GID INTERFACE FOR THE PARAMETRIC GENERATION OF SIMPLIFIED BRAIDED-WIRE SHIELDS GEOMETRIES[*]

O. Fruitos, R. Isanta, R. Otín and R. Méndez

CIMNE - International Center for Numerical Methods in Engineering
Parque Tecnológico del Mediterráneo, Edificio C3 - Av. del Canal Olímpic, s/n
E-08860 Castelldefels (Barcelona, Spain)
e-mail: metalform@cimne.upc.edu, web page: http://www.cimne.com/

Key words: Finite element methods, Numerical analysis, Electromagnetic compatibility, cable shielding, GiD, braided-wire geometries, TCL-TK.

1 INTRODUCTION

In this paper we are going to present a program that generates automatically geometries of braided-wire shields inside GiD[i]. The general objective of this work is to develop a numerical tool able to calculate the transfer impedance of these braided wires shields. The transfer impedance of a braided wire is a magnitude that characterizes the effectiveness of its shielding. Low transfer impedance indicates a good shielding. The development of this numerical tool will help to optimize the design of braided wires (better shield with less material) and to gain insight in the physics involved. Also, the advantage of using a numerical tool based on the finite element method instead of analytical calculations is that we will be able to apply it for general complex geometries and materials. The numerical tool we are going to customize to this purpose is called ERMES[ii,iii], which is a finite element code which uses GiD as pre and post processor.

To compute the transfer impedance, ERMES needs the GiD geometry of a braided wire shield. But the generation of these geometries is, by far, the most time consuming task in the whole computational process. This is why we need a program able to make these geometries automatically. In this paper is presented a basic module which generates perforated tubes. In a future work this program will be extended to generate real braided-wire geometries.

The geometry generator presented here is fully integrated in GiD and has a user-friendly interface. After introducing the parameters of the perforated tube in a window, the geometry is created. This geometry is then ready to be used by ERMES. TCL-TK and TKWIDGETS had been used to make the windows, while the geometry is generated with a C++ code and imported to GiD as a batch file. We will show how this program has been developed, its relation with GiD and ERMES and some examples of the type of geometries that can be made with it.

[*] *This work is part of the EU project HIRF-SE (High Intensity Radiated Fields Synthetic Environment), European Commission, 7th European Union Marco Program, ref.: 205294*

2 OVERVIEW OF THE PARAMETRIC TOOL

The Problem Type has been programmed in Tool Command Language (Tcl) and Tool Kit (Tk) and it is fully integrated in the GiD environment. The interface allows the creation of simplified braided-wire geometries. To do so, it is provided users with user-friendly windows to define the specific geometry parameters for each case. There are four study cases altogether (see fig. 1).

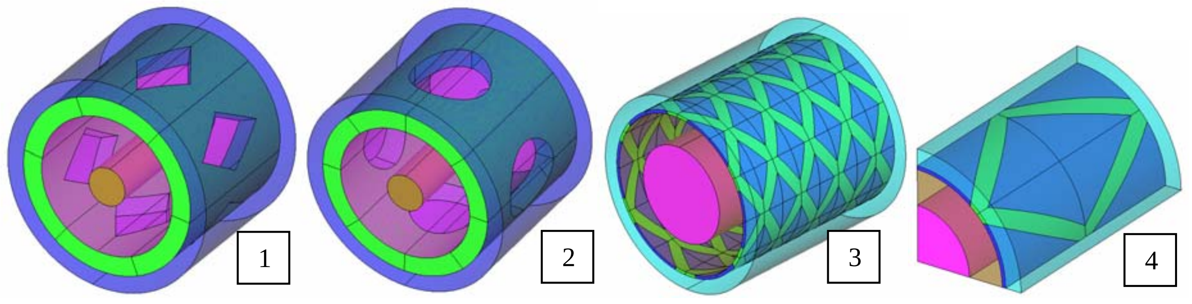
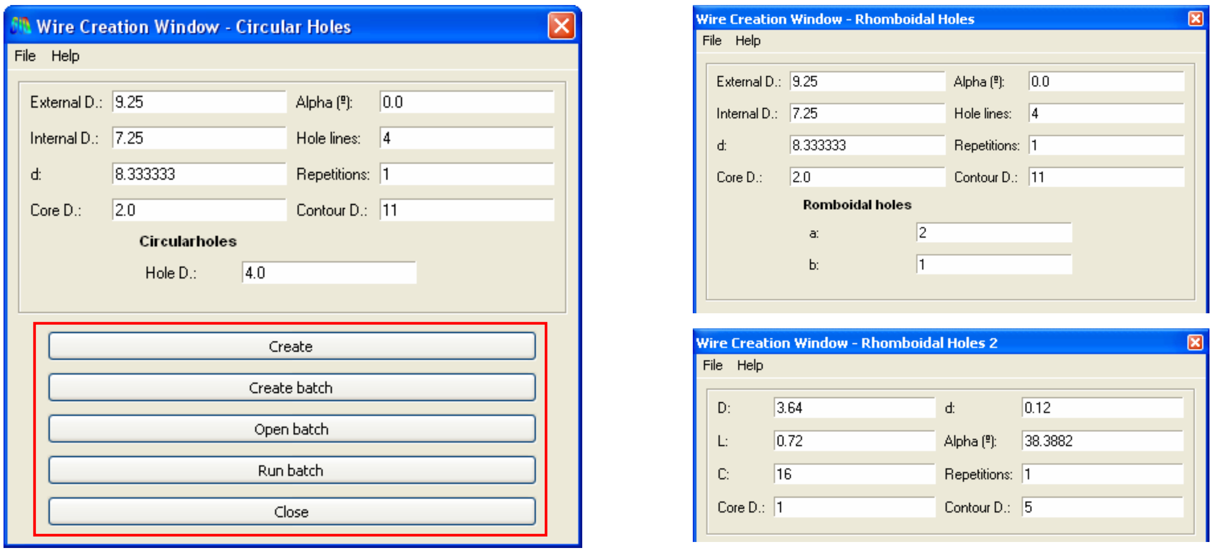


