2011-03

Driving Sonnet through a Python-based interface

Becerra-Pérez, Daniel; Rayas-Sánchez, José E.

D. Becerra-Pérez and J. E. Rayas-Sánchez, “Driving Sonnet through a Python-based interface,” in Int. Review of Progress in Applied Computational Electromagnetics (ACES 2011), Williamsburg, VA, Mar. 2011, pp. 412-417.

Driving Sonnet through a Python-based Interface

Daniel Becerra-Pérez and José E. Rayas-Sánchez

Department of Electronics, Systems and Informatics, ITESO (*Instituto Tecnológico y de Estudios Superiores de Occidente*), 45604 Tlaquepaque, Jal., Mexico
daniel_becerra@hotmail.com, erayas@iteso.mx

Abstract: In this paper, we present an alternative for driving Sonnet's 3D planar electromagnetic simulator from Python. Such a driver facilitates parameter sweeps and optimization with Sonnet. In general, the proposed Python driver enables users to control and automate many functionalities of Sonnet's EM simulator. The main advantage of using Python is that it is free to use and distribute, in contrast to other solutions such as Matlab, which requires the user to have a license. For a user who is working with Sonnet Lite, which can also be obtained for free (registration required), a Python-based interface increases the flexibility of parameterization at no financial cost, which can be an excellent alternative for educational purposes.

Keywords: Parameterization, Sonnet Lite, Python, Sonnet-Python Interface, EM Analysis, EM-Based Optimization, Microstrip Notch Filter.

1. Introduction

This paper presents a Python-based interface for driving Sonnet's 3D planar full-wave electromagnetic simulator. The main advantage of driving Sonnet Lite from Python is that it facilitates the user to do parameter sweeps and optimization without the need to invest in software, since both Sonnet Lite[1] and Python[2] are free to use, which makes an excellent scenario for educational environments. Another important reason for using Python as a numerical tool and driver is that this software is increasingly used in industry and academia, as confirmed in [1-3].

There are several reasons for doing parameterization outside of Sonnet, including version restrictions, or needs for special cases of parameterization that are not possible with Sonnet alone. In this paper, a brief introduction to Sonnet Lite and the corresponding version restrictions regarding parameter sweeps is presented in Section 2. An introduction to Python is presented on Section 3. Our proposed Python-Sonnet interface is described in Section 4. Finally in Section 5, we show the results of a parameterization example of a microstrip notch filter with mitered bends, realized with the proposed Python-based interface.

2. Introduction to Sonnet Lite

Sonnet Lite is a free version of Sonnet's professional Suites. It is limited in features with respect to other commercial versions of Sonnet' EM simulator, but still provides full-wave electromagnetic 3D (three dimensional) analysis of two-port planar circuits, which enables the

[1] Sonnet Lite™ ver. 12.53, Sonnet Software Inc., North Syracuse, NY, 2010.
[2] Python™ ver 2.5.5, Python Software Foundation, Wolfeboro Falls, NH. 2010.

possibility of simulating a huge set of structures, including matching networks, filters, interconnects, etc., in microstrip, stripline, coplanar waveguide, and other popular planar technologies, provided they do not exceed the memory size limitation of Sonnet Lite.

Designers working with Sonnet often need to sweep through one or more parameters across different values to identify trends in the EM performance of the simulated structure, or to estimate the optimal solution for problems with one or two design variables. Sonnet Lite supports equations and variables within parameter sweeps, but analysis is limited to 1 parameter only. Sonnet Level 2 (Sonnet Basic™ and Sonnet Silver™) and Level 3 (Sonnet Gold™) suites add in a capability of sweeping up to 2 parameters, but only Sonnet Professional™ allows unlimited parameter sweeps.

Additionally, internal Sonnet's parameterization is restricted in the following scenarios: there are special cases where not only the geometry is changing, but also the enclosing box is changing, the number of cells is changing (to vary the resolution), and possibly other important variables are changing simultaneously too. It is under such cases that the need of additional software to do parameterization arises.

