Using MATLAB to Control Commercial Computational Electromagnetics Software

R. L. Haupt

Applied Research Laboratory, State College, PA 16801, haupt@ieee.org

Abstract – This paper provides details on how to use MATLAB to control some commercial electromagnetics software packages. FEKO is an example that can be directly called from MATLAB. Other commercial software, such as CST Microwave Studio and Ansoft HFSS, require a scripting language interface. An example of a design of an inset rectangular patch antenna is presented using a direct call to FEKO and a Visual Basic for Applications interface to CST Microwave Studio are presented.

***Keywords*:** MATLAB, optimization, microstrip antenna, and genetic algorithm.

I. INTRODUCTION

MATLAB [1] has become a ubiquitous math, data manipulation, signal processing, and graphics software package. Engineers use its powerful functions for analysis and design in many areas including antenna design. MATLAB is general-purpose software, so many arcane applications, like antenna design, are done using special purpose commercial software. Although these packages can model very complex electromagnetics systems, they lack some of the powerful analysis tools in MATLAB. Using MATLAB to control these commercial electromagnetics solvers creates a powerful tool for design, analysis, and control.

There are a number of applications where a MATLAB-commercial electromagnetics solver interface is critical. Numerical optimization is one example. Although most commercial electromagnetics codes now come with some numerical optimization, they lack the versatility of optimization routines in MATLAB. Another example is in the use of signal processing software in conjunction with beamforming in an antenna array. For instance, the commercial computational electromagnetics software models the antenna elements while MATLAB takes the signals from the elements and performs the signal processing. Other applications include modeling wireless systems, radar cross section reduction, and electromagnetic band gap material design.

This paper provides systematic instructions to interface MATLAB with FEKO [2] or via a scripting language, such as Visual Basic for Applications (VBA), to a commercial software package like CST Microwave Studio [3] or Ansoft HFSS [4]. These software combinations are used to design an inset rectangular microstrip patch antenna that is resonant at 2.0 GHz. Both combinations result in successful patch antenna designs.

II. MICROSTRIP PATCH OBJECTIVE FUNCTION

The example used in this paper is the design of an inset rectangular patch that is resonant at 2.0 GHz. Fig. 1 shows a diagram of the patch with the design variables labeled. The substrate is 1.6 mm thick and has a relative dielectric constant of 2.2. An 8 mm border (E) surrounds the metallic patch. The ground plane has the same area as the substrate. There is a 1 mm gap (G) between the feed line and the patch. The microstrip feed line is F=25 mm long and has a voltage feed 3 mm from its left end. The values of L, M, B, and W are found using numerical optimization.

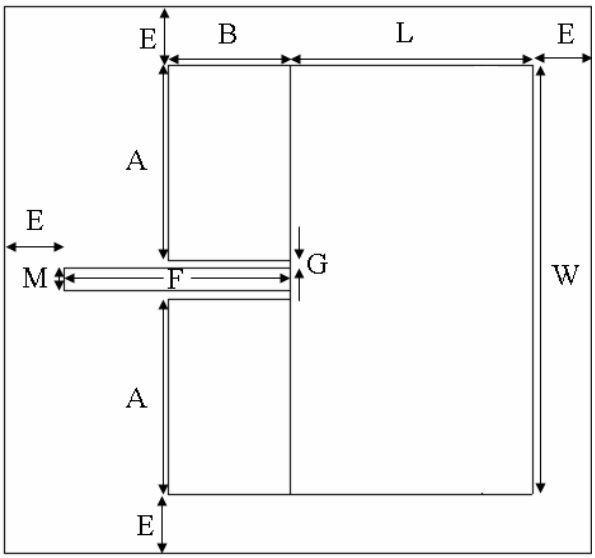

Fig. 1. Diagram of the inset patch antenna design.

