Scripting the CST Studio Suite® with the Python®

3DEXPERIENCE®

Version 1.0 - 4/27/2020

Written by: Yun Xu

Introduction

This application note illustrates a workflow for scripting the CST Studio Suite® version 2020 with the Python® Programming Languages.

CST Studio Suite® Version 2020 installation comes with the Python® Version 3.6, which requires no further setup to start using it with the CST Python Libraries. The package provides a Python® interface to the CST® Studio Suite for automation of tasks such as controlling a running CST Studio Suite® Version 2020, accessing results to simulation, optimizing the results etc. There are several CST Python Libraries included in the CST Studio® Suite version 2020. In addition to the CST Python Libraries, all the VBA scripts can be utilized as well to provide even more powerful solution for automation. More details and commands of CST Python Libraries can be found in CST Studio Suite® 2020 Help → Automation and Scripting → Python → CST Python Libraries.

In this application note, we illustrate the ease of scripting and simulating a T-splitter in the CST Microwave Studio® software by using the Python®. The workflow controlled by the Python® contains these steps:

1. Setup and Import CST Python Libraries
2. Open CST Studio Suite® version 2020 Design Environment and create a CST Microwave Studio® project;
3. Add a parameter, define units, and define frequency range;
4. Build models (with using the parameter defined previously) and add field monitors;
5. Define solver, run the simulation, access and plot results;
6. Define and start a parameter sweep.

Each of these steps are described in details in the following sections.

1. Setup and import CST Python Libraries

The workflow is written in the Jupyter® Notebook. CST Python Libraries of CST Studio Suite® version 2020 support only Python® version 3.6. After a Python® 3.6 environment is properly setup in the Jupyter® Notebook, CST Python Libraries can be imported with the following commands:

```
import sys
sys.path.append("C:\Program Files (x86)\CST Studio
Suite2020\AMD64\python_cst_libraries")
import cst
import cst.results
import cst.interface
print(cst.__file__) # should print ' C:\Program Files (x86)\CST Studio
Suite 2020\AMD64\python_cst_libraries\cst\__init__.py'
```

If CST Studio Suite® version 2020 is not installed at C:\Program Files (x86), which is the default installation path, please replace `C:\Program Files (x86)` with your installation path. You have successfully set up your Python environment when *"C:\Program Files (x86)\CST Studio Suite 2020\AMD64\python_cst_libraries\cst__init__.py"* is printed without any error.

2. Open CST Studio Suite version 2020 Design Environment and create a Microwave Studio project

This section gives useful examples to manage the CST Studio Suite® projects with the Python®, including commands to open the CST Studio Suite®, create a new CST Microwave Studio® project, save the activated project as user's desired path and name.

The example shown below is for a Microwave Studio project for High Frequency applications. Commands to create new projects for other applications can be found in CST Studio Suite® version 2020 by Python can be found in CST Studio Suite® 2020 Help → Automation and Scripting → Python → CST Python Libraries → cst.interface package.

```
project = cst.interface.DesignEnvironment() #Open CST Design Environment
my_CST = project.new_mws() #Create a new Microwave Studio project
folder_path = "E:\\"
my_CST.save(folder_path + "Python_T-splitter.cst") #Save the CST project
```

In this workflow, all files are saved in E Drive (E:). Please replace `E:\\` with your desired location.

3. Add a parameter, define units, and define frequency range

3.1. Parameter handling

Parameter List in the CST Studio Suite® offers an easy way to perform simulations with different parameter values. Parameters defined in Parameter list can be easily adjusted by Parameter Sweep and Optimizer in the CST Studio Suite®, or by storing parameters from scripting. After each simulation, 0D and 1D results will be saved by default for later comparison.

