A Geometry Description Language for 3D Electromagnetic Analysis Codes

Todd Hubing, Clara Hong-Him Lim, and James Drewniak
University of Missouri-Rolla

With the proliferation of 3D electromagnetic modeling software, many researchers are finding that the process of defining the input and interpreting the output is overly complex and unintuitive. Graphical user interfaces (GUIs) can help the user to visualize input geometries and output current and field distributions, but representing three dimensional data on a two-dimensional computer screen is inherently a complex process. The steep learning curve associated with most user interfaces prevents many EM modelers from experimenting with different EM modeling codes.

An Initial Solution

In a previous paper [1], a user interface (input only) was described that employed commercial Computer Aided Drawing (CAD) software. The idea was to isolate the development of the numerical EM modeling software from the development of the user interface. In this way, EM modeling code developers could concentrate on developing the best modeling software and development of the CAD interface could be left to programmers who specialized in that area. A relatively simple translation code was used to link the CAD software to the EM modeling code. The translation code read the 3D geometry information, prompted the user for any additional information (e.g. source amplitudes or material constants), checked for errors, and wrote a file suitable for exporting to the EM modeling code.

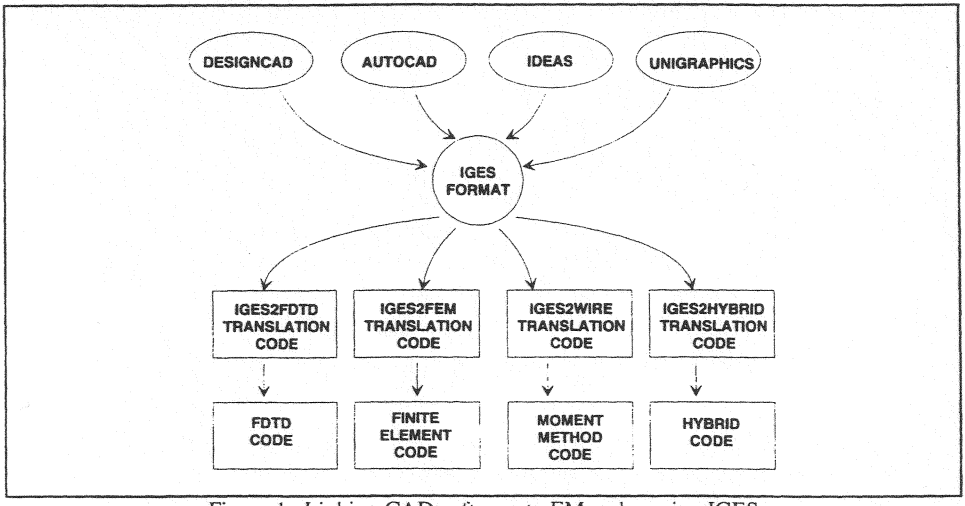

Figure 1: Linking CAD software to EM codes using IGES

In theory, a translation code written to interpret a standard graphics format, such as IGES or DXF, could accept input from a variety of CAD software packages. This approach, illustrated in Figure 1, would allow EM modelers to select the CAD software best-suited to their own needs to use as a graphical front-end to their EM modeling codes. If each 3D EM modeling code used was provided with a similar translation code, the modeler could analyze a given geometry using more than one technique without having to enter the geometry data each time. Also the code user could become proficient with a single CAD package and avoid having to learn a new user interface for every new EM modeling code. Using CAD software to create an IGES file is usually much more efficient than using a text editor and/or special subroutines to create a native format file. With the translation code, the standard graphics format effectively replaces the native input format of the modeling code.

This approach is not without drawbacks however. Currently there is no truly standard graphic format. IGES and DXF files can vary significantly depending on the CAD software used, the computer platform, and the techniques used to create a particular geometry. Also, IGES and DXF files were not optimized for storing the data required by most EM modeling software. The files are long and complex and they contain much information that is unusable by EM modeling codes. In addition, the IGES and DXF formats do not provide a standard method for conveying EM source or material information. In order to store an input geometry that the user may want to modify later, both the CAD file and a file containing the source and material information must be saved.

