BEYKENT UNIVERSITY JOURNAL OF SCIENCE AND ENGINEERING
Volume 5(1-2), 2012, 131-144

ANALAZING USE of PROGRAMMING LANGUAGE for FINITE ELEMENT MODELING of ELECTROMAGNETICS

Berna SULU*, Turhan KARAGULER**
*Beykent University, Institute of Science and Engineering Postgraduate Student, bernaslu@gmail.com

**Beykent University, Department of Energy Systems Engineering, Ayazaga Maslak Campus, 34396, Maslak-Istanbul, Turkey
turank@beykent.edu.tr

ABRACTS

The problems of electromagnetism are modeled by means of various numerical methods. Among them Finite Element Method (FE) is considered as a superior one despite its rather complex coding requirements. As programming environment, until recently FORTRAN and C/C++ have been the usual choice but now MATLAB and even JAVA are becoming alternative options. This paper briefly discusses some programming issues and outlines superiorities and weakness of the development environments regarding FE modeling of electromagnetics over an example problem.

Keyword: *Finite elements modelling, Numerical Electromagnetism*

ÖZET

Elektromanyetizma Problemlerinin Sonlu Elemanlar Yöntemiyle Modellenmesinde Programlama Dillerinin Analizi; Elektromanyetik problemler bir çok farklı sayısal yöntemler yardımıyla modellenmektedir. Bu yöntemler arasında, kod yapısının ve algoritmasının görece karmaşıklığına karşın Sonlu Elemanlar Yöntemi (FE) diğerlerine göre daha popülerdir. Yakın zamana kadar Sonlu Eleman modellemesinde, FORTRAN ve C/C++ programlama dil seçeneği olarak yaygın olarak tercih edilmesine karşın son yıllarda MATLAB ve JAVA dilleri de bu dillere alternatif olarak ortaya çıkmaktadır. Bu çalışmada, Sonlu Elemanlar Yöntemiyle modellemede karşılaşılabilecek programlama konuları tartışılarak bir örnek problem üzerinden bazı dillerin karşılaştırılması yapılmıştır.

Anahtar Kelimeler: *Sonlu Elemanlar Modellemesi, Sayısal Elektromanyetizma*

1 INTRODUCTION

Today the finite element method (FE) is highly popular numerical technique employed in modeling scientific and engineering problems. Regarding the field of electrical engineering, FE is mostly applied to electromagnetism and related areas.

The common way of using the FE technique is through pre-developed software packages. This is simply because coding FE is a real task and requires extensive programming ability. On the other hand, using a software package can be restrictive and may not address the specific problems of interest. Due to this fact, one might consider to write the FE program from the scratch. In this case, the choice of programming environment becomes significant. This paper will attempt to emphasize on the programming issue by means of introducing an example problem and steps of FE procedure which is coded in both conventional programming (C Language) and script programming (MATLAB) environments. Additionally, recent FE coding applications with popular Java environment are discussed.

1.1 Brief History

The root of Finite Element Method (FE) is mostly associated with Courant's work published in 1943 in which he introduced piecewise-linear approximation on triangles called as "elements" to solve a 2D potential problem [1]. These elements in fact are non-overlapping small regions obtained through division of solution domain into sub-domains. However the real take off the method had to wait for development of digital computers. From the mid 50s to early 60s only few FE papers covering mainly structural analysis were published [2-3]. The impact of Zienkiewicz's book in which the FE method was explained in details and applied to various field problems was quite extensive such that all other disciplines apart from building engineering also started adopting the technique to their own fields [4].

2 FE AND ELECTROMAGNETİCS

The introduction of FE Method to electrical engineering problems was about a decade later than to early civil and mechanical engineering applications. The Silvester's works on modeling hollow waveguide and potential problems are widely considered as the first real FE application to electromagnetics [5, 6]. These papers were immediately followed by several new ones covering magnetostatics, dielectric waveguides, and other well known boundary value problems of electromagnetics. Today it is possible to find FE modeling on almost any kind of electromagnetic application. Although the early models were mainly in the area of high frequency applications, the method later was found more in low frequency and DC applications.

