

# OptiSPICE

## Python Post Processing

Optoelectronic Circuit Design Software

---

Version 5.2





---

# OptiSPICE

## Python Post Processing

Optoelectronic Circuit Design Software

---

### **Copyright © 2016 Optiwave**

All rights reserved.

All OptiSPICE documents, including this one, and the information contained therein, is copyright material.

No part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means whatsoever, including recording, photocopying, or faxing, without prior written approval of Optiwave.

### **Disclaimer**

Optiwave makes no representation or warranty with respect to the adequacy of this documentation or the programs which it describes for any particular purpose or with respect to its adequacy to produce any particular result. In no event shall Optiwave, its employees, its contractors or the authors of this documentation be liable for special, direct, indirect or consequential damages, losses, costs, charges, claims, demands, or claim for lost profits, fees or expenses of any nature or kind.



# Table of Contents

---

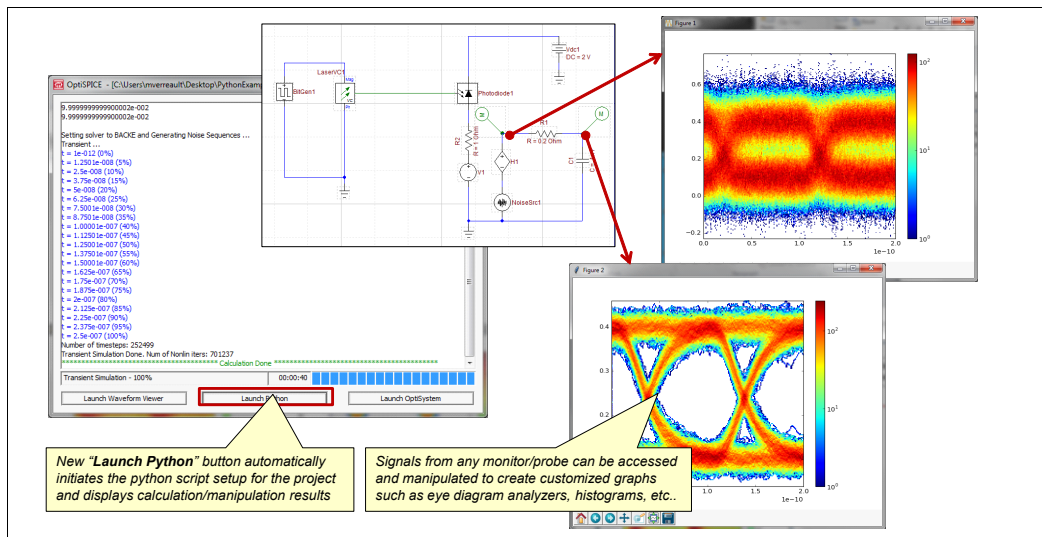
Python Post Processing.....	3
How to install and set up the Python post processing feature.....	4
Python Post Processing Example 1 .....	6
Python Post Processing Example 2 .....	13



# Python Post Processing

After completing a simulation, users can initiate Python scripts to access any active data ports, manipulate the data and create a variety of 2D and 3D views [Figure 1](#)

**Figure 1 OptiSPICE Python Post Processing feature**



This document describes how to setup the Python post-processing feature for OptiSPICE and provides examples on how to access and view simulation data using Python signal retrieval and plotting functions.

This chapter is divided into three sections:

- *How to install and set up the Python post processing feature* - This procedure shows you how to download and install WinPython so that you can view simulation data from OptiSPICE using Python's feature rich processing and graphing tools
- *Python Post Processing Example 1* - This tutorial shows you how to setup a Python script that can be called after running a project simulation and demonstrates how to launch and view electrical and optical signals using Python
- *Python Post Processing Example 2* - This tutorial shows you how Python can be used to display multi-window plots; in this case the characteristics of a long metal contact modeled as a transmission line used for a traveling wave modulator with a 50 ohm load.