In this section, an example is given of adding a new parameter in Parameter List. This parameter "offset" will be used in Section 4 to create models and in Section 6 to sweep parameters. The code quoted as a string is a VBA code to store the parameter in Parameter List. The VBA code is assigned to the variable "`ParameterDefineString`" as a multiline string. Then the VBA code is executed by using a command "`execute_vba_code()`" defined in CST Python Libraries. After

"`my_CST.schematic.execute_vba_code(ParameterDefineString)`" is executed, a new parameter "offset" with the value of 3 can be found in Parameter List in the CST Studio Suite® version 2020 as shown in Figure 1.

```
ParameterDefineString = """
Sub Main
StoreParameter("offset","3")
End Sub"""
my_CST.schematic.execute_vba_code(ParameterDefineString)
```

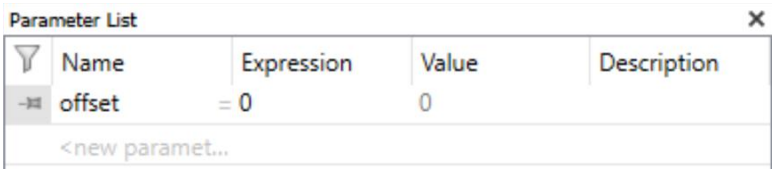

Figure 1. A parameter "offset" defined in CST Studio Suite by Python

Later, the parameter "offset" will be used when creating a model. Furthermore, it can be used for parametric and/or optimization simulations based on the user's application.

More commands about parameter handling can be found in CST Studio Suite 2020 Help → Automation and Scripting → Visual Basic (VBA) → 3D Simulation VBA → VBA Objects → Global → Project → Parameter Handling.

3.2. Define units and frequency range

Important actions (i.e. modeling, solver setup, excitation, etc.) should be recorded in the History List of the CST Studio Suite®. With this list, you may apply changes to your model, restore a previous project state and record macros. The actions are stored as VBA-Commands and their names are listed in History List. Figure 2 shows an example of History List. Actions such as define unit and define frequency range were recorded.

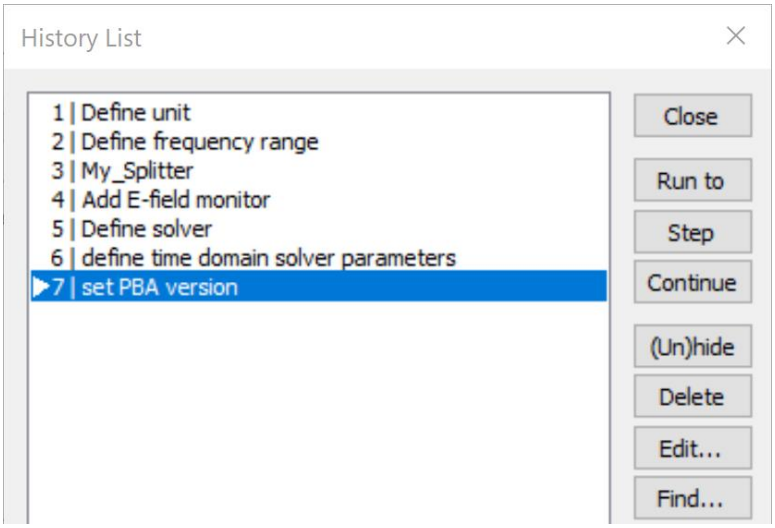

Figure 2. History List built in the CST Microwave Studio® by Python

When an item in the History List is double clicked, it will open the VBA code as shown in Figure 3. The simulation can be edited by editing the VBA commands in the History List Item in the graphical user interface (GUI) or adding new Items in History List with scripting.

