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:
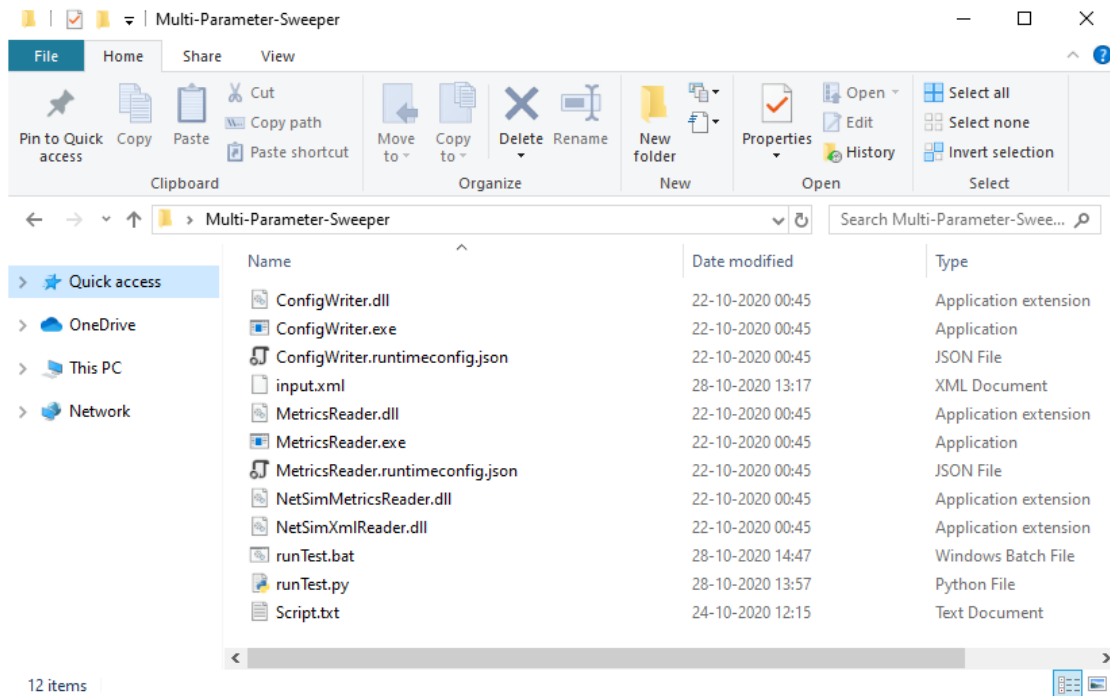
1.   **input.xml**: This file contains the base NetSim network configuration that is to be simulated. This file is created by copy pasting the Configuration.netsim file that can be obtained by saving a network configuration in NetSim and renaming it to input.xml.
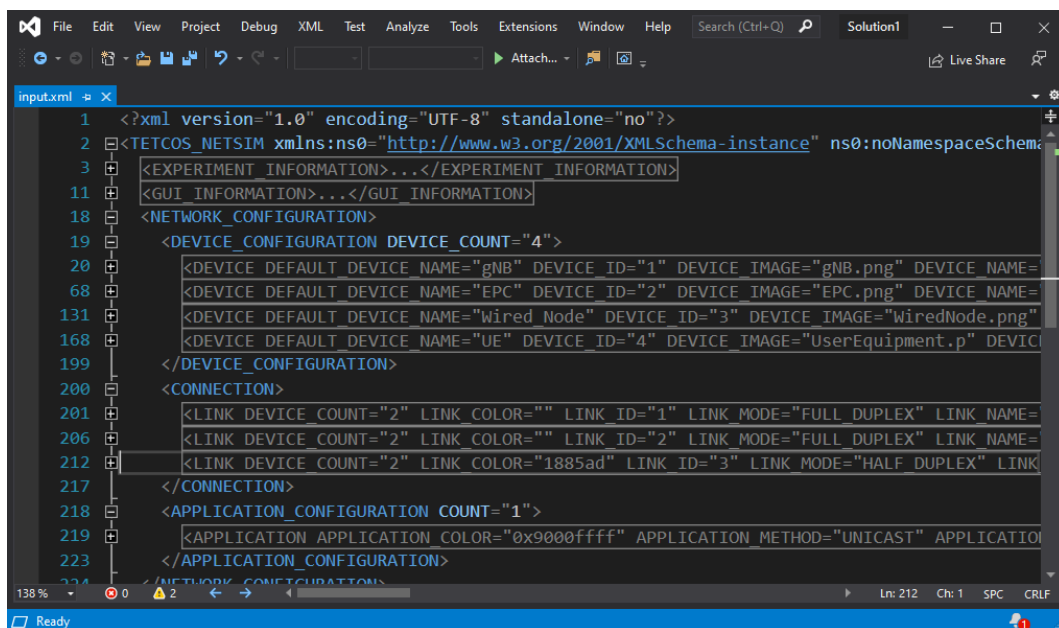
The values of parameters which are to be varied during each simulation run needs to be specified as {0}, {1}, {2}, etc. respectively.