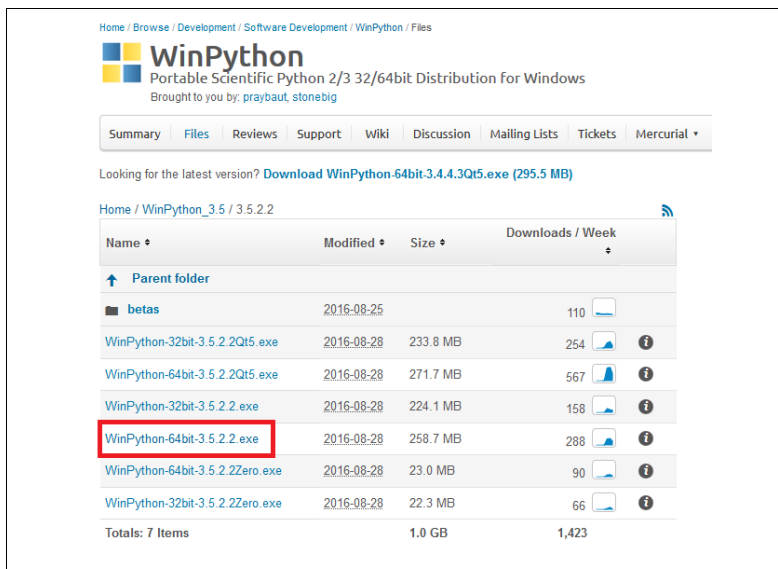


## How to install and set up the Python post processing feature

Before being able to perform Python post processing of simulation results, the following steps need to be performed:

- | Step | Action   |
|------|--|
| 1    | Proceed to the following web link:<br><a href="https://sourceforge.net/projects/winpython/files/WinPython_3.5/">https://sourceforge.net/projects/winpython/files/WinPython_3.5/</a>                            |
| 2    | Under the <i>Parent folder</i> heading, select (left-click) the link for “3.5.2.2”   |
| 3    | <b>Select</b> (left-click) the following link: “WinPython-64bit-3.5.2.2.exe” (see <a href="#">Figure 2</a> )<br><i>A dialog box will appear confirming if you would like to save the file to your computer</i> |

**Figure 2 WinPython Distributable File to download**



The screenshot shows the SourceForge page for WinPython. The page title is "WinPython Portable Scientific Python 2/3 32/64bit Distribution for Windows". Below the title, there are navigation tabs: Summary, Files, Reviews, Support, Wiki, Discussion, Mailing Lists, Tickets, and Mercurial. A link for "Download WinPython-64bit-3.4.4.3Qt5.exe (295.5 MB)" is visible. The main content area shows a file list for the "3.5.2.2" version. The file "WinPython-64bit-3.5.2.2.exe" is highlighted with a red box. The table below shows the details of the files:

Name	Modified	Size	Downloads / Week
betas	2016-08-25		110
WinPython-32bit-3.5.2.2Qt5.exe	2016-08-28	233.8 MB	254
WinPython-64bit-3.5.2.2Qt5.exe	2016-08-28	271.7 MB	567
WinPython-32bit-3.5.2.2.exe	2016-08-28	224.1 MB	158
<b>WinPython-64bit-3.5.2.2.exe</b>	2016-08-28	258.7 MB	288
WinPython-64bit-3.5.2.2Zero.exe	2016-08-28	23.0 MB	90
WinPython-32bit-3.5.2.2Zero.exe	2016-08-28	22.3 MB	66
Totals: 7 Items		1.0 GB	1,423

- |   |  |
|---|--|
| 4 | <b>Select Save File</b>  |
| 5 | After completion of the download proceed to the location where the WinPython executable was downloaded and <b>left double-click</b> on the executable program: “WinPython-64bit-3.5.2.2.exe”   |
| 6 | <b>Select Run</b> and then <b>select I Agree</b> (for the license agreement)   |
| 7 | <b>Confirm</b> or set a new Destination Folder where you would like to install WinPython and <b>select “Install”</b><br><i>By default WinPython will be installed on your Windows Desktop.</i> |



- 8 Proceed to the following folder location “C:\Program Files\Optiwave Software\OptiSPICE 5\scripts\Python”, **right-click** on the file “getSignals.py” and select **Copy**
- 9 Proceed to the folder location where WinPython has been installed (WinPython-64bit-3.5.2.2) and open the folder: “python-3.5.2.amd64”
- 10 **Select** and **open** the folder “Lib”.
- 11 **Right-click** and select “Paste” to copy the file “getSignals.py” to this directory.