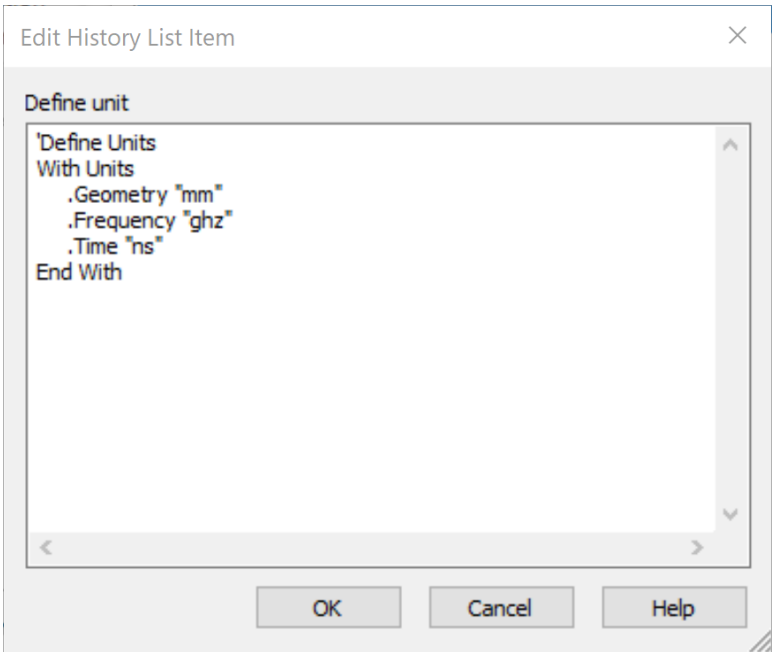

Figure 3. History List Item of Define unit

Below we show an example of adding an action of define unit to History List. A multiline string of VBA commands is assigned to a variable and it will be added to the History List by executing "`add_to_history ("Define unit", Full_History)`". The name of the action is "Define unit".

```
Full_History = """
With Units
     .Geometry "mm"
     .Frequency "ghz"
     .Time "ns"
End With
"""
my_CST.modeler.add_to_history ("Define unit", Full_History)
```

Similarly, frequency range can be defined by adding a History List Item with the following commands.

```
Full_History = """
Solver.FrequencyRange 8, 10
"""
my_CST.modeler.add_to_history ("Define frequency range", Full_History)
```

4. Build models (with using the parameter defined previously) and add field monitors

Figure 4 shows a T-splitter with a pin (used to minimize reflection coefficient) in the center and waveguide ports created by adding an item “My_Splitter” in History List with Python. The position of the pin on X-direction is controlled by the parameter “offset”. The value of “offset” will be adjusted in a later section with Parameter Sweep in CST Studio Suite®.

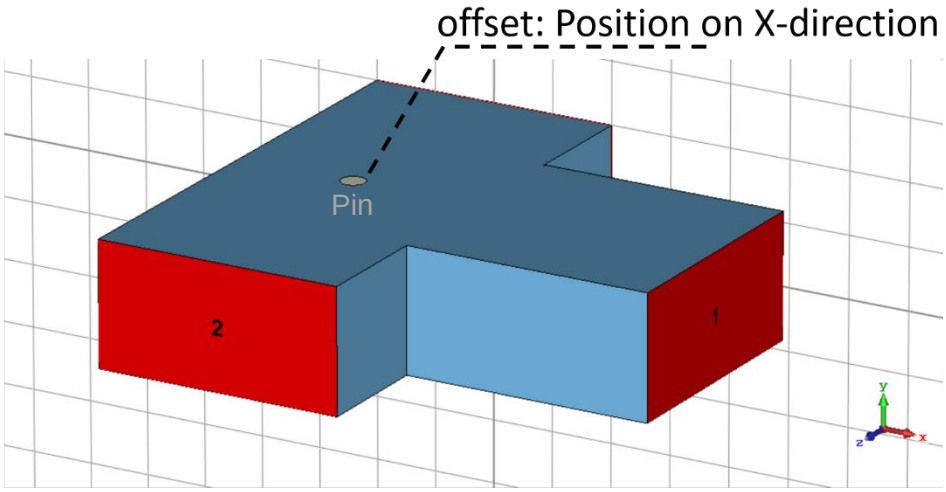

Figure 4. A T-splitter built in the CST Studio Suite® by Python

Python script can be used to build the model of T-Splitter and to add the item “My_Splitter” to the History List of the CST Studio Suite®. “My_Splitter” includes the actions of creating the T-spliiter model, defining materials, boundaries and background, and adding waveguide ports. More details of the code for this can be found in the Section 4 of the Jupyter® Notebook file.

In order to view 3D field result, we use the command below to add an E-field monitor at the frequency of 9 GHz for the entire simulation domain. Section 5.2.2 will show how to export and plot the 3D E-field result.

```
Full_History = """
'Define Monitor
With Monitor
     .Reset
     .Name "e-field (f=9)"
     .Dimension "Volume"
     .Domain "Frequency"
     .FieldType "Efield"
     .Frequency 9
     .Create
End With
"""
my_CST.modeler.add_to_history ("Add E-field monitor", Full_History)
```