Many individuals and research groups have come up with solutions using Matlab[3] as a driver for Sonnet, as confirmed in [4-6]. Recently, a much more developed and free to use Matlab driver for Sonnet has been proposed[4], which enables the automation of many advanced features of Sonnet. However, these approaches still require the user to have access to a Matlab license, which may be difficult for many.

3. Introduction to Python

Python was selected for this work due to several advantages. Python is under an open source license that makes it freely usable and distributable. It runs over many different operating systems, including but not limited to, Windows, Linux, Unix, OS/2, Mac, and Amiga. It is a dynamic object-oriented language based on a real time interpreter, such that there is no need to compile to get working code. Some of its features include clear readable syntax, intuitive object orientation, natural expression of procedural code, exception based error handling, extensive standard libraries and third party modules that nicely complement Python to achieve almost any typical task at hand. Extensions or modules not available are easily written in C or C++ and can be integrated into Python code. There is, also, a large community of Python users and extensive documentation, making it easy for someone to get familiar with Python quickly.

Among the many additional packages available for Python, Table 1, briefly, describes those that were used in the development of our Python-Sonnet parameterization interface.

4. Sonnet-Python Interface

The Sonnet-Python interface was developed to provide the user with parameterization capabilities through a graphical application. It allows the user to perform parameter sweeps of one or more items. An item can be anything contained in the Sonnet project file whose typical extension is '.son'. Sample items include vertex coordinates, material properties, dielectric materials, metal materials, a configuration parameter such as box size, number of cells, etc. In general, any portion of the file regardless of being a string or a number can be parameterized. The interface, also, provides the capability to plot the results obtained, and save them into a graphic file with a portable network graphics (PNG) format. For each of the values that are swept, the interface generates a Sonnet project file that can be later opened directly from Sonnet Lite, or use Sonnet response viewer to analyze results if that is preferred.

[3] MATLAB™, The MathWorks Inc., Natick, MA, 2008.
[4] SonnetLab Toolbox for Matlab, http://www.sonnetsoftware.com/support/sonnet-suites/sonnetlab.html.

Table 1: Description of packages used to develop the proposed Python-Sonnet interface

Package	Description
SciPy and NumPy	SciPy[5] is a Python based open-source software for mathematics, science and engineering. The SciPy library is built to work with arrays defined in another library called NumPy[6]. NumPy provides N-dimensional array manipulation in a fast, convenient way. SciPy also provides many user-friendly and efficient numerical routines for numerical integration and optimization. Both are free of charge, easy to use and powerful enough to be valuable tools for scientists and engineers.
PyLab	PyLab is a group of Python packages that includes SciPy, NumPy, Matplotlib[7] and IPython[8]. Matplotlib is basically a library for 2D and 3D plotting and IPython is a shell interface. These packages together, provide the user with many of the capabilities that Matlab, or other math software provides.
wxPython	wxPython[9] is a Graphical User Interface (GUI) cross-platform toolkit for Python. It is designed to allow the user the creation of programs with robust highly functional graphical user interfaces. Like Python, wxPython is open source, meaning it is free to use and the source code is available too. Currently supported platforms are 32-bit Microsoft Windows, Unix, Linux and Macintosh OS X.
Boa Constructor	Boa Constructor[10] is a Python Integrated Developer Environment (IDE) that offers visual frame creation and manipulation. It features an object inspector, object browsers, documentation, an advanced debugger and integrated help. It is written in Python and uses the wxPython library. Boa Constructor was used to develop the GUI Python-Sonnet Interface presented in this paper.
Win32 extensions for Python	PyWin32[11] is a series of Microsoft Windows extensions for Python that allow the programmer to access Win32API, COM support, etc.

The Python-Sonnet interface main screen, shown in Fig. 1, allows the user to load a Sonnet project file and display it in a text editor window within the main screen, where the user can select the items that need to be parameterized and configure them appropriately. To know which portions of the Sonnet project file format to replace, the user must be familiar with Sonnet project file format [7] and with the geometries contained in the structure that the user is working on.

