Graphical Interface for Electromagnetic Problem Solving Using Meshless Methods

Leonardo Augusto Ferreira, Alexandre Ramos Fonseca,
Department of Computer at Federal University of Vales do Jequitinhonha e Mucuri, UFVJM, 39100-000 Diamantina – MG, Brazil (e-mail: leauferreira@gmail.com, arfonseca@ufvjm.edu.br)
Naïsses Zoia Lima, Renato Cardoso Mesquita, Gabriel Cardoso Salgado
Department of Electrical Engineering at Federal University of Minas Gerais, UFMG, 31270-000 Belo Horizonte – MG, Brazil (e-mail: naisseszoia@gmail.com, renauto@cpdee.ufmg.br, gabrielcardososalgado@gmail.com)

Abstract— It is common to find a software with a graphical interface that uses the finite element method or the finite difference method to solve electromagnetic problems. However, there are just few software which use meshless methods to solve these problems, and none with a graphical interface. This project aims at developing a graphical software capable of solving Electromagnetic Problems by meshless methods. Among these meshless methods, it is possible to mention the Element Free Galerkin Method (EFG), Meshless Local Petrov Galerkin (MLPG) and Point Interpolation Methods (PIM). The main idea is to enable the user to design the geometry of a problem, choose materials in the domain, impose boundary conditions to choose a meshless method to obtain a solution and view the results in a graphical form. This software-user interaction will be intermediated in the domain by a graphical user interface that makes transparent the using of the meshless method using an intuitive toolkit responsible for the problem design and for the solution visualization.


I. INTRODUCTION

Electromagnetic problems are usually solved through numerical methods like Finite Elements Method (FEM) or Finite Difference Method (FDM). These traditional methods use a mesh to obtain the solution of the problem. The FEM uses an unstructured mesh to model the problem while the FDM uses a structured mesh, a grid. The Figure 1 shows elements of an unstructured mesh on the left and a structured one on the right.
Fig. 1. Elements of unstructured mesh using FEM (left) and structured mesh using FDM (right).

An alternative for these traditional methods are the meshless methods [2]. These methods don’t require a mesh for problem solving. Meshless methods, which inherit many advantages of FEM and yet need no explicit mesh structure to discretize geometry. The Meshless method uses a nodes distribution into the domain and its boundaries to obtain the solution, as shown in Fig. 2.

Fig. 2. Nodes distribution in meshless method

Features of absence of a mesh:
- No mesh alignment sensitivity. This is a serious problem in mesh based calculations.
- In problems with large deformations of the domain or moving discontinuities a frequent remeshing is needed in mesh-based methods.
- No mesh generation at the beginning of the calculation is necessary. This is still not a fully automatic process, especially not in complex three-dimensional domains, and may require major human interactions.

Meshless Methods, but not all of them, need just background mesh for integration without care about mesh quality. The accuracy of meshless methods is, generally, greater than traditional methods.

The goal of this project is to implement a software called MFree Editor capable of obtaining a solution for an Electromagnetic problems through meshless methods, offering a friendly interface which encapsulates the presented complexity in all applications involved in this project. The MFree Editor provides a toolkit that allows an electromagnetic problem to be designed by the user, solve it and analyze results through graphs and color maps.

II. TOOLS

This section brings up the support tools and it shows their importance in the software development.
A. Qt Framework

Qt is a C++ framework for software implementation with Graphical User Interface (GUI) software implementation [2]. This framework follows the concept “write once, compile anywhere” [3] which allows the applications implemented with Qt to run in several operating systems (OS). Thus, the user can implement a function using Qt without worrying about which OS will compile it because the Qt Framework will solve it internally. Beside this features, the Qt uses an object oriented approach which is presented in this project.

The Qt Framework is important to the MFree Editor due to all of the features presented in this framework that guarantee a cross platform application, simplifying the implementation and allowing the creation of a GUI.

B. CGAL

The language used to write the Computational Geometry Algorithms Library (CGAL) is C++ and it comprises three main parts: The first part is the kernel, which consists of a primitive geometric objects fixed set, non-modifiable, and the operations on these objects. The objects are expressed both as stand-alone classes that are parameterized by a representation class, which specifies the types of underlying number used for calculations and as members of the kernel classes, which allows for more flexibility and adaptability of the kernel. The second part is a collection of basic geometric algorithms as well as data structures which are parameterized by trait classes defining the interface between the data structure or algorithm and the primitive objects used by them. In many cases, the kernel classes provided in CGAL can be used as trait classes for these data structures and algorithms. The third part of the library encompasses a support for some non-geometric facilities, such as circulators, random sources, I/O support for debugging and for interfacing CGAL to several visualization tools [4].

This library was widely used in both the MFree Editor and MFree Framework applications development in this project, from points, edges and polygon representations up to complex algorithms that are offered by CGAL.

C. OpenGL

The Open Graphics Library (OpenGL) [5] is a graphic library which offers 120 distinct commands for creating graphical representations. It is possible to create 2D and 3D designs through a set of primitive available in OpenGL.

Ensuring the performance, this library do not implement complex commands, but through the basic primitive is possible create simple or complex structures.