End of procedure

## Python Post Processing Example 1

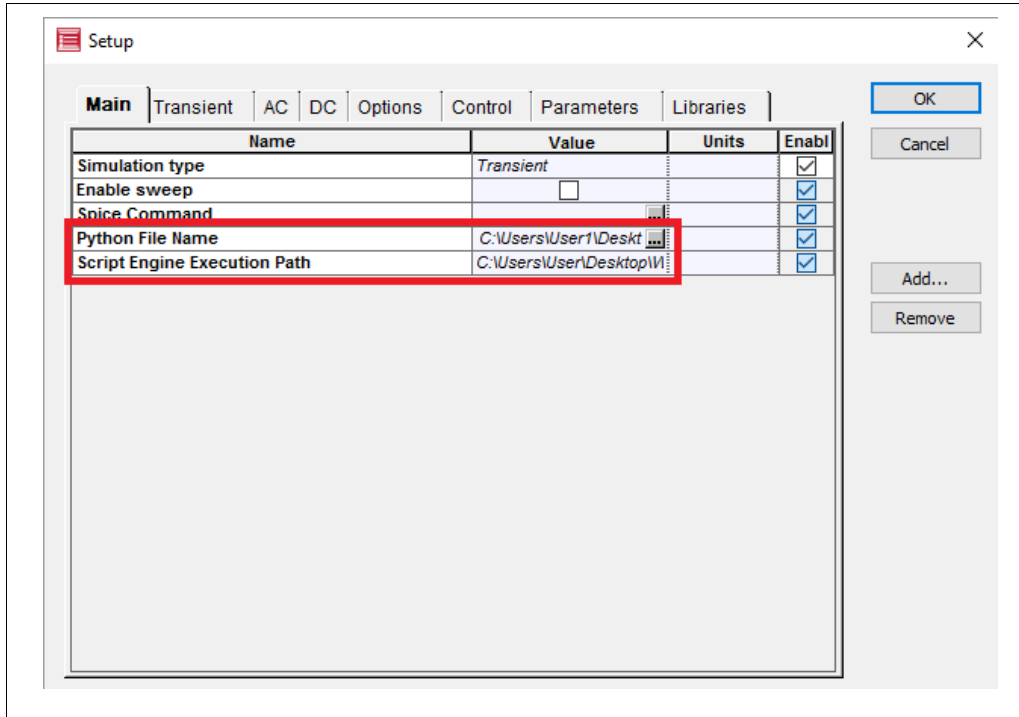
This tutorial shows you how to setup a Python script that can be called after running a simulation and demonstrates how to launch and view electrical and optical signals using Python.

### **Configuring a simulation to link a Python post processing script with an OptiSPICE design file**

- | Step | Action  |
|------|---|
| 1    | <b>Start</b> OptiSPICE Schematics and <b>open</b> the file “PythonPostProcessingExample1.osch” located at <i>OptiSPICE 5.2 Samples\Python script examples\Example1_ElectricalAndOpticalSignals</i>  |
| 2    | <b>Double left-click</b> anywhere on the design to open the Simulation Setup parameters dialog box.   |
| 3    | Under the <b>Main</b> tab select the gray box next to the parameter <b>Python file name</b>   |
| 4    | <b>Ensure</b> that the file “PlotSignals.py” has been selected and <b>select Open</b><br><b>Note:</b> The file “PlotSignals.py” should be located within the same folder as the OptiSPICE Schematic “PythonPostProcessingExample1.osch”   |
| 5    | <b>Copy</b> the directory location information for WinPython (for example <i>C:\Users\User1\Desktop\WinPython-64bit-3.5.2.2\python-3.5.2.amd64</i> )  |
| 6    | <b>Select</b> the <i>Value</i> field next to the parameter <b>Script Engine Execution Path</b> and paste the WinPython directory information into the field box (see <a href="#">Figure 3</a> )<br><b>Note:</b> It is important to ensure that any text information previously contained in the field box (if this is the case) is completely overwritten by the new directory information. |

- 7 Select OK to close the Simulation Setup parameters dialog box

Figure 3 Setup information for Python Post Processing



### Running the simulation and viewing post processed data using Python

- | Step | Action   |
|------|--|
| 1    | <b>Select Analysis/Run</b><br><i>The Simulation Progress dialog box will appear</i>  |
| 2    | After completion of the simulation, select the <b>Launch Python</b> action button at the bottom of the dialog box (see <a href="#">Figure 4</a> )<br><i>The terminal output of the executed Python script will be captured via a windows command prompt (see <a href="#">Figure 5</a>)</i> |

