NetSim Multi-Parameter Sweep Program

Software Recommended:

NetSim v12.2 (32/64 bit), DOT NET CORE SDK 3.1, Python 3.7.4

Follow the instructions specified in the following link to clone/download the project folder from GitHub using Visual Studio:
https://tetcos.freshdesk.com/support/solutions/articles/14000099351-how-to-clone-netsim-file- exchange-project-repositories-from-github-

Other tools such as GitHub Desktop, SVN Client, Sourcetree, Git from the command line, or any client you like to clone the Git repository.

Note: It is recommended not to download the project as an archive (compressed zip) to avoid incompatibility while importing workspaces into NetSim.

Secure URL for the GitHub repository:

https://github.com/NetSim-TETCOS/Multi-Parameter-Sweeper_v12.2.git

Introduction:

When users want to *sweep* one or more parameters, they change their values between simulation runs, and compare and analyse the performance metrics from each run. NetSim multi-parameter sweeper enables users to automate the sweep process.

Consider an example, where a user wishes to create and simulate a network scenario for all possible values of one or more parameters in combination and analyse a set of performance metrics across the simulation runs. This is be extremely time consuming to do manually using the NetSim GUI.

The multi-parameter sweep program enables users to automate the sweep process across multiple input parameters, simulate each run, save each result, and compare specific output metrics via a spreadsheet software like MS Excel.

The sweep program runs NetSim via its CLI interface.

File Organization

The project directory consists of several binaries which are responsible for different tasks during a multi-parameter sweep:
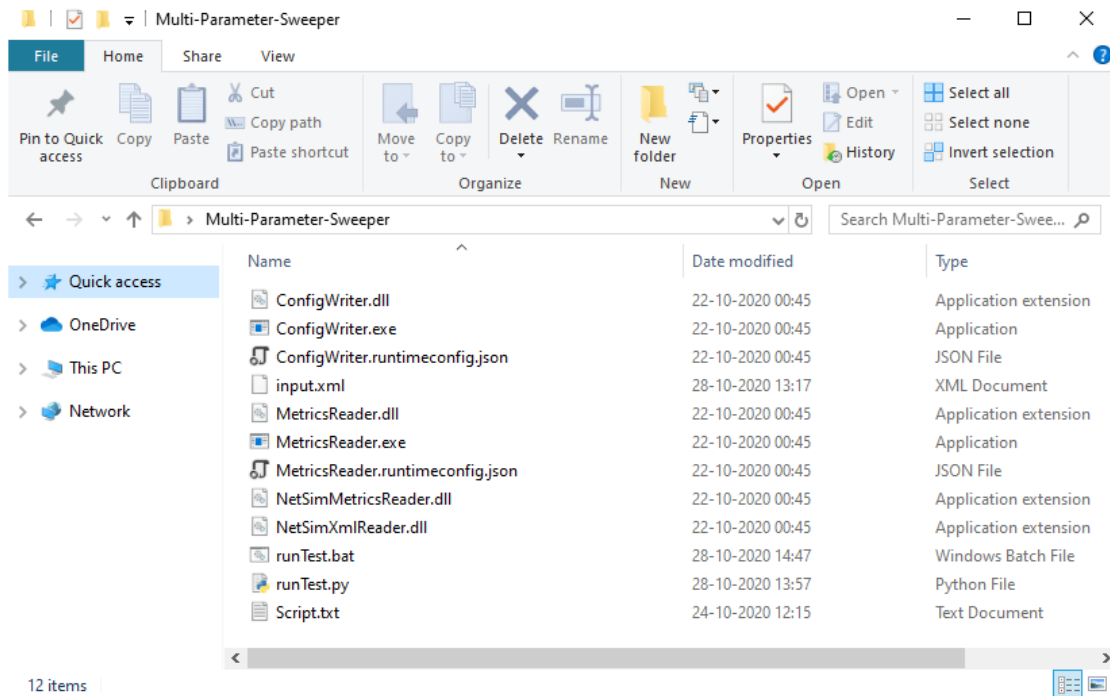
1.   **input.xml**: This file contains the base NetSim network configuration that is to be simulated. This file is created by copy pasting the content of the Configuration.netsim file that can be obtained by saving a network configuration in NetSim.

The values of parameters which are to be varied during each simulation run needs to be specified as {0}, {1}, {2}, etc. respectively.

