

NetSim[®]

Accelerate Network R & D

Multi-Parameter Sweeper Python

Version 15.0

A Network Simulation & Emulation Software

By

TETCOS LLP

Contents

1	Introduction	3
2	File Organization	3
2.1	Input Configuration	3
2.2	Metric Script	4
2.3	Helper Executables and Python Script	5
3	Running the Sweeper	6
4	Example 1: Single Input Parameter	10
4.1	Python Script Changes	12
5	Example 2: Multiple Input Parameters	17
6	Advanced Use Cases	25
6.1	Running the Sweeper with Modified Workspace Binaries	25
6.2	Configurations with Supporting Files	26

Software: NetSim v15.0 (64 bit), .NET Core SDK 3.1, Python 3.11.1 or later.

Project download link:

<https://github.com/NetSim-TETCOS/Multiparameter-Sweeper-Python-v15.0/archive/refs/heads/main.zip>

1 Introduction

NetSim Multi-Parameter Sweeper automates the network simulation process by running NetSim through the command line interface. Instead of changing parameters and analyzing results manually in the GUI, users mark the input parameters to vary, run the Python script, and collect output metrics in CSV files.

2 File Organization

The project directory contains the helper executables, supporting DLL files, input configuration, metric scripts, and Python script used during a multi-parameter sweep.

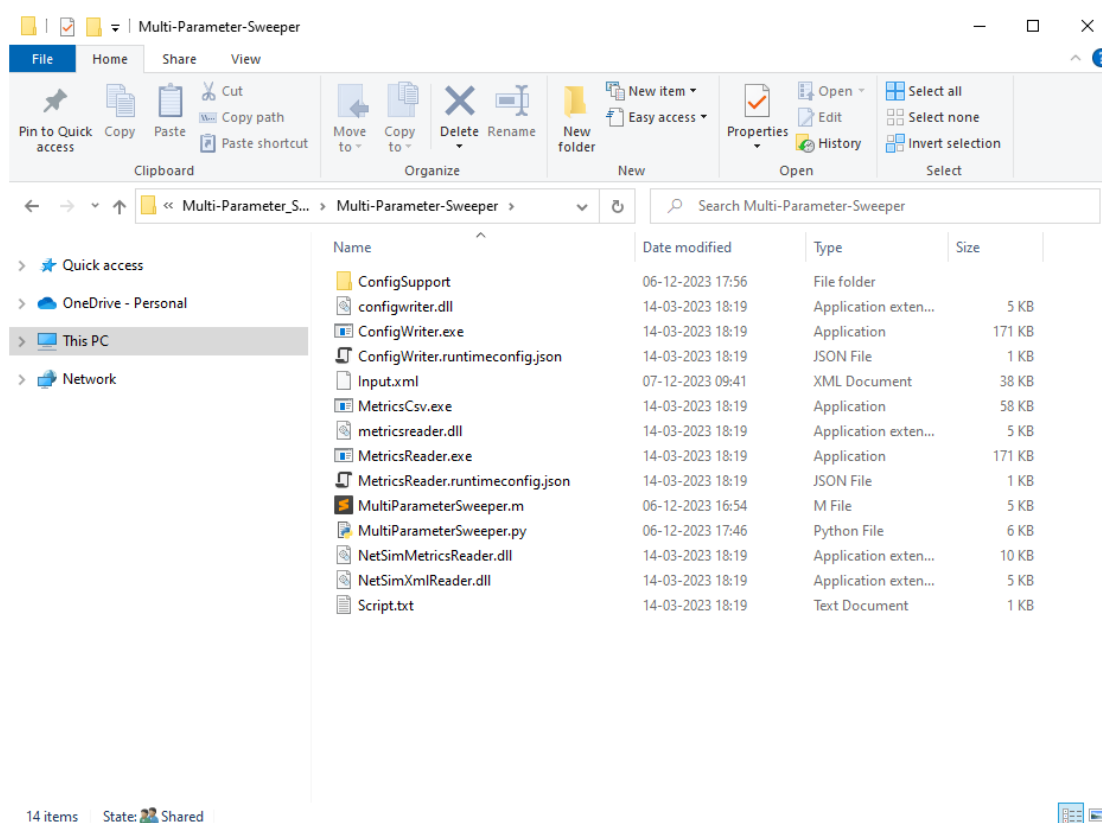


Figure 2-1: Project directory with sweeper binaries and script files

2.1 Input Configuration

Input.xml contains the base NetSim network configuration. Create this file by saving a network scenario in NetSim v15.0, copying Configuration.netsim, and renaming it to Input.xml. Copy the ConfigSupport folder along with Input.xml.

