENT OF
%    field_t

%        OP 4=
%    INNER PRODUCT of
%    OP2
%    AND
%    OP3

H_X_W_X=+(dif(kyy*Fi(i),X)-dif(kyx*Fi(i),Y))*dif(Fi(j),X)
H_X_W_Y=+(-dif(kxy*Fi(i),X)+dif(kxx*Fi(i),Y))*dif(Fi(j),X)
H_Y_W_X=+(dif(kyy*Fi(i),X)-dif(kyx*Fi(i),Y))*dif(Fi(j),Y)
H_Y_W_Y=+(-dif(kxy*Fi(i),X)+dif(kxx*Fi(i),Y))*dif(Fi(j),Y)
```
(15)

From this example, someone should see that every file saved in the KF format has a header which describes all operations done until the generation of the final expression. In fact, the header is the step by step manipulation of the operators, and assure the user about what was done.

For both cases illustrated before, (14)-(15), it must be seen that the final expressions have transverse components, which, from the FEM theory explained in Section II, are the so-called element matrices, EM's.

However, this is not the most common way to define an EM because, for the examples above, our results consider the material characteristics (k_{yy}, k_{yx}, k_{xx}, etc).

In the books, it is usual the definition of an EM without any conditions, or material characteristics embedded on it. The authors say that if these characteristics changes inside of a single element *e*, a finite element program running will produce wrong results. That's true!

$$SS1_{ij}^{e} = \iint_{\Omega e} \frac{\partial\{\phi_j\}^T}{\partial x}\frac{\partial\{\phi_i\}}{\partial y}dxdy \quad (16.a)$$

$$SS2_{ij}^{e} = \iint_{\Omega e} \frac{\partial\{\phi_j\}^T}{\partial y}\frac{\partial\{\phi_i\}}{\partial x}dxdy \quad (16.b)$$

$$SS3_{ij}^{e} = \iint_{\Omega e} \frac{\partial\{\phi_j\}^T}{\partial x}\frac{\partial\{\phi_i\}}{\partial x}dxdy \quad (16.c)$$

$$SS4_{ij}^{e} = \iint_{\Omega e} \frac{\partial\{\phi_j\}^T}{\partial y}\frac{\partial\{\phi_i\}}{\partial y}dxdy \quad (16.d)$$

$$FF1_{ij}^{e} = \int_{\Gamma e} \frac{\partial\{\phi_j\}^T}{\partial x}\{\phi_i\}d\Gamma_e \quad (16.e)$$

$$FF2_{ij}^{e} = \int_{\Gamma e} \frac{\partial\{\phi_j\}^T}{\partial y}\{\phi_i\}d\Gamma_e \quad (16.f)$$

i, j=1, 2, 3 (linear triangular element) or
i, j= 1,2, 3, ... 6 (quadratic triangular element)

However for classical and practical considerations in optics, when someone is building a source-code program he will certainly deal with expressions similar to those one found by our developer.

Then, if someone decide to extract the element matrices of (13) in their simplest form, he will find only six basic element matrices, as those one showed in (16).

In fact, to finalize this brief discussion, our developer attends any conditions. It means, if the user decides to calculate the element matrices directly from the output of (14) or (15), for example, there is no problem. This example was done using (13.c) and for the sake of brevity, its partial result is shown below. It is exported to the MATLAB format, linear field approximation, and surface integral.

```
%%%%%  2D-AREA INTEGRATION OF %%%%%
%%    (-k0^2*fi(j)*fi(i))

H_X_W_X(1,1)=-1/6*Ae*k0^2;
H_X_W_X(1,2)=-1/12*Ae*k0^2;
.
.
.
H_X_W_X(3,2)=-1/12*Ae*k0^2;
H_X_W_X(3,3)=-1/6*Ae*k0^2;

%%%%%  2D-AREA INTEGRATION OF %%%%%
%%    (0)

H_X_W_Y(1,1)=0;
.
.
.
H_X_W_Y(3,3)=0;

%%%%%  2D-AREA INTEGRATION OF %%%%%
%%    (0)

H_Y_W_X(1,1)=0;
.
.
.
H_Y_W_X(3,3)=0;

%%%%%  2D-AREA INTEGRATION OF %%%%%
%%    (-k0^2*fi(j)*fi(i))

H_Y_W_Y(1,1)=-1/6*Ae*k0^2;
H_Y_W_Y(1,2)=-1/12*Ae*k0^2;
H_Y_W_Y(1,3)=-1/12*Ae*k0^2;
H_Y_W_Y(2,1)=-1/12*Ae*k0^2;
H_Y_W_Y(2,2)=-1/6*Ae*k0^2;
H_Y_W_Y(2,3)=-1/12*Ae*k0^2;
H_Y_W_Y(3,1)=-1/12*Ae*k0^2;
H_Y_W_Y(3,2)=-1/12*Ae*k0^2;
H_Y_W_Y(3,3)=-1/6*Ae*k0^2;
```

Also, if the user wants to calculate the EM just like in (16), it will be done using the layout of Fig. 2b. The switching between one program or another is done by a simple click of mouse, without leaving the ambient.

To illustrate the functionality of our second program (see Fig. 2b), we will now calculate (16.c) and save the EM in C format (file called *SS1.c*), evaluating it with a

quadratic field's approximation. The partial file's contents is shown below.