For Example, if the X and Y coordinates of a device is to be varied the values can be modified in the input.xml file as shown below:
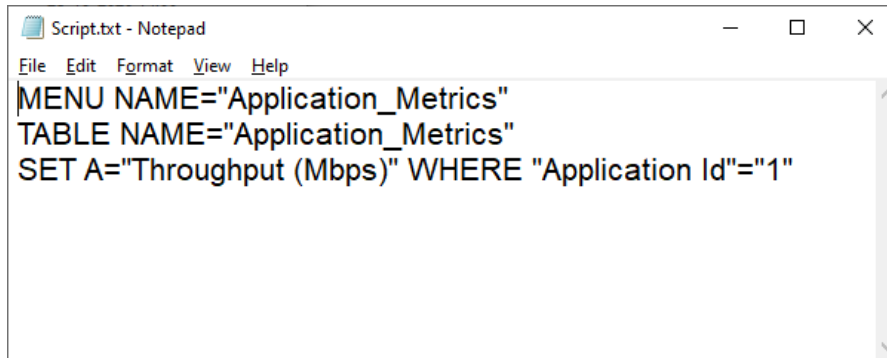
2.   **Script.txt:** This file should be updated with the parameter from the output metrics of NetSim that is to be logged at the end of each simulation run for the purpose of analysis.
At the end of every simulation, NetSim generates a Metrics.xml file which contain the performance metrics written in a specific format based on which it is loaded in the results dashboard.
Each Metric is part of a results table which can be accessed using a menu in the results dashboard.
A NetSim Metrics.xml file is shown below:

For Example, if the application throughput is to be logged for each simulation run then the scrip file can be updated as shown below:

```
Script.txt - Notepad                          —    □    ×
File  Edit  Format  View  Help
MENU NAME="Application_Metrics"
TABLE NAME="Application_Metrics"
SET A="Throughput (Mbps)" WHERE "Application Id"="1"
```
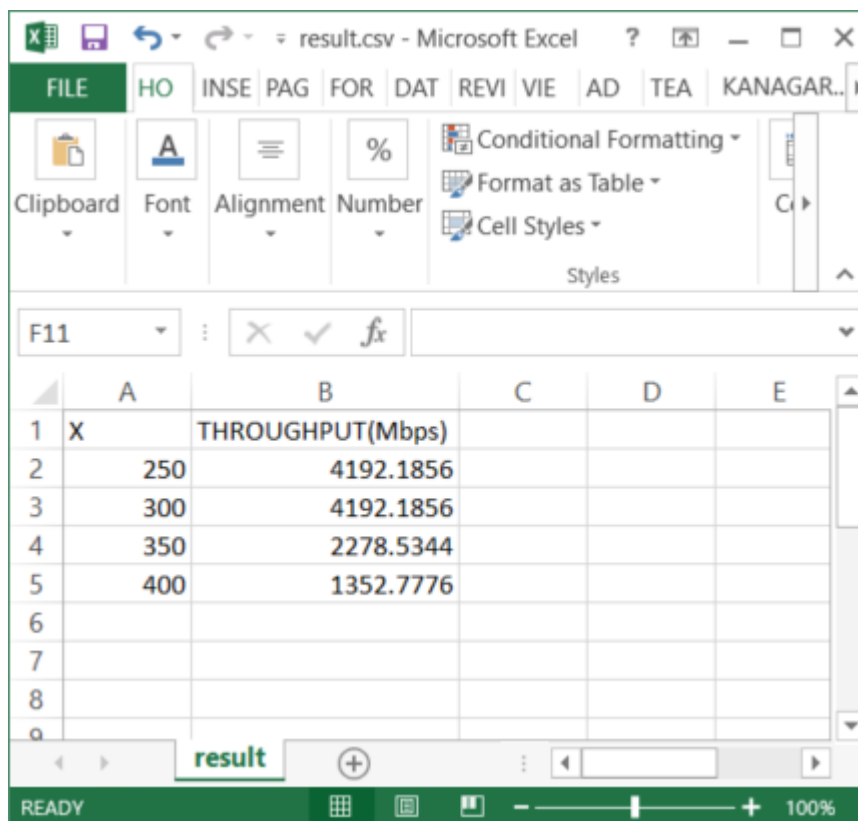
3.   **ConfigWriter.exe**: This executable takes one or more command line arguments as input and generated Configuration.netsim file by replacing the arguments in place of the variable parameters specified in the input.xml file.
If there are two variable parameters specified in the input.xml file ({0} and {1}) then two arguments need to be passed while calling ConfigWriter.exe.

4.   **MetricsReader.exe**: This executable is responsible for reading the output parameter from the Metrics.xml file generated after each simulation and logging it to the results file.
Users the Script.txt file to determine which parameter to read from the Metrics file.
If multiple parameters are to be read and logged then the MetricsReader.exe can be called multiple times with Script.txt file having information about the parameter to be read each time.

A	B
X	THROUGHPUT(Mbps)
250	4192.1856
300	4192.1856
350	2278.5344
400	1352.7776

5.   **Supporting DLL's**: Some the supporting files such as ConfigWriter.dll, MetricsReader.dll, NetSimMetricsReader.dll, NetSimXmlReader.dll, etc. which are present in the project folder are used by other executable such as ConfigWriter.exe and MetricsReader.exe for various purposes during a multi-parameter sweep.

6.   **runTest** script files: The files runTest.bat, runTest.py are the main script files that can be used to start a multi-parameter sweep process. Both files differ in the programming language used for the script.

runTest.bat uses Windows commands that can be executed by the windows command line interpreter. Batch scripts may get complex as the number of input and output parameters increases.

runTest.py uses python programming language which is less complex and offers more flexibility as the number of input and output parameters increases.