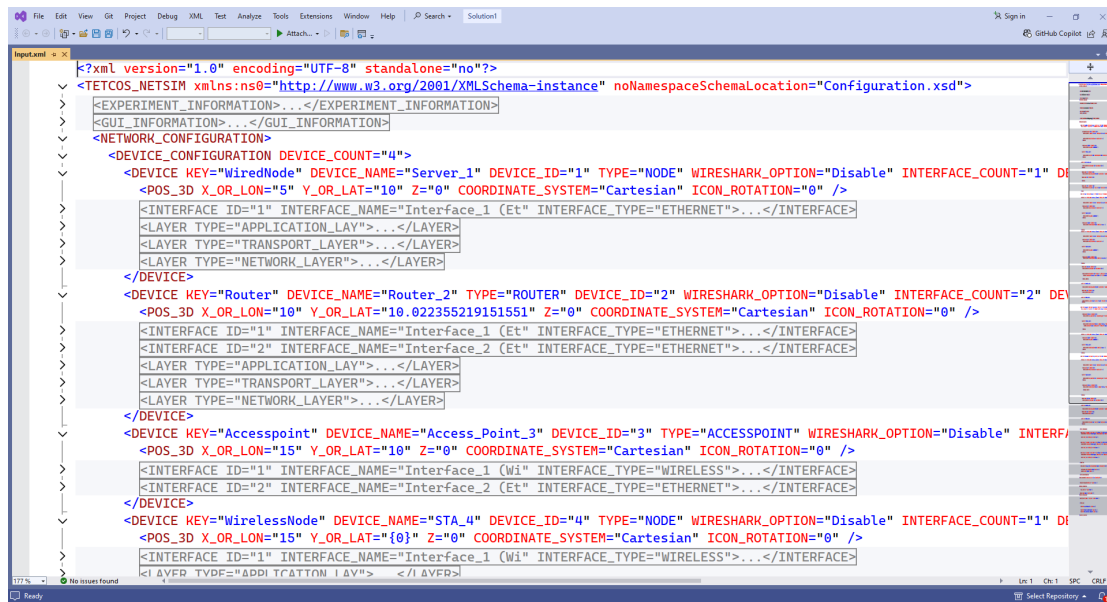


Figure 2-2: Input.xml file with NetSim configuration parameters to sweep

The values of parameters to vary during each simulation run must be specified as {0}, {1}, {2}, and so on. For example, the Y coordinate of a device can be replaced with {0}.