Figure 1: Interface study cases: rhomboidal holes (1), circular holes (2), rhomboidal holes type 2 (3) and rhomboidal holes type 2 with symmetry (4).

Every case has a different window and input data but they have five buttons in common to manage the tool (fig. 2 on the left, framed in red). Fig. 2 shows the windows of three cases namely cylinders with circular holes (left), cylinders with rhomboidal holes (right and above) and cylinders with rhomboidal holes type 2 (right and below).



Wire Creation Window - Circular Holes

File Help

External D.:	9.25	Alpha (º):	0.0
Internal D.:	7.25	Hole lines:	4
d:	8.333333	Repetitions:	1
Core D.:	2.0	Contour D.:	11

Circularholes

Hole D.: 4.0

Wire Creation Window - Rhomboidal Holes

File Help

External D.:	9.25	Alpha (º):	0.0
Internal D.:	7.25	Hole lines:	4
d:	8.333333	Repetitions:	1
Core D.:	2.0	Contour D.:	11

Romboidal holes

a: 2

b: 1

Wire Creation Window - Rhomboidal Holes 2

File Help

D:	3.64	d:	0.12
L:	0.72	Alpha (º):	38.3882
C:	16	Repetitions:	1
Core D.:	1	Contour D.:	5

Figure 2: Appearance of three of the input data windows.

The last one of the windows (rhomboidal holes type 2 with symmetry) has exactly the same parameters than the Rhomboidal Holes 2 window, shown in figure 2 (right and below). In this case, however, the resulting geometry is already prepared to be used in ERMES environment. In both “Rhomboidal Holes type 2” and “Rhomboidal Holes type 2 with symmetry” cases, the parameters that the user manage are those commonly used in the coaxial wires industry viz:

D: Inner diameter of the shield.
L: Width of carriers.
C: Number of carriers in the braid.
Core D: Diameter of the core.
d: Thickness of the shield.
Contour D.: Diameter of the contour cylinder.
Alpha: Angle between the main cable axis and wires.
Repetitions: Repetitions of the periodic portion.

3 THE IMPLEMENTED C++ CODE IN BRAIDED WIRES

Braided Wires kernel consists of four performer files, one for each type of geometry, and one file which manage the others. The input data of the C++ code is a file with the properties filled in the different interface windows. The interface creates this file as soon as the user pushes either to “create” or to “create batch” buttons (fig. 2 left) and just before the C++ code is executed. The kernel creates a batch file (“.bch” extension) with all the necessary steps to generate the desired geometry in GiD. From the interface windows, users can create the geometry directly, open the batch file, close the batch file, or run the batch file (framed buttons in red in fig. 2).

Braided Wires C++ code writes in the batch file a list of commands which create every case taking into account the parameters that the user has inputted. This is extremely simple using the GiD environment because it is possible to know, after doing any action, the commands that this software has used during these actions. Braided Wires C++ code acts as a GiD user but automatically and much faster. The libraries which the kernel uses are iostream, fstream, string and cmath. The calculations performed internally by the C++ code are mainly geometrical.