In general FE method involves 4 main stages which are deriving governing equation, discretizing the model region into elements, assembling of all the elements and building the matrixes and solving the system of equations.

The first stage of FE modeling is to obtain the governing equation. Since following Maxwell Equations together with the constitutive equations define electromagnetic phenomena completely, the governing equation can be obtained by using these equations.

$$Curl\vec{E} + \frac{\partial\vec{B}}{\partial t} \tag{1}$$

$$Curl\vec{H} = \vec{J} + \frac{\partial\vec{D}}{\partial t} \tag{2}$$

$$div\vec{B} = 0 \tag{3}$$

$$div\vec{D} = \rho \tag{4}$$

In these equations, electric field (E) with displacement vector (D), and magnetic field (B) with magnetic induction (H) are related with constitutive equations such that the medium properties are taken into consideration. Additionally the expression of Ohm’s law in which the current density vector (J) and electric field are related by conductivity (σ) completes full set of electromagnetic equations.

$$\vec{B} = \mu\vec{H} \tag{5}$$

$$\vec{D} = \varepsilon\vec{E} \tag{6}$$

$$\vec{J} = \sigma\vec{E} \tag{7}$$

It is quite common that in FE modeling, the Electric Scalar Potential (V) and the Magnetic Vector Potential (A) are employed in the governing equations instead of E and B respectively. This is mainly because the potential functions mostly lead to fewer unknowns in the final set of system equations and easier handling of the boundary conditions.

$$\vec{E} = gradV \tag{8}$$

$$\vec{B} = curl\vec{A} \tag{9}$$

In the frame of this work, 2D representation of a capacitor connected to a DC source is chosen as the test problem (see figüre 1).

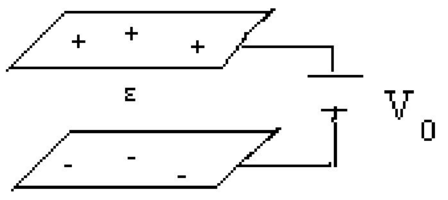

Fig. 1 Test problem

In order to let the problem be simple, the 2D model is assumed as linear and source charge free. The governing equation which is well known Laplace Equation can be easily obtained from the Maxwell Equation (4), the constitutive equation (6) and the definition of Scalar Electric Potential (8).

$$\frac{\partial}{\partial x}\varepsilon\frac{\partial V}{\partial x} + \frac{\partial}{\partial y}\varepsilon\frac{\partial V}{\partial y} = 0 \tag{10}$$

For the FE solution, as well as the governing equation, certain boundary conditions should be considered too. These conditions can be either interior or outer type. The interior conditions arises from the fact that the model may have several regions with different properties such as air, dielectric, conducting etc.

The following equations related with tangential and normal components of Electric Field will be sufficient to take the boundary conditions into account at the interface.

$$E_{t1} = E_{t2}\,, \qquad \varepsilon_1 E_{n1} = \varepsilon_2 E_{n2} \tag{11}$$

However the outer conditions are defined only on the external boundaries and required to complete the set of equations. Depending on the nature of a problem, on the boundary, either a variable can be set to a fixed value (Dirichlet Type) or its derivatives can be specified (Neumann Type). In the test model, the Dirichlet boundaries do exist as the conductors have fixed potentials of V_0 and 0.

It is common that FE Method is applied to a governing equation by means of either variational methods or weighted residual methods. In the variational method, a variational expression called functional would be introduced. The method seeks the minimum of this functional which represents the governing equation under the boundary conditions. Thus minimizing the functional with respect to unknown variable will result with the approximate solution. The variational procedure is also named as Ritz method and mostly used in the early FE applications [7]. However finding a functional for certain problems is not always as easy as Laplace problem which uses potential energy directly to be minimized. Furthermore the variational methods do not deal with the physical equation directly, instead, use the corresponding functional. Contrarily, weighted residual methods are applied directly to the physical equation and comparatively simpler to understand and implement therefore lately majority of FE works are carried out by the weighted residual methods. The Galerkin method which is one form of these weighted residual methods is the one mostly used in electromagnetism [8]. Due to this fact, in this work, the governing equation which is a differential one is converted to the numerical equations by applying the weighted residual method from Galerkin's point of view. This procedure of conversion is briefly explained in the following sub section. Although variational and weighted residual methods have different paths to discretize the governing equation, both methods usually end up with the same set of equations to solve.