Users can also write the script to run the multi-parameter sweep process in a preferred programming language as per the convenience.

The script runs multiple simulation iterations based on the number of parameters to be varied and the range of values of each parameter.

Running a Multi-Parameter Sweep process:

Consider the following network 5G network scenario in NetSim, comprising of a Wired Node, EPC, gNB and a UE.
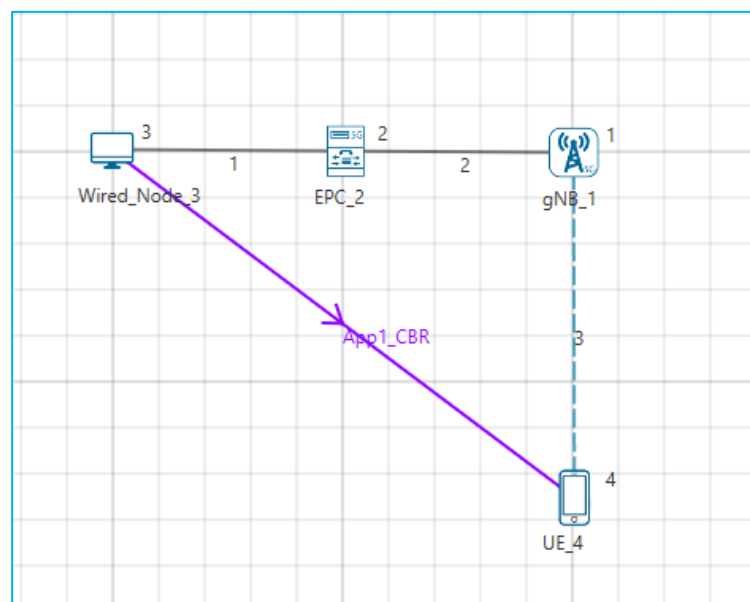
The network configuration has the initial distance between the gNB and UE as 50 meters with the gNB located at (1000,200) and UE located at (1000,250).
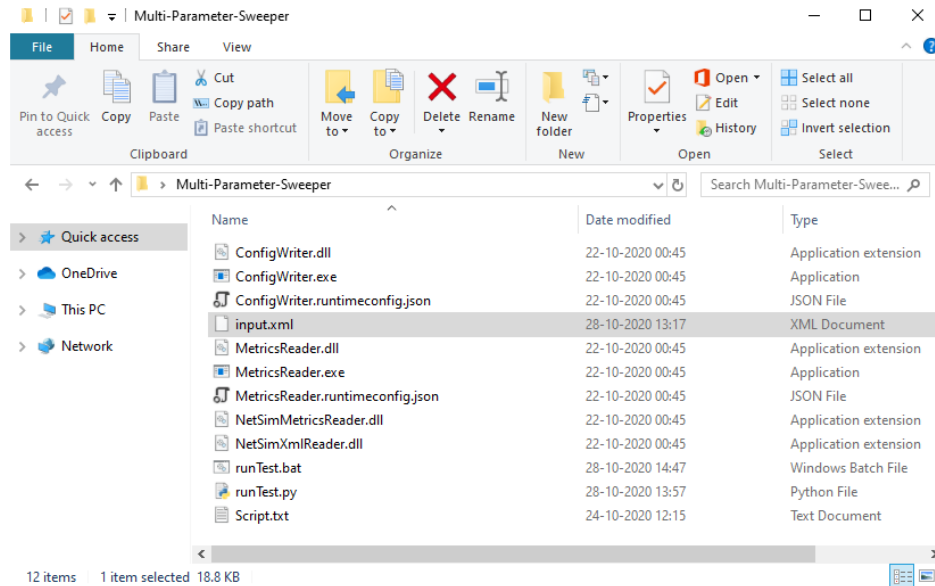
Multi-Parameter Sweeper is configured to run simulations for different distance between the gNB and UE by varying the UE Y coordinate value from 250 to 400 in steps of 50 meters.

1. The network configuration information is copied from the saved Configuration.netsim file to the input.xml file present in the Multi-Parameter-Sweeper project folder.