Figure 4 Launch Python button

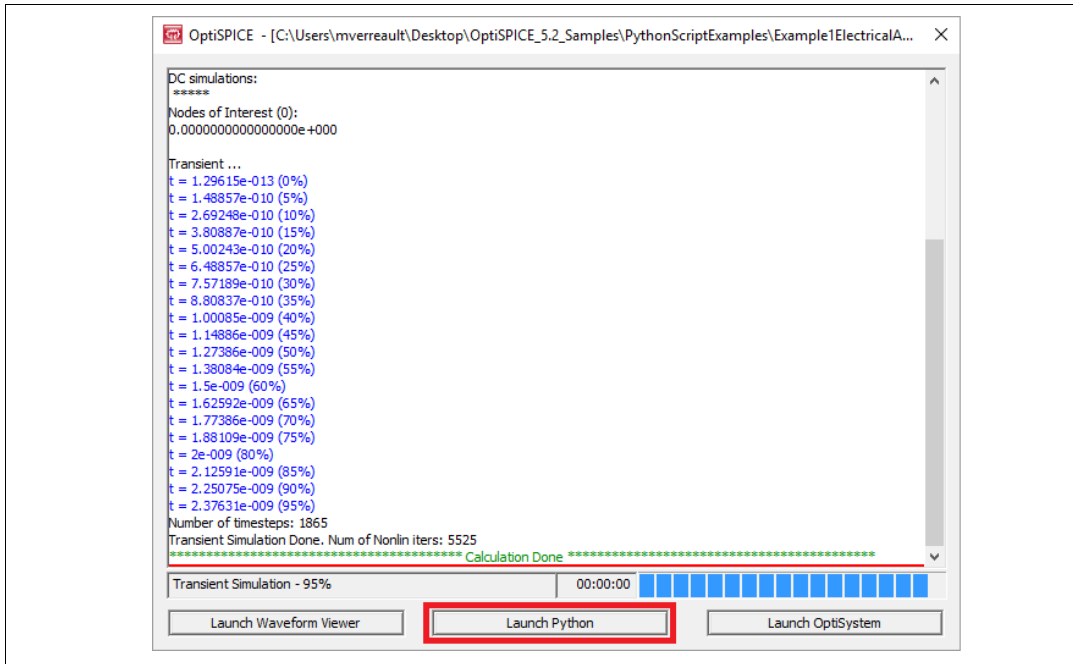
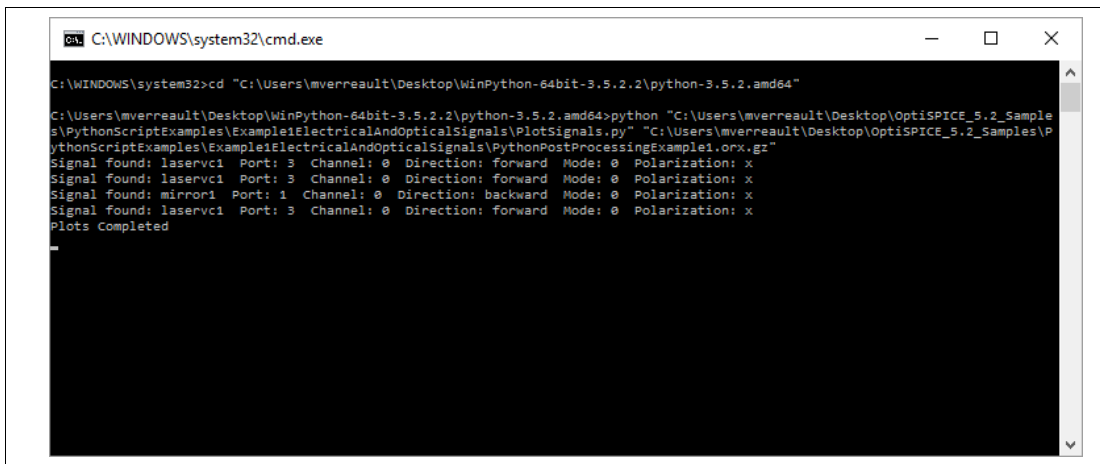