C++ Braided Wires kernel uses the parameters shown in fig. 4 to create “rhomboidal holes type 2 with symmetry” geometries. The key points to create this type of geometries are “a”, “b”, “c”, and “d”, which are pointed in fig. 3. When the “a”, “b”, “c” and “d” points are found, the kernel uses these to create work planes. Then, there are intersections between lines and surfaces and finally several volumes are created from surfaces. The parts of the final geometry are the core, the dielectric, a simulation contour and the wires.

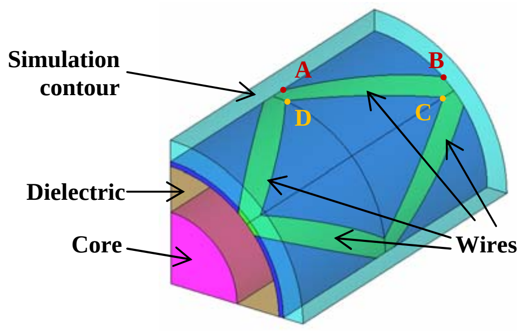


Figure 3: Key points and parts of the geometry.

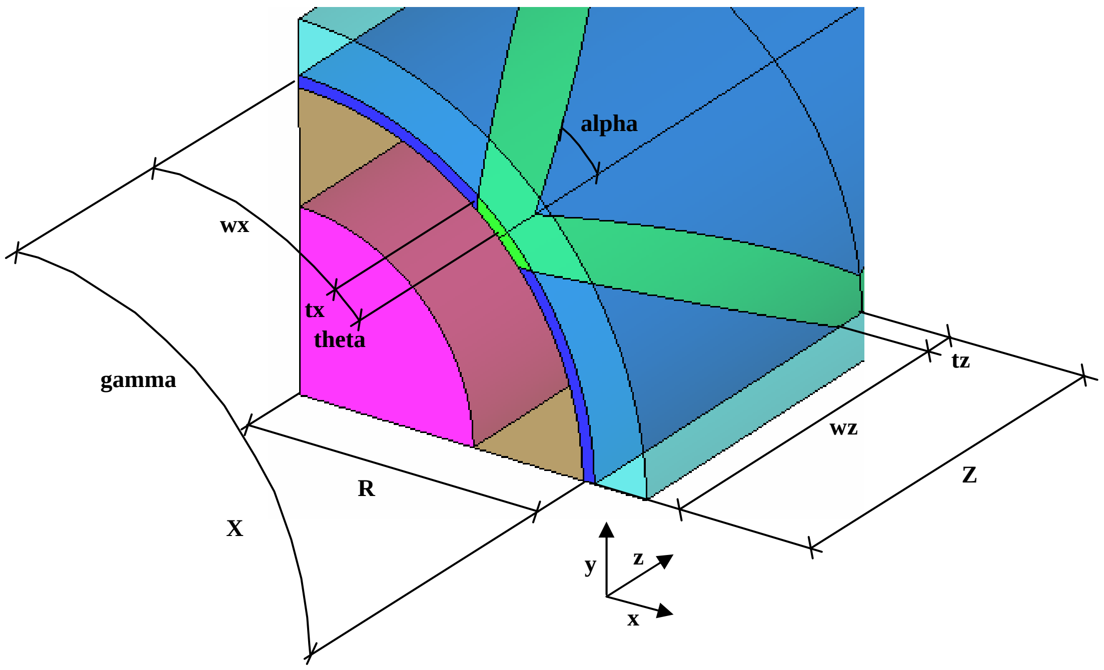


Figure 4: Geometry parameters of rhomboidal holes type 2 with symmetry case.

5 CURRENT WORK

The future C++ kernel and tcl/tk developments will be focused in extending the tool to generate real braided-wire geometries. There are two ways to work in this issue. On the one hand it is possible to work with the whole carriers as if the several wires were fused together. On the other hand it is possible to work with the wires separately. In both cases, the geometrical equations involved are under study. Also, the finite element model used to compute the transfer impedance is under study[ii] and new geometrical developments might be required. The interface will also be improved to include these new geometric typologies and to make it more user-friendly and pleasing to the eye.

REFERENCES

---

[i] http://www.gidhome.com

[ii] R. Otin, J. Verpoorte, and H. Schippers, “A Finite Element Model for the Computation of the Transfer Impedance of Cable Shields”. IEEE Transactions on Electromagnetic Compatibility. Submitted.

[iii] R. Otin, “Regularized Maxwell Equations and Nodal Finite Elements for Electromagnetic Field Computations”, Electromagnetics, vol. 30, pp. 190-204.