The value of the Y coordinate of UE that is to be modified during each simulation run is updated ("{0}") in the configuration file as shown below:
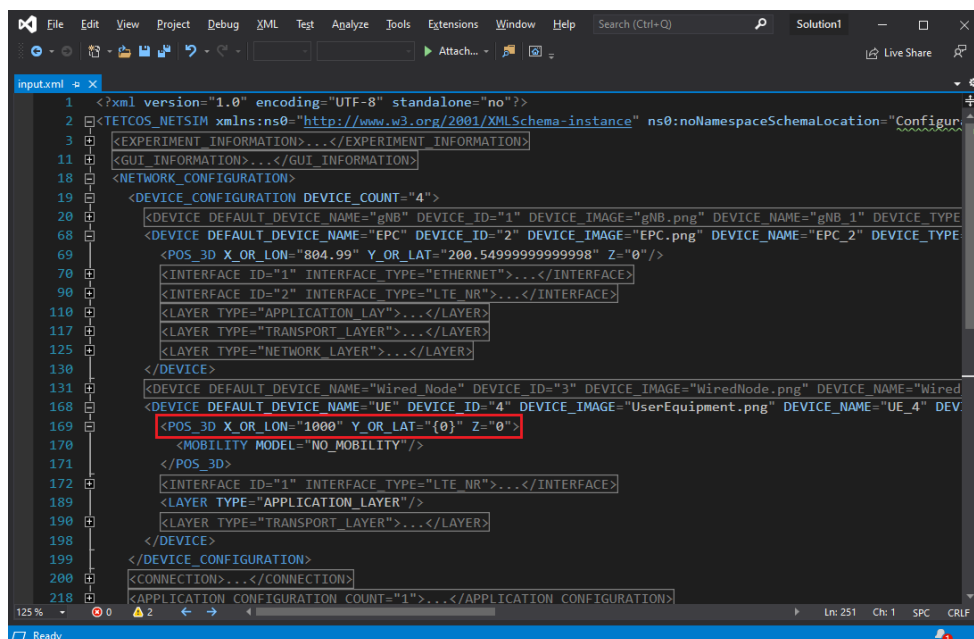
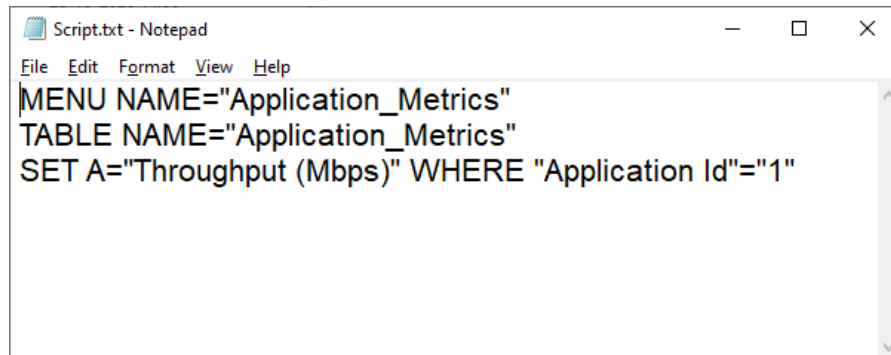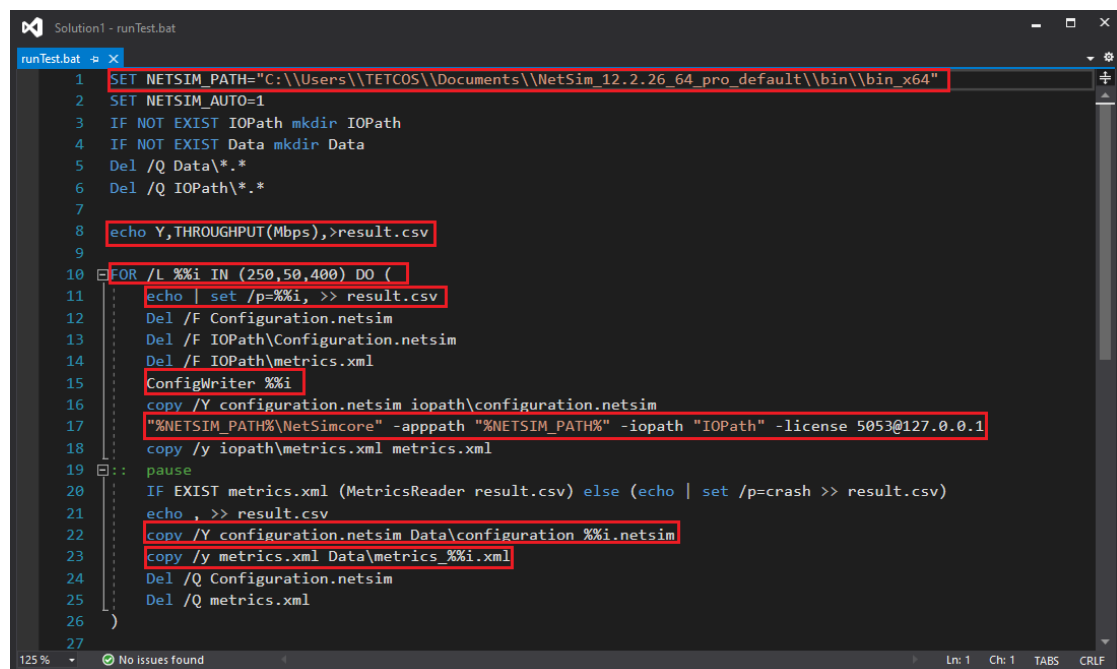2. The Script.txt file is updated with the details of the output parameter to be read from the Metrics.xml file and added to the result csv log file. In this case the Application throughput is to be logged for each simulation run.

3. runTest.py/runTest.bat is updated to pass the Y coordinate value during each iteration to generate Configuration file run simulation and update the result csv log.

