Control CST Studio 2020 with Python

Marc Bodem
Version: October 2020

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Inhaltsverzeichnis

1 Introduction

This small tutorial will help you tu use Python in connection with CST Studio Suite 2020. Therefore the implementation with Malab will be redundant. It is possible to open files, change Parameters, start solvers (even Parameter Sweep) and access the results in appropiate way. Possbile is this through the new Python Interface from CST Studio 2020 and some VBA scripts, which you can write and execute in your Python code directly.

You can access the Python Help of CST via

```
C:\Program Files (x86)\CST Studio Suite 2020\Online Help\Python
```

If you want to know more about VBA scripting for CST checkout

http://www.mweda.com/cst/cst2013/vba/vba_macro_language_overview.htm

2 Programs and versions used

For my implentation I used the following programs and versions:

- Anaconda
- Spyder 4.1.5
- Python 3.6.12
- CST Studio Suite 2020 (no Student version!)

It is really important to use Python 3.6.x and CST Studio Suite 2020 because just these two versions will work together properly.

3 Installation

Install CST 2020 as usual. There is nothing to note here. Then download Anaconda. Spyder comes with it and install it. It is possible, that you do not have the latest Spyder version. To Update Spyder open *Anaconda Prompt* from your Windows menu. Then update everything with:

```
conda update conda
conda update anaconda
conda update spyder
```

Because Anaconda provides a newer Python version than 3.6.x you have to download the Python version from *Python.org*. Then install the Python version you downloaded.
In the next step a new Anaconda environment has to be created. To do this open your *Anaconda Prompt* once again and type

```
conda create -n py36 python=3.6 spyder-kernels
```

to create a new environment for your Python 3.6.x. To add the new Python interpreter to your Spyder open Spyder and go to *Tools* → *Preferences* → *Python Interpreter* → *Use the following Python Interpreter*. Choose your new Python 3.6.x installation. For me the path was the following:

```
C:\Users\marc\anaconda3\envs\py36\python.exe
```

Restart Spyder and check in the console if Python 3.6.x is used.
Now everything is set up to use Python in connection with CST 2020. To check everything works fine use the following short routine.

```
import sys
sys.path.append(r"C:\Program Files (x86)\CST Studio Suite 2020\AMD64\...
python_cst_libraries")
import cst
print(cst.__file__)
# should print '<PATH_TO_CST_AMD64>\python_cst_libraries\cst\__init__.py'
```

If the path is returned correctly you can start to use CST with Python.

4 Implementation

4.1 Setup

You have already seen some code brackets you will use in your Python implementation. We are going a step back and start with the small check routine you have already used. To start your implementation you have to add CST to your Python. You will do that by using

```
import sys
sys.path.append(r"C:\Program Files (x86)\CST Studio Suite 2020\AMD64...
\python_cst_libraries")
```

Here the CST path is added to our little program. In the next step you have to import CST to use it. So

```
import cst
```

will import it.
In the following we have to use some interfaces which are provided by the CST Python implementation. You will use the interface `cst.interface` to control the running CST and the interface `cst.results` to provide access to the 0D and 1D Results of your cst file. So import both interfaces via

```
import cst.interface
import cst.results
```

To work with the results we also import numpy.

```
import numpy as np
```

4.2 Start CST, open and close files

Now you can start to open CST with

```
mycst=cst.interface.DesignEnvironment()
```

and the desired project with

```
mycst1=cst.interface.DesignEnvironment.open_project(mycst,r'Path to your cst file')
```

You can close CST with

```
cst.interface.DesignEnvironment.close(mycst)
```

4.3 Change Parameters

Now it is possible to start a solver or what we do here to change parameters in the first place and then starting the solver. Because there is no published method to change parameters yet, we have to work with a VBA script here. But no worries, we can implement that easily in our Python routine. So it is possible to do anything what you can do in VBA with CST with Python as well. So now you can write some VBA code, which we can start later, as a normal string variable. You can either use the classic way, where you change Parameters and update the model on your own, just the way you would do it in CST, or you can use the build-in Parameter Sweep Option, where you can define parameter combinations you want to test. This decision will lead in different ways how to solve and even access the results.