Figure 5 Terminal output from Python script

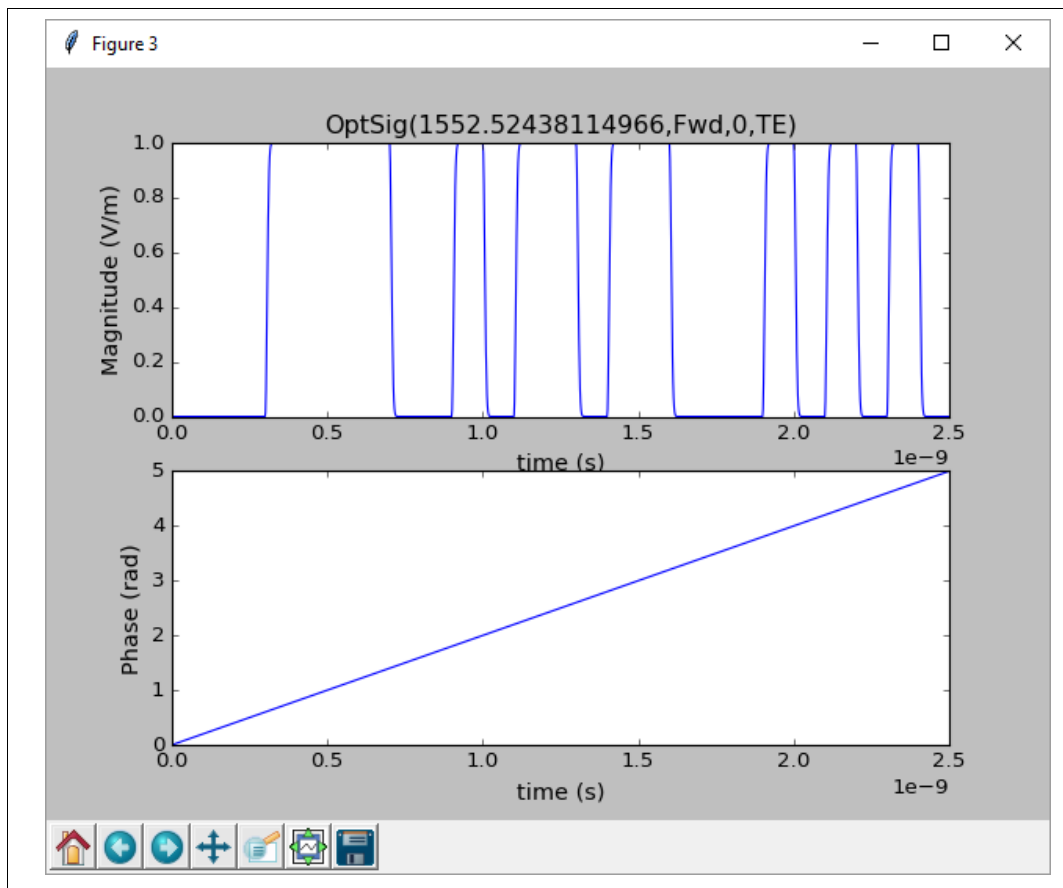


A total of seven figures will be produced (an example of Figure 3 is shown in [Figure 6](#))

- Figure 1: Voltage vs. time for bitgen1 port 1
- Figure 2: Current output of bitgen1 port 1
- Figure 3: Magnitude and phase of the laser output
- Figure 4: Real and imaginary parts of the laser output
- Figure 5: Reflection from mirror1
- Figure 6: Output power of laser in frequency domain

These figures are created from the **matplotlib.pyplot** library (which is included with the installation of WinPython). For further information on how to use and program various plots based on this library please go to: <http://matplotlib.org/>

**Figure 6** Magnitude and phase of the laser output (from PythonPostProcessingExample1.osch)



## Overview of the Python script program *PlotSignals.py*

In this example we use the Python script file *PlotSignal.py*. This script has been created specifically to access and display both optical and electrical signal results from any OptiSPICE simulation. The following describes the contents of the file and how to configure its functions for your own simulation models.

### Accessing simulation results

The following code is used at the beginning of *PlotSignal.py*. to access the simulation results:

```
import getSignals as sig #getSignals library processes the xml file and gets the data
import sys
import matplotlib.pyplot as plt #OPTIONAL:Plotting library

xmlFileName = sys.argv[1] #Getting the xml file name from the schematic editor
#The xml file has the probe information and where the data file is located

#xmlFileName = 'PythonPostProcessingBasics.orx.gz'
#The xml file name can be used directly
#in the python script to use it independently from the OptiSpice schematic editor

[root, data] = sig.getXMLFileStrAndData(xmlFileName) #Processing the xml and getting the data
Sim = root.getchildren()[0].tag #OPTIONAL: Gets the simulation type, AC,DC,Transient etc..
```

The function `sys.argv[1]` is used to access the XML file name associated with the simulation results (this action is performed after selecting the **Launch Python** button). It is also possible to directly use the name of the XML file directly to run the python script.

### Signal retrieval functions

There are two main functions that can be used to retrieve the simulation data, *getElectricalSignal* and *getOpticalSignal*.

#### *getElectricalSignal* function

The *getElectricalSignal* is structured as follows:

`[x, y1, y2, Formats, labels] = getElectricalSignal(root, data, DevName, SigType, PortNum, Domain):`

The right-hand side of the *getElectricalSignal* function specifies the probe location and port number and the type and format of the electrical input data:

- “root” stores the processed XML file.
- “data” contains all the simulation results.
- “DevName” specifies the name of the device or signal where a probe is located.
- “SigType” defines whether to access current data ('I') or voltage data ('V').
- “PortNum” specifies the port number where the probe is located (set to '-1' if the probe is on a signal).
- “Domain” can be set to 'time' (for time domain data) or 'FFT' (for frequency domain results).