This is a narrow band antenna, so minimizing the reflection coefficient at 2.0 GHz results in a very sharp decrease in s_{11} at 2.0 GHz. Finding this sharp decrease is difficult for optimization algorithms. Small changes in the patch dimensions can significantly move the resonant frequency. Local search algorithms work well with four

variables when the starting point is very close to the best solution. If a good guess is not available, then a genetic algorithm written in MATLAB is used to first find a good initial first guest for a MATLAB Nelder Mead downhill simplex algorithm (fminsearch.m). The genetic algorithm used here is described in detail in [5]. The genetic algorithm was stopped once it found a solution that had an s_{11} < -10 dB. This stopping point was chosen, because an antenna is considered matched to a transmission line when the reflection coefficient is less than -10 dB. These optimizations routines call MATLAB functions that interface with the commercial software package or scripting language. A diagram of the optimization process is shown in Fig. 2. The next two sections demonstrate the optimum design of an inset patch using a combination of MATLAB and FEKO or Microwave Studio.

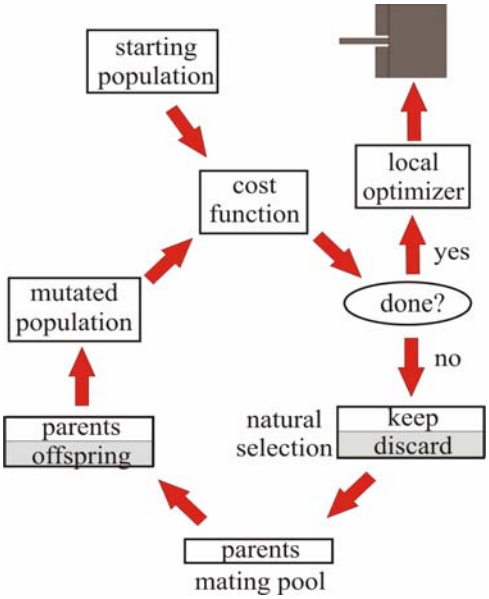

Fig. 2. Flowchart of the optimization algorithm.

Before trying the numerical algorithms, the patch is first designed using standard analytical approaches [6]-7] The patch and microstrip line are designed for 50 ohms. These values can be used to seed the numerical optimization algorithm to find dimensions that are more accurate, as shown in Table 1.

Table 1. Patch dimensions from the analytical design.

Dimension	L	B	W	M
Size in mm	32.32	17.48	59.29	4.61

III. CONTROLLING FEKO WITH MATLAB

The relevant ASCII files that are used by MATLAB and FEKO are shown in Table 2. There are a few other files generated by MATLAB and FEKO but are not important to the user. All the files have the same name but a different extension, so they are easy to associate with the same project. The MATLAB commands are in mpatch.m. Data written from MATLAB to be used by FEKO is stored in the mpatch.txt file. FEKO commands are in the mpatch.pre file. Data written by FEKO for use by MATLAB is written in the mpatch.ffe and mpatch.out files. Figure 1 is a flowchart of the MATLAB-FEKO software configuration. MATLAB can directly call FEKO to calculate s_{11} at 2.0 GHz.

Table 2. Relevant MATLAB-FEKO computer files.

File name	contents
mpatch.m	MATLAB m file
mpatch.txt	ASCII file with variable values
mpatch.pre	PREFEKO file
mpatch.out	FEKO output file
mpatch.ffe	FEKO file with angles, electric field, gain

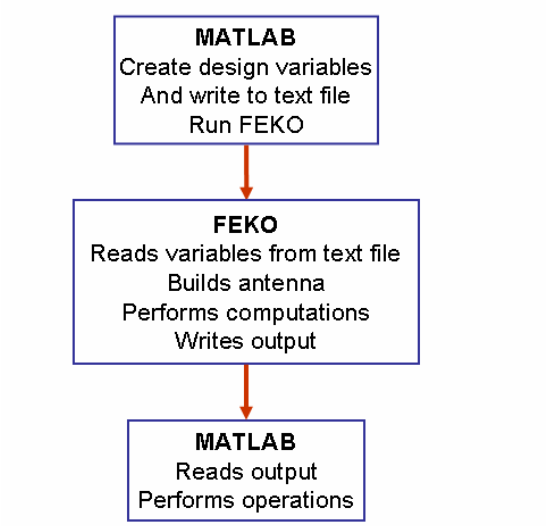

Fig. 3. MATLAB-FEKO flowchart.