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.
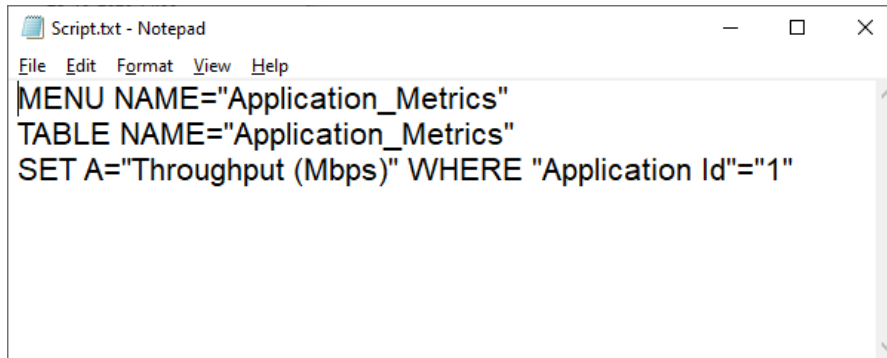A NetSim Metrics.xml file is shown below:

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"
```
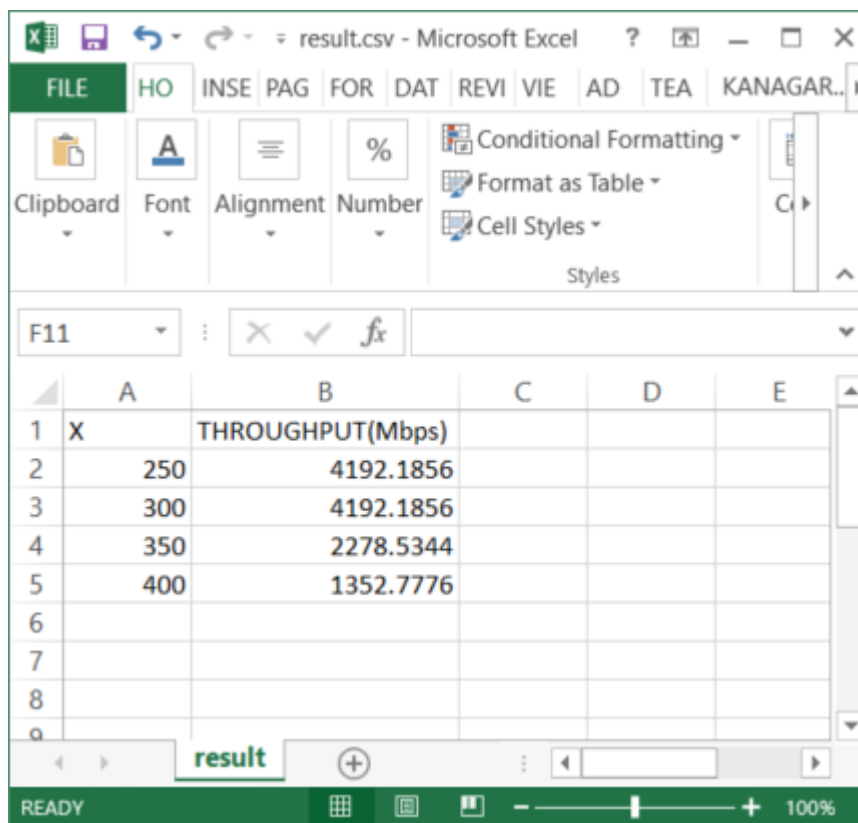
3.   **ConfigWriter.exe**: This executable takes one or more command line arguments as input and generated Configuration.netsim file by replacing the arguments in place of the variable parameters specified in the input.xml file.