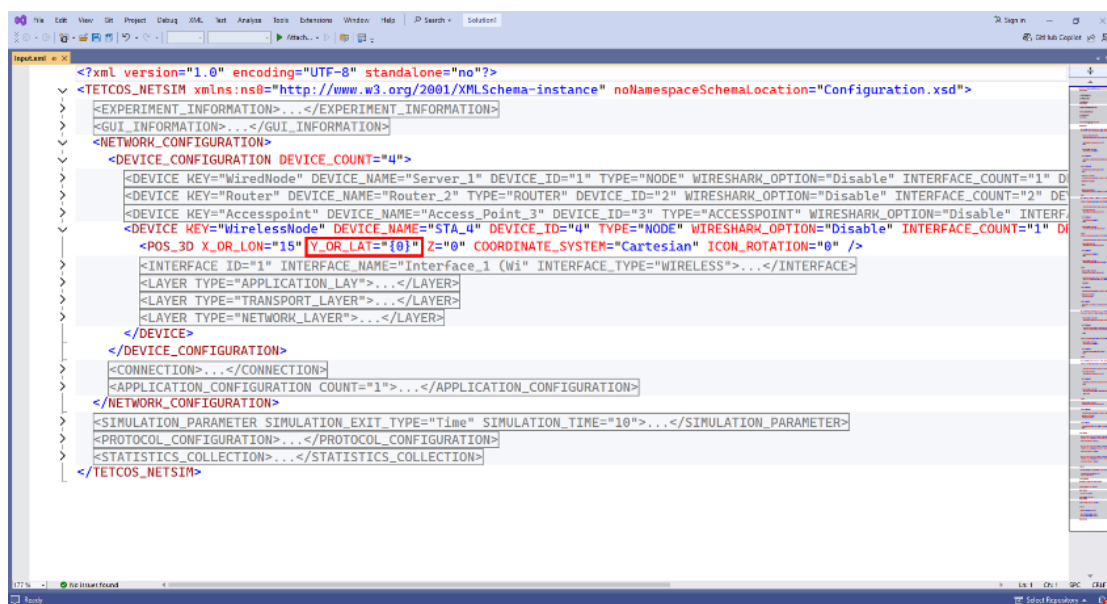


Figure 2-3: Y coordinate placeholder modified in Input.xml

2.2 Metric Script

Script.txt is updated with the output metric that must be logged at the end of each simulation run. NetSim writes metrics to Metrics.xml; the metrics reader uses the script file to locate the selected metric and append it to the result CSV.

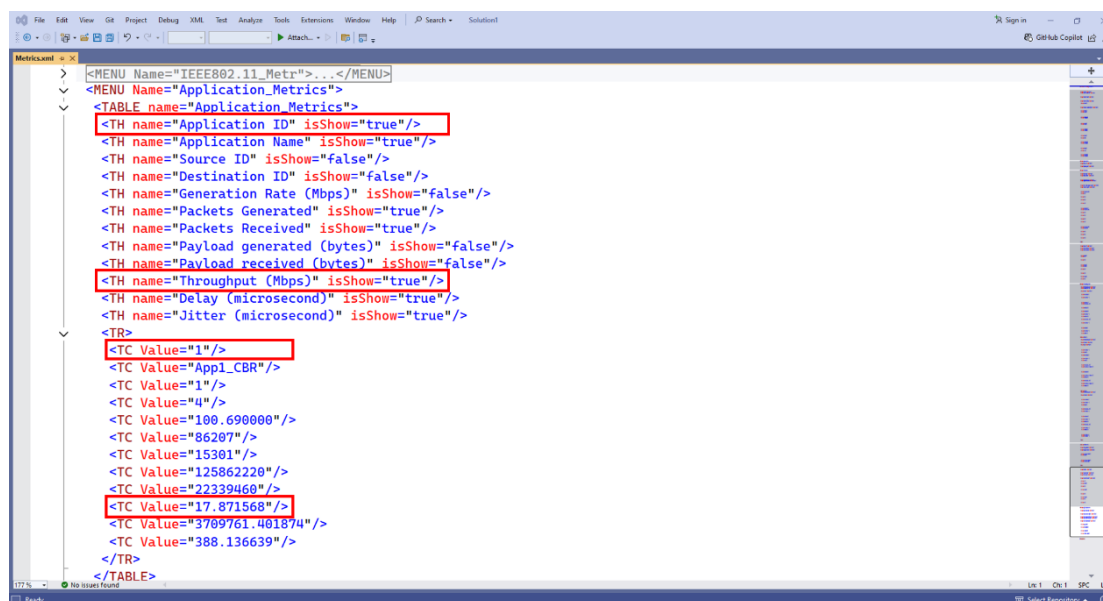


Figure 2-4: NetSim output Metrics.xml file

For example, to log application throughput for each simulation run, update Script.txt with the application metrics table and throughput field.



Figure 2-5: Application throughput metric configured in Script.txt

2.3 Helper Executables and Python Script

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.

MetricsCSV.exe: This executable is used to convert the Metrics.xml file present in the output folder into a comma separated file, MetricsPrint.csv.

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.

It uses the Script.txt file to determine which parameter to read from the Metrics file.