The microstrip patch is represented as a lossless metal polygon in FEKO. All the polygon corners are generated by MATLAB. MATLAB plots the patch and substrate shape and labels the points. An example of a plot of half of the patch (the other half is a mirror image) is shown in Fig. 4. If the coordinates of the numbered points in Fig. 4 are (x,y), then the MATLAB code to draw the outline of the patch is given by

```
figure(1);plot(x,y,'-o');
axis equal
for ii=1:length(x)
text(x(ii),y(ii)+z(ii),num2str(ii))
end
```

This plot is useful in troubleshooting and watching the convergence of the optimization algorithm.

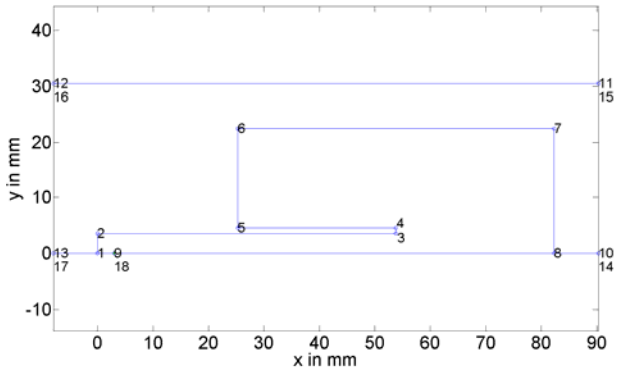

Fig. 4. Figure drawn by MATLAB before passing point to FEKO.

The numbered points in Fig. 4 are calculated from the values of the variables shown in Fig. 1. Some of these values are set while others can vary between predetermined limits. Once MATLAB has created an antenna design, all the (x,y,z) coordinates are written to the text file, `mpatch.txt` using,

```
fid=fopen('mpatch.txt','w');
N=length(x);
fprintf(fid,'%6.2f\n',N);
for q=1:N
fprintf(fid,'3%6.2f\n',x(q),y(q),z(q));
end
fclose(fid);
```

The file, mpatch.txt, has N+1 line. The first line contains the number of points. The following N lines contain the coordinates of the points.

Next, the following commands run PREFEKO and FEKO from MATLAB,

```
!prefeko mpatch > output.txt
!runfeko mpatch > output.txt
```

the "`> output.txt`" part of the commands places output generated in the running of PREFEKO and FEKO into a file rather than displaying them on the computer screen. If this part of the command is skipped, then the computer screen is filled with a lot of run data that is usually of little interest.

The lines in mpatch.pre that read from the text file and create the points defining the outline of the antenna are given by,

```
#N= fileread("mpatch.txt",1,1)
!!for #i = 1 to #N
#ax[#i]= fileread("mpatch.txt",1+#i,1)
#ay[#i]= fileread("mpatch.txt",1+#i,2)
#az[#i]= fileread("mpatch.txt",1+#i,3)
DP : p#i : #ax[#i] : #ay[#i] : #az[#i]
!!next
```

The defining points on the antenna are created by the define point (DP) command and are labeled `p1` to `pN`. After these points are formed in FEKO, then the structure is built out of triangles, polygons, wires, etc. FEKO performs the calculations and writes the output to `mpatch.out` and possibly to other files, such as `mpatch.ffe`. MATLAB can easily read the ASCII file where the far field information is written using the textread command,

```
[t,p,rEt,iEt,rEp,iEp,gt,gp,g]
=textread('mpatch.ffe','2%f (%f,%f)
(%f,%f) 3%f');
```

Reading from mpatch.out is more difficult but possible using m-files downloaded from the MATLAB website [1] (e.g. findstring.m).