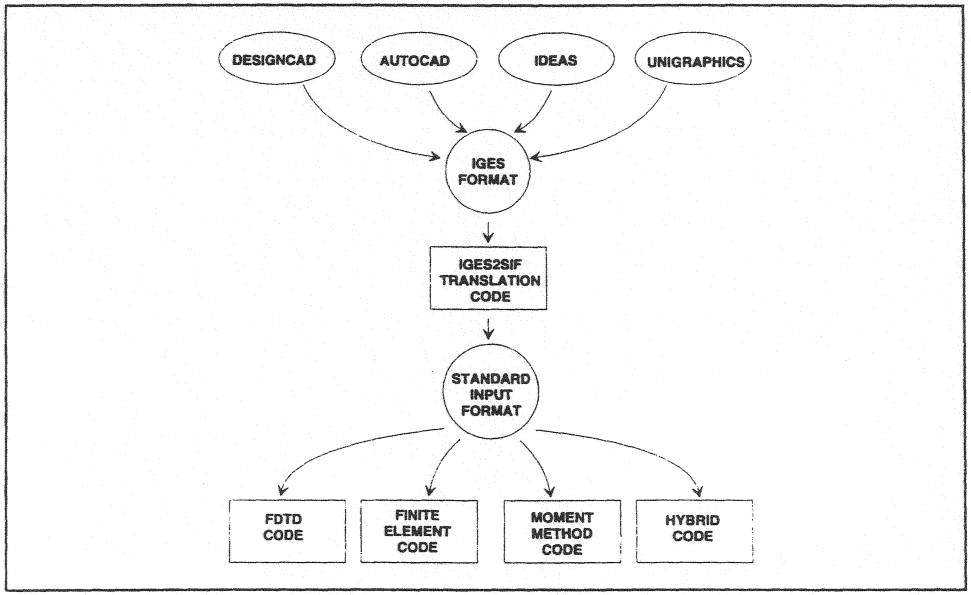

Figure 2: Linking CAD software to standard input codes

A New Approach

If there were a standard input file format that could be read by a variety of EM modeling codes, then only a single translation code would be required to link CAD software to the electromagnetic modeling software as indicated in Figure 2. Furthermore, if the standard input format were concise and simple to understand, then perhaps in many cases, it would be possible to eliminate the graphical user interface altogether.

To some extent, this is the approach used by the Numerical Electromagnetics Code (NEC). The relatively simple text input format employed by NEC allows many users to create and/or edit input geometry information without the aid of a graphical user interface. For complex geometries, a number of graphical interfaces are available all of which generate NEC input files in the same text format. Because of this, the NEC code is relatively platform independent. Code developers can make changes in the NEC code without worrying about the user interface, and users can choose the particular user interface best-suited to their own needs.

Of course, the NEC input format is optimized for the NEC code, which is a surface integral technique. It would not be suitable for another EM modeling code that used a different analysis technique (e.g. a finite element code or an FDTD code).

The remainder of this paper describes a new input file format (or, more specifically, a "geometry description language") that is optimized for use with EM modeling codes, but flexible enough to be used in codes that employ a variety of modeling techniques. This input format is currently used by three separate EM modeling codes at the University of Missouri-Rolla. One is a finite-element modeling code employing nodal elements. One is a hybrid FEM/MOM edge-element code. The third code employs a finite-difference time-domain technique.

The Input File Format

The new input file format was developed with three primary objectives:

1. That it be as simple and intuitive as possible. For simple geometries, the user should be able to visualize the input configuration simply by reading the input file.
2. That it be concise. In most cases, the user should be able to type in the entire input file using only a text editor.
3. That it be flexible. EM codes using different modeling techniques have vastly different capabilities and requirements.

Each line of the input file is either a data line or a comment (blank lines are ignored). Comment lines begin with a # sign and may be ignored or copied to the output file by the modeling code. All data lines consist of a *keyword* followed by space-delimited parameters. The format and the number of parameters depends on the keyword. For example, the line

```
dielectric 1 1 1 8 2 8 4.2 .002 1.0 d
```

Geometry Keywords

boundary	x1	y1	z1	x2	y2	z2		- *surface of a meshed volume*
box	x1	y1	z1	x2	y2	z2		- *hollow pec surface*
conductor	x1	y1	z1	x2	y2	z2	rad seg# ntag	- *pec volume, surface, or wire*
aperture	x1	y1	z1	x2	y2	z2	name	- *hole in pec*
dielectric	x1	y1	z1	x2	y2	z2	eps sig mu m1	- *dielectric region*
esource	x1	y1	z1	x2	y2	z2	freq dir mag ph	- *electric field source*
msource	x1	y1	z1	x2	y2	z2	freq dir mag ph	- *magnetic field source*
vsource	x1	y1	z1	x2	y2	z2	freq dir mag ph	- *voltage source*
isource	x1	y1	z1	x2	y2	z2	freq dir mag ph	- *current source*
gndplane	orient	value						- *ground plane*
iterate	x1	y1	z1	x2	y2	z2	p1	- *repeat geometry*