If there are two variable parameters specified in the input.xml file ({0} 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.

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:

Example 1: Modifying a single input parameter and logging a single output parameter

Consider the following network 5G network scenario in NetSim, comprising of a Wired Node, EPC, gNB and a UE.
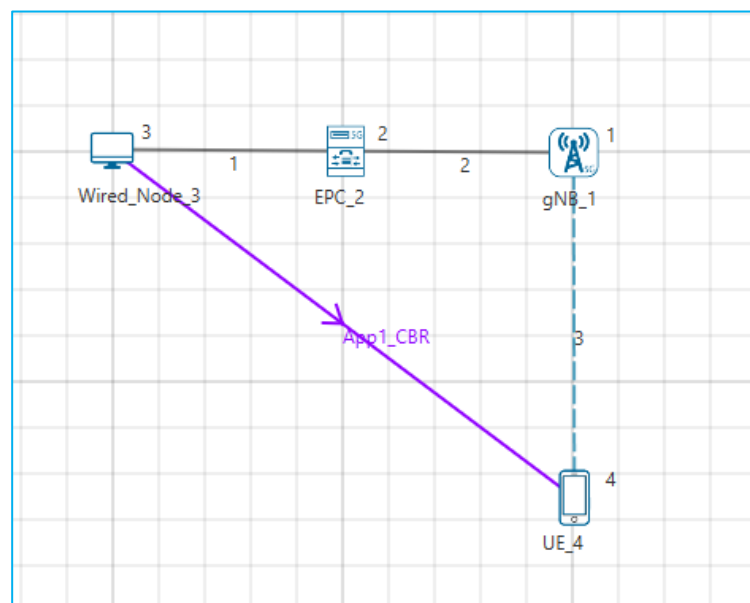
The network configuration has the initial distance between the gNB and UE as 50 meters with the gNB located at (1000,200) and UE located at (1000,250).

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 scenario is saved and the content of the Configuration.netsim file is copied to the Multi-Parameter-Sweeper directory and renamed as input.xml.
Refer to the Example 1 directory which is part of the project folder (Multi-Parameter-Sweeper_v12.2\Examples\Multi-Parameter-Sweeper-Example-1)
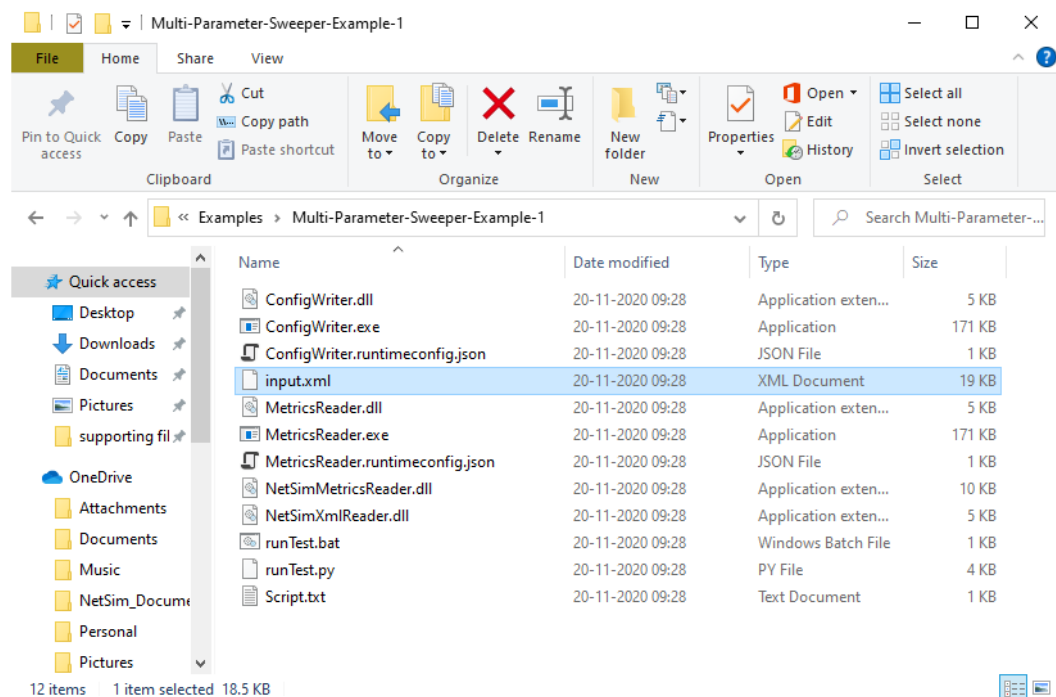
The value of the Y coordinate of UE that is to be modified during each simulation run is updated ("{0}") in the configuration file as shown below:

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.