5. Define solver, run the simulation, access and plot results

5.1. Define and start solver

The CST Studio Suite® offers a wide variety of different solvers and solver types. Time domain solver (TD Solver) is selected for the simulation of this example of T-splitter. The solver setup should be added by the history list by using the code shown below. More information of defining solvers in the CST Studio Suite® with scripting can be found in the CST Studio Suite® 2020 Help → Automation and Scripting → Visual Basic (VBA) → 3D Simulation VBA → VBA Objects → Solver.

```
Full_History = """
'Define Solver
With Solver
     .CalculationType "TD-S"
     .StimulationPort "1"
     .StimulationMode "1"
     .MeshAdaption False
     .CalculateModesOnly False
     .SParaSymmetry False
     .StoreTDResultsInCache False
     .FullDeembedding False
     .UseDistributedComputing False
End With
"""
my_CST.modeler.add_to_history ("Define solver", Full_History)
```

To confirm the selected solver, we can execute

```
my_CST.modeler.get_active_solver_name()
```

and it will print 'HF Time Domain' for TD solver.

Once TD solver has been confirmed, we can start the simulation by execute the command the below:

```
my_CST.modeler.run_solver()
```

While executing "`run_solver()`", the Python® will be paused before executing any other commands until the simulation finishes.

5.2. Access and plot results

After the simulation is finished, we want to plot the results for further analysis. In this simulation, S-parameters will be saved by default. To view 2D/3D field results, we need to

add Field Monitors before running the simulation. As shown in Figure 2, an E-field monitor was defined by adding a History List item with the Python®.

5.2.1. 0D/1D results

The package `cst.result` in CST Python Libraries provides access to 0D/1D results of CST files. It is not necessary to open the CST Studio Suite® while using functions defined in `cst.result`, which can save time from loading the project.

If the simulation project is already open, we can work in interactive mode to access 0D/1D results. This can be done by using the commands as shown below:

```
result_project = cst.results.ProjectFile(folder_path + "Python_T-
splitter.cst", allow_interactive=True)
```

The code below is an example of accessing S1,1 of the T-splitter and plotting magnitude and phase of S1,1 as shown in Figure 5.

```
%matplotlib qt
#Please comment out %matplotlib qt if not using Jupyter Notebook
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
s11 = result_project.get_3d().get_result_item("1D Results\S-
Parameters\S1,1")
ss = np.asarray([s11.get_xdata() , s11.get_ydata()])
fig1 = plt.figure(figsize=(10,10))
plt.subplot(2,1,1)
plt.plot(s11.get_xdata(),20*np.log10(np.absolute(np.asarray(s11.get_ydata(
))))))
plt.title('S-Parameter Magnitude')
plt.ylabel('S11 (dB)')
plt.xlabel('Frequency (GHz)')
plt.grid(True)
plt.xlim((8,10))
plt.subplot(2, 1, 2)
plt.plot(s11.get_xdata(),np.angle(np.asarray(s11.get_ydata()),deg=True))
plt.title('S-Parameter Phase')
plt.ylabel('S11 Phase (degree)')
plt.xlabel('Frequency (GHz)')
plt.grid(True)
plt.xlim((8,10))
```