Note: In VBA code `\n` *stands for a new line. You can either write it just before your commands like here or you can add a blank after it.*

4.3.1 Classic

In the classic way you start the VBA code with

```
Sub Main ()
```

Then you are able to change some parameters of interest with

```
\nStoreParameter("wg_h", '+str(0.3)+'),
```

where wg_h is the parameter you are changing and 0.3 is the new value.
You can repeat this for all parameters you want to change. In the end you have to update the model which you can do with

```
\nRebuildOnParametricChange (bfullRebuild, bShowErrorMsgBox)
```

and you are closing the VBA script with

```
\nEnd Sub
```

As a whole you get the following code for 2 parameters.

```
par_change = 'Sub Main () \nStoreParameter("wg_h", '+str(0.3)+')...
\nStoreParameter("wg_w",'+str(28)+')...
\nRebuildOnParametricChange (bfullRebuild, bShowErrorMsgBox)...
\nEnd Sub'
```

After defining your VBA code you can execute it with

```
mycst1.schematic.execute_vba_code(par_change, timeout=None)
```

Now all parameters will be updated as you defined them.

4.3.2 Parameter Sweep Option

Just like in the classic way every VBA script starts with

```
Sub Main ()
```

You can add a Sequence to the Parameter Sweep via

```
\nParameterSweep.AddSequence('Name of Sequence')
```

Note: Please make sure that there are no old sequences you do not want to use in your CST file before adding new sequences. To make sure that all sequences are deleted use

```
delete ='Sub Main() \nParameterSweep.DeleteAllSequences() \nEnd Sub'
mycst1.schematic.execute_vba_code(delete, timeout=None)
```

to delete all sequences before adding new ones.

If you want to test specific parameter combinations every combination has to have its own sequence.
You can add a parameter to the sequence using

```
\nParameterSweep.AddParameter('Name of Sequence',"Name of Parameter",...
'+str(True)+','+str(from)+','+str(to)+','+str(steps)+')
```

where you can use the from-to function with a number of steps.
Close the VBA script with

```
\nEnd Sub
```

Here you can see a small example how to use the functions:

```
createSequence = 'Sub Main () \nParameterSweep.AddSequence('+str(Seq1)+') \nEnd Sub'
add_Para_1 = 'Sub Main () \nParameterSweep.AddParameter('+str(Seq1)+',"d",...
'+str(True)+','+str(1)+','+str(5)+','+str(50)+') \nEnd Sub'
add_Para_2 = 'Sub Main () \nParameterSweep.AddParameter('+str(Seq1)+',"w",...
'+str(True)+','+str(14)+','+str(19)+','+str(20)+')
```

You can execute the VBA scripts via

```
mycst1.schematic.execute_vba_code(createSequence, timeout=None)
mycst1.schematic.execute_vba_code(add_Para_1, timeout=None)
mycst1.schematic.execute_vba_code(add_Para_2, timeout=None)
```

If you want to use the same file again with other combinations, be sure to delete all sequences after you solved your problem. Use

```
delete ='Sub Main() \nParameterSweep.DeleteAllSequences() \nEnd Sub'
mycst1.schematic.execute_vba_code(delete, timeout=None)
```

to delete all sequences.

4.4 Solve

In the next step you can start the solver with

4.4.1 Classic

```
mycst1.modeler.run_solver()
```

4.4.2 Parameter Sweep Option

```
solve = 'Sub Main () \nParameterSweep.Start \nEnd Sub'
mycst1.schematic.execute_vba_code(solve, timeout=None)
```

4.5 Access S-Parameters

After the simulation is done you can access the S-Parameter data. To do this you have to define your CST file path once again for the results interface.

```
project = cst.results.ProjectFile(r'Path to your CST file')
```

In order to make data (e.g. S-parameter) accessible while the CST file is opened, the additional command `allow_interactive=True` has to be included, i.e.,

```
project = cst.results.ProjectFile(r'Path to your CST file', allow_interactive=True)
```

Now you can choose from the different S-Parameter results. Here we want to access the S21 Parameters for our object.