```
//Differentiation of Fi-i with respect x
//multiplied by
//Differentiation of Fi-j with respect x

ss1 [0][0] = pow(-y3+y2,2.0)/Ae/4.0;
ss1 [0][1] = (y1-y3)*(-y3+y2)/Ae/12.0;
ss1 [0][2] = -(-y3+y2)*(y1-y2)/Ae/12.0;
ss1 [0][3] = -(y1-y3)*(-y3+y2)/Ae/3.0;
ss1 [0][4] = 0.0;
.
.
ss1 [5][3] = -2.0/3.0*(y1-y3)*(y1-y2)/Ae;
ss1 [5][4] = -2.0/3.0*(y1-y3)*(-y3+y2)/Ae;
ss1 [5][5] = 2.0/3.0*(y1*y1-y1*y3-y1*y2+y3*y3-
y3*y2+y2*y2)/Ae;
```

From this example, let's pay attention the EM's name, the same of the file saved. It is done automatically. As expected, the EM is in the C format, including the head. In fact, it always happens independent if it is in C, FORTRAN or MATLAB.

In our next examples, we will show the result of a difficult and not common EM: the boundary integral. For the 2D case, an integral of line. For this case, let's save the file in MATLAB format, called *FF1.m*, and consider a linear field approximation.

```
%Differentiation of Fi-j with respect x
%multiplied by
%Fi-i

FF1_12(1,1)=1/4*h12/Ae*(-y3+y2);
FF1_12(1,2)=1/4*h12/Ae*(-y1+y3);
FF1_12(1,3)=1/4*h12/Ae*(y1-y2);
FF1_12(2,1)=1/4*h12/Ae*(-y3+y2);
FF1_12(2,2)=1/4*h12/Ae*(-y1+y3);
FF1_12(2,3)=1/4*h12/Ae*(y1-y2);
FF1_12(3,1)=0;
FF1_12(3,2)=0;
FF1_12(3,3)=0;

FF1_23(1,1)=0;
FF1_23(1,2)=0;
FF1_23(1,3)=0;
FF1_23(2,1)=1/4*h23/Ae*(-y3+y2);
FF1_23(2,2)=1/4*h23/Ae*(-y1+y3);
FF1_23(2,3)=1/4*h23/Ae*(y1-y2);
FF1_23(3,1)=1/4*h23/Ae*(-y3+y2);
FF1_23(3,2)=1/4*h23/Ae*(-y1+y3);
FF1_23(3,3)=1/4*h23/Ae*(y1-y2);

FF1_31(1,1)=1/4*h31/Ae*(-y3+y2);
FF1_31(1,2)=1/4*h31/Ae*(-y1+y3);
FF1_31(1,3)=1/4*h31/Ae*(y1-y2);
FF1_31(2,1)=0;
FF1_31(2,2)=0;
FF1_31(2,3)=0;
FF1_31(3,1)=1/4*h31/Ae*(-y3+y2);
FF1_31(3,2)=1/4*h31/Ae*(-y1+y3);
FF1_31(3,3)=1/4*h31/Ae*(y1-y2);
```

In this case *h12, h23, h31* are the length of the edges of the triangular element, and the three integrals expressed by *_12*, *_23*, *_31* are the results along the path going from 1 to 2, 2 to 3 and 3 to 1, respectively.

Although no one example on 3D was done, we would like to call attention that our programs were already tested on 3D formulations. In fact, EM were calculated using our developer and specifics programs in FEM used them, giving consistent results.

As a second point, we would like to state that our developer works very fast and it is running in both Windows-MS and Li(UNIX) platforms.

And as a last point, to say that the developer is available with a user's guide in the internet, or through the authors's email.

V. Conclusions

In this paper, it was presented a GUI using MATLAB to design and to manipulate differential operators in FEM, as well as for the evaluation of element matrices on the nodal FEM. The developer works with both 2D and 3D domains, using triangular and tetrahedral elements, respectively. For the 2D domain, it calculates flow (line) and surface integral, while for the 3D it calculates volume and flow (surface) integrals. It also makes possible to export results in a compatible format with MATLAB, FORTRAN, and C. Finally, it provides matrices considering linear or quadratic elements for both 2D and linear 3D domains.

Finally, because of its structure, designing and simplicity, the developer here presented can be used also in other fields of engineering, physics and chemistry, being their results very easy to be interpreted. Therefore, this developer may be beneficial for either professional and educational FEM activities. Here, the usefulness of the present developer is demonstrated through key electromagnetic examples.

References

[1] Wen X. Zhou, Chien H. Hsiung, Robert E. Fulton, Xun Fei Yin et al. "CAD_Based analysis tools for electronic packaging design", *INTERpack Innovations in CAD/CAE Integration in Electronic Packing, Kohala, June 15-19,* 1997.

[2] Donald Koo, Russel S. Peak and Robert E. Fulton "An object-oriented parser-based finite element analysis tool interface", *SPIE Photonics East,* pp. 215-237, September 1999.

[3] Roger J. Dejus and Manuel Sanchez del Rio, "XOP: A graphical user interface for spectral calculations and x-ray optics", *AIP Rev. Sci. Instrum.* , Vol. 67, Nº 9, pp. 1-4, September 1996.

[4] M. Koshiba, *Optical Waveguide Theory by the Finite Element Method*. Kluwer Academic Publishers: Boston, 1992, pp. 13.

[5] H. E. Hernández Figueroa, F. A. Fernández, Y. Lu and J. B. Davies, "Vectorial finite element method of 2D leaky waveguides", *IEEE Transaction on Magnetics*, vol. 31, pp. 1710-1713, May 1995.