Running the MATLAB-FEKO algorithm to minimize s_{11} at 2.0 GHz resulted in an s_{11} of -31 dB. The plot of s_{11} over a 4% frequency range is shown in Fig. 5. This narrow band resonance was achieved from the dimensions shown in Table 3. The values found for L and B are close to those in Table 1, while the values for W and M are considerably smaller.

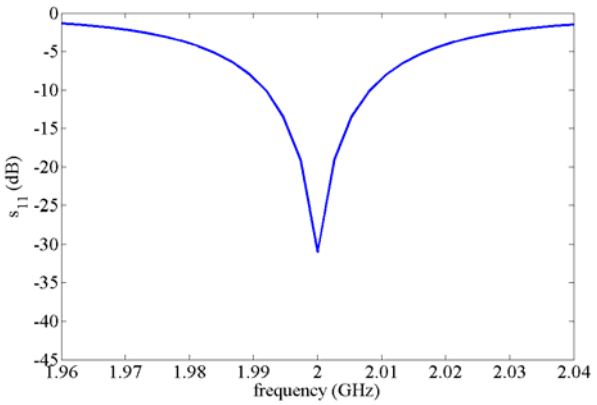

Fig. 5. Plot of s_{11} for patch antenna optimized by FEKO.

Table 3. Patch dimensions from the MATLAB-FEKO design.

Dimension	L	B	W	M
Size in mm	32.61	18.55	34.24	2.91

IV. CONTROLLING COMMERCIAL SOFTWARE PROGRAMS WITH MATLAB VIA A SCRIPT

MATLAB can control some commercial software via a scripting language. A script is a text file containing instructions written in a scripting language. The commands in the script are executed when the scripting language opens the file. VBA is widely used and Microwave Studio and Ansoft HFSS have VBA editors built in. It is a good idea to get familiar with the VBA

editor in the software package before attempting to interface with MATLAB. Commands that call various functions in the commercial software are placed in the *.bas file using the VBA editor. Types of commands include building geometry, passing variables, and engaging the main software engine.

The example described in this section uses MATLAB to control Microwave Studio via a VBA script. Microwave Studio generates a huge number of files (74) with each run. The relevant ASCII files are shown in Table 4. Again, all the files have the same name but a different extension. The s_{11} data written by Microwave Studio for use by MATLAB is written in the mspatch^d1(1)1(1).sig file. Fig. 6 shows the flow chart.

Table 4. Relevant MATLAB-Microwave Studio computer files.

File name	contents
mpatch.m	MATLAB m file
mpatch.txt	ASCII file with variable values
mpatch.bas	VBA program
mpatch.mod	Microwave Studio model file
mpatch^d1(1)1(1).sig	File containing s_{11} data

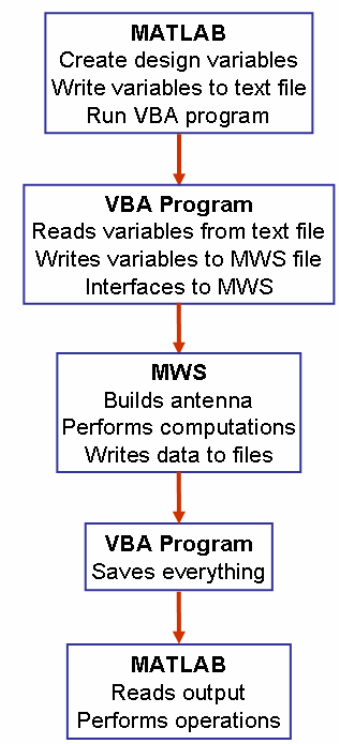

Fig. 6. MATLAB-CST Microwave Studio flowchart.