The left-hand side of the *getElectricalSignal* function specifies the structure of the electrical output data:

- “x” represents time (s) or frequency (Hz) depending on the simulation.
- “y1” represents the magnitude or real part of the signal.
- “y2” represents the phase or imaginary part of the signal (it will be 0 for transient simulations).
- “Formats” defines the output format. It can be scalar (transient), rectangular (frequency domain) or polar (frequency domain).
- “Labels” is used to specify the label of the signal.

For example the following function example is used to retrieve the time-domain, output current from port 1 of the device “bitgen1”:

```
[t,Im1,Ip1,formats,labels] = sig.getElectricalSignal(root,data,'bitgen1','I',1,'time')
```

### **getOpticalSignal function**

The *getOpticalSignal* function is structured as follows:

```
[x, y1, y2, Lambda, Formats, Labels] = getOpticalSignal(root, data, DevName, Port, OutType, ChanNum, Direction, ModeNum, Pol, Domain, Format)
```

The right-hand side of the *getOpticalSignal* function specifies the probe location and port number and the type and format of the optical input data:

- “root” stores the processed XML file.
- “data” contains all the simulation results.
- “DevName” specifies the name of the device or signal where a probe is located.
- “Port” specifies the port number where the probe is located.
- “OutType” follows the same conventions as the optical probe: OptFields, OptPower, OptPhase, OptChirp.
- “ChanNum” specifies the wavelength channel number (0 to n).
- “Direction” defines whether to access the forward or backward field data.
- “ModeNum” specifies the mode number (0 to n).
- “Pol” defines whether to access the ‘x’ or ‘y’ polarization data
- “Domain” can be set to ‘time’ (for time domain data) or ‘FFT’ (for frequency domain results).
- “Format” specifies the data format (MAGPHI: Mag-Phase; CMPLX: Real/Imag; MAG: Mag only).

The left-hand side of the *getOpticalSignal* function specifies the output structure of the optical probe data:

- “x” represents time (s) or frequency (Hz) depending on the simulation.
- “y1” represents the magnitude or real part of the signal.
- “y2” represents the phase or imaginary part of the signal
- “Lambda” is used to specify the list of wavelengths.

- “Formats” defines the output format. It can be scalar (transient), rectangular (frequency domain) or polar (frequency domain).
- “Labels” is used to specify the label of the signal.

For example the following function example is used to retrieve the time-domain, field magnitude and phase from port 3 of the laser device “laserc1”:

```
[t,LaserOutMag,LaserOutPhase,Lambda1,formats,labels] =  
sig.getOpticalSignal(root,data,'laserc1',3,'OptFields',0,'forward',0,'x','time','MAGPHI')
```

Also specified in this function is the forward propagating signal ('forward'), the X polarization data ('x'), channel 0 (0) and mode number 0 (0).





## Python Post Processing Example 2

In this example an AC simulation is used to measure the characteristics of a long metal contact modeled as a transmission line used for a traveling wave modulator with a 50 ohm load.

### **Configuring a simulation to link a Python post processing script with an OptiSPICE design file**

- | Step | Action   |
|------|--|
| 1    | <b>Start</b> OptiSPICE Schematics and <b>open</b> the file “PythonPostProcessingExample2.osch” located at <i>OptiSPICE 5.2 Samples\Python script examples\Example2_Sparameters</i>   |
| 2    | <b>Double left-click</b> anywhere on the design to open the Simulation Setup parameters dialog box.  |
| 3    | Under the <b>Main</b> tab select the gray box next to the parameter <b>Python file name</b>  |
| 4    | <b>Ensure</b> that the file “SparamScript.py” has been selected and <b>select Open</b><br><b>Note:</b> The file “SparamScript.py” should be located within the same folder as the OptiSPICE Schematic “PythonPostProcessingExample2.osch”  |
| 5    | <b>Copy</b> the directory location information for WinPython (for example <i>C:\Users\User1\Desktop\WinPython-64bit-3.5.2.2\python-3.5.2.amd64</i> )   |
| 6    | <b>Select</b> the <i>Value</i> field next to the parameter <b>Script Engine Execution Path</b> and paste the WinPython directory information into the field box (see <a href="#">Figure 3</a> )<br><b>Note:</b> It is important to ensure that any text information previously contained in the field box (if this is the case) is completely overwritten by the new directory information |
| 7    | <b>Select</b> OK to close Simulation Setup parameters dialog box   |