4.5.1 One parameter combination

If you just have one parameter combination you can access the results with

```
results = project.get_3d().get_result_item(r"1D Results\S-Parameters\S2,1")
```

From the results we can derive the frequencies with

```
freq = np.array(S21_data.get_xdata())
```

and the S21 Parameters with

```
S21 = np.array(S21_data.get_ydata())
```

4.5.2 More than one parameter combination or Parameter Sweep Option

No Tasks

If you are not using tasks in your CST file you can start to access the results as follows. After defing your project path you can access the results via

```
results = project.get_3d().get_result_item(r"1D Results\S-Parameters\S2,1",k)
```

where k defines the k^{th} parameter combination you tested. The number of parameter combinations start with 1 and not with 0 because 0 is used for the Current Run.
Further you can access the frequencies with

```
freqs = results.get_xdata()
```

and the S-Parameters with

```
S21 = results.get_ydata()
```

Tasks

If you are using Tasks in your CST file you have to use the schematic tree. Therefore you have to open it via

```
schematic = project.get_schematic()
```

Then you can access the results with

```
results = schematic.get_result_item('Tasks\\SPara1\\S-Parameters\\S2,1',k)
```

where k defines, as in the part without tasks, the k^{th} parameter combination you tested. The number of parameter combinations start with 1 and not with 0 because 0 is used for the Current Run.
Further you can access the frequencies with

```
freqs = results.get_xdata()
```

and the S-Parameters with

```
S21 = results.get_ydata()
```

4.6 Prepare CST file

The solver settings should be done in the CST file, i.e., the choice of the solver, the mesh or the number of frequency data points.

4.6.1 Define number of result data samples

The S-Parameter will be evaluated in a given number of frequency result data samples, which can be specified in your cst file.

Classic

In the classic way of the implementation you can change the number of result data samples at *Solver Setup → Method → Properties... → Result Data Sampling → Number of result data samples*.

Tasks

Using Tasks you can define the number of result data samples if you select your Task and then go to *Task Parameter List (Your Task) → S-Parameters → Simulation Settings → Samples*.

4.6.2 Activate Parameter Sweep

If the Parameter Sweep option should be used, it seems to be necessary to activate this option in the CST file. This can be done by adding and deleting a new sequence: *Home → Par. Sweep → New. Seq. → Delete→ Close*. Then the CST file can be saved and closed and the Parameter Sweep Option can be run from Python.

4.6.3 Delete old results

Using the Parameter Sweep Option, a list of result data is imported to Python. Since the import starts in the beginning of the result data, old result data have to be deleted. This can be done by deleting the result folder within the created project folder. Using Python, this could be done by running

```
import shutil
try:
 shutil.rmtree(cst_result_folder)
except OSError as e:
 print(e)
```

```
else:
  print("The results directory is deleted successfully")
```

before the project is opened.

5 Examples

In the following we present two examples from the CST Component Library: the *S-Parameter Lowpass* (as Design Studio example) and the *Lossy Load Waveguide* (as Microwave Studio example). For both, the frequency domain solver has been applied. The lowpass filter uses tasks and the Parameter Sweep Option. The waveguide examples use the Parameter Sweep Option for more than one parameter and the classic way for one parameter. Both examples have two different templates you can access to get a better understanding how to use the Python CST interface. One template always shows you how to use the techniques of this tutorial when you want to change only one parameter. The other template shows you how to change a set of parameters.