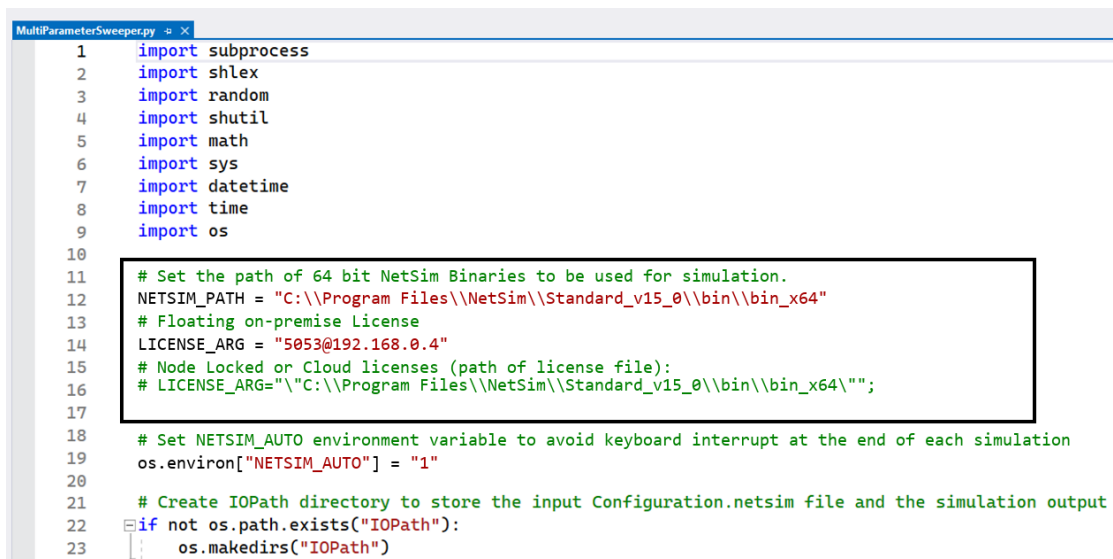
If multiple parameters are to be read and logged, then the MetricsReader.exe can be called multiple times with Script.txt file having information about the parameter to be read each

time.

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.

MultiParameterSweeper.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 can be configured to run multiple simulation iterations based on the number of parameters to be varied and the range of values of each parameter.
- NETSIM_PATH variable can be set to the path of NetSim 64-bit binaries (bin_x64) in the install directory or workspace which is to be used to run Simulations.
- LICENSE_ARG variable can be set to License server port and IP details in case of floating on premise licenses or the path of license file in case of node locked or cloud licenses.



```

MultiParameterSweeper.py - x
1 import subprocess
2 import shlex
3 import random
4 import shutil
5 import math
6 import sys
7 import datetime
8 import time
9 import os
10
11 # Set the path of 64 bit NetSim Binaries to be used for simulation.
12 NETSIM_PATH = "C:\\Program Files\\NetSim\\Standard_v15_0\\bin\\bin_x64"
13 # Floating on-premise License
14 LICENSE_ARG = "5053@192.168.0.4"
15 # Node Locked or Cloud licenses (path of license file):
16 # LICENSE_ARG="\"C:\\Program Files\\NetSim\\Standard_v15_0\\bin\\bin_x64\"";
17
18 # Set NETSIM_AUTO environment variable to avoid keyboard interrupt at the end of each simulation
19 os.environ["NETSIM_AUTO"] = "1"
20
21 # Create IOPath directory to store the input Configuration.netsim file and the simulation output
22 if not os.path.exists("IOPath"):
23     os.makedirs("IOPath")

```

Figure 2-6: NetSim path in the updated Python script

For NetSim v15.0, set:

```

NETSIM_PATH = "C:\\Program Files\\NetSim\\Standard_v15_0\\bin\\bin_x64"
LICENSE_ARG = "5053@192.168.0.4"
# Node Locked or Cloud licenses (path of license file):
# LICENSE_ARG="\"C:\\Program Files\\NetSim\\Standard_v15_0\\bin\\bin_x64\"";

```

3 Running the Sweeper

- The MetricsCSV.exe will convert the Metrics.xml file inside the output folder into a csv file.

```

MultiParameterSweeper.py
89
90     # print(cmd)
91     os.system(cmd)
92
93     # Create a copy of the output Metrics.xml file for writing the result log
94     if os.path.isfile("IOPath\Metrics.xml"):
95         shutil.copy("IOPath\Metrics.xml", "Metrics.xml")
96         os.system("MetricsCsv.exe IOPath")
97
98     # Number of Script files i.e Number of Output parameters to be read from Metrics.xml
99     # If only one output parameter is to be read only one Script text file with name Script.txt to
100    # If more than one output parameter is to be read, multiple Script text file with name Script1
101    # ...Scriptn.txt to be provided
102    OUTPUT_PARAM_COUNT = 1
103
104    if os.path.isfile("Metrics.xml"):
105        # Write the value of the variable parameters in the current iteration to the result log
106        csvfile = open("result.csv", "a")
107        csvfile.write("\n" + str(i) + ",")
108        csvfile.close()
109

