A Matlab Application Programmer Interface For Educational Electromagnetics

*Stefano Selleri**
Department of Electronics and Telecommunications, University of Florence
Via C. Lombroso 6/17, 50134 Firenze – Italy
selleri@ det.unifi.it

Abstract - *An application programmer interface is devised to allow for an easy creation of virtual electromagnetic laboratories. The API provides the programmer with a fully automatic graphical user interface, freeing him from the daunting task of programming it directly through Matlab objects.*

Introduction - Engineering education in general, and The Electromagnetic Engineer education in particular requires the assimilation of large amount of math by the students. This assimilation is well known to be largely enhanced by appropriate, interactive, computer examples showing numbers and graphics relevant to the math under study.

Many packages exist which prove to be extremely powerful and useful in this approach, among these [1-3], but they are so powerful and general that they often need an University course by themselves to be mastered, and even their basic utilization can be too long to learn within a standard course devoted to Electromagnetics.

A solution can be custom made specific programs, with a simplified - possibly graphic - interface, which can be learned very quickly leaving the greatest possible time to learn Electromagnetism. The drawback of this is that building a custom program is a long and demanding task, especially if a good graphical user interface (GUI) is desired, even in the very high level programming languages provided by aforementioned packages ([1-3]).

The purpose of this work is to present a Matlab [2] suite of scripts providing an application programmer interface (API) for a virtual laboratory, defining an experiment framework (EF) able to host different kind of virtual experiments, or experiment applets (EA), allowing the educator to write just the minimum required of code, leaving all GUI and user interaction to the framework itself, yet presenting the user with a nice, easy to use, interface.

Experiment Framework API - The EF is a suite of Matlab scripts providing the basic functionality for the virtual laboratory. It opens a main window and populates the “Labs” (Fig. 1) menu by automatically browsing the directory tree where it resides, seeking for EA definitions. All other menus are defined within an appropriate, external, text file so that localization to languages different than English is immediate.

The directory tree containing the EA is organized into two levels below the root directory. Each directory of the first level defines a class of

0-7803-7846-6/03/$17.00 ©2003 IEEE

experiments. In each of these directories there is a text file, defining “Labs” menu and submenu entries, and as many subdirectories as there are experiments. EA definition and code is stored within the pertinent subdirectory. Since the EF API allows for an upward search in the directory tree for subroutines, code common to more than a single experiment need not to be duplicated but can be stored in the first level directory.

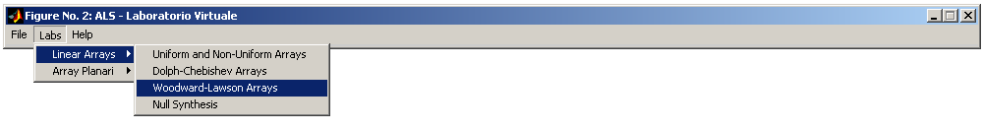

Figure 1 *– EF framework example.*

Experiment Applet – When an experiment is ran the EF reads and run the relevant EA. An EA is an object defined by a text file stating the parameters of the experiment and the name of the various Matlab user defined functions needed to perform the operations.

When an EA is invoked EF reads this definition file and creates a Matlab data structure whose entries are the experiment parameters defined in the text itself. It then creates two windows, one containing a toolbar, with one button for each possible operation defined in the experiment (Fig. 2, left) and one for the experiment outputs (Fig. 2, right). Besides these user defined operation buttons some standard actions – Save data, Load data from disk, help, exit – are also provided in the toolbar.