   The runTest.bat batch script modified for running simulations for different values of Y coordinates starting from 250 up to 400 in steps of 50 is shown below:

- NETSIM_PATH variable is set to the path of NetSim 32-bit/64-bit binaries in the install directory or workspace in the system.
- A result.csv file is created and added with headings Y and Throughput(Mbps)
- For loop is set to iteratively run simulations for values starting from 250 to 400 in steps of 50.
- The value of the parameter Y in the current iteration is written to the result log file for analysis.

- The value of the parameter Y in the current iteration is passed as input to ConfigWriter executable to generate Configuration.netsim file for each simulation.
- NetSim simulation is run via CLI mode by passing the apppath, iopath and license server information
- Configuration file and Metrics file are copied and renamed appending the value of the parameter in the current iteration.

The runTest.py python script modified for running simulations for different values of Y coordinates starting from 250 up to 400 in steps of 50 is shown below:
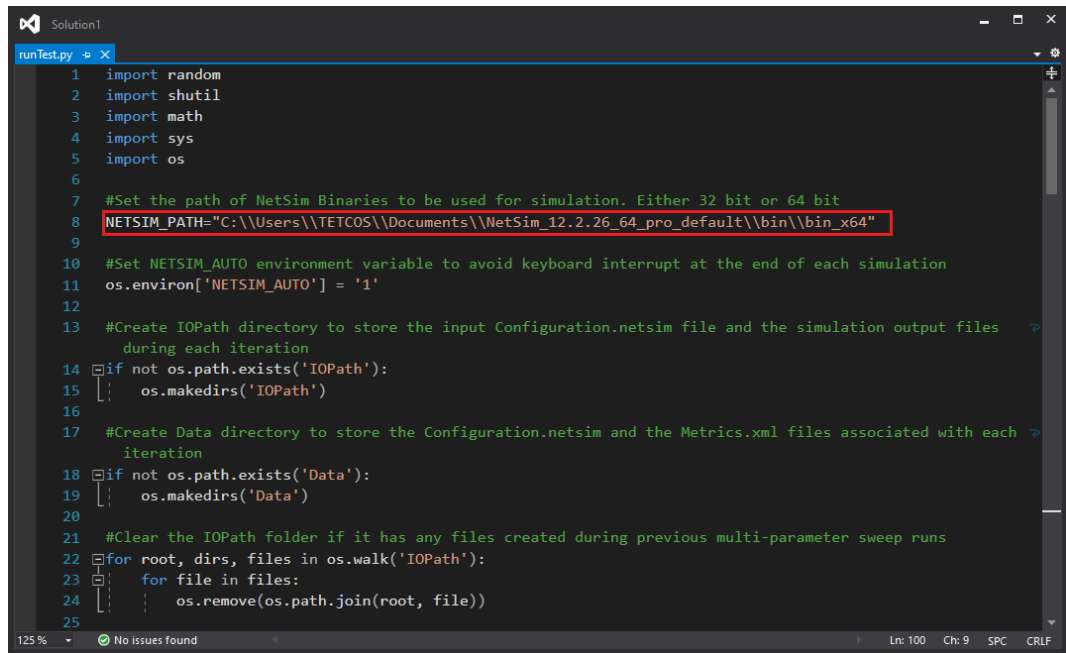


```python
import random
import shutil
import math
import sys
import os

#Set the path of NetSim Binaries to be used for simulation. Either 32 bit or 64 bit
NETSIM_PATH="C:\\Users\\TETCOS\\Documents\\NetSim_12.2.26_64_pro_default\\bin\\bin_x64"

#Set NETSIM_AUTO environment variable to avoid keyboard interrupt at the end of each simulation
os.environ['NETSIM_AUTO'] = '1'

#Create IOPath directory to store the input Configuration.netsim file and the simulation output files
    during each iteration
if not os.path.exists('IOPath'):
    os.makedirs('IOPath')

#Create Data directory to store the Configuration.netsim and the Metrics.xml files associated with each
    iteration
if not os.path.exists('Data'):
    os.makedirs('Data')

#Clear the IOPath folder if it has any files created during previous multi-parameter sweep runs
for root, dirs, files in os.walk('IOPath'):
    for file in files:
        os.remove(os.path.join(root, file))
```

- NETSIM_PATH variable is set to the path of NetSim 32-bit/64-bit binaries in the install directory or workspace in the system.

```
26    #Clear the Data folder if it has any files created during previous multi-parameter sweep runs
27    for root, dirs, files in os.walk('Data'):
28        for file in files:
29            os.remove(os.path.join(root, file))
30
31    #Delete result.csv file if it already exists
32    if(os.path.isfile("result.csv")):
33        os.remove("result.csv")
34
35    #create a csv file to log the output metrics for analysis
36    csvfile = open("result.csv", 'w')
37
38    #Add headings to the CSV file
39    csvfile.write('Y,THROUGHPUT(Mbps),')
40    csvfile.close()
41
42    #Iterate based on the number of time simulation needs to be run and the input parameter range
43    for i in range(250, 401, 50):
44
45        if(os.path.isfile("Configuration.netsim")):
46            os.remove("Configuration.netsim")
47
48        if(os.path.isfile("IOPath\Configuration.netsim")):
49            os.remove("IOPath\Configuration.netsim")
50
51        if(os.path.isfile("IOPath\Metrics.xml")):
52            os.remove("IOPath\Metrics.xml")
```