2.1 Application of Galerkin Method

Unlike analytical methods, numerical methods produce approximate solutions which are different from the exact solution. This difference between exact and approximate solutions is a residual R(x,y). The residual form of the governing Laplace equation can be obtained by replacing V with the approximate Va.

$$\frac{\partial}{\partial x}\varepsilon\frac{\partial V_a}{\partial x}+\frac{\partial}{\partial y}\varepsilon\frac{\partial V_a}{\partial y}=R(x,y) \qquad (11)$$

The weighted residual methods in general attempt to force the weighted integral of this residual over the entire domain to be zero.

$$\int_S W R(x,y)dS=0 \qquad (12)$$

In the above equation, W is known as a weighting function.

In general, in the FE model, the solution domain is divided in small regions called elements. These elements, for instance in 2D, can be triangles or quadrilaterals. For the sake of simplicity, the first order triangles are used in discritizing the model assuming that the potential varies linearly within a triangle. While the corners of a triangle are named as the nodes or degree of freedom and assembly of the triangles (elements) is named as the mesh, FE method tries to find potential values for these nodes and later for each element by using the approximation below:

$$V_a=\sum_{i=1}^{3}N_iV_i \qquad (13)$$

where Ni, and Vi are known geometry dependent shape functions and unknown potentials for each node respectively. The Galerkin procedure specifies the weighting function W as the shape function N. Thus equation (12) can be rewritten for k. element as

$$\sum_{i=1}^{3}\int_{S_k}N_iR(x,y)dS=0 \qquad (14)$$

This equation above can be extended to all the elements in the mesh.

At this stage, another fundamental step of FE procedure which is assembly of elements should be implemented. While having K as the number of nodes in the model, assembling of elements at the common nodes between adjoin elements leads to K number of equations and unknowns. In order to simplify and automate the assembling elements, a special node numbering scheme is used [4].

The last step to the solution is to solve the system of equations which are normally obtained in a matrix form as $[S]*[V]=[b]$. The solution $[V]$can be found by either applying a direct method such as Gauss elimination or an iterative method such as conjugate gradient method. Since the FE equations involve only nodal variables belonging to the same element, it is apparent that FE methods yield sparse stiffness matrix $[S]$. In order to exploit this sparseness, usually iterative methods are employed. However in this work, the number of unknowns of 2D model is rather low therefore Gauss elimination is opted out due to its simplicity.

All these fundamental steps and application of boundary conditions are well evaluated and detailed in references [9, 10].

3 Programming Aspect

Regarding FE implementation, despite there has been so much published work on numerical application and algorithm sides of the method, not much has been said on the programming side. This is reasonable as not many options were available in the early years of the method.

The evaluation of FE programming has been very much inline with the evaluation of hardware and programming languages in general. Naturally, the early coding language for FE method was purely FORTRAN as it was probably the only language designed and developed for scientific and engineering related programming practices. This dominance went on until mid 70s when C was born and started becoming also a familiar language. The main superiority of C over FORTRAN was the speed thus performance. This is provided partly by introduction of pointers enabling direct access to the memory space and avoiding time consuming address conversion. It is well known fact that FE codes particularly during pre-processing and post- processing stages require extensive use of memory and CPU time so that C language got wide attention among FE programmers. Furthermore, initially C was created as the development language of UNIX operating system and then later became almost the default language for most new operating systems. This lets C take full advantage of operating systems and hardware and produce machine independent and portable programs. Despite all these superior points of C, for FE programming, a big shift from FORTRAN to C did not materialize until mid 80's.