In this case, it is easier to have MATLAB write the four unknown patch dimensions to a file. The commands in mpatch.m that do this are,

```
fid=fopen('mspatch.txt','w');
fprintf(fid,'%f\r',xmin);
fprintf(fid,'%f\r',xt);
fprintf(fid,'%f\r',py);
fprintf(fid,'%f\r',cp);
fclose(fid);
```

Next, MATLAB calls the VBA program via (all on one line),

```
!"c:\program files (x86)\cst studio
suite 2006\cst design environment.exe" -
m mspatch.bas > output.txt
```

The VBA program has two parts. The first part reads the data from the mpatch.txt data file generated by MATLAB. This file contains the values for L, B, W, and M. The code that reads this data is given by,

```
Open "d:<dir>\mpatch.txt" For Input As
#1
Input#1,v(1)
Input#1,v(2)
Input#1,v(3)
Input#1,v(4)
Close #1
```

Once the data is read, then the Microwave Studio model file is opened using the command,

```
openfile("d:<dir>\mpatch.mod")
```

The model with the previously stored dimension values appears on the computer screen. In order to change the dimension values, they must be stored in the model file using the following commands,

```
storeparameter("xmin",v(1))
storeparameter("xt",v(2))
storeparameter("py",v(3))
storeparameter("cp",v(4))
```

Next, the data is saved and the model rebuild using,

```
save
Rebuild
```

The picture of the model on the screen is redrawn to reflect the new dimension values. Finally, the solver (in this case, the transient solver) is started and the results saved through the commands,

```
Solver.start
save
```

When the solver finishes and the data is stored, control returns to MATLAB and the Microwave Studio window closes. The Microwave Studio window will reopen every

time the program is called from MATLAB. MATLAB reads the s_{11} data using the line,

```
[f,s11]=textread('mspatch^d1(1)1(1).sig'
,'','headerlines',4);
```

Running the MATLAB-Microwave Studio algorithm resulted in an s_{11} of -70.8 dB at 2.0 GHz. The plot of s_{11} over a 4% frequency range is shown in Fig. 7. The final dimensions for the patch are shown in Table 5. The values of L, B, and W are very close to those predicted by Table 1.

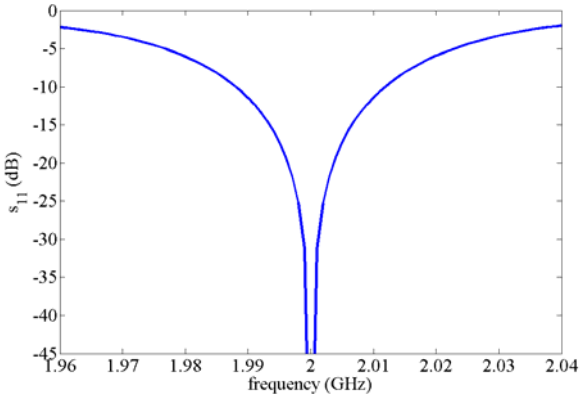

Fig. 7. Plot of s_{11} for patch antenna optimized by Microwave Studio.

Table 5. Patch dimensions from the MATLAB-Microwave Studio design.

Dimension	L	B	W	M
Size in mm	32.68	16.40	59.18	3.06

V. CONCLUSIONS

Using MATLAB to control commercial electromagnetics software creates a powerful design and systems analysis environment. This paper describes how to create the interface between MATLAB and commercial software via direct calls and via a scripting language. The different approaches to the design of an inset fed microstrip patch produced similar results. Fig. 8 shows s_{11} calculated using FEKO and the dimensions found in Tables 1, 3, and 5. The results are very close to each other (within 2.5%). Fig. 9 shows s_{11} calculated using Microwave Studio and the dimensions found in Tables 1, 3, and 5. The results are not as close together. Refining the models would likely produce better results.

There are some lessons learned here. First, the VBA interface requires learning VBA (if you did not know it already – like me). Second, the variable can be passed in a number of ways. With FEKO, it seemed easier to pass the points outlining the patch, while with Microwave Studio, it seemed easier to pass the dimensions. Third, optimizing large structures would be very time-consuming. Fourth, a number of different variables can be passed other than dimensions. For instance, material properties, type of source, source voltage, etc.

Although there are several papers that have interfaced MATLAB to particular software packages, none provide the details on how to create that interface. The purpose of this paper is to give readers enough information to create useful interactions between MATLAB and commercial electromagnetics software.