- A result.csv file is created and added with headings Y and Throughput(Mbps)
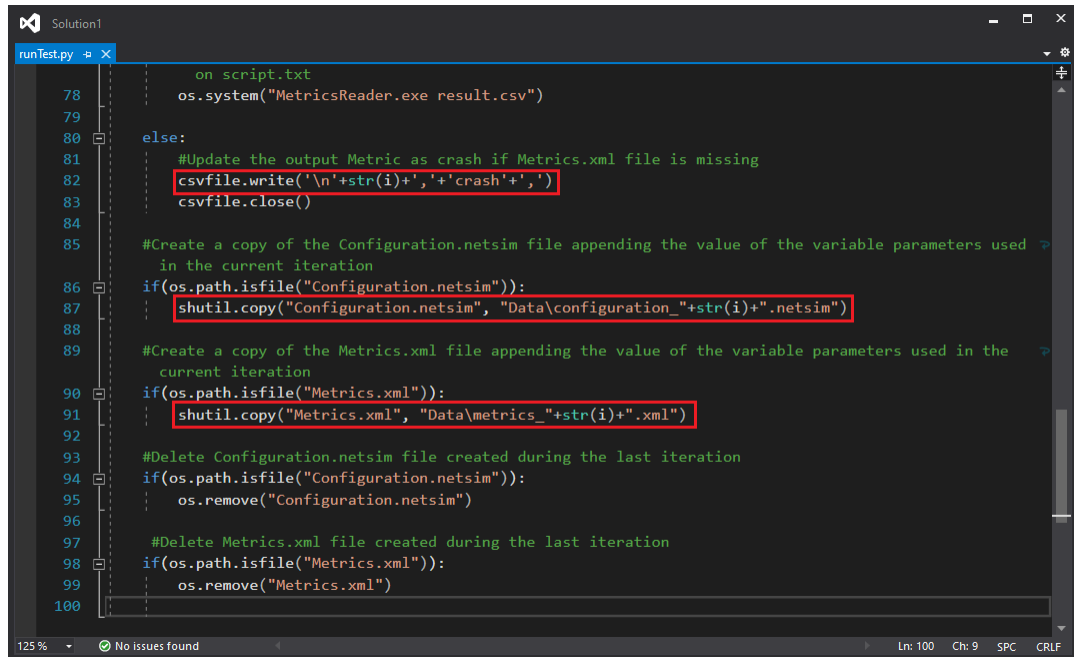- For loop is set to iteratively run simulations for values starting from 250 to 400 in steps of 50.


```
54        #Call ConfigWriter.exe with arguments as per the number of variable parameters in the input.xml file
55        cmd='ConfigWriter.exe '+str(i)
56        print(cmd)
57        os.system(cmd)
58
59        #Copy the Configuration.netsim file generated by ConfigWriter.exe to IOPath directory
60        if(os.path.isfile("Configuration.netsim")):
61            shutil.copy("Configuration.netsim","IOPath\Configuration.netsim")
62
63        #Run NetSim via CLI mode by passing the apppath iopath and license information to the NetSimCore.exe
64        cmd=NETSIM_PATH+"\\NetSimcore.exe -apppath "+NETSIM_PATH+" -iopath IOPath -license 5053@127.0.0.1"
65        os.system(cmd)
66        #print(cmd)
67
68        #Create a copy of the output Metrics.xml file for writing the result log
69        if(os.path.isfile("IOPath\Metrics.xml")):
70            shutil.copy("IOPath\Metrics.xml","Metrics.xml")
71
72        #Write the value of the variable parameter in the current iteration to the result log
73        csvfile = open("result.csv", 'a')
74        if(os.path.isfile("Metrics.xml")):
75            csvfile.write('\n'+str(i)+',')
76            csvfile.close()
77            #Call the MetricsReader.exe passing the name of the output log file for updating the log based
                on script.txt
78            os.system("MetricsReader.exe result.csv")
79
```

- The value of the parameter Y in the current iteration is passed as input to ConfigWriter executable to generate Configuration.netsim file for each simulation.
- NetSim simulation is run via CLI mode by passing the apppath, iopath and license server information.