```

Figure 3-1: *MetricsPrint.csv* file will be created in the *IOPath* and then copied into the respective output folder

- 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:

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.3693]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Mil\Downloads\Multi-Parameter_Sweeper_v15.0-main\Multi-Parameter-Sweeper>Python MultiParameterSweeper.py

```

Figure 3-2: *Running the Python script from Command Prompt*

- This starts the Multi-Parameter-Sweeping process which runs NetSim simulations iteratively for different values of Y parameter of Wireless Node.
- At the end of the process the Multi-Parameter-Sweeping folder will have the following file and folders created:

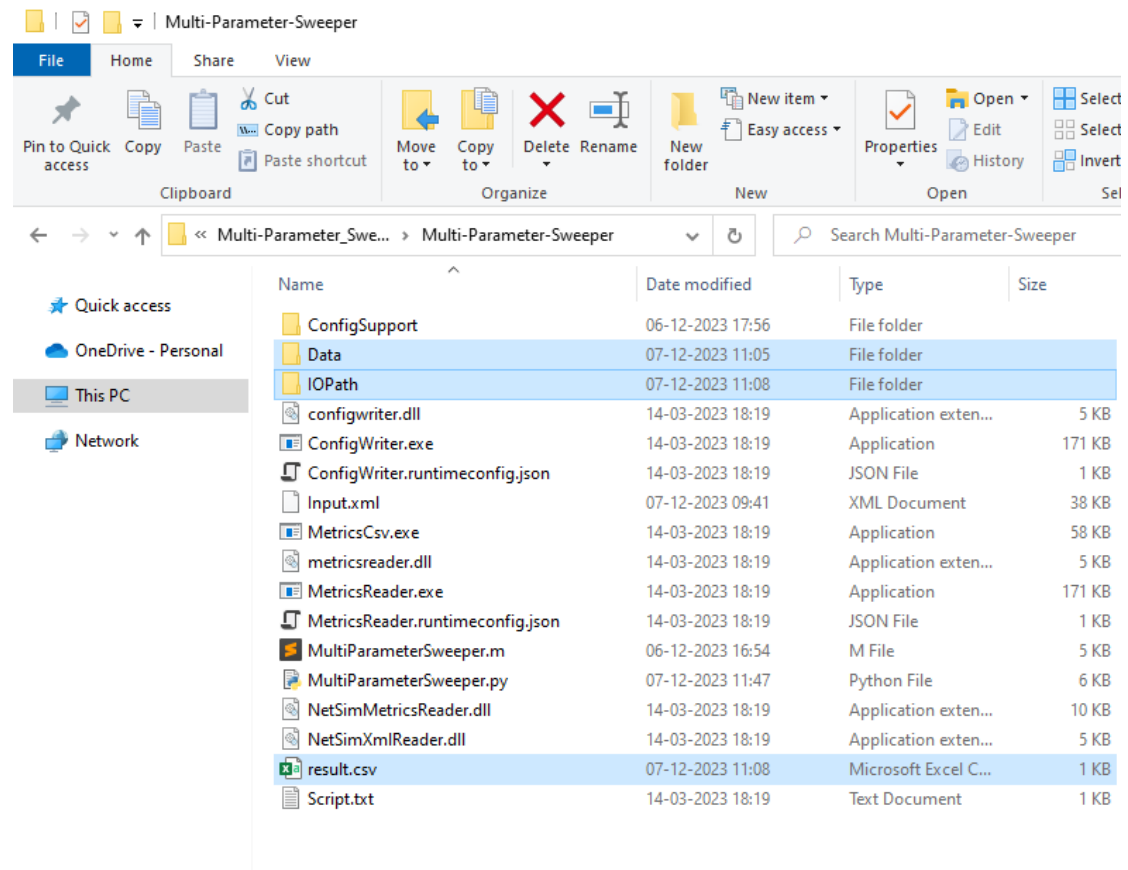


Figure 3-3: After Simulation Multi-Parameter-Sweeping folder contains output files like result.csv, Data etc