### Running the simulation and viewing post processed data using Python

- | Step | Action   |
|------|--|
| 1    | <b>Select Analysis/Run</b><br><i>The Simulation Progress dialog box will appear</i>  |
| 2    | After completion of the simulation, select the <b>Launch Python</b> action button at the bottom of the dialog box (see <a href="#">Figure 7</a> )<br><i>The terminal output of the executed Python script will be captured via a windows command prompt (see <a href="#">Figure 8</a>)</i> |

Figure 7 Launch Python button

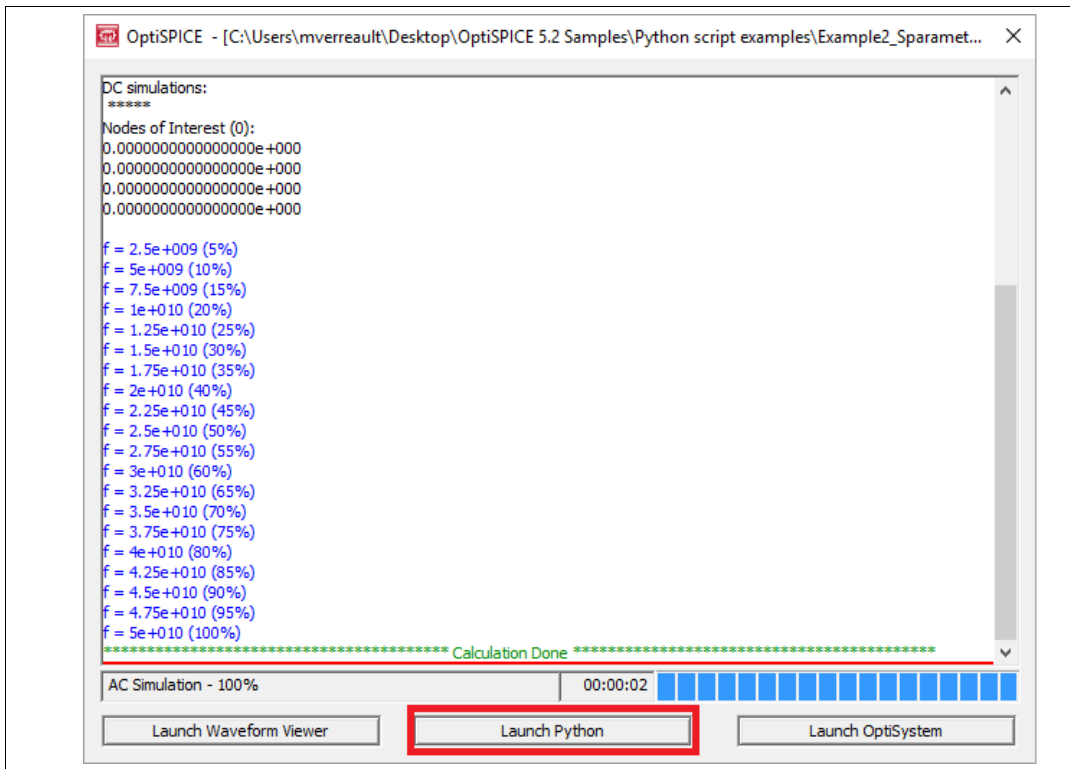
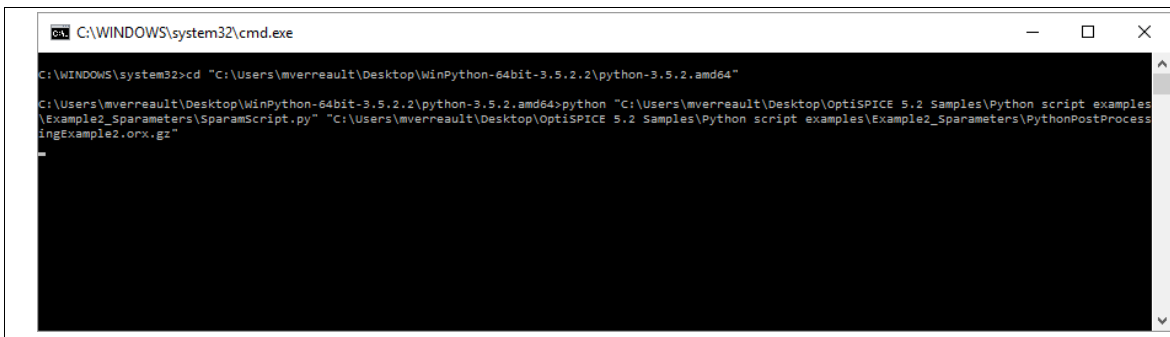