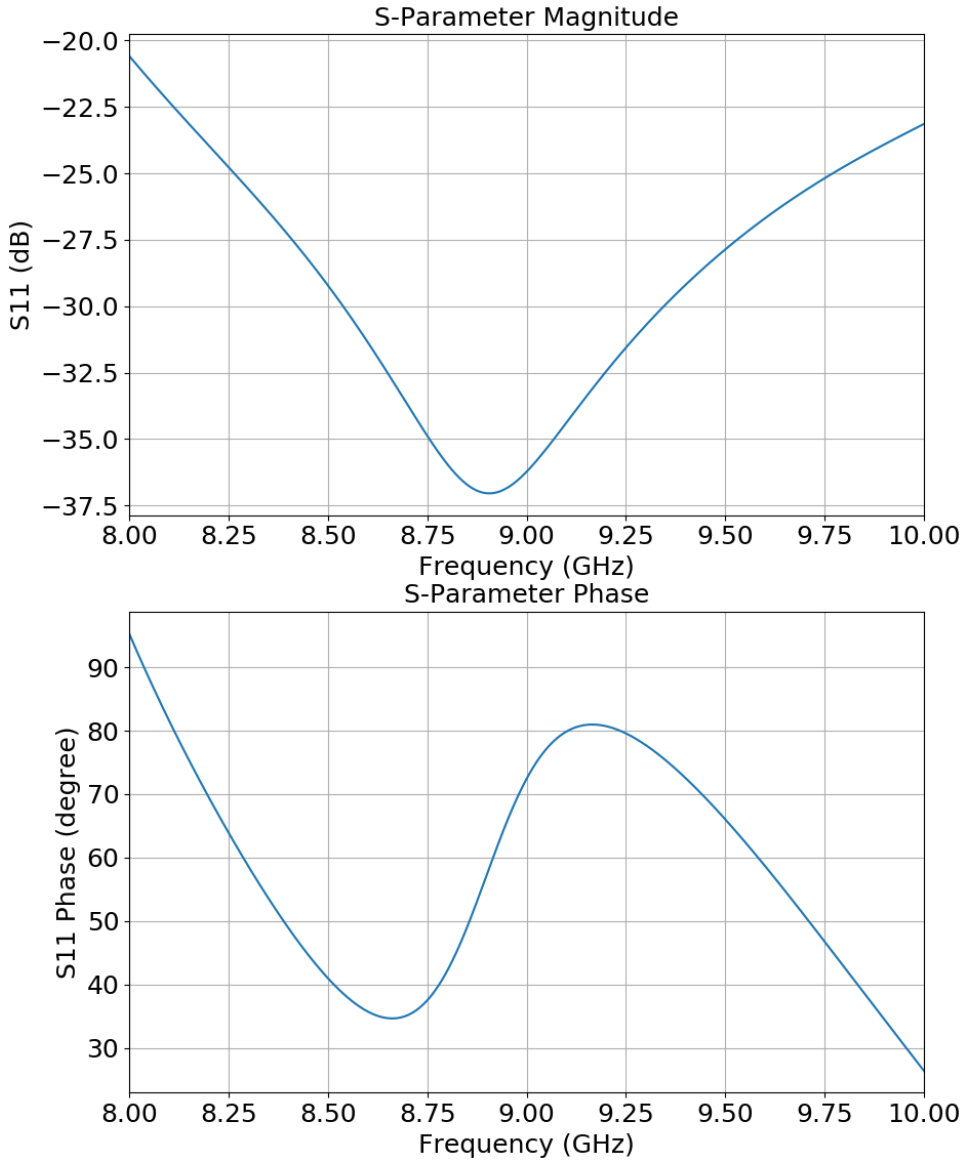

Figure 5. Magnitude and phase of S1,1 accessed and plotted with the Python®

5.2.2. 2D/3D results

To export 2D/3D results, a VBA code has to be executed. Below shows an example of exporting the 3D E-field result with the frequency of 9 GHz into an hdf5 file.

```
Export2DResult = """
Sub Main
SelectTreeItem("2D/3D Results\E-Field\e-field (f=9) [1]")
With ASCIIExport
.Reset
.SetFileType ("hdf5")
.FileName (\""""+folder_path+"""E_field.h5")
.Mode ("FixedWidth")
.StepX (0.5)
```

```
.StepY (0.5)
.StepZ (0.5)
.Execute
End With
End Sub"""
my_CST.schematic.execute_vba_code(Export2DResult)
```