After the Sonnet project file has been loaded, the user can proceed to define reference parameters or directly replace portions of the Sonnet project file format with parameters or groups of parameters.

Parameters and reference parameters can be vectors with uniformly distributed values (created by specifying the start, stop and step values), fixed values, an arbitrary list of values (numerical or strings) or a function of reference parameters. Such options are shown in Fig. 2. Figure 3 shows an example of a parameter defined as a function of a reference parameter.

Once the user defines the portions of the Sonnet project file that will change and defines how these portions will change, the Python-Sonnet interface creates a set of Sonnet files that are used to launch Sonnet EM simulator and finally plot the results obtained.

[5] SciPy, ver. 0.7.2, SciPy.org (2010), April 22, 2010, http://www.scipy.org/
[6] NumPy, ver. 1.4.1, NumPy.org (2010), April 22, 2010, http://www.numpy.org/
[7] Matplotlib, ver. 0.99.1, (2010), Nov. 2009, http://matplotlib.sourceforge.net/
[8] IPython, ver. 0.10, (2010), Jan. 2010, http://ipython.scipy.org/moin/Documentation
[9] wxPython, ver. 2.8.9.2 (2009), Feb. 2009, http://www.wxpython.org/
[10] Boa Constructor, ver. 0.2.3, (2010), April 2010, http://boa-constructor.sourceforge.net/
[11] PyWin32, ver 214, (2010), July 2009, http://sourceforge.net/projects/pywin32/

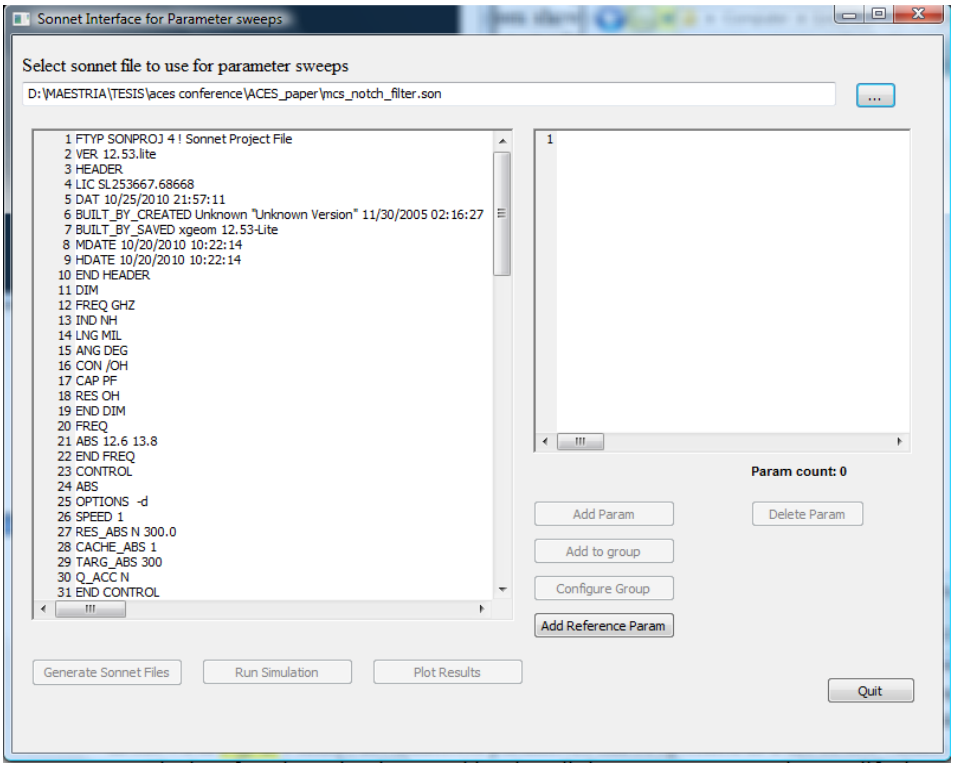

Fig. 1. Sonnet-Python interface main window. The user can add reference parameters, or directly replace portions of the Sonnet project file format with parameters or groups of parameters.