A Example programs

./../Template_Lowpass.py

```
# -*- coding: utf-8 -*-
"""
Created on Thu Sep  3 16:18:47 2020

@author: Marc Bodem

Template for Calling CST from Python
- Lowpass Filter (Design Studio)
- Parameter Sweep to load several sample points
- change 6 parameters
- Obtain S-Parameter values

"""
import sys
sys.path.append(r"C:\Program Files (x86)\CST Studio Suite 2020\AMD64\
    python_cst_libraries")
import cst
import cst.interface
import cst.results
import numpy as np
import scipy.stats
import time
import shutil

start = time.time()

# Local path to CST project file --> Please adapt
cst_path = r'D:\Documents\Vorlagen\CST_Python_Interface' # path
cst_project = '\S-Parameter Lowpass' # CST project

cst_project_path = cst_path + cst_project + '.cst'
cst_result_folder = cst_path + cst_project + '\Result'


# Delete all old results (if exist)
try:
    shutil.rmtree(cst_result_folder)
except OSError as e:
    print(e)
else:
    print("The results directory is deleted successfully")


# Define random list of unifrom distributed sample points
Nmc = 3 # number of sample points
Nuq = 6 # number of parameters per sample point
means = [6.8, 5.1, 9.0, 1.4, 1.4, 1.3] # mean values for parameters
sample_list = []
```

```
for nuq in range(Nuq):
    mu = means[nuq]; uq = 0.2;
    samples_k = scipy.stats.uniform.rvs(mu-uq*mu,2*uq*mu,Nmc)
    sample_list.append(samples_k)
sample_list = np.array(sample_list).T


# Define list with frequency points of interest, here 0-7 GHz, Number of calculated points are set to 7001 in CST file, to get the desired frequencies.
freq_range = [0,1,2,3,4,5,6,7]
freq_range_pos = np.array(freq_range)*1000


# Initialize lists with S-parameter results
S_all = []
S_real_all = []
S_imag_all = []
SdB_all = []


# Open CST as software
mycst=cst.interface.DesignEnvironment()

# Open CST Project
mycst1=cst.interface.DesignEnvironment.open_project(mycst,cst_project_path)

#Delete All Sequences before starting to get a fresh file
delete ='Sub Main() \n ParameterSweep.DeleteAllSequences() \nEnd Sub'
mycst1.schematic.execute_vba_code(delete, timeout=None)

for i in range(len(sample_list)):
   #VBA Add Sequence
   createSequence = 'Sub Main () \n ParameterSweep.AddSequence('+str(i)+') \n End Sub'
   #VBA Add Parameters to Sequence. For each Parameter combination you want to test, create 1 sequence
   add_Para_L1 = 'Sub Main () \n ParameterSweep.AddParameter('+str(i)+',"L1",'+str(True)+','+str(sample_list[i][0])+','+str(sample_list[i][0])+','+str(1)+') \n End Sub'
   add_Para_L2 = 'Sub Main () \n ParameterSweep.AddParameter('+str(i)+',"L2",'+str(True)+','+str(sample_list[i][1])+','+str(sample_list[i][1])+','+str(1)+') \n End Sub'
   add_Para_L3 = 'Sub Main () \n ParameterSweep.AddParameter('+str(i)+',"L3",'+str(True)+','+str(sample_list[i][2])+','+str(sample_list[i][2])+','+str(1)+') \n End Sub'
   add_Para_W1 = 'Sub Main () \n ParameterSweep.AddParameter('+str(i)+',"W1",'+str(True)+','+str(sample_list[i][3])+','+str(sample_list[i][3])+','+str(1)+') \n End Sub'
   add_Para_W2 = 'Sub Main () \n ParameterSweep.AddParameter('+str(i)+',"W2",'+str(True)+','+str(sample_list[i][4])+','+str(sample_list[i][4])+','+str(1)+') \n End Sub'
   add_Para_W3 = 'Sub Main () \n ParameterSweep.AddParameter('+str(i)+',"W3",'+str(True)+','+str(sample_list[i][5])+','+str(sample_list[i][5])+','+str(1)+') \n End Sub'