- **Data:** Contains multiple folders created based on date and time of simulation inside which multiple output folders corresponding to each simulation run, with its name including the value of the parameters in that iteration gets created.

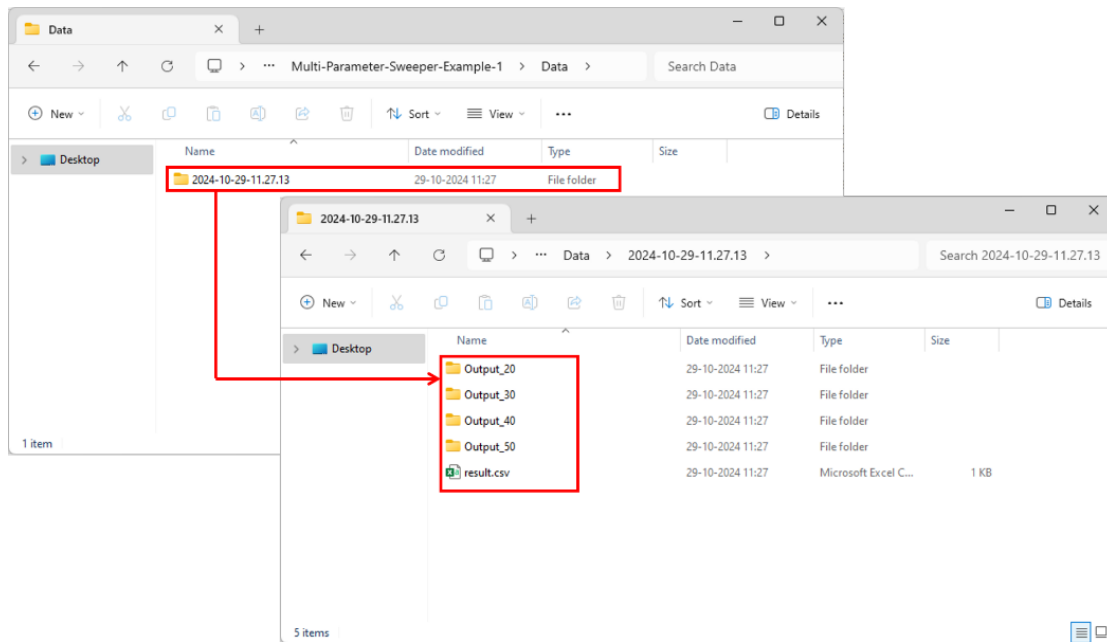


Figure 3-4: Based on values in the iteration, Output folder gets created in the Data directory inside a folder named as per date and time of simulation along with a copy of result.csv file.

- Each folder contains the all the output files associated with the simulation run.

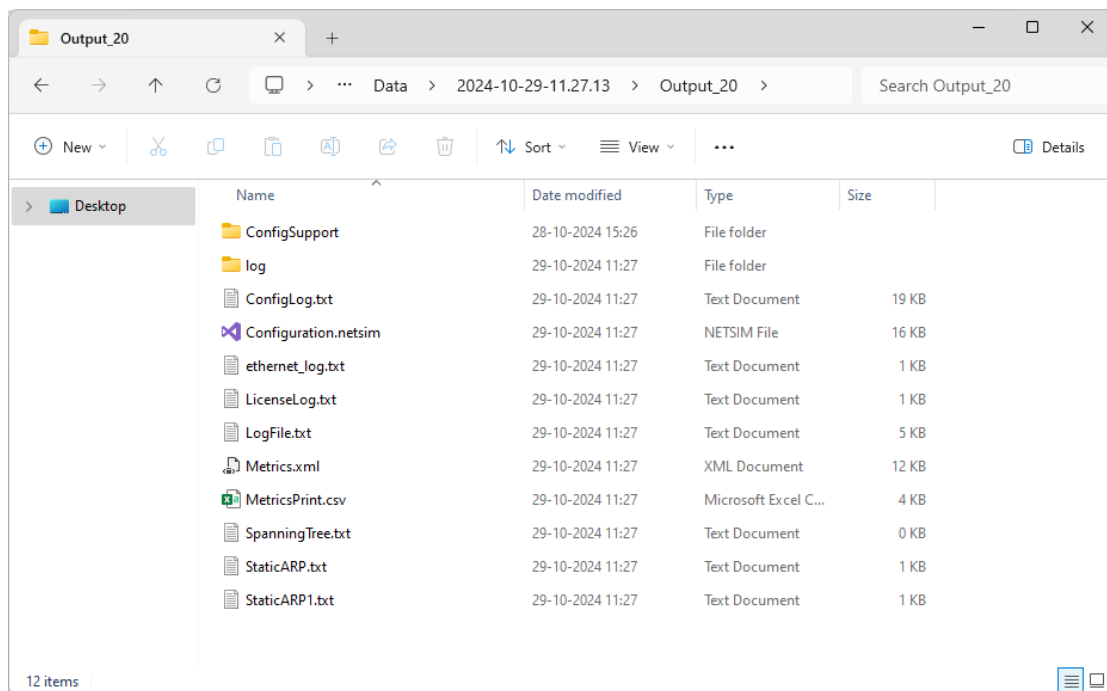


Figure 3-5: Each folder contains all the output files and the Metrics.xml file converted to MetricsPrint.csv file

Note: User should keep a back-up of the data folder to avoid data loss.

- **IO Path:** 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. The `result.csv` file will also be copied into the output folder after each simulation.