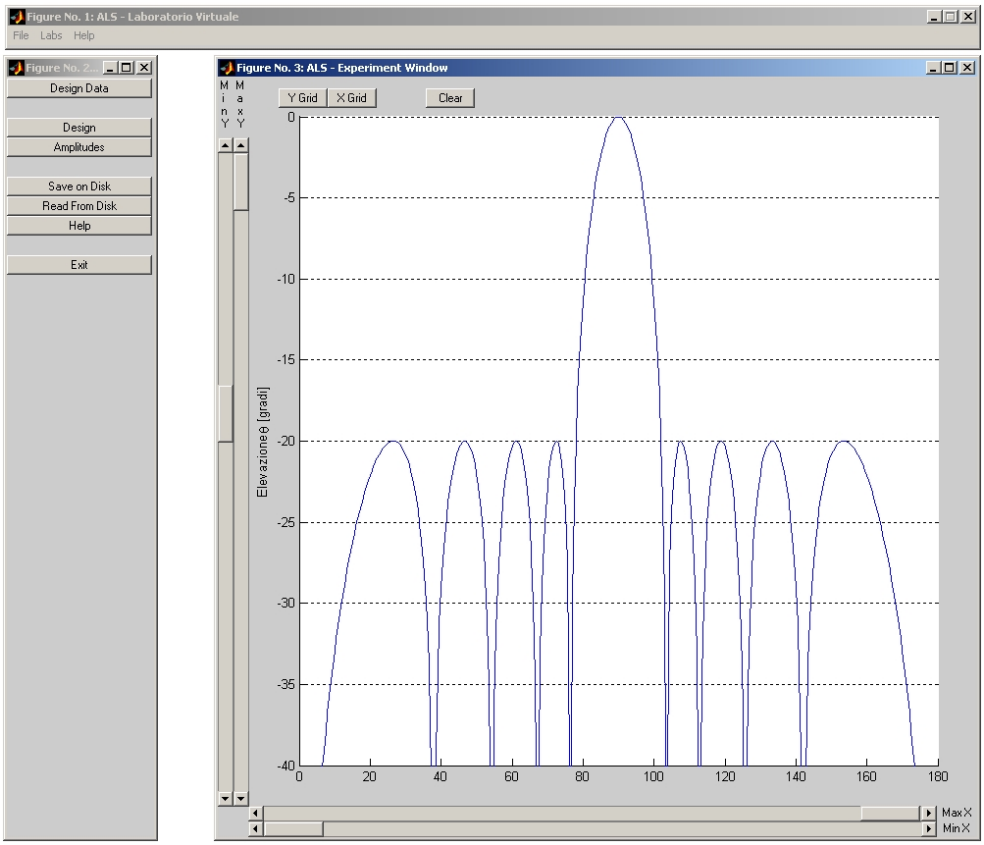

Figure 2 *– ALV running an EA featuring Dolph-Chebychev array design.*

As an example of the simplicity of an EA the design of an Array with given Side Lobe Level (SLL) with the Dolph-Chebyshev technique [4] is here analyzed.

The Listing in Fig. 3 shows the EA definition file for the experiment at hand. The first block **DataStruct** defines the parameters of the EA. Each parameter can be a scalar value, an array or a matrix, of Integer, real or complex type, with a given range of variation. This description allows the EF to automatically validate each users input. The syntax is

ParameterName=DefaultValue;Type;[Range]

In the present case an integer number, indicating the number of elements in the array. A real number, indicating the Side Lobe Level (SLL) in dB and an array of amplitudes is necessary. The amplitudes need not to be complex, strictly speaking, but this allow for more freedom in experimenting with phased arrays.

