Solving the Laplacian Equation in 3D using Finite Element Method in C# for Structural Analysis

BedrEddine Ainseba[1], Mostafa Bendahmane[2] and Alejandro López Rincón[3]

EPI, Anubis INRIA Bordeaux Sud-Ouest. Institut Mathematiques de Bordeaux, Universite Victor Segalen Bordeaux 2 Place de la Victoire 33076 Bordeaux, France.
[1]bedreddine.ainseba@u-bordeaux2.fr, [2]mostafa.bendahmane@u-bordeaux2.fr and [3]alejandro.lopez_rincon@inria.fr

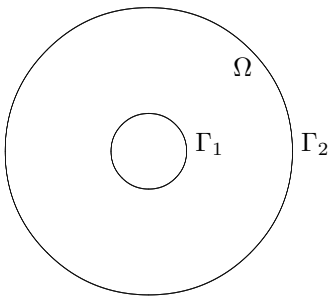
***Abstract*- In this paper, we present the implementation of a Finite Element Method (FEM) software developed using $C\#$ for Three Dimensional Laplace Equation, considering inhomogeneous Dirichlet and Neumann Boundary Conditions. This can be used for structural analysis.**

***Keywords*- Finite Element Method (FEM), $C\#$, Boundary Conditions, Structural Design, software.**

1 Implementing the Finite Element Method

Having the following system:

$$\begin{cases} -\nabla \cdot \left(\mathbf{c}\nabla u\right) = f, & \text{in } \Omega, \\ u = g, & \text{on } \Gamma_1, \\ c\frac{\partial u}{\partial n} = h, & \text{on } \Gamma_2, \end{cases} \tag{1.1}$$



where Ω is a polygonal domain and $\partial\Omega = \Gamma_1 \cup \Gamma_2$.

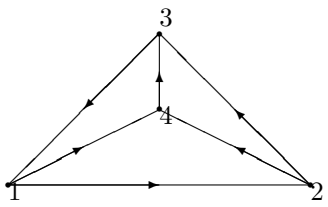
The Finite Element Method [13] (FEM), when applied to (1.1), produces a matrix-vector equation $KU = F$ whose solution vector U contains the nodal values of the approximate solution function.

There are three steps to implement the FEM:

- Create the mesh for Ω,
- Calculate the Stiffness Matrix K and Load Vector F,
- Solve the system $KU = F$.

1.1 Mesh

The different Meshes were created using the Meshes from *.Tri Files, *.STL Files, and *.Msh Files. The Tetrahedra elements are created from the given surfaces using GMESH [1], or by programming a specified geometry, like two spheres, and transforming it into a *.Msh File. To represent the geometries we programmed a graphic environment using OpenGL [2] (Open Graphics Library) and a visual interface of 30 fps (frames per second). To implement OpenGL we used the Tao Framework [3].



1.2 Stiffness Matrix

The Stiffness Matrix is calculated using the following:

$$K_{ij} = \sum_{i=0}^{N} \int_{Tet_k} c\nabla\phi_i \cdot \nabla\phi_j, \tag{1.2}$$

where $\phi = \begin{cases} 1, & i = j, \\ 0, & i \neq j, \end{cases}$
$i, j = 1, 2, 3, 4$

where the function ϕ is represented by

$$\phi_i(x, y, z) = a_i + b_i x + c_i y + d_i z, (x, y, z), \tag{1.3}$$

$$\text{for } i, j = 1, 2, 3, 4.$$

The coefficients $a_i + b_i + c_i + d_i$ can be found by solving a 4 x 4 system of equations. For example, for $i = 1$ we get the following system:

$$\begin{aligned} a_1 + b_1 x_1 + c_1 y_1 + d_1 z_1 &= 1 \\ a_1 + b_1 x_2 + c_1 y_2 + d_1 z_2 &= 0 \\ a_1 + b_1 x_3 + c_1 y_3 + d_1 z_3 &= 0 \\ a_1 + b_1 x_4 + c_1 y_4 + d_1 z_4 &= 0. \end{aligned} \tag{1.4}$$

If we combine the 4 systems and put them in matrix form we have the next system:

$$\begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{bmatrix} \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{bmatrix} = \tag{1.5}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{bmatrix}, \tag{1.6}$$

is the inverse of

$$\begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{bmatrix}. \tag{1.7}$$

To assemble the Stiffness Matrix is necessary to calculate the following integral:

$$\int_{Tet_k} c\nabla\phi_i \cdot \nabla\phi_j. \tag{1.8}$$

since ϕ_i is linear over Tet_k (the k Tetrahedra), the gradients are constant:

$$\nabla\phi_i = \begin{bmatrix} b_i \\ c_i \\ d_i \end{bmatrix}. \tag{1.9}$$

Therefore

$$\int_{Tet_k} c\nabla\phi_i \cdot \nabla\phi_j = (\nabla\phi_i \cdot \nabla\phi_j) \int_{Tet_k} c, \tag{1.10}$$

and the integral of c can be estimated with the 1-point rule.