```
MENU NAME="Application_Metrics"
TABLE NAME="Application_Metrics"
SET A="Throughput (Mbps)" WHERE "Application Id"="1"
```
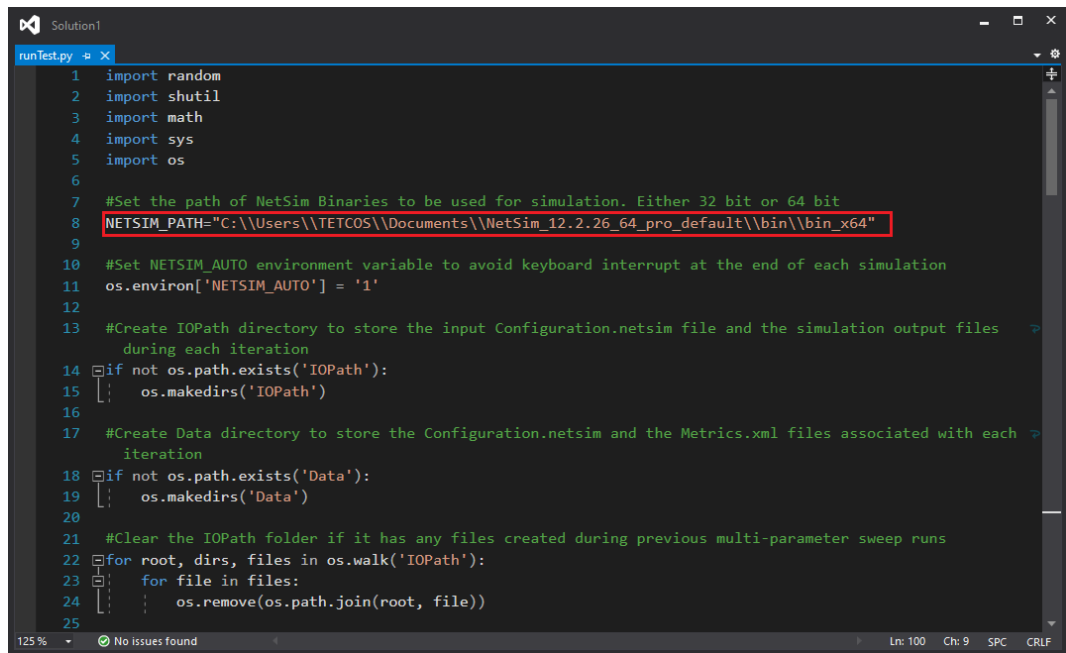
3. runTest.py/runTest.bat is updated to pass the Y coordinate value during each iteration to generate Configuration file run simulation and update the result csv log.