After the data structure there can be as many blocks **InputMask** and **ExecButton** as desired. The order in which they are given states the order of the buttons in the toolbar. The **InputMask** blocks define the parameters to be presented to the user for input. The Input window is built automatically by the API, and a check on the values provided against type and range is automatically performed (Fig. 4, left). In case that some operations need to be made on the newly entered values before any other is performed it is possible to define a Matlab function to be executed after the input is successfully completed via a **UserCallback** definition. The **ExecButton block**, on the other hand, just defines a user defined Matlab function to be executed.

```
% Dolph-Chebyshev Design
%%%%%%%%%%%%%%%%%%%%%%%%%%%
Begin DataStruct
% Number of Elements
N=2;Int;[2,20]
% Side Lobe level [dB]
SLL=-20;Real;[-100,0]
% Element Amplitudes
A=[1.,1.];Cplx;[0,Inf,0,6.2832,0]
End DataStruct
%%%%%%%%%%%%%%%%%%%%%%%%%%%
Begin InputMask
% Mask for User input
Mask=Design Data
Title=Mumber of Elements
Variable=N;N
Variable=SLL;SLL
UserCallBack=ResizeA
End InputMask
%%%%%%%%%%%%%%%%%%%%%%%%%%%
Begin ExecButton
% Button for Executing the Design
Label=Design
Command=Design
End ExecButton
%%%%%%%%%%%%%%%%%%%%%%%%%%%
Begin ExecButton
% Button to show the Amplitudes
synthesized
Label=Amplitudes
Command=ShowA
End ExecButton
%%%%%%%%%%%%%%%%%%%%%%%%%%%
Begin HelpMask
Title=Array di Dolph-Chebyshev
Line= Some theory here…
End HelpMask
```

Figure 3 *– Example of EA description file.*

The user defined Matlab functions have access to the parameters of the EA via a global Matlab Structure named **AppletData**, which holds all the data defined in the EA **DataStruct** block. In the present case the **AppletData.N** and **AppletData.SLL** scalar values and the

AppletData.A array. Functions called by an ExecBlock defined button are also passed, as a parameter, a Window Handler relative to the output window (Fig.2, right) where graphs can be drawn. The Output window automatically presents an area for graphs and four sliders to allow the user to interactively adjust axes limits as well as buttons to hide/show grid lines. There are also API functions to show windows containing numerical results with just one command these windows are automatically built and their appearance can be seen in the example presented in Fig. 4 on the right.

For the given example the functions needed are indeed three: **ResizeA** which is a 5-line function resizing array **AppletData.A** accordingly to the current number of elements **AppletData.N**; this is needed to prevent undesired results in having the Design function operating over an array of dimension different than *N*. **ShowA** is a two line function calling the appropriate API function to show the amplitudes (Fig.4 right) of the synthesized array. The most complex function is **Design**, which actually implements the Dolph Chebyshev design process [4]. As stressed in the introduction the preparation of the experiment, from the educator point of view, requires basically to write down this subroutine.

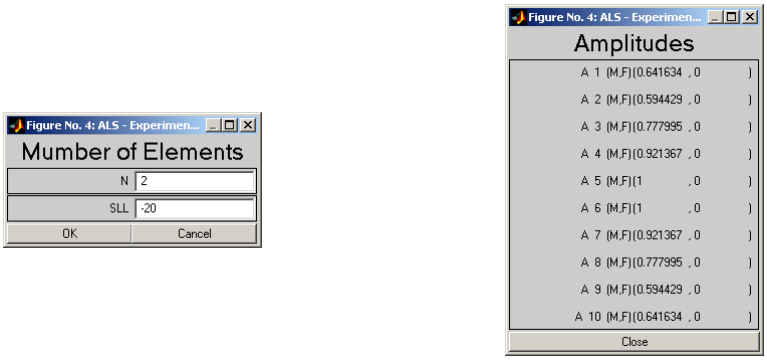

Figure 4 *– Example of automatically generated input (right) and output (left) masks.*

Conclusions – an experiment framework API has been devised and created in Matlab, allowing the simple creation of classes of experiment. By resorting to the presented API the educator is able to prepare its own experiments with little or no concern on the user interaction, but only focusing on the math and physics, hence saving much time.

The API source code is freely available, together with its full documentation, under the terms of the GNU General Public License [5] on the authors site http://www.selleri.org/ or by e-mail request.

References

[1] Waterloo Maple - http://www.maplesoft.com.
[2] Mathematica – Wolfram Research – http://www.wolfram.com
[3] Matlab - The MathWorks Inc. – http://www.mathworks.com
[4] C.A. Balanis, *Antenna theory, analysis and design,* John Wiley & Sons, New York (NY) 1982, pp.247-254.
[5] http://www.gnu.org/licenses/gpl.html