Conceptually both FORTRAN and C belong to the family of procedural languages. Therefore from computing point of view, FE programming model kept its original form until object oriented programming being introduced [11]. Forte and co-worker's paper is possible one of the earliest work on object-oriented FE application in which basic FE related classes such as elements, nodes, matrices, etc. are developed and used [12]. Most of the object oriented FE programs are written in C++ which is derived from C with object oriented features such as encapsulation, inheritance and polymorphism. These features allow producing more reliable and reusable FE codes. These codes are also more practical to manage and modify. Classes which comprise data and functions are the fundamental pieces of C++ and object oriented programming languages in general. Objects created from classes with new powerful functionalities replace data variables used in structural programming. C++ programmers mostly do not write the code from scratch but rather take advantage of the rich existing classes and templates in the C++ standard library. A book written by Mackie details how object oriented approach to FE programming fundamentally differs from conventional structural language approach [13]. Today almost all the FE programmers consider object oriented approach undoubtedly the first choice if only the classes are properly designed and supported with graphical user interface facilities. However still a few developers claim that new version FORTRAN 90 should be considered and as it is superior over C++ regarding execution time efficiency [14].

Java is also an object oriented language, simpler than C++ and possesses some extra capabilities such as built-in data structures and functions for designing graphical user interfaces and communicating with other devices over a network. New Java versions contain highly developed 2D and 3D graphical packages which make graphical programs be developed easily. Moreover standard Java has a mechanism of garbage collection for preventing memory leaks thus automatic release of memory back to the operating system. Another important point about Java is that Java platform was initially designed to achieve the total independency from hardware by means of developing Java Virtual Machine (JVM). The essential aim of introducing JVM is to make Java the default programming platform for the web. Because of this, the programs in Java are first compiled to obtain only java specific bytecodes which are later interpreted by the JVM to get ready for the execution. Java bytecodes also provide strong checking on the code for safe and secure programs. All these important gains by Java are achieved at the expense of speed. Unlike compiled code which includes a series of microprocessor instructions, an interpreter must first translate the java bytecodes into the equivalent processor instructions therefore leading to obviously slower running operations. This is possible the main reason that FE programmers have not considered Java as a developing tool until a few years ago. Recent introduction of Just-in-Time compiler designed as the integral part of JVM significantly increased the running speed of applications and applets by taking the bytecodes and compiling them into the native code. The works for FE modeling using Java started appearing in conferences and journals. Nikiskhov's papers are significant for achieving FE programming with Java. One of his publications outlines all the details of object oriented design of FE in Java [15], and the second claims that the performance of Java is in comparable range with C language therefore should be considered seriously for FE modeling [16].

3.1 Script Language Versus Conventional Language

In the previous section, programming languages in a groups of structural (FORTRAN, C) and object oriented (C++, Java) are compared for FE modeling. However there are also script languages such as PERL, MATLAB, Python, Ruby, etc which have relatively limited resources and are mainly used in areas like utilization, text processing, report writing, etc. Among these languages MATLAB (also MAPLE and Mathematica) is considered as scientific computing environment and provides more than simple tasks of scripting languages therefore is a real option for FE programming practices.

MATLAB is extremely powerful in matrix and vector intensive operations and 2D and 3D graphical visualization which all together form the backbone of FE modeling. As a language, it is simple, effective and flexible therefore becoming a popular tool for developing FE applications. In reference [17], it is shown that how a short MATLAB FE program, less than 50 lines, can bring out graphical solution of different type engineering problems. Mesh generation which is also an essential part of FE modeling can be achieved by using MATLAB too. Regarding the matter, the work of Persson and Strang in [18] explains how to use well known distmesh2D and distmesh3D functions which are based on Deleaunay triangulation algorithms.