$$\int_{Tet_k} c = V * c(\overline{x}, \overline{y}, \overline{z}), \tag{1.11}$$

where

$$\begin{aligned} \overline{x} &= \frac{x_1 + x_2 + x_3 + x_4}{4}, \\ \overline{y} &= \frac{y_1 + y_2 + y_3 + y_4}{4}, \\ \overline{z} &= \frac{z_1 + z_2 + z_3 + z_4}{4}, \end{aligned} \tag{1.12}$$

and

$$V = \frac{1}{6} det \begin{vmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix}. \tag{1.13}$$

1.3 Load Vector

To compute the load vector F the process is similar:

$$\int_{Tet_k} f\phi_i = \frac{1}{4} V f(\overline{x}, \overline{y}, \overline{z}). \tag{1.14}$$

1.4 Dirichlet conditions

To deal with the Inhomogeneous Dirichlet conditions the Load Vector F becomes:

$$F_i = \int_\Omega f\phi_i - \int_\Omega c\nabla G \cdot \nabla\phi_i, i = 1, 2, .., N, \tag{1.15}$$

where G is defined by

$$G(n) = \begin{cases} g(n), & n \in \Gamma_1, \\ 0, & n \neq \Gamma_1. \end{cases}$$

The function G will be 0 in all the tetrahedra that does not have constrained nodes. In the tetrahedra that have

at least one constrained node, the calculation will have the next form:

$$-\int_{Tet} c\nabla G\nabla\phi_i = -\nabla G\nabla\phi_i \int_{Tet} c, \tag{1.16}$$

where

$$\nabla G = \sum_{i=0}^{4} w_i \nabla\phi_i,$$

and w_i are the values in each Node. The contributions to F_i will have the form of

$$F_i- = c(\overline{x},\overline{y},\overline{z}) * G * V, \tag{1.17}$$

where G has the form of

$$G = \begin{bmatrix} b_1 & c_1 & d_1 \\ b_2 & c_2 & d_2 \\ b_3 & c_3 & d_3 \\ b_4 & c_4 & d_4 \end{bmatrix} * \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{bmatrix} * \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}. \tag{1.18}$$

and will contribute to each Node i in the tetrahedra.

1.5 Inhomogeneous Neumann conditions

To add the Inhomogeneous Neumann conditions the load vector F becomes:

$$F_i = \int_\Omega f\phi_i - \int_\Omega c\nabla G\cdot\nabla\phi_i + \int_{\Gamma_2} h\phi_i, \qquad i = 1,2,..,N \tag{1.19}$$

where

$$\int_{\Gamma_2} h\phi_i = \int_{e_1} h\phi_i + \int_{e_2} h\phi_i + \int_{e_3} h\phi_i. \tag{1.20}$$

being e_1 and e_2 and e_3 the Nodes of the outside triangle belonging to the the tetrahedra in the Γ_2.

To apply the Neuman Data vector in each node, we do the following steps:
First we add the values of the $NeumanData_i$,for$i = 1,2,3$ and we will call this value h and divide it by 3. Then we calculate the Normal vector,then we generate two temporal vectors in the next form

$$\begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{bmatrix}, \tag{1.21}$$

$$\begin{bmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - z_1 \end{bmatrix}. \tag{1.22}$$

then we do the cross product of the vectors (1.21) and (1.22), and calculate the Euclidian norm($n = \|x\| = \sqrt{x^2+y^2+z^2}$) of the vector. The area of the triangle will be

$$A = \frac{n}{2}, \tag{1.23}$$

The contributions to F_i will have the form of

$$F_i+ = \frac{h*A}{3}. \tag{1.24}$$

1.6 Solving the System

We have a system $KU = F$, where the Stiffness Matrix is a positive definitive matrix, we can solve the system by implementing the Conjugate Gradient algorithm. To improve the performance of the algorithm we use sparse matrix multiplication for each step in the CG calculation. This reduces the calculation up to 90% in time.

The Stiffness Matrix , for the moment has a limit. In Visual Studio and Visual C# Express the limit in 64 bits systems is an object (in this case an array) of 2 GB. In practice, the object ,to be manageable, should not be greater of 1 GB. To solve this problem for large calculations the information is written in disk and loaded where necessary to have enough free memory through SQL implementation.

2 Numerical Results for FEM

To test the program we used the following equations:

$$u = (x-1)*(x-1)*(y-1)*(y-1)*(z-1)*(z-1), \tag{2.1}$$

$$\mathbf{c} = 1.$$

As Mesh we have two spheres as is Figure 1. Knowing the analytical solution in each of the nodes is (2.1), we have the next result (Figure 2) in the Nodes in Γ_2 where the difference between the calculated and the analytical solution is shaded. The generated figures of the test are Figure 3 and 4 for the calculated and analytical solution respectively.