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:

```
1    SET NETSIM_PATH="C:\\Users\\TETCOS\\Documents\\NetSim_12.2.26_64_pro_default\\bin\\bin_x64"
2    SET NETSIM_AUTO=1
3    IF NOT EXIST IOPath mkdir IOPath
4    IF NOT EXIST Data mkdir Data
5    Del /Q Data\*.*
6    Del /Q IOPath\*.*
7
8    echo Y,THROUGHPUT(Mbps),>result.csv
9
10   FOR /L %%i IN (250,50,400) DO (
11       echo | set /p=%%i, >> result.csv
12       Del /F Configuration.netsim
13       Del /F IOPath\Configuration.netsim
14       Del /F IOPath\metrics.xml
15       ConfigWriter %%i
16       copy /Y configuration.netsim iopath\configuration.netsim
17       "%NETSIM_PATH%\NetSimcore" -apppath "%NETSIM_PATH%" -iopath "IOPath" -license 5053@127.0.0.1
18       copy /y iopath\metrics.xml metrics.xml
19   ::  pause
20       IF EXIST metrics.xml (MetricsReader result.csv) else (echo | set /p=crash >> result.csv)
21       echo , >> result.csv
22       copy /Y configuration.netsim Data\configuration %%i.netsim
23       copy /y metrics.xml Data\metrics_%%i.xml
24       Del /Q Configuration.netsim
25       Del /Q metrics.xml
26   )
27
```
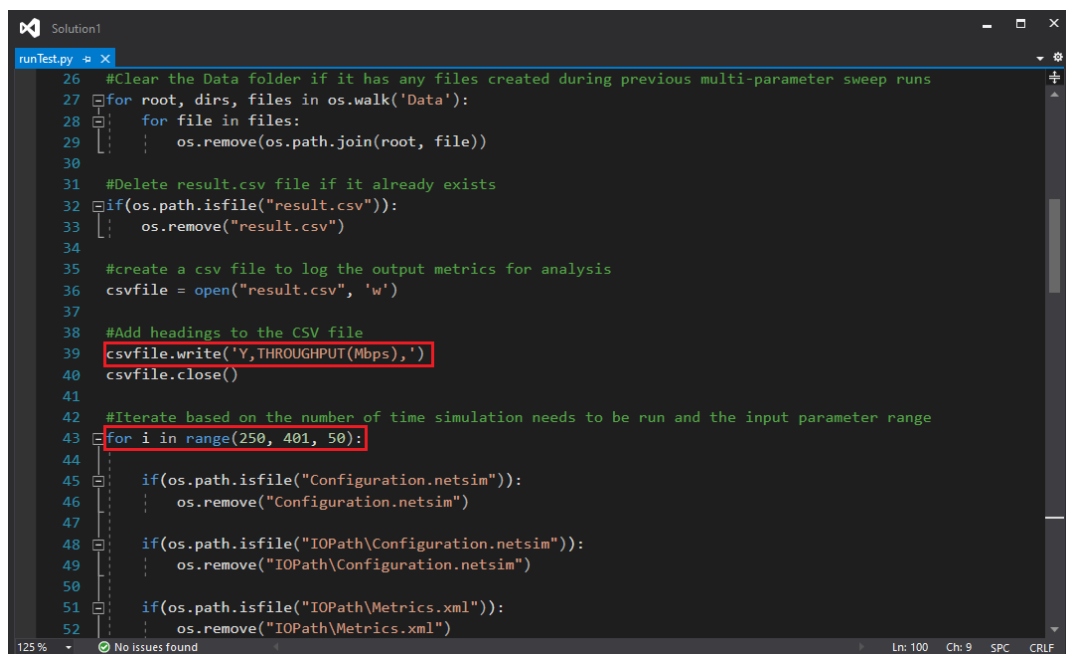
- NETSIM_PATH variable is set to the path of NetSim 32-bit/64-bit binaries in the install directory or workspace in the system.
- A result.csv file is created and added with headings Y and Throughput(Mbps)
- For loop is set to iteratively run simulations for values starting from 250 to 400 in steps of 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:

- 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.

```
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.