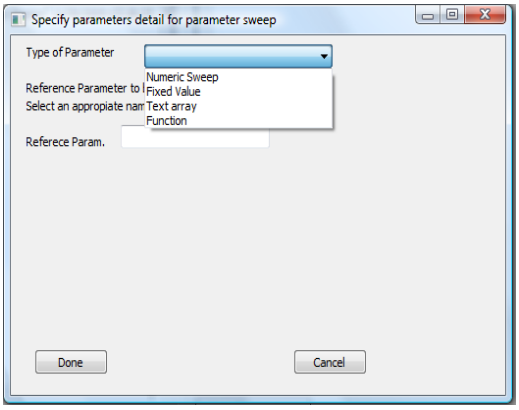

Fig. 2. Type of parameters include numeric sweeps, fixed values, list of values or functions.

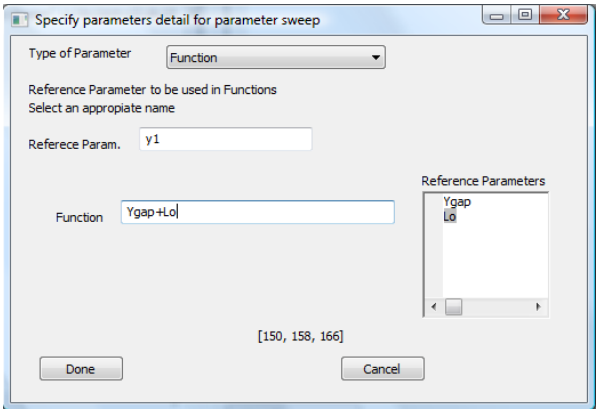

Fig. 3. Example of a parameter y_1 defined as a function of two other reference parameters: $y_1 = y_{gap} + L_o$

5. Testing the Sonnet-Python Interface with a Microstrip Notch Filter

To test the interface, a microstrip notch filter with mitered bends is used. The filter, as shown in Fig. 4, is enclosed in a metallic box with lossless metals, where H_{air} is the distance between the microstrip layer and the top of the box, y_{gap} is the spacing between the open stubs and the box walls, and L_p is the length of the input and output lines used for de-embedding. A substrate with thickness H and relative dielectric constant ε_r is used. The dimensions of the filter, shown in Fig. 5, are the width of the traces W, the length of the input and output lines L_p, the open stubs length L_o, the length of the coupled lines L_c and the separation gap S_g. These geometries are described by pairs of vertices (x_n, y_m), with each vertex being a function of the previous dimensions.

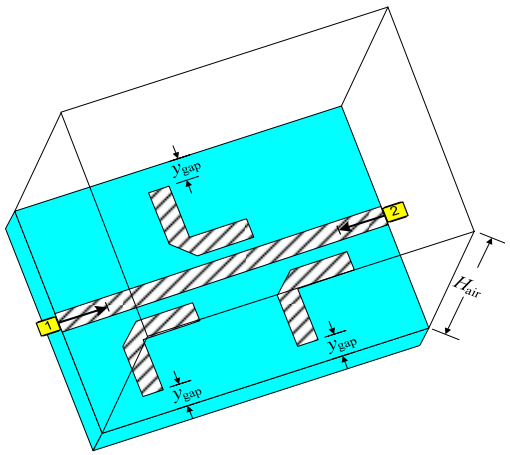

Fig. 4. Microstrip notch filter with mitered bends inside Sonnet's box.