```

```
    #excute VBA Code above
    mycst1.schematic.execute_vba_code(createSequence, timeout=None)
    mycst1.schematic.execute_vba_code(add_Para_L1, timeout=None)
    mycst1.schematic.execute_vba_code(add_Para_L2, timeout=None)
    mycst1.schematic.execute_vba_code(add_Para_L3, timeout=None)
    mycst1.schematic.execute_vba_code(add_Para_W1, timeout=None)
    mycst1.schematic.execute_vba_code(add_Para_W2, timeout=None)
    mycst1.schematic.execute_vba_code(add_Para_W3, timeout=None)

#Start Solver
solve = 'Sub Main () \n ParameterSweep.Start \nEnd Sub'
mycst1.schematic.execute_vba_code(solve, timeout=None)

#Delete All Sequences
mycst1.schematic.execute_vba_code(delete, timeout=None)

#get Project for results
project = cst.results.ProjectFile(cst_project_path, allow_interactive=True)

#get schamatic
schematic = project.get_schematic()


# Evaluate each sample point in CST
for i in range(len(sample_list)):
    #k = i+1 because run Ids start at 1 and 0 is the current run
    k=i+1
    #get the actual results
    results = schematic.get_result_item('Tasks\\SPara1\\S-Parameters\\S2,1',k)
    #get frequencies
    freqs = results.get_xdata()
    #get S-Parameter values
    S_Para = results.get_ydata()

    # Initialize value list for one MC sample point over all frequency points
    freq_pos = []
    freq = []
    S = []
    SdB = []
    S_real = []
    S_imag = []


    # Get results for each freq. point of interest from CST
    for j in range(len(freq_range)):
        freq_pos_j = freq_range_pos[j]
        freq_pos.append(freq_pos_j)

        freq_value_j = freqs[freq_pos_j]
        freq.append(freq_value_j)

        S_real_j = S_Para[freq_pos_j].real
        S_real.append(S_real_j)

        S_imag_j = S_Para[freq_pos_j].imag
```

```
143         S_imag.append(S_imag_j)
144
145         S_j=np.sqrt(S_real_j**2+S_imag_j**2)
146         S.append(S_j)
147
148         S_dB_j = 20*np.log10(S_j)
149         SdB.append(S_dB_j)
150
151
152     # Add results to the lists for all sample points
153     S_all.append(S)
154     S_real_all.append(S_real)
155     S_imag_all.append(S_imag)
156     SdB_all.append(SdB)
157
158
159 #close CST
160 cst.interface.DesignEnvironment.close(mycst)
161
162 # Return results
163 print('\n frequency range in GHz:', freq_range)
164 print('frequency range as positions in CST:', freq_pos)
165 print('S-parameter:',S_all)
166 print('S-parameter (real part):',S_real_all)
167 print('S-parameter (imag part):',S_imag_all)
168 print('S-parameter (in dB):',SdB_all)
169
170 # Return runtime
171 end = time.time()
172 print('\n Runtime: {:5.3f}seconds'.format(end-start))
```

./../Template_Lowpass_1_Para.py

```python
# -*- coding: utf-8 -*-
"""
Created on Mon Oct 19 15:02:27 2020

@author: Marc Bodem

Template for Calling CST from Python
- Lowpass Filter (Design Studio)
- Parameter Sweep (load only one sample point at a time)
- change one parameter at a time
- Obtain S-Parameter values

"""
import sys
sys.path.append(r"C:\Program Files (x86)\CST Studio Suite 2020\AMD64\
    python_cst_libraries")
import cst
import cst.interface
import cst.results
import numpy as np
import time
import shutil

start = time.time()

# Local path to CST project file --> Please adapt
cst_path = r'D:\Documents\Vorlagen\CST_Python_Interface' # path
cst_project = '\S-Parameter Lowpass' # CST project

cst_project_path = cst_path + cst_project + '.cst'
cst_result_folder = cst_path + cst_project + '\Result'


# Delete all old results (if exist)
try:
    shutil.rmtree(cst_result_folder)
except OSError as e:
    print(e)
else:
    print("The results directory is deleted successfully")


# Define parameter value for change
sample_point = 6.8


# Define list with frequency points of interest, here 0-7 GHz, Number of
    calculated points are set to 7001 in CST file, to get the desired frequencies.
freq_range = [0,1,2,3,4,5,6,7]
freq_range_pos = np.array(freq_range)*1000


# Initialize lists with S-parameter results
S_all = []
```

```
S_real_all = []
S_imag_all = []
SdB_all = []


# Open CST as software
mycst=cst.interface.DesignEnvironment()

# Open CST Project
mycst1=cst.interface.DesignEnvironment.open_project(mycst,cst_project_path)

#Delete All Sequences before starting to get a fresh file
delete ='Sub Main() \n ParameterSweep.DeleteAllSequences() \nEnd Sub'
mycst1.schematic.execute_vba_code(delete, timeout=None)


#VBA Add Sequence
createSequence = 'Sub Main () \n ParameterSweep.AddSequence("Seq1") \n End Sub'
#VBA Add Parameters to Sequence. For each Parameter combination you want to test,
    create 1 sequence
add_Para_L1 = 'Sub Main () \n ParameterSweep.AddParameter("Seq1","L1",'+str(True)+
    ','+str(sample_point)+','+str(sample_point)+','+str(1)+') \n End Sub'

#excute VBA Code above
mycst1.schematic.execute_vba_code(createSequence, timeout=None)
mycst1.schematic.execute_vba_code(add_Para_L1, timeout=None)


#Start Solver
solve = 'Sub Main () \n ParameterSweep.Start \nEnd Sub'
mycst1.schematic.execute_vba_code(solve, timeout=None)

#Delete All Sequences
mycst1.schematic.execute_vba_code(delete, timeout=None)

#get Project for results
project = cst.results.ProjectFile(cst_project_path, allow_interactive=True)

#get schamatic
schematic = project.get_schematic()

#get the actual results
results = schematic.get_result_item('Tasks\\SPara1\\S-Parameters\\S2,1')

#get frequencies
freqs = np.array(results.get_xdata())

#get S21 Parameter
S_Para = np.array(results.get_ydata())

# Initialize value list for one MC sample point over all frequency points
freq_pos = []
freq = []
S = []
SdB = []
```

```
S_real = []
S_imag = []


# Get results for each freq. point of interest from CST
for j in range(len(freq_range)):
    freq_pos_j = freq_range_pos[j]
    freq_pos.append(freq_pos_j)

    freq_value_j = freqs[freq_pos_j]
    freq.append(freq_value_j)

    S_real_j = S_Para[freq_pos_j].real
    S_real.append(S_real_j)

    S_imag_j = S_Para[freq_pos_j].imag
    S_imag.append(S_imag_j)

    S_j=np.sqrt(S_real_j**2+S_imag_j**2)
    S.append(S_j)

    S_dB_j = 20*np.log10(S_j)
    SdB.append(S_dB_j)


#close CST
cst.interface.DesignEnvironment.close(mycst)

# Return results
print('\n frequency range in GHz:', freq_range)
print('frequency range as positions in CST:', freq_pos)
print('S-parameter:',S)
print('S-parameter (real part):',S_real)
print('S-parameter (imag part):',S_imag)
print('S-parameter (in dB):',SdB)

# Return runtime
end = time.time()
print('\n Runtime: {:5.3f}seconds'.format(end-start))
```