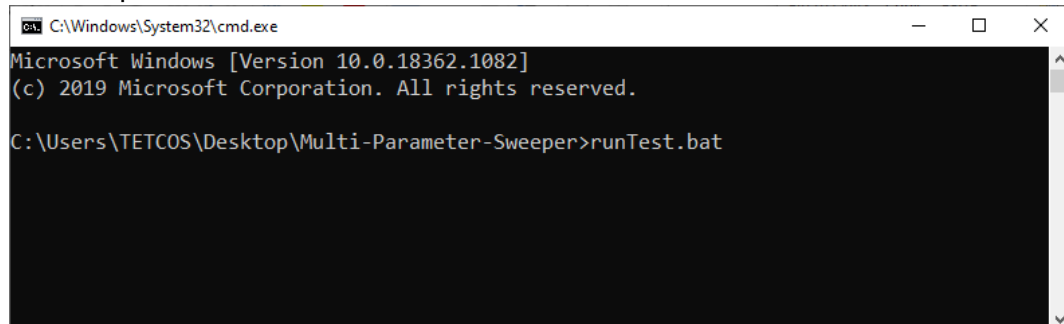


```
              on script.txt
78         os.system("MetricsReader.exe result.csv")
79
80     else:
81         #Update the output Metric as crash if Metrics.xml file is missing
82         csvfile.write('\n'+str(i)+','+'crash'+',')
83         csvfile.close()
84
85     #Create a copy of the Configuration.netsim 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:
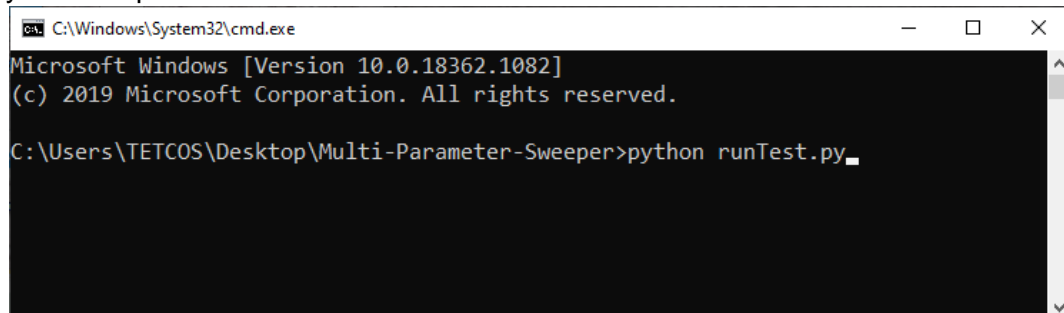
Batch Script:

Python Script:

This starts the Multi-Parameter-Sweeping process which runs NetSim 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.

Name	Date modified	Type
configuration_250.netsim	28-10-2020 19:18	NetSim_File
configuration_300.netsim	28-10-2020 19:18	NetSim_File
configuration_350.netsim	28-10-2020 19:19	NetSim_File
configuration_400.netsim	28-10-2020 19:20	NetSim_File
metrics_250.xml	28-10-2020 19:18	XML Document
metrics_300.xml	28-10-2020 19:18	XML Document
metrics_350.xml	28-10-2020 19:19	XML Document
metrics_400.xml	28-10-2020 19:20	XML Document

- **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.

	A	B
1	Y	THROUGHPUT(Mbps)
2	250	4192.1856
3	300	4192.1856
4	350	2278.5344
5	400	1352.7776

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.

To log multiple parameters, 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.

Example 2: Modifying multiple input parameters and logging multiple output parameter

Consider the following network 5G network scenario in NetSim, comprising of a Wired Node, EPC, gNB and a UE.
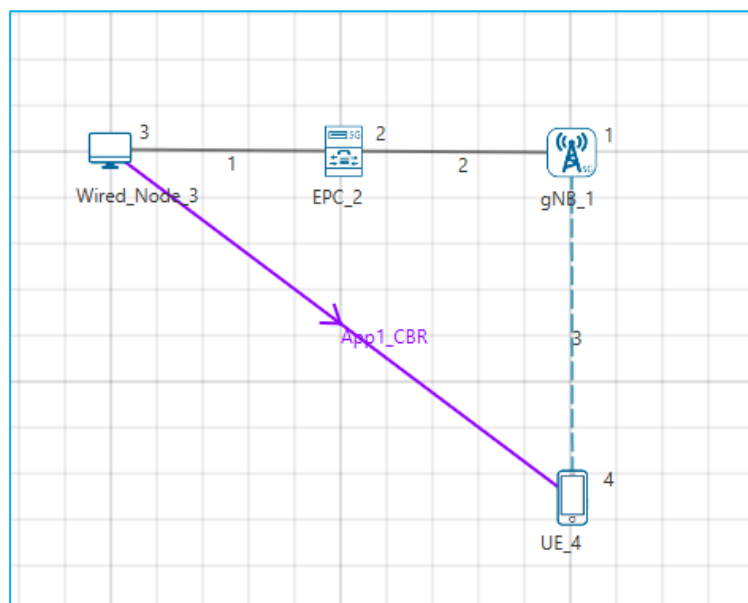
Properties configured in the LTE_NR interface of the gNB is shown in the table below:

Interface(LTE-NR) Properties	
Tx_Power(dBM)	40
Downlink_MIMO_Layer_Count	2
Uplink_MIMO_Layer_Count	1
CA_Type	Single Band
CA_Configuration	n78
CA_Count	1
MU	0

Channel Bandwidth (MHz)	10
PRB Count	52
MCS Table	QAM64
CQI Table	Table 1
X_Overhead	XOH0
DL UL Ratio	4:1
Outdoor Scenario	Rural Macro
LOS Mode	Standard
Wireless Link Properties	
Channel Characteristics	No_Pathloss
Wired Link Properties	
Link Speed (Mbps)	10000
BER	0
Propagation Delay (µs)	0
Application Properties	
Packet Size (Byte)	1460
Inter Arrival Time (µs)	166
Generation Rate (Mbps)	100
Transport Control	UDP
Start Time (s)	1
QoS	BE
Simulation Parameters	
Simulation Time (s)	1.1

Traffic is generated at a rate of 70 Mbps and upon running simulation, the throughput achieved is 59.95 Mbps.