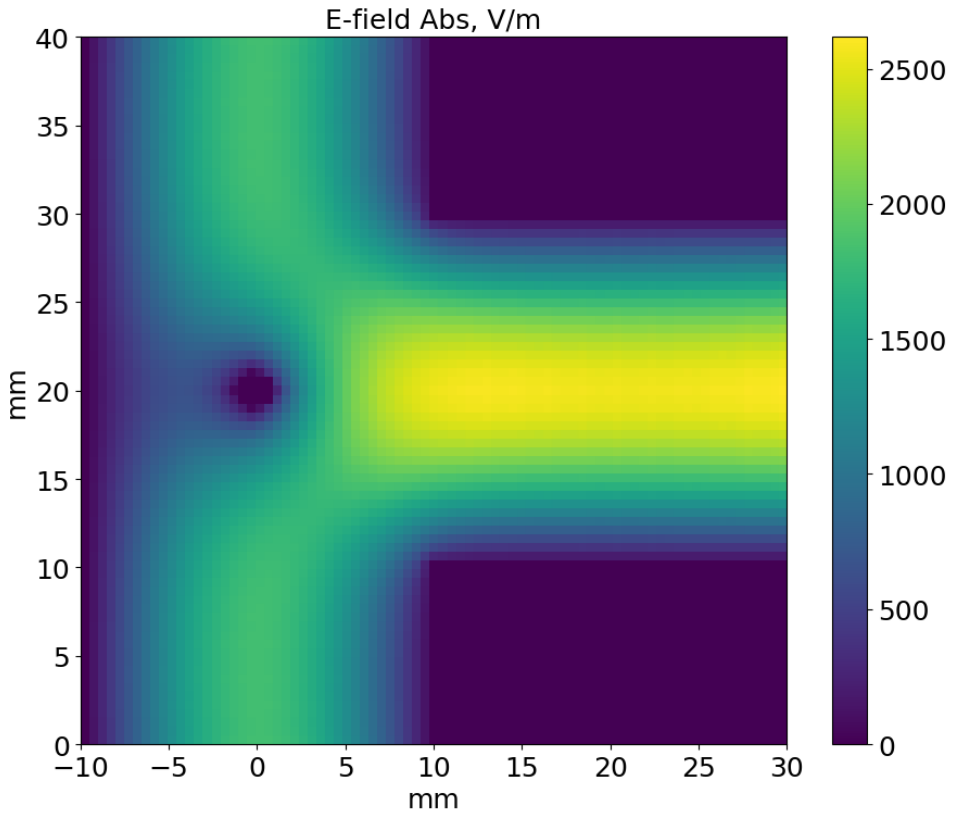

Figure 6. Absolute value of E-field exported and plotted by the Python®

After the E-field result is saved in E_field.h5, it can be opened and plotted by the Python® with the code shown below. Figure 6 shows the plotted result on X-Z plane.

```
import h5py
f = h5py.File(folder_path + 'E_field.h5', 'r')
list(f.keys())
dset = f['E-Field']
meshX = np.asarray(f['Mesh line x'])
meshY = np.asarray(f['Mesh line y'])
meshZ = np.asarray(f['Mesh line z'])
dset.shape
dset
dset.dtype
dset['x']['re']
Ex = (dset['x']['re']+1j*dset['x']['im'])
Ey = (dset['y']['re']+1j*dset['y']['im'])
Ez = (dset['z']['re']+1j*dset['z']['im'])
Ez.shape
fig2 = plt.figure(figsize=(8,6))
plt.title('E-field Abs, V/m',fontsize=18)
```

```
plt.ylabel('mm',fontsize=18)
plt.xlabel('mm',fontsize=18)
im = plt.imshow(np.abs(Ey[:,11,:]), extent=[meshX[0], meshX[-1], meshZ[0],
meshZ[-1]], origin='lower',
           cmap='viridis')
plt.colorbar(im)
```

More information about result export by VBA can be found in the CST Studio Suite® 2020 Help → Automation and Scripting → Visual Basic (VBA) → 3D Simulation VBA → VBA Objects → Import/Export → ASCII Export.

6. Define and start a parameter sweep

A parameter “offset” was defined in Section 3.1 and used to build the T-splitter in Section 4. The Parameter Sweep offers an easy and efficient way to perform several simulations with different structure parameter values. In this section we demonstrate sweeping this parameter and compare the results for different values. Parameter Sweep can be accessed in the CST Studio Suite® GUI from Home → Simulation → Par. Sweep as shown in Figure 7.