./../Template_LLwaveguide.py

```
# -*- coding: utf-8 -*-
"""
Created on Mon Oct 19 12:12:06 2020

@author: Marc Bodem

Template for Calling CST from Python
- Lossy Load Waveguide (Microwave Studio)
- Parameter Sweep to load several sample points
- change 2 parameters
- Obtain S-Parameter values

"""
```

```
import sys
sys.path.append(r"C:\Program Files (x86)\CST Studio Suite 2020\AMD64\
    python_cst_libraries")
import cst
import cst.interface
import cst.results
import numpy as np
import scipy.stats
import time
import shutil

start = time.time()

# Local path to CST project file --> Please adapt
cst_path = r'D:\Documents\Vorlagen\CST_Python_Interface' # path
cst_project = '\Lossy Loaded Waveguide' # CST project

cst_project_path = cst_path + cst_project + '.cst'
cst_result_folder = cst_path + cst_project + '\Result'


# Delete all old results (if exist)
try:
    shutil.rmtree(cst_result_folder)
except OSError as e:
    print(e)
else:
    print("The results directory is deleted successfully")


# Define random list of unifrom distributed sample points
Nmc = 3 # number of sample points
Nuq = 2 # number of parameters per sample point
means = [0.05,0.9] # mean values for parameters
sample_list = []
for nuq in range(Nuq):
    mu = means[nuq]; uq = 0.2;
    samples_k = scipy.stats.uniform.rvs(mu-uq*mu,2*uq*mu,Nmc)
    sample_list.append(samples_k)
sample_list = np.array(sample_list).T


# Define list with frequency points of interest, here 0-7 GHz, Number of
    calculated points are set to 7001 in CST file, to get the desired frequencies.
freq_range = [80,90,100,110,120]
freq_range_pos_float = 0.1*(np.array(freq_range)-freq_range[0])*250
freq_range_pos= freq_range_pos_float.astype(int)


# Initialize lists with S-parameter results
S_all = []
S_real_all = []
S_imag_all = []
SdB_all = []
all_S_Para=[]
```

```
67
68
69  # Open CST as software
70  mycst=cst.interface.DesignEnvironment()
71
72
73  # Open CST Project
74  mycst1=cst.interface.DesignEnvironment.open_project(mycst,cst_project_path)
75
76
77  #Delete All Sequences before starting to get a fresh file
78  delete ='Sub Main() \n ParameterSweep.DeleteAllSequences() \nEnd Sub'
79  mycst1.schematic.execute_vba_code(delete, timeout=None)
80
81
82  for i in range(len(sample_list)):
83      #VBA Add Sequence
84      createSequence = 'Sub Main () \n ParameterSweep.AddSequence('+str(i)+') \n End
            Sub'
85      #VBA Add Parameters to Sequence. For each Parameter combination you want to
            test, create 1 sequence
86      add_Para_width = 'Sub Main () \n ParameterSweep.AddParameter('+str(i)+',"width
            ",'+str(True)+','+str(sample_list[i][0])+','+str(sample_list[i][0])+','+str
            (1)+') \n End Sub'
87      add_Para_dist = 'Sub Main () \n ParameterSweep.AddParameter('+str(i)+',"dist",
            '+str(True)+','+str(sample_list[i][1])+','+str(sample_list[i][1])+','+str
            (1)+') \n End Sub'
88
89      #excute VBA Code above
90      mycst1.schematic.execute_vba_code(createSequence, timeout=None)
91      mycst1.schematic.execute_vba_code(add_Para_width, timeout=None)
92      mycst1.schematic.execute_vba_code(add_Para_dist, timeout=None)
93
94  #Start Solver
95  solve = 'Sub Main () \n ParameterSweep.Start \nEnd Sub'
96  mycst1.schematic.execute_vba_code(solve, timeout=None)
97
98  #Delete All Sequences
99  mycst1.schematic.execute_vba_code(delete, timeout=None)
100
101 #get project for results
102 project = cst.results.ProjectFile(cst_project_path, allow_interactive=True)
103
104
105 # Evaluate each sample point in CST
106 for i in range(len(sample_list)):
107     #k = i+1 because run Ids start at 1 and 0 is the current run
108     k=i+1
109     #get the actual results
110     results = project.get_3d().get_result_item(r"1D Results\S-Parameters\S1,1",k)
111     #get frequencies
112     freqs = results.get_xdata()
113     #get S-Parameter values
114     S_Para = results.get_ydata()
115
```

```
    # Initialize value list for one MC sample point over all frequency points
    freq_pos = []
    freq = []
    S = []
    SdB = []
    S_real = []
    S_imag = []


    # Get results for each freq. point of interest from CST
    for j in range(len(freq_range)):
        freq_pos_j = freq_range_pos[j]
        freq_pos.append(freq_pos_j)

        freq_value_j = freqs[freq_pos_j]
        freq.append(freq_value_j)

        S_real_j = S_Para[freq_pos_j].real
        S_real.append(S_real_j)

        S_imag_j = S_Para[freq_pos_j].imag
        S_imag.append(S_imag_j)

        S_j=np.sqrt(S_real_j**2+S_imag_j**2)
        S.append(S_j)

        S_dB_j = 20*np.log10(S_j)
        SdB.append(S_dB_j)


    # Add results to the lists for all sample points
    S_all.append(S)
    S_real_all.append(S_real)
    S_imag_all.append(S_imag)
    SdB_all.append(SdB)


#close CST
cst.interface.DesignEnvironment.close(mycst)

# Return results
print('\n frequency range in GHz:', freq_range)
print('frequency range as positions in CST:', freq_pos)
print('S-parameter:',S_all)
print('S-parameter (real part):',S_real_all)
print('S-parameter (imag part):',S_imag_all)
print('S-parameter (in dB):',SdB_all)

# Return runtime
end = time.time()
print('\n Runtime: {:5.3f}seconds'.format(end-start))
```