This library is applied when the user designs the problem and during the phase of results visualization. The OpenGL provides graphics representations for the model and solution as the color of the maps and geometry problem. The CGAL library offers computational representation to represent the same thing.
D. **Boost**

The Boost is a collection of several libraries having a wide use spectrum and being compatible with Standard Template Library (STL) in C++ [6].

In order to ensure efficiency and flexibility, the Boost makes extensive use of templates, becoming a source of research for the generic programming development which is the programming strategy used in CGAL and MFree Framework.

E. **Triangle**


The user-entered modeling problem is edited by the MFree Editor in a format compatible with Triangle. Then the MFree Editor makes the calls to the Triangle, interprets the geometry of the problem and creates two files that are relevant to the application namely one with a distribution of nodes (which are used by the meshless method) and other with a distribution of elements (that could be used or not by the meshless method). Thus, Triangle performs the communication between the MFree Editor and the MFree Framework.

F. **Doxygen**

The Doxygen creates code documentation generated from comments and it made on the source code during the implementation. The documentation can be generated in several formats and, the most used is html and Latex. This software can be used to document codes written in several programming languages such as C, C++, Java, Objective-C, Python, IDL (CORBA and Microsoft variations), Fortran, VHDL, PHP, C # and others [10].

III. **PROGRAMMING PARADIGM**

Programming paradigms are implementation models that determine how the software source code must be created. An application is written over rules of a programming technique that presents a standard behavior, which is easier to systematize.

A. **Object Oriented Programming**

The Object Oriented Programming (OOP) is used in Qt Framework, therefore, it is also used for the MFree Editor development. The OOP allows the software of being implemented by using characteristics presented in the real world. The Object Oriented Programming models the real-world objects with software counterparts [12]. This technique is based on three main features: polymorphism, inheritance and encapsulation.

The polymorphism allows the system to decide which method containing multiple implementations will be used at a time. The method choice is made at runtime leaving the decision upon the situation.

Inheritance is the mechanism used by which an object accumulates the methods and properties of another object. It supports the concept of hierarchical classification.
Encapsulation means hiding the internal details of an object, i.e. how an object does something. This is a technique used to protect the information in an object from the other object. It hides the data for security such as making the variables private, and exposes the property to access the private data which would be public.

B. Generic Programming

The generic programming is about generalizing software components so they can be easily reused in a wide variety of situations. This process focuses on finding commonality among similar implementations of the same algorithm, providing suitable abstractions so that a single, generic algorithm can cover many concrete implementations.

The generic programming deals with finding abstract representations of efficient algorithms, data structures, and other software concepts, and with their systematic organization. The generic programming focuses on representing families of domain [12].

IV. DESIGN PATTERNS

A design pattern is a language-independent codification of a solution to a common programming problem expressed, in such way that it can apply to many contexts. Firstly, you can think of a pattern as a particular clever and insightful way to solve a particular class of problem. It appears that a team has worked out all the angles of a problem and it has come up with the most general, flexible solution for that type of problem [13].

A. Singleton

Among the design patterns, the most common is called Singleton. This pattern ensures that a class has only one instance, and provides a global point of access to it [14].

The Singleton also prevents the user from making an unsuitable use of the code. This pattern is implemented in connection with Factories in the MFree Editor. This matching (Singleton – Factories) is common in several applications.

B. Abstract Factory

It provides an interface for creating families of related or dependent objects without specifying their concrete classes [14]. The Abstract Factory consists of the implementation of the Factory Method inside an independent class which provides more liberty regarding to object manipulation. This one is a generalization of the Factory Method.

Abstract Factory is usually combined with Singleton because only one instance of the Factory is needed in the code with a global point of access to it.

C. Prototype

It specifies the kinds of objects to create using a prototypical instance, and it creates new objects by copying this prototype [14].

One of the main characteristics of the Prototype is to create an instance of each class automatically...
by the time the application is loaded. This pattern is used to create a number of tools available in the MFree Editor. The user uses fewer lines of code by using the prototype, consequently improving productivity.

D. Memento

Without violating encapsulation, it captures and externalizes the internal state of an object so that the object can be restored later to this state [14]. It is possible, by the implementation of this Memento pattern, to build functionality undo (Ctrl + Z) and redo (Ctrl + Y) in the MFree Editor.

E. State

It allows an object to alter its behavior when its internal state changes. The object will appear to change its class [14].

This design pattern can be applied in two conditions. The first occurs when the object behavior changes at runtime and this behavior depends on the state of the object. The second occurs when an object has several states and several complex conditional operations.

Through the implementation of the State, the transition between the states becomes clear due to introduction of separated objects for different states. This pattern may be noticed when the user switches the tools in menu bar: the application is the same, but its internal state changes.

F. Command

It encapsulates a request as an object, thereby letting you parameterize clients with different requests, row or log requests, and support undoable operations [14].

The purpose of this implementation is to encapsulate a request as an object, enabling the configuration of different clients with different requests and call sites. In other words, it means releasing the choice of a function from where this function was called.