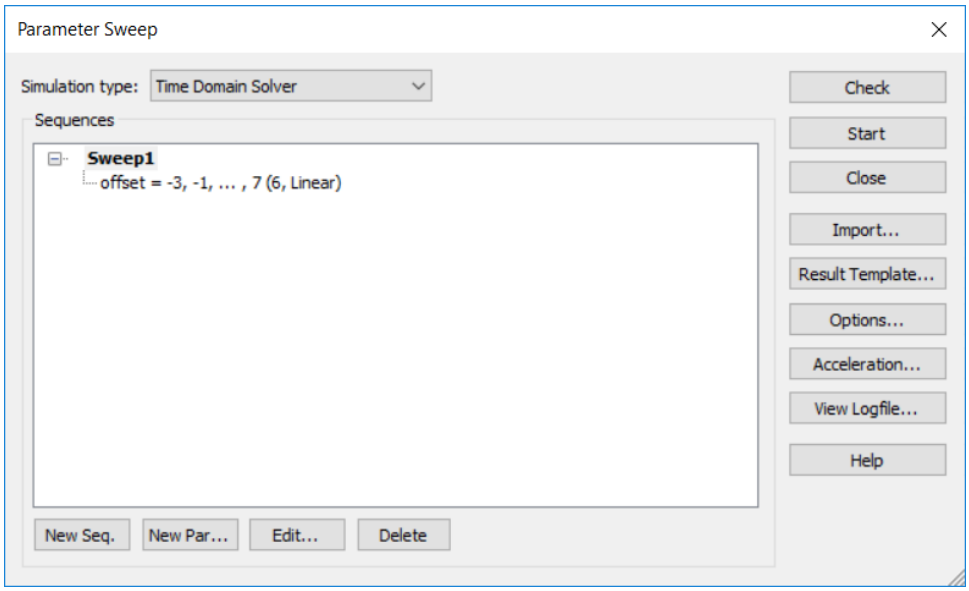

Figure 7. Parameter Sweep in the CST Studio Suite® GUI

Such similar task can be performed using the Python® scripting. Below is the code that shows an example of adding and running Parameter Sweep in the CST Studio Suite® version 2020 with the Python®. The parameter “offset” is swept from values -3 to 7 at a step width of 2.

```
par_sweep = """
Sub Main
With ParameterSweep
    .DeleteAllSequences
    .SetSimulationType ("Transient")
    .AddSequence ("Sweep1")
```

```
    .AddParameter_Stepwidth ("Sweep1", "offset", -3, 7, 2)
    .Start
End With
End Sub
"""
my_CST.schematic.execute_vba_code(par_sweep)
```

Figure 8 (a) and (b) show the 3D models of a T-splitter with the value of parameter “offset” is set as -3 and 7, respectively. The location of the pin in the T-splitter on X-direction is controlled by the value of “offset”. Figure 8 shows that the distance of the pins is equal to 10 mm.

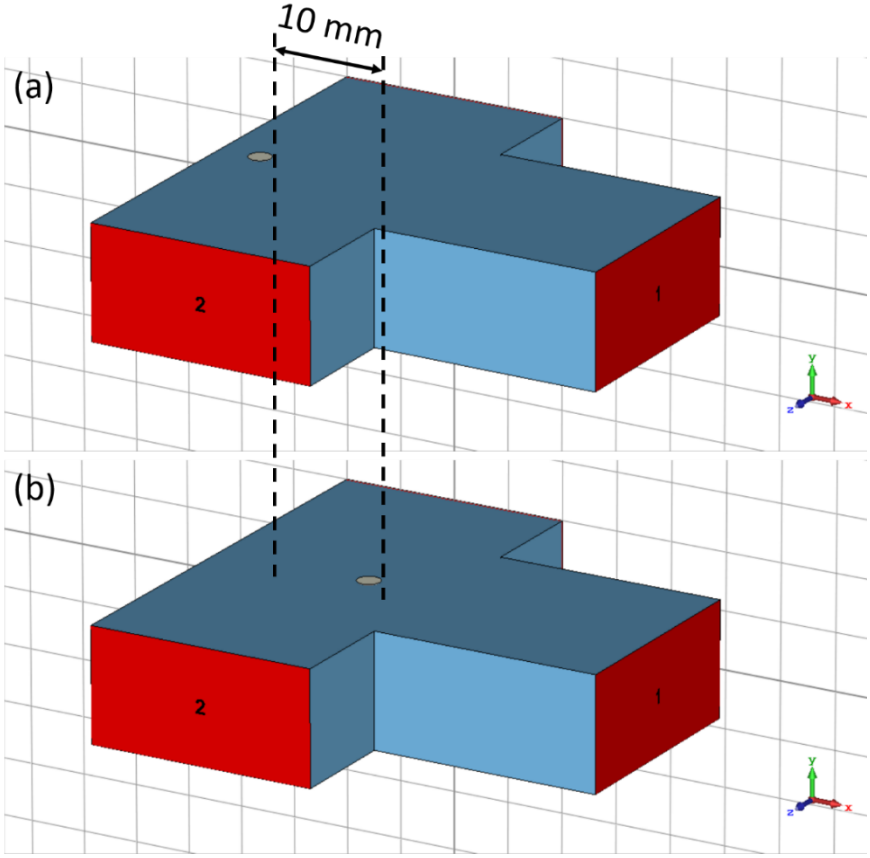

Figure 8. T-splitter with different values of parameter “offset”. (a) offset=-3 and (b) offset=7