There is also some weakness of MATLAB comparing with conventional languages. Firstly, MATLAB like all other script and interpreted languages performs slower. Secondly, to run MATLAB programs, usually it is required that MATLAB to be installed on the machine thus the portability on any other machine is a problem. Thirdly, developing interactive GUI's with MATLAB is not an easy. MATLAB's commonly used GUIED tool may be sufficient for simple FE applications but may not so for the cases in which user friendly interfaces are required.

4 RESULTS AND DISCUSSIONS

In order to make a comparison between a conventional language and script language, the simple test problem described earlier is modeled and coded both in C language and in MATLAB. For the programs, the steps of standard FE procedure are followed in the same manner as described in earlier section.

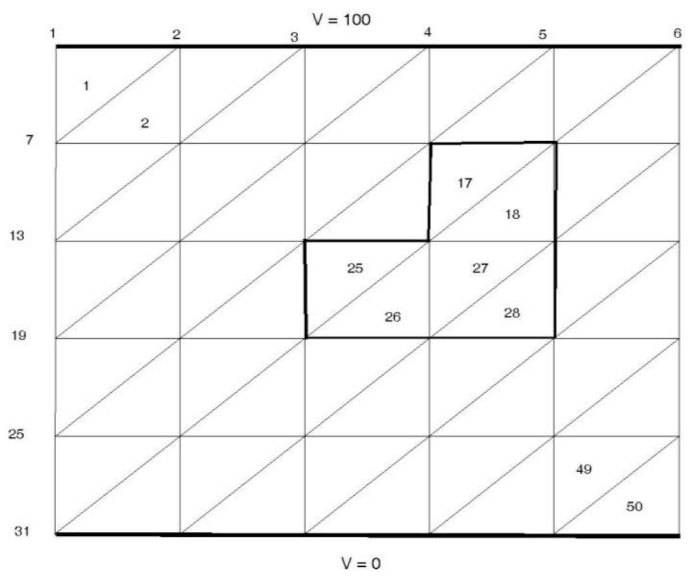

Fig.2 FE mesh of test problem

As seen from fig.2, a 2D mesh of the test problem with 50 elements, and 36 nodes is designed. The thick lines indicate the Dirichlet type fixed potential boundaries which represent conductors. In between conductors, air and a dielectric region with εr=5 specified by elements 17, 18, 25,26,27,28 exist. The length of conductors and the gap between them are unrealistically taken as 10 units just for clarity. Both programs use a data file for input which comprises node numbers and region number for each element, coordinates of each node, the node numbers on the dirichlet boundaries and their fixed potential values, etc. As expected, the results from both programs are almost identical. For instance at node 10 which is upper left corner of dielectric region, the potential value is 71.4533, the magnitude of electric field in element 17 (dielectric) is 5.4002, and just outside the dielectric, in element 16, is 9.4433.The colored potential distribution is displayed in figure 3.