We now find the max throughput for each possible bandwidth, DL MIMO count combination varying the generation rate based accordingly.

Two more parameters to be taken care include, the PRB Count and Guard Band(KHz) which vary with respect to the bandwidth.

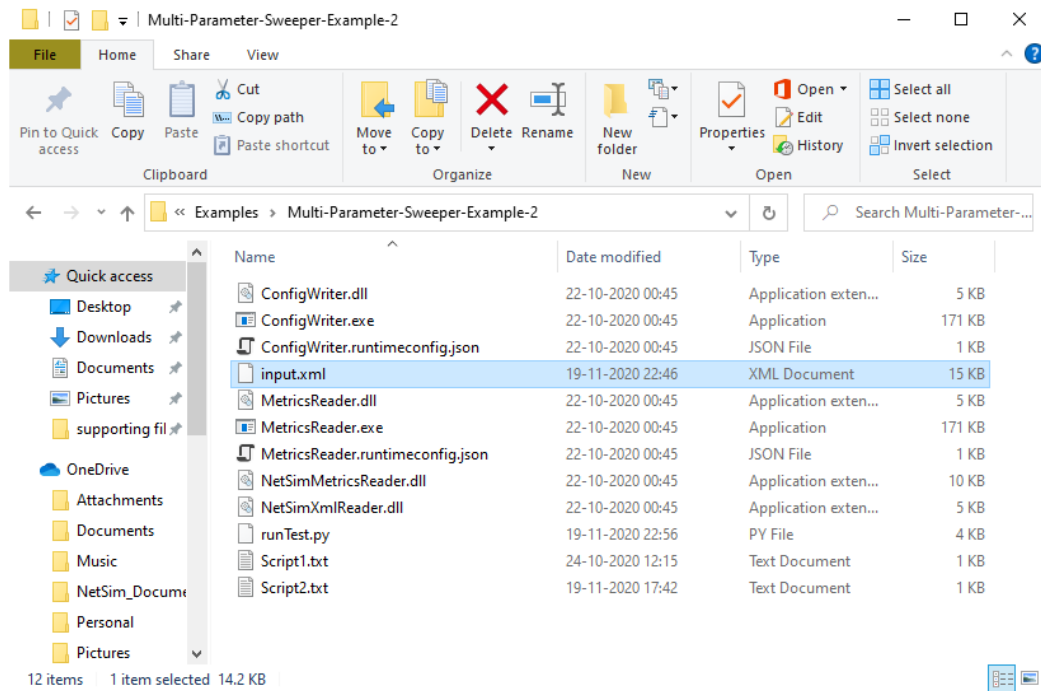| Input Variables	Value Range
Channel Bandwidth (MHz)	10,15,20,25,30,40,50
Downlink_MIMO_Layer_Count	2,4,8
PRB Count	52,79,106,133,160,216,270
Guard Band (KHz)	312.5,382.5,452.5,522.5,592.5,552.5,692.5
Reference Inter Arrival Time (Micro Seconds)	166
Reference Bandwidth	10
Reference DL MIMO Layer Count	2

Inter Arrival Time for each case is calculated based on the Reference IAT Bandwidth and DL MIMO Layer Count as shown below:

$$Inter\ Arrival\ Time\ (Micro\ Seconds) = \frac{Ref\ IAT}{\left(\frac{Curr\ BW}{Ref\ BW}\right) * \left(\frac{Curr\ DL\ MIMO\ Count}{Ref\ DL\ MIMO\ Count}\right)}$$

For E.g. In case of Bandwidth of 20 MHz and DL MIMO Count of 4 inter arrival time is

$$Inter\ Arrival\ Time\ (Micro\ Seconds) = \frac{166}{\left(\frac{20}{10}\right) * \left(\frac{4}{2}\right)} = 41.5\ Mbps$$

1. The network scenario is saved and the content of the Configuration.netsim file is copied to the Multi-Parameter-Sweeper directory and renamed as input.xml.