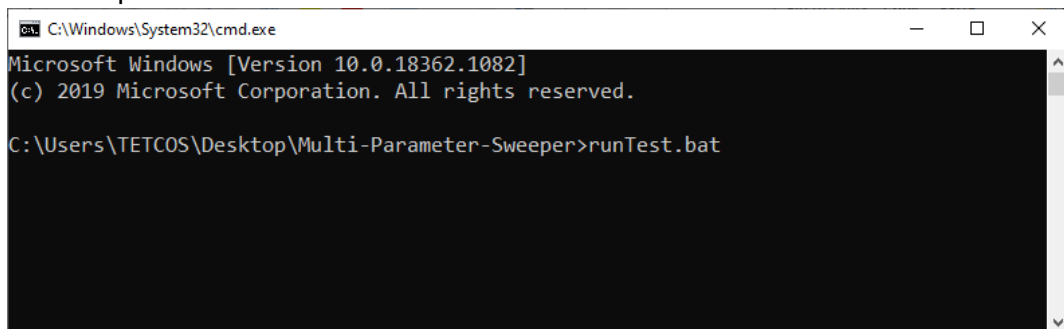- The value of the parameter Y in the current iteration is written to the result log file for analysis.


```python
               on script.txt
78            os.system("MetricsReader.exe result.csv")
79
80    else:
81            #Update the output Metric as crash if Metrics.xml file is missing
82            csvfile.write('\n'+str(i)+','+'crash'+',')
83            csvfile.close()
84
85    #Create a copy of the Configuration.netsim file appending the value of the variable parameters used
         in the current iteration
86    if(os.path.isfile("Configuration.netsim")):
87            shutil.copy("Configuration.netsim", "Data\configuration_"+str(i)+".netsim")
88
89    #Create a copy of the Metrics.xml file appending the value of the variable parameters used in the
         current iteration
90    if(os.path.isfile("Metrics.xml")):
91            shutil.copy("Metrics.xml", "Data\metrics_"+str(i)+".xml")
92
93    #Delete Configuration.netsim file created during the last iteration
94    if(os.path.isfile("Configuration.netsim")):
95            os.remove("Configuration.netsim")
96
97     #Delete Metrics.xml file created during the last iteration
98    if(os.path.isfile("Metrics.xml")):
99            os.remove("Metrics.xml")
100
```

- Configuration file and Metrics file are copied and renamed appending the value of the parameter in the current iteration.

4. Multi-Parameter Sweeping process is started by opening command prompt in the directory of the Multi-Parameter-Sweeping project and starting the batch script or the python script as shown below:
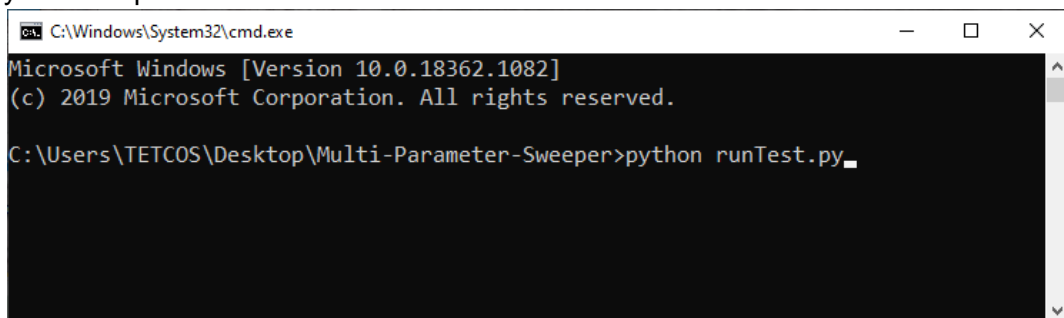
Batch Script:


```
Microsoft Windows [Version 10.0.18362.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\TETCOS\Desktop\Multi-Parameter-Sweeper>runTest.bat
```

Python Script:


```
Microsoft Windows [Version 10.0.18362.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\TETCOS\Desktop\Multi-Parameter-Sweeper>python runTest.py
```

This starts the Multi-Parameter-Sweeping process starts and run simulations iteratively for different values of Y parameter of UE.