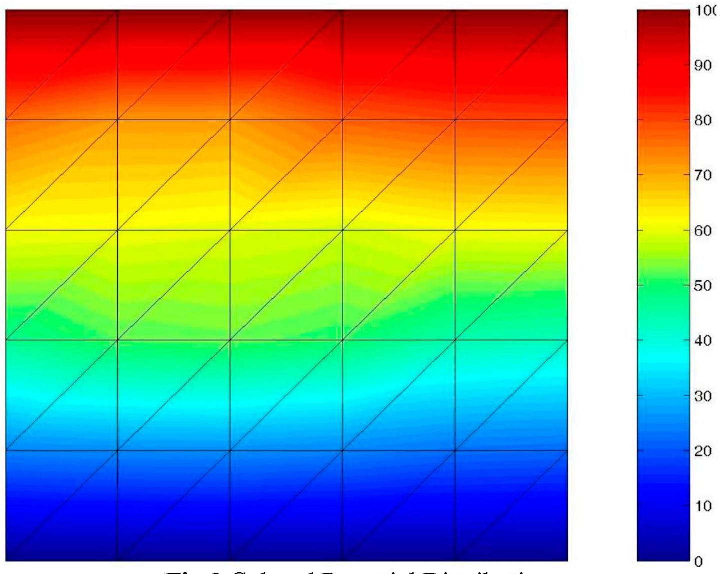

Fig.3 Colored Potential Distribution

It is quite interesting that the programs producing the same results differ hugely in length. Ignoring the comment and empty lines, while C program is coded with 193 lines, the MATLAB program is coded with only 57 lines. On the other hand, the execution time for the C program is almost undetectable whereas for the MATLAB program, 0.0375 CPU time is registered by using cputime command. This is considerably high even for a very small sized test problem. For MATLAB programming, the performance can be a real issue for FE models having complex 3D geometries. But at the same time easiness and effectiveness of graphical display facilities of MATLAB, with only few lines of code, should be taken into account. In order to achieve the same graphical display with C, complex graphical library routines, such as openGL, need to be integrated with the main FE program. This means that the main computation effort may be spent on secondary issues rather than the actual modeling and solving of the problem.

5 Conclusion

FE modeling is extensively used for solving electromagnetic problems. There are few options available in programming of FE discretization. In order to make a solid decision on choosing the best developing environment, some computational issues should be considered together with the capabilities and specific features of programming languages. Only an initial and introductory work is carried out regarding comparison of structural and scripting languages over a 2D FE test problem. As expected, the C program runs faster however the MATLAB program was shorter and effective. The work would be extended to the inclusion of object oriented (C++) and web based (Java) platforms in comparison. Further improvement can also be achieved by considering a more complex 3D geometry as the mesh of test problem.

REFERENCES

[1] Courant RL. Variational methods for the solution of problems of equilibrium and vibration. Bulletin of the American Mathematical Society 1943; 49: 1-23.

[2] Turner MJ, Clough RW, Martin HC, Topp LC. Stiffness and Deflection Analysis of Complex Structures. Journal of Auronautical Science 1956; 23: 507-510.

[3] Clough RW. The Finite Element Method in Plane Stress Analysis. Proceedings of 2nd ASCE Conference on Electronic Computation. September 1960; Pittsburgh.

[4] Zienkiewicz OC. The Finite Element Method. 1967; McGraw-Hill.

[5] Silvester P. Finite-element solution of homogeneous waveguide problems. Alta Frequenza 1969; 38: 313-317.

[6] Silvester P. High-order polynomial triangular finite elements for potential problems. International Journal for Engineering Science 1969; 7: 849-861.

[7] Ritz W. Über eine neue methode zur losung gevissen variations – probleme der mathematischen physic. J.Reine Angew. Math 1909; 135: 1-61.

[8] Galerkin BG. Series solution of some problems of elastic Equilibrium of rods and plates. Vestn. Inzh. Tech 1915; 19: 897-908.

[9] Jin J. The Finite Element Method in Electromagnetics. US: John Wiley & Sons; 2002.

[10] Pedro J, Bastos A, Sadowski N. Electromagnetic Modeling by Finite Element Methods. US: Marcel Dekker; 2003.

[11] Bathe KJ. Finite Element Procedure. NJ: Prentice-Hall; 1996

[12] Forte BWR, Foschi RO, Steimer SF. Object-oriented finite element analysis. Computers and Structures 1990: 34; 355-374.

[13] Mackie RI. Object oriented methods and finite element analysis. UK: Saxe-Coburk Publication; 2001.

[14] Akin JE, Singh M. Object Oriented Fortran 90 P- adaptive finite element method. Advances in Engineering Software 2002: 33; 461-468.

[15] Nikishkov GP. Object oriented design of a finite element code in Java. CMES-Computer Modeling in Engineering & Sciences 2006: 11(2); 81-90.

[16] Nikishkov GP, Nikishkov YG, Savchenko VV. Comparison of C and Java in finite element computations. Computers and Structures 2003: 81; 2401-2408.

[17] Alberty J, Carstensen C, Funken SA. Remarks around 50 lines of Matlab: short finite element implementation. Numerical Algorithms 1999: 20;117-137.

[18] Persson PO, Strang G. A simple mesh generator in Matlab. SIAM Review 2004: 16; 329-345.