2. Refer to the Example 2 directory which is part of the project folder (Multi-Parameter-Sweeper_v12.2\Examples\Multi-Parameter-Sweeper-Example-2)

3. In the Input.xml file the value of the input variables are modified as shown in the table below:

Input Variables	
Channel Bandwidth (MHz)	{0}
Downlink_MIMO_Layer_Count	{1}
Inter Arrival Time (Micro Seconds)	{2}
PRB Count	{3}
Guard Band (KHz)	{4}

4. The python script runTest.py is modified to run simulation for all possible combinations of Bandwidth and Downlink MIMO Layer Count with the respective values of Guard Band, PRB Count and the IAT that is calculated.

5. Multiple parameters are read from the Metrics.xml file and logged in the results.csv file along with the input parameters such as CHANNELBANDWIDTH_MHz, DOWNLINK_MIMO_LAYER_COUNT, INTER_ARRIVAL_TIME(micro sec)

Output Parameters
Throughput(Mbps)
Data Packets transmitted

6. Two script text files namely Script1.txt and Script2.txt are created with information to read each of the parameters from the Metrics.xml file.
Script1.txt

Scritp2.txt

7. In the python script runTest.py, MetricsReader is called twice, once to log each parameter by renaming the script files to Script.txt and back to its original name and separating the entries with a comma (","). The input parameters that were varied during each simulation run are also logged in the results.csv file.

8. The Input Configuration.netsim and the output Metrics.xml files associated with each simulation run is renamed including the bandwidth and DL MIMO count values that were used during each simulation run.

```
75          csvfile.write(',')
76          csvfile.close()
77          os.rename("Script.txt","Script1.txt");
78          os.rename("Script2.txt","Script.txt");
79          os.system("MetricsReader.exe result.csv")
80          os.rename("Script.txt","Script2.txt");
81
82      else:
83          csvfile.write('\n'+str(i)+','+str(j)+','+str(iat)+','+'crash'+','+'crash'+',')
84          csvfile.close()
85
86
87      if(os.path.isfile("Configuration.netsim")):
88          shutil.copy("Configuration.netsim", "Data\configuration_"+str(i)\
89          +'_'+str(j)+".netsim")
90
91      if(os.path.isfile("Metrics.xml")):
92          shutil.copy("Metrics.xml", "Data\metrics_"+str(i)+'_'+str(j)+".xml")
93
94      if(os.path.isfile("Configuration.netsim")):
95          os.remove("Configuration.netsim")
96
97      if(os.path.isfile("Metrics.xml")):
98          os.remove("Metrics.xml")
99
```

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

Python Script:

This starts the Multi-Parameter-Sweeping process which runs NetSim simulations iteratively for different combinations of input parameters.

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.

CHANNELBANDWIDTH_MHz	DOWNLINK_MIMO_LAYER_COUNT	INTER_ARRIVAL_TIME(micro sec)	THROUGHPUT(Mbps)	Data Packets transmitted
10	2	166	59.8016	1790
10	4	83	120.4208	3518
10	8	41.5	238.0384	6932
15	2	110.6666667	90.9872	2663
15	4	55.33333333	183.1424	5260
15	8	27.66666667	362.1968	10405
20	2	83	123.4576	3542
20	4	41.5	248.5504	7023
20	8	20.75	491.4944	13922
25	2	66.4	154.6432	4403
25	4	33.2	311.272	8765
25	8	16.6	615.6528	17393
30	2	55.33333333	185.8288	5282
30	4	27.66666667	373.9936	10508
30	8	13.83333333	739.8112	20866
40	2	41.5	250.7696	7042
40	4	20.75	504.8096	14038
40	8	10.375	998.5232	27901
50	2	33.2	314.4256	8780
50	4	16.6	632.9392	17542
50	8	8.3	1251.9792	34889