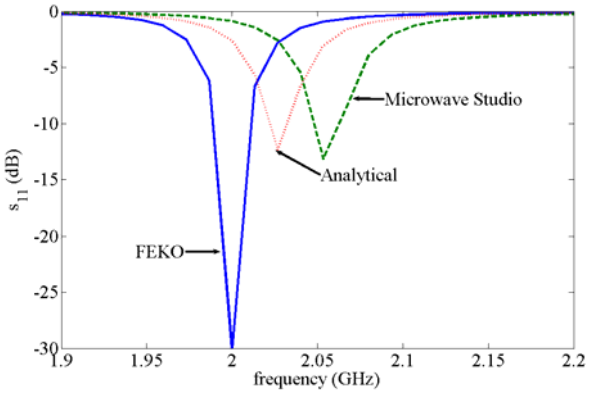

Fig. 8. The dimensions in Tables 1, 3, and 5 are used to build a patch antenna in FEKO and calculate these values of s_{11}.

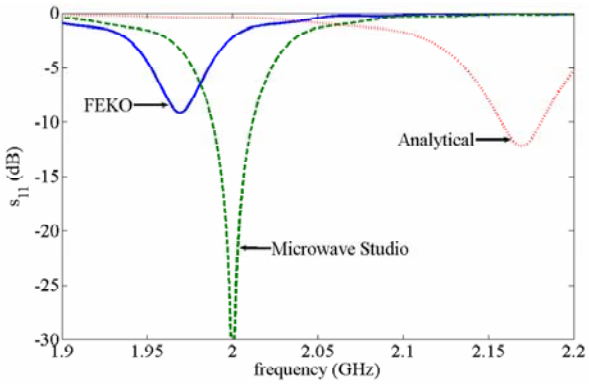

Fig. 9. The dimensions in Tables 1, 3, and 5 are used to build a patch antenna in Microwave Studio and calculate these values of s_{11}.

REFERENCES

[1] MATLAB Version 7.3.0.267, The (www.mathworks.com), Aug 3, 2006.

[2] FEKO Suite 5.1, EM Software and Systems (www.feko.info), 2005.

[3] CST Microwave Studio, Version 2006.05, April 19, 2006.

[4] High Frequency Structure Simulation (HFSS), ANSOFT Co., Pittsburgh, PA, USA.

[5] R. L. Haupt, "A mixed integer genetic algorithm for electromagnetics applications," *IEEE AP-S Trans.*, vol. 55, no. 3, pp. 577-582, Mar. 2007.

[6] http://www1.sphere.ne.jp/i-lab/ilab/tool/ms_line_e.htm
[7] C.A. Balanis, *Antenna Theory Analysis and Design*, New York: John Wiley & Sons, 1997.

Randy L. Haupt is an IEEE Fellow and Department Head of Computational Electromagnetics and Senior Scientist at the Penn State Applied Research Laboratory. He has a Ph.D. in Electrical Engineering from the University of Michigan, MS in Electrical Engineering from Northeastern University, MS in Engineering Management from Western New England College, and BS in Electrical Engineering from the USAF Academy. He was Professor and Department Head of Electrical and Computer Engineering at Utah State University from 1999-2003. He was a Professor of Electrical Engineering at the USAF Academy and Professor and Chair of Electrical Engineering at the University of Nevada Reno. In 1997, he retired as a Lt. Col. in the USAF. Dr. Haupt was a project engineer for the OTH-B radar and a research antenna engineer for Rome Air Development Center. He was the Federal Engineer of the Year in 1993 and is a member of Tau Beta Pi, Eta Kappa Nu, URSI Commission B, and Electromagnetics Academy. He served on the board of directors for the Applied Computational Electromagnetics Society and is on the IEEE Antenna and Propagation Society Administrative Committee. He has published journal articles, conference publications, and book chapters on antennas, radar cross section and numerical methods and is co-author of the book Practical Genetic Algorithms, 2 ed., John Wiley & Sons, 2004 and Genetic Algorithms in Electromagnetics, John Wiley & Sons, 2007. He has eight patents in antenna technology.