At the end of the process the Multi-Parameter-Sweeping folder will have the following file and folders created:
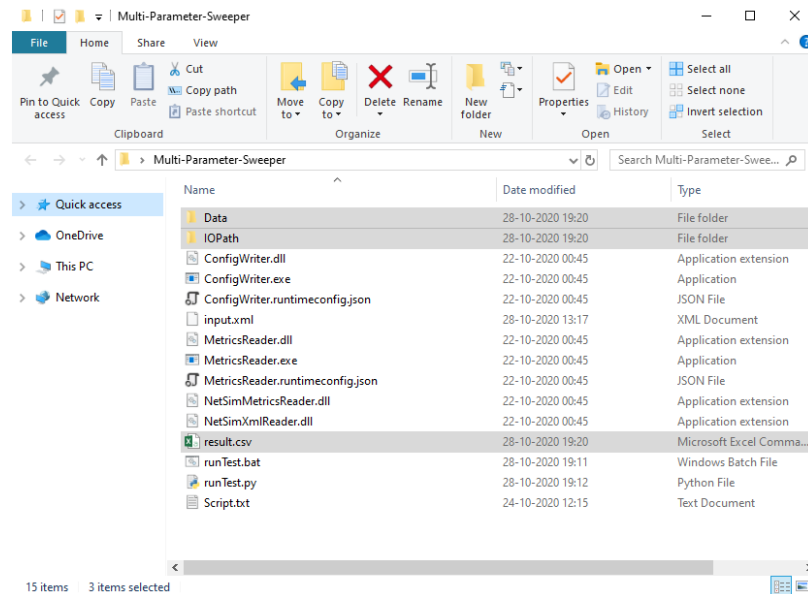
- **Data**: The Data directory contains the Configuration.netsim and the Metrics.xml files associated with each simulation run, renamed including the value of the parameter in the file name.
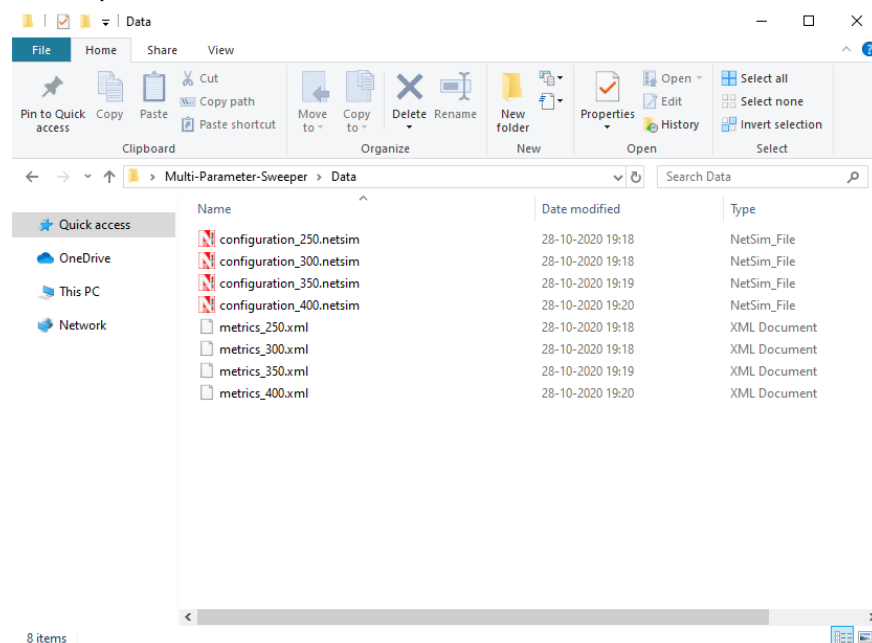
- **IOPath**: Used for storing the Configuration.netsim file and the simulation files generated during each simulation run.
- **Result.csv**: This is the output log which contains the parameter varied during each simulation run and the output parameter associated with each run.

Varying multiple network parameters:

In order to vary multiple network parameters during the multi-parameter sweep process each parameter in the input.xml file can be modified as {0},{1},{2},{3},…..{n} respectively.

Logging multiple output parameters:

Each output parameter that is to be logged should be part of the Script.txt file. However, the Script.txt file should contain only the details of one output parameter during the call to MetricsReader.exe.

In order to log multiple parameter, MetricsReader.exe can be called multiple times with Script.txt file having information about different parameter during each call.

For Example, there can be two Script files as shown below:
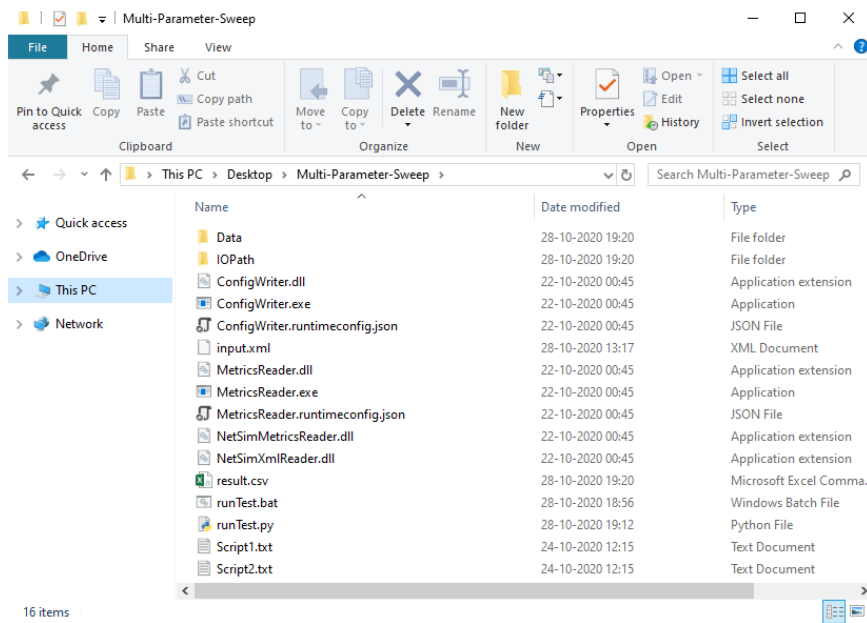
During each call to MetricsReader each of the Script files (Script1.txt and Script2.txt) can be renamed to Script.txt and renamed back.