The code below shows how to access and plot S1,1 with different values of “offset”. The plot is shown in Figure 9 and the values of “offset” are labeled in the lower right corner. According to Figure 9, only with offset = 3, S1,1 is lower than -20dB from 8 to 10 GHz.

```
run_id = result_project.get_3d().get_all_run_ids()
fig3 = plt.figure(figsize=(10,7))
offset = [];
for id in run_id:
    s11 = result_project.get_3d().get_result_item("1D Results\S-
Parameters\S1,1",id)
    ss = np.asarray([s11.get_xdata() , s11.get_ydata()])
    par = s11.get_parameter_combination()
    S11_mag_dB = 20*np.log10(np.absolute(np.asarray(s11.get_ydata())))
```

```
    if id!=0:
      plt.plot(s11.get_xdata(),S11_mag_dB,label="offset="+str(par['offset'
])) #Please be careful with indentation
plt.legend(loc='lower right')
plt.title('S-Parameter Magnitude')
plt.ylabel('S11 (dB)')
plt.xlabel('Frequency (GHz)')
plt.grid(True)
plt.xlim((8,10))
```

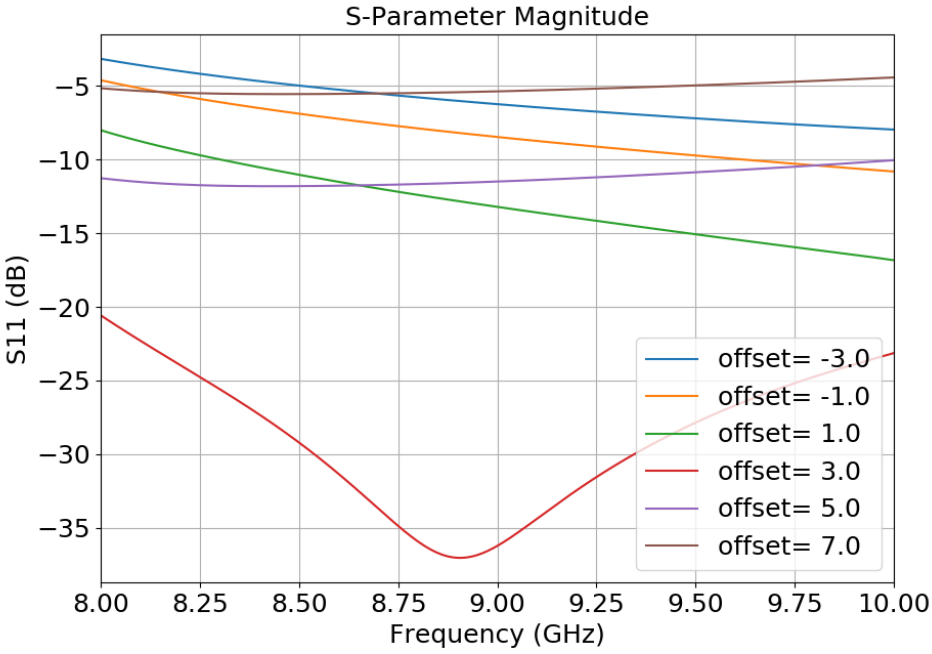

Figure 9. S1,1 plot of Parameter Sweep

Thus, using Parameter Sweep offers an easy and efficient way to perform several simulations with different structure parameter values.

Conclusion

In this application note, we showed a workflow for scripting a T-Splitter with the Python® and simulating it with the CST Studio Suite® version 2020. The steps involved creating a file, running the simulation, accessing and plotting the results. In addition, the model was parameterized and we demonstrated how to sweep a parameter and obtain results using the Python®.

More workflows of controlling the CST Studio Suite® version 2020 by the Python® can be realized by combining functions of both the Python® and VBA together. For example, by updating parameters and accessing to results, an external optimizer can be integrated to simulations.