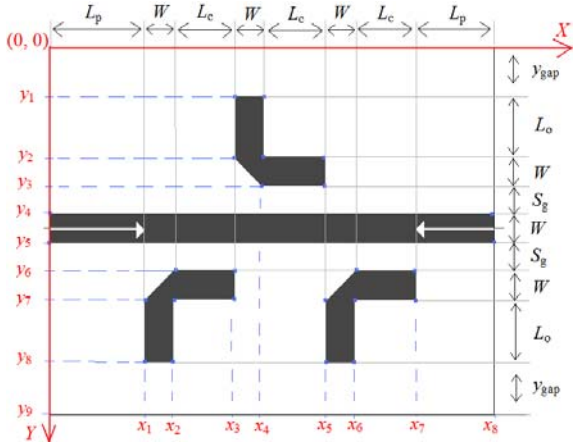

Fig. 5. Microstrip notch filter parameters and vertices definition (x_n, y_m).

The substrate used is RT Duroid 5880 with ε_r = 2.2, loss tangent = 0.0009 and H = 10 mils. For all parameterization cases, y_{gap}, H_{air}, W, L_p, and S_g are kept constant, with the following values: y_{gap} = 32 mils, H_{air} = 130 mils, W = 32 mils, L_p = 34 mils and S_g = 8 mils.

Three parameterizations of different dimensions of the microstrip notch filter using the Python-Sonnet interface are illustrated. The first parameterization is a linear sweep over 3 different equally spaced values of L_0. In this case, the start, stop, and step size are introduced and the vector is automatically created by the interface. The second parameterization is a sweep over 3 arbitrary values of L_c entered through a list. The third parameterization is a sweep of both L_0 and L_c changing as a function of a reference parameter being linearly swept over 3 different equally spaced values. In all cases, the geometries, the box, ports, and de-embedding lines are adjusted appropriately as the swept parameters change.

Figure 6 shows the result of the first parameterization targeted at L_o = [150, 158, 166] mils with the vector defined through start, stop, and step. In this case, L_c is fixed to L_c = 144 mils.

Figure 7 shows the result of the second parameterization targeted at L_c = [136, 144, 147] mils, with the vector defined by an arbitrary list of values, using the 'Text array' option. In this case, L_o is fixed to L_o = 158 mils.

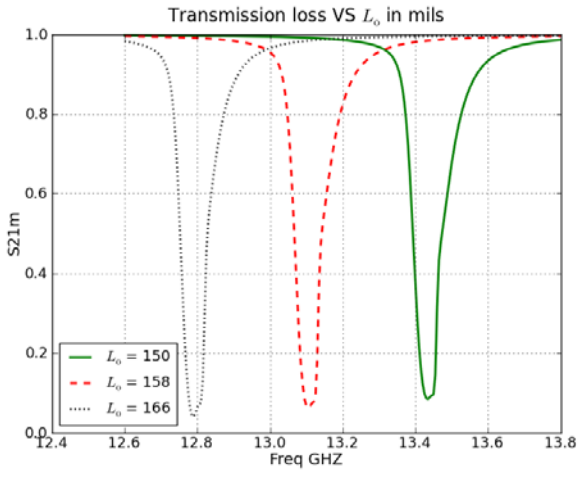

Fig. 6. Filter response for a parameterization of L_o = [150, 158, 166] mils with L_c = 144 mils.

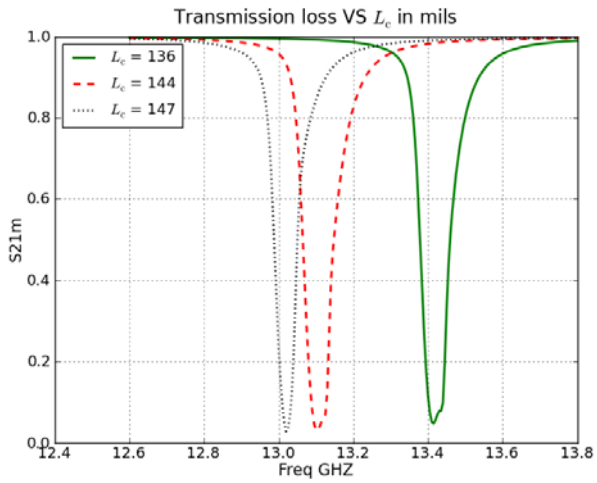

Fig. 7. Filter response for a parameterization of L_c = [136, 144, 147] mils with L_o = 158 mils.