V. MAIN SOFTWARE

The main software involved in this project are the MFree Editor and the MFree Framework. The Fig. 3 represents the main application model with its distinct modules.
The modules of the MFree Editor are Editor and Visualization. The applications that represent the modules of the MFree Framework are MFree Editor App, MFree Nodes Femm and MFree Cell Grid Creator. The Triangle (section II, item E) is responsible for helping the communication between the MFree Editor and the MFree Framework.

A. MFree Editor

The MFree Editor is a tool that, not only offer the possibility of electromagnetism problem solving (electrostatic and magnetostatic) using the main meshless methods, but also provides a graphical interface that presents pre-processing, processing and post-processing elements for users [15].

1) Editor

The editor is the biggest module involved in this application. This module has an intuitive interface that improves the user experience by using the software. It is not necessary for the user to have knowledge on how to operate the MFree Framework to use the application, as its complexity is abstracted by the interface created in the Editor Module.

Fig. 4 shows the home screen editor. With the editing tools, the creation and modification of problems became easier, once that the problem geometry is previously created.
The MFree Editor is responsible for calling all applications when necessary. Fig. 5 represents a model with the management of all applications. Applications related to MFree Framework are represented by squares and MFree Editor, circles.

2) Visualization
The display module allows the solution to be interpreted graphically by the user. This graphical representation is shown through several components such as color map, field lines and three-dimensional perspectives.
The graphical representation of the solution simplifies its understanding. If this integration among the applications was not made, the user would have to use a third application to obtain the solution through graphics, contours or any other strategy.

B. MFree Framework

A framework is a reusable, “semi-complete” application that can be specialized to produce custom applications [16].

The MFree Framework is a framework for meshless methods implemented with generic programming [17]. This framework is crucial for the data processing. There are three applications that are directly linked to solve the problems created by the user. The three applications derived from this framework are MFree App Editor, MFree Nodes Femm and MFree Cell Grid Creator.

Today the MFree Framework provides the following meshless methods:

- Element Free Galerkin (EFG) [1]
- Point Interpolation Method (PIM) [1]
- Meshless Local Petrov-Galerkin (MLPG) [1]
- MLPG mixed [18]

1) MFree App Editor

The MFree App Editor is a MFree Framework application that is designed to be integrated with the MFree Editor. The MFree Editor App is already prepared to receive the files produced by the Triangle and properly processed by the projects MFree Nodes Femm and MFree Cell Grid Creator.

2) MFree Nodes Femm

The use of MFree Nodes Feem comes from the necessity of meshless methods to use a distribution of nodes in the problem domain. Thus, the vertices were chosen by of a triangulation performed by Triangle as our distribution. The MFree Nodes Femm translates the file containing the nodes of the triangulation made by Triangle in a way that is understood by the MFree Framework.

3) MFree Cell Grid Creator

The MFree Cell Grid Creator works in the same way as the MFree Nodes Femm does, however in this case it is the interpretation of the data made by the grid integration. Some of the meshless methods that are made available for the MFree Framework depend on a grid for doing numerical integration, for instance.

VI. Results

We have presented results for the same problem presented in [19]. The local point interpolation method (LPIM) is used with visibility criteria changed to treat boundaries among different materials.

Fig. 6 shows the initial modeling often electromagnet. As the geometry of this model is symmetric, only half of the problem is represented.
After modeling the problem, the user needs to set the materials and boundaries. These settings are made in configuration dialog showed in Fig. 7.

There are three types of material involved in this problem: air, iron and copper. The materials are set in all problem faces.

In the next step the meshless method is going to be set into dialog method. There is a default configuration that allows user uses a standard solution, but changes may be done if necessary. Fig. 8 brings up the dialog for doing this.
Fig. 8. Dialog to configure the meshless method to be used.

After the settings, all data related to the geometry problem, materials, boundaries, and choice of meshless method are sent to the MFree Framework. In turn, the framework processes the data and sends the solution to the visualization module. The results may be seen in several representations.

Fig. 9 shows the representation of equipotential lines and distribution of the magnetic vector potential.

Fig. 9. Representation of equipotential lines and distribution of the magnetic vector potential.
Fig. 10 shows a tridimensional perspective of distribution of the magnetic vector potential.

![Fig. 10. Tridimensional perspective of distribution of the magnetic vector potential](image)

Fig. 11 shows a magnetic induction distribution.

![Fig. 11. Magnetic induction distribution](image)

VII. CONCLUSION

The aim of this project implementation was attended the demand and proved its functionality. The graphical interface allows the user to describe the domain of an electromagnetism problem (electrostatic and magnetostatic) in two dimensions, detailing the problem and its solution. The geometry is processed by the MFree Framework according to the user's needs and the graphical
interface makes transparent its use. After the problem processing, the MFree Framework, through multiple files, sends the solution to MFree Editor, which in turn graphically details solution in the visualization module.

As future works, we propose:

- Figure translation;
- New editing tools;
- New meshless methods implementation;
- Expansion for three dimensions problems.

ACKNOWLEDGMENT
The authors acknowledge the financial support of FAPEMIG.

REFERENCES