Execution Keywords

celldim	value units	- *mesh units*
execute	p1	- *run program (y or n)*

Output Keywords

efield	x1	y1	z1		- *evaluate electric field*
hfield	x1	y1	z1		- *evaluate magnetic field*
pplot	distance		a-init	a-delta	- *generate polar plot data*
output	p1				- *generate default output*

Figure 3: Sample keyword definitions

contains the keyword "dielectric". This keyword must always be followed by at least 8 parameters. The first 6 parameters are integers that define the shape and position of the dielectric. The next 2 parameters are floating point numbers that define the relative permittivity and conductivity of the dielectric. The next parameter is optional and defines the relative permeability. The last parameter, also optional, is a character that, in this case tells the mesh generator to use an extra fine mesh in this region of the geometry.

A number of keywords have been defined, some of which are briefly described in Figure 3. Note that in addition to geometry keywords, there are keywords that affect the execution of the program and keywords that determine the output parameters.

Not all EM modeling codes will take advantage of every keyword. For example, a surface-integral code like NEC would probably ignore any *boundary* statements in its input file. EM modeling codes should be written to accept all valid keywords, but warn the user if any keywords in the input file are being ignored.

The basic keywords described in Figure 3 and their parameters have been chosen so that they are meaningful to EM codes based on a variety of EM modeling techniques. Codes that employ surface grids, codes with fixed volume meshes, and codes that require mesh generators, can all read the same input file and interpret it in a similar manner.

```
#  a shorted waveguide with dielectric loading
box          0 0 0 8 5 20
dielectric   0 0 10 8 5 20  4.0  0
esource      1 1 0 7 4 0 3000 y 1.0 0.0
```

Figure 4: Input file for a dielectric-loaded waveguide

```
# a simple printed circuit board configuration
# two rectangular loops above a ground plane
boundary     0 0 0 10 11 5
conductor    1 1 1 9 10 1
dielectric   1 1 1 9 10 2 4.0 0.0
conductor    2 2 2 8 3 2
conductor    2 4 2 8 5 2
conductor    2 2 2 3 5 2
conductor    7 2 2 8 5 2
conductor    2 6 2 8 7 2
conductor    2 8 2 8 9 2
conductor    2 6 2 3 9 2
conductor    7 6 2 8 9 2
esource      2 3 2 3 4 2 80 y 5.0 0.0
pplot        3 0 180
```

Figure 5: Input file for a printed circuit configuration

Examples

Examples of this geometry description language are provided in Figure 4 and Figure 5. Figure 4 shows the input file corresponding to a dielectric-loaded waveguide that is driven at one end with a uniform electric field and shorted at the other end. An IGES format description of this geometry is 195 lines long and does not include the source information. The input file in Figure 4, on the other hand, is 4 lines long and contains all the information required by the EM modeling code.

Figure 5 shows the input file corresponding to a printed circuit board configuration. The IGES format description requires 159 lines without the source. The input file in Figure 5 is 15 lines. Both

input files in these examples were easily generated with a simple text editor and are intuitive enough to visualize without a graphical user interface.

Conclusions

The geometry description language outlined above allows EM code users to create standard input files that can potentially be used with a variety of EM modeling codes. By establishing a standard input file format, code developers are isolated from user interface issues and they are free to develop codes that are relatively platform independent. Code users also benefit from a standard input format because they are able to choose a single compatible user-interface and use it with a variety of EM modeling codes.

So far, this input format is being used with three different EM modeling codes. One is a finite element code, one is an FDTD code, and the other is a hybrid code. Although a CAD interface that employs this input format has been developed, the authors typically find that simply creating the input files using a text editor is quicker and simpler. Development of this geometry description language is still under way. Enhancements continue to be made as the authors gain experience using the new input format and as they receive suggestions from other code users.

Reference

[1] T. H. Hubing, " Using Commercial CAD Packages to Interface with 3-D EM Modeling Codes," Proceedings of the Eighth Annual Review of Progress in Applied Computational Electromagnetics, Monterey, CA, March 1992, pp. 607-610.