	A	B	C
1	Y	THROUGHPUT(Mbps)	
2	20	21.616176	
3	30	14.703952	
4	40	9.561248	
5	50	0	
6			
7			

Figure 3-6: Iterated value and Throughput obtained listed in result.csv

4 Example 1: Single Input Parameter

Consider an Internetwork scenario in NetSim v15.0 with a wired node, router, access point, and wireless node.

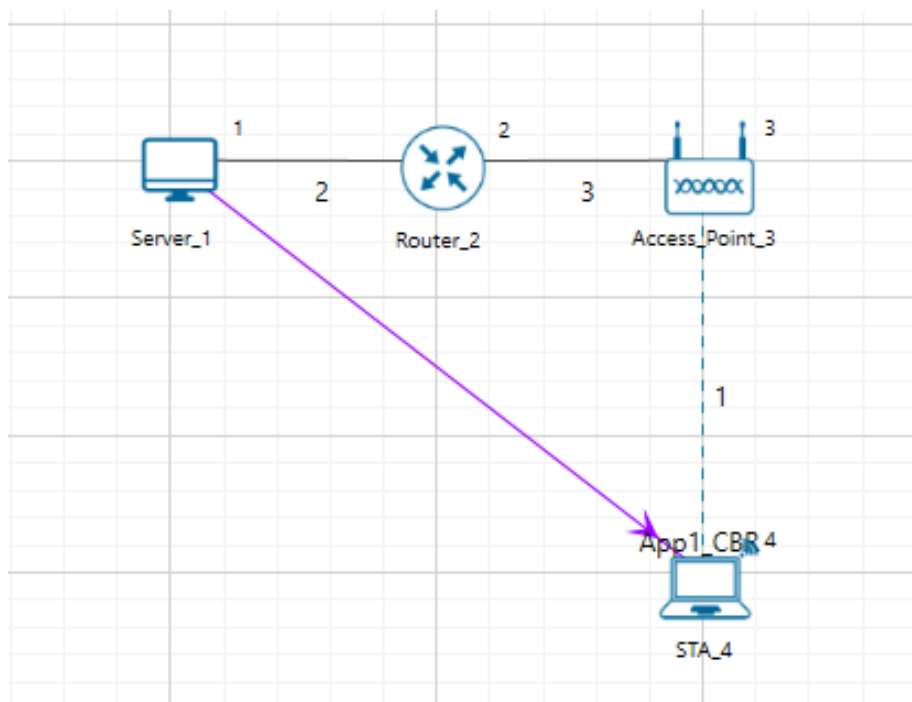


Figure 4-1: Network topology for the single-parameter sweep

The network configuration has the initial distance between the Access Point and Wireless Node

as 10 meters with the Access Point located at (15, 10) and Wireless Node located at (15, 20). Multi-Parameter Sweeper is configured to run simulations for different distance between the Access Point and Wireless Node by varying the Wireless Node Y coordinate value from 20 to 50 in steps of 10 meters.

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:

Examples\Multi-Parameter-Sweeper-Example-1