Figure 8 Terminal output from Python script

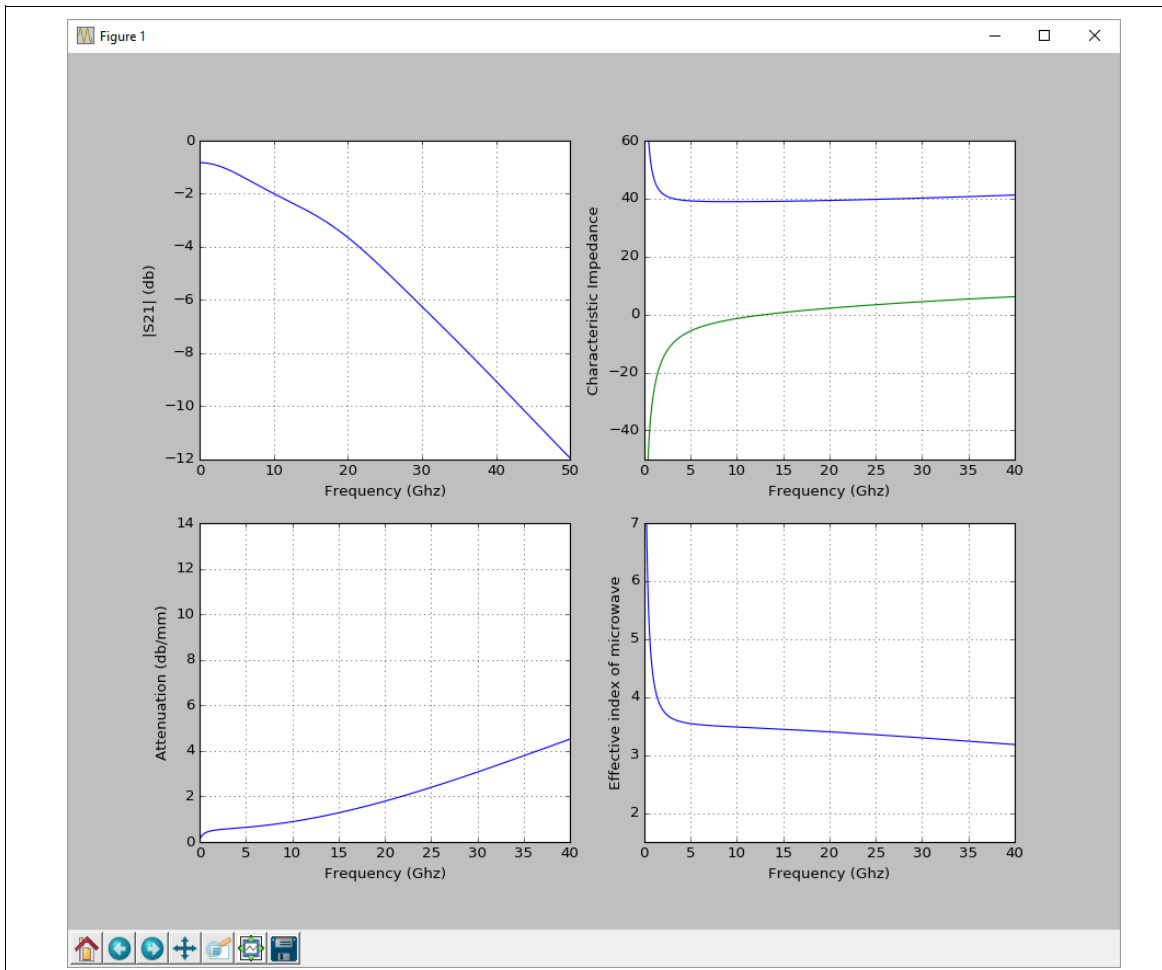


The simulation output is processed in Python to generate various characteristics for a transmission line such as attenuation, effective index of microwave and characteristic impedance. A total of four figures are produced (integrated into one display window - see [Figure 9](#)).

- $S_{21}$  (dB) vs Frequency (GHz)
- Characteristic impedance vs Frequency (GHz)
- Attenuation (dB/mm) vs Frequency (GHz)
- Effective index of microwave vs Frequency (GHz)

These figures are created from the **matplotlib.pyplot** library (which is included with the installation of WinPython). For further information on how to use and program various plots based on this library please go to: <http://matplotlib.org/>

**Figure 9** Frequency domain characteristics of the TWMZM generated by Python post processing









**Optiwave**  
7 Capella Court  
Ottawa, Ontario, K2E 7X1, Canada

**Tel.: 1.613.224.4700**  
**Fax: 1.613.224.4706**

**E-mail: [support@optiwave.com](mailto:support@optiwave.com)**  
**URL: [www.optiwave.com](http://www.optiwave.com)**