With a calculated error of $3.78\mathbf{x}10^{-4}$ using the following

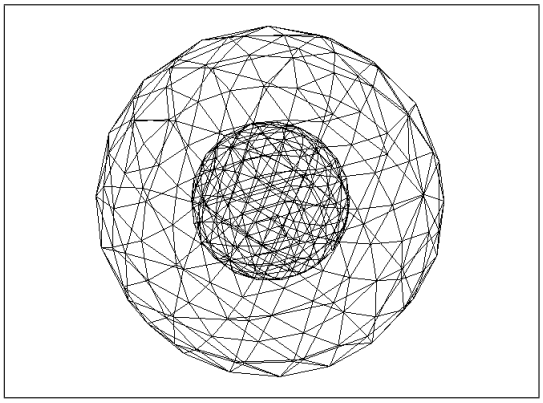


Figure 1: Mesh use for the FEM Test 1

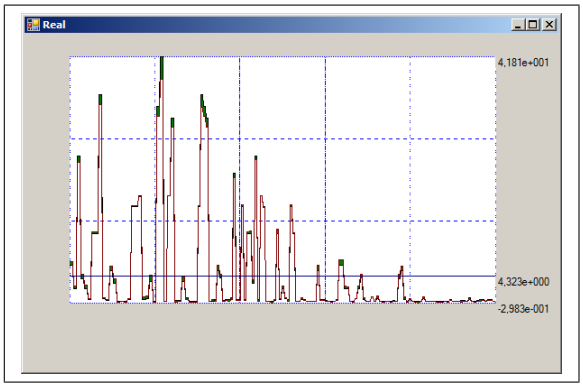


Figure 2: Results of FEM Test 1

formula:

$$e = \frac{\frac{\sum_{i=0}^{N_t}(|u_{ta}[i]-u_{tc}[i]|)}{\sum_{j=1}^{N_t}(|u_{ta}[i]|)}}{n_t}. \qquad (2.2)$$

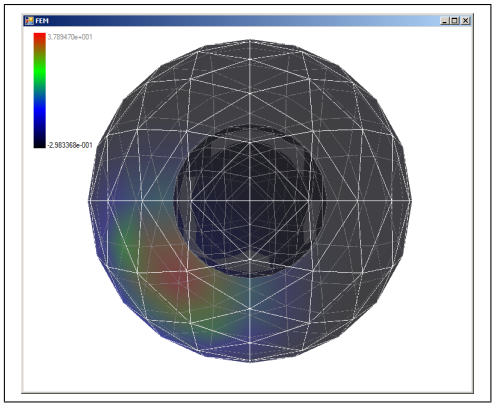


Figure 3: Calculated solution of FEM Test

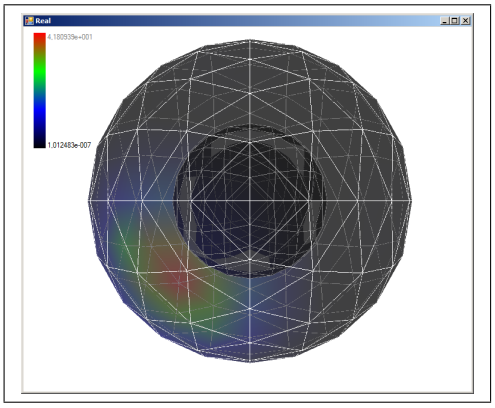


Figure 4: Analytical solution of FEM Test

The software can be applied to any geometry and with different mathematical functions as in Figure 5. With the Boundary Conditions described in Figure 6 where the color red Represents the Neuman Boundary Conditions and the black the Dirichlet Boundary Conditions.We applied the FEM and we have a calculated error of $5.42\mathbf{x}10^{-3}$. We can see the result in the Figure 7.