Figure 8 shows the result of the third parameterization targeted at both L_o and L_c, both of which are defined as functions of a reference parameter N = [2, 4, 6], as follows: L_o = 154 + N and L_c = 140 + N, resulting in (L_o, L_c) = [(156, 142), (158, 144), (160, 146)].

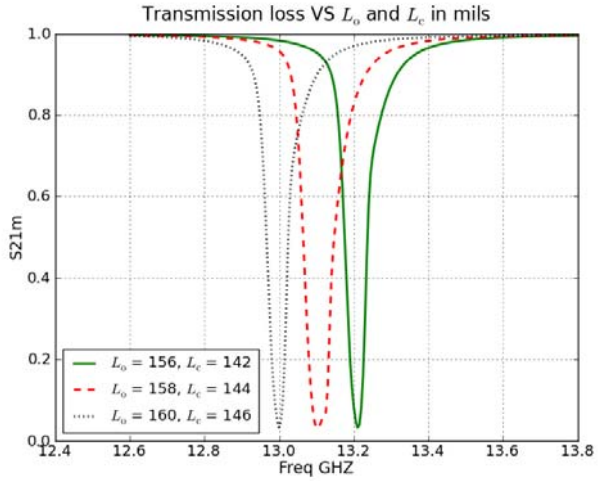

Fig. 8. Filter response for a parameterization of design variables L_o and L_c as a function of a reference parameter N = [2, 4, 6], where L_o = 154 + N and L_c = 140 + N (mils).

6. Conclusions

Python provides a wide range of capabilities as demonstrated in the development of this interface, and it is free to use and distribute, based on the Open Source license that governs Python. Parameterization of Sonnet Lite through external software helps overcome the limitations for parameter sweeps due to license restrictions, and enables free to use advanced EM analysis using Sonnet, which can be beneficial in academic environments. This interface was designed to isolate the user from having to learn how to program in Python while providing a friendly graphic user interface that offers great flexibility in the types of parameterization that can be achieved. The user can basically sweep anything contained in the Sonnet project file, with options to do fixed values, numerical sweeps, text array sweeps, and functional dependencies on reference parameters. One downside to this Python-Sonnet interface is the requirement that the user must be familiar with the Sonnet project file syntax. This is required for the user to be able to identify the elements of interest within the Sonnet project that should be parameterized. In principle, the proposed Python driver can be used with any version of Sonnet.

References

[1] P. Bienstman, L. Vanholme, W. Bogaerts, P. Dumon, P. Vandersteegen, "Python in nanophotonics research," *Computing in Science and Engineering*, vol. 9, no. 3, pp. 46-47, May/June 2007.

[2] F. Pérez, B. E. Granger, J. D. Hunter, "Python: an ecosystem for scientific computing," *Computing in Science and Engineering*," pre-print, pp. N/A, Nov. 2010.

[3] M. Summerfield, *Programming in Python 3: A Complete Introduction to the Python Language*, Boston, MA: Addison-Wesley Professional, 2009.

[4] S. Koziel, J. W. Bandler and K. Madsen, "A space mapping framework for engineering optimization: theory and implementation," *IEEE Trans. Microwave Theory Tech.*, vol. 54, pp. 3721-3730, Oct. 2006.

[5] J. E. Rayas-Sánchez and V. Gutiérrez-Ayala, "EM-based Monte Carlo analysis and yield prediction of microwave circuits using linear-input neural-output space mapping," *IEEE Trans. Microwave Theory Tech.*, vol. 54, pp. 4528-4537, Dec. 2006.

[6] V. L. Subrahmanya, *Pattern Analysis of the Rectangular Patch Antenna*, Master's Thesis, Dept. of Electrical Engineering, University College of Boras, Boras, Sweeden, 2009.

[7] Sonnet Software Inc., *Sonnet Project Format Release 12*, North Syracuse, NY: Sonnet Software Inc., May 2010.