./../Template_LLwaveguide_1_Para.py

```
# -*- coding: utf-8 -*-
```

```
"""
Created on Mon Oct 19 14:10:05 2020

@author: Marc Bodem

Template for Calling CST from Python
- Lossy Load Waveguide (Microwave Studio)
- Store Parameter to load one sample point at a time
- change one parameter at a time
- Obtain S-Parameter values

"""
import sys
sys.path.append(r"C:\Program Files (x86)\CST Studio Suite 2020\AMD64\
    python_cst_libraries")
import cst
import cst.interface
import cst.results
import numpy as np
import time
import shutil

start = time.time()

# Local path to CST project file --> Please adapt
cst_path = r'D:\Documents\Vorlagen\CST_Python_Interface' # path
cst_project = '\Lossy Loaded Waveguide' # CST project

cst_project_path = cst_path + cst_project + '.cst'
cst_result_folder = cst_path + cst_project + '\Result'


# Delete all old results (if exist)
# This option is only necessary in the Parameter Sweep Option - here it could only
    be a nice feature in order to save computing time
try:
    shutil.rmtree(cst_result_folder)
except OSError as e:
    print(e)
else:
    print("The results directory is deleted successfully")


# Define parameter value for change
sample_point = 0.06


# Define list with frequency points of interest, here 80-120 GHz, Number of
    calculated points are set to 1001 in CST file, to get the desired frequencies.
freq_range = [80,90,100,110,120]
# This line mights be adapted, depending on the number of frequency result points
    and the specified frequency range
freq_range_pos_float = 0.1*(np.array(freq_range)-freq_range[0])*250
freq_range_pos= freq_range_pos_float.astype(int)