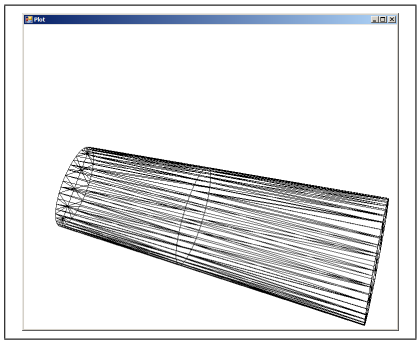


Figure 5: A simplified Mesh for a Wing with 223 Nodes and 1498 Tetrahedra

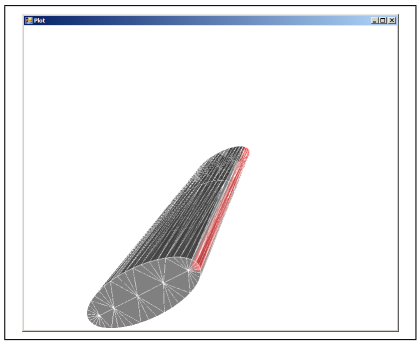


Figure 6: Boundary Conditions for the Wing Mesh

We refine the mesh as in Figure 5. and make the anal-

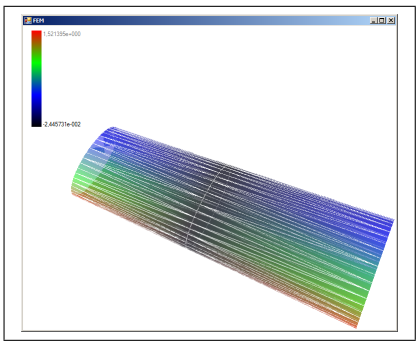


Figure 7: FEM Results for the Wing Mesh

ysis again, we can see the results in Figure 9, having an error of $1.79\mathbf{x}10^{-3}$, we can conclude that error gets smaller as we have more elements.

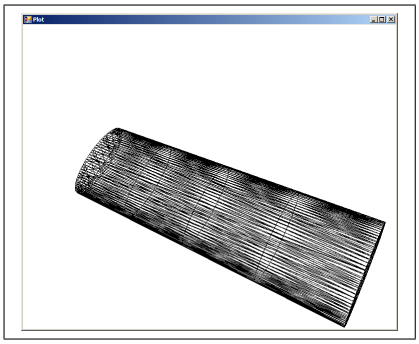


Figure 8: Second Mesh for a Wing with 1803 Nodes and 10864 Tetrahedra

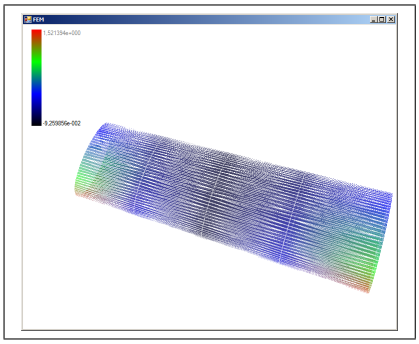


Figure 9: FEM Results for the Wing Mesh 2

3 Conclusions and Future Work

The software was created using the Visual C# Express 2010, a free open compiler from Microsoft. It was tested over a Window 7 computer with a 64 bit processor. The time needed to make the analysis is about of 2 minutes, and it has been tested with up to 50,000 elements, which takes about 15 minutes.

The software was successful in solving the Laplace equation in 3 dimensions which is very useful in calculating the propagation of heat and electrostatic energy which is used in mechanical and mechatronical design and can be applied to the design of aero spatial equipment. It was tested with several geometries and different test functions.

As future work the second version of the software will implement Vibration analysis and Steady State analysis and the implementation of the Delaunay triangulation for local Mesh Refinement.

References

[1] Geuzaine C. and Remacle J. Gmsh: a three- dimensional finite element mesh generator with built-in pre- and post- processing facilities. Web.1997-2008.$< http : //www.geuz.org/gmsh >$.

[2] OpenGL - The Industry Standard for High Performance Graphics. Web. 21 May 2011. $< http : //www.opengl.org/ >$.

[3] R. Ridge, " Tao framework.". Web. 21 May 2011. $< http : //www.taoframework.com/Home >$.

[4] Gockenbach, Mark S. Understanding and Implementing the Finite Element Method. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2006. Print.

[5] Blum, Howard. "Solid Mechanics and Its Applications # 157: MATLAB Codes for Finite Element Analysis: Solids and Structures by A. J. M. Ferreira - Powell's Books." Web. 09 May 2011.

[6] Reddy, J. N. An Introduction to the Finite Element Method. McGraw-Hill, 1993. Print.

[7] Petyt, M. Introduction to Finite Element Vibration Analysis. New York: Cambridge UP, 2010.

[8] Dhondt, Guido D. C. The Finite Element Method for Three-dimensional Thermomechanical Applications. Hoboken, NJ: Wiley, 2004. Print.

[9] Liu, Gui-Rong, and S. S. Quek. The Finite Element Method: a Practical Course. Oxford [u.a.: Butterworth-Heinemann, 2006. Print.

[10] Polycarpou, Anastasis C. Introduction to the Finite Element Method in Electromagnetics. [San Rafael, Calif.]: Morgan & Claypool, 2006. Print.

[11] Press, William H. Numerical Recipes: the Art of Scientific Computing. Cambridge, UK: Cambridge UP, 2007. Print.

[12] Lucquin, Brigitte, and Olivier Pironneau. Introduction Au Calcul Scientifique. Paris: Masson, 19 9. Print.

[13] "Finite Element Method." Wikipedia, the Free Encyclopedia. Web. 09 May 2011. $< http : //en.wikipedia.org/wiki/Finite_element_method >$.