```

```
53
54 # Initialize lists with S-parameter results
55 S_all = []
56 S_real_all = []
57 S_imag_all = []
58 SdB_all = []
59 all_S_Para=[]
60
61
62 # Open CST as software
63 mycst=cst.interface.DesignEnvironment()
64
65 # Open CST Project
66 mycst1=cst.interface.DesignEnvironment.open_project(mycst,cst_project_path)
67
68
69 #Delete All Sequences before starting to get a fresh file
70 dele = 'Sub Main () \n dim objName as object \n set objName = Result1D("S-
       Parameters") \n DeleteAt("truemodelchange") \nEnd Sub ()'
71 mycst1.schematic.execute_vba_code(dele, timeout=None)
72
73 #VBA Code for Parameter change and rebuild
74 par_change = 'Sub Main () \n StoreParameter("width", '+str(sample_point)+') \
       nRebuildOnParametricChange (bfullRebuild, bShowErrorMsgBox)\nEnd Sub'
75
76 #execute VBA Code above
77 mycst1.schematic.execute_vba_code(par_change, timeout=None)
78
79 #start solver
80 mycst1.modeler.run_solver()
81
82 #get project for results
83 project = cst.results.ProjectFile(cst_project_path, allow_interactive=True)
84
85 #get the results
86 results = project.get_3d().get_result_item(r"1D Results\S-Parameters\S1,1")
87
88 #get frequencies
89 freqs = np.array(results.get_xdata())
90
91 #get S21 Parameter
92 S_Para = np.array(results.get_ydata())
93
94 # Initialize value list for one MC sample point over all frequency points
95 freq_pos = []
96 freq = []
97 S = []
98 SdB = []
99 S_real = []
100 S_imag = []
101
102
103 # Get results for each freq. point of interest from CST
104 for j in range(len(freq_range)):
105     freq_pos_j = freq_range_pos[j]
```

```
106     freq_pos.append(freq_pos_j)
107
108     freq_value_j = freqs[freq_pos_j]
109     freq.append(freq_value_j)
110
111     S_real_j = S_Para[freq_pos_j].real
112     S_real.append(S_real_j)
113
114     S_imag_j = S_Para[freq_pos_j].imag
115     S_imag.append(S_imag_j)
116
117     S_j=np.sqrt(S_real_j**2+S_imag_j**2)
118     S.append(S_j)
119
120     S_dB_j = 20*np.log10(S_j)
121     SdB.append(S_dB_j)
122
123
124 #close CST
125 cst.interface.DesignEnvironment.close(mycst)
126
127 # Return results
128 print('\n frequency range in GHz:', freq_range)
129 print('frequency range as positions in CST:', freq_pos)
130 print('S-parameter:',S)
131 print('S-parameter (real part):',S_real)
132 print('S-parameter (imag part):',S_imag)
133 print('S-parameter (in dB):',SdB)
134
135 # Return runtime
136 end = time.time()
137 print('\n Runtime: {:5.3f}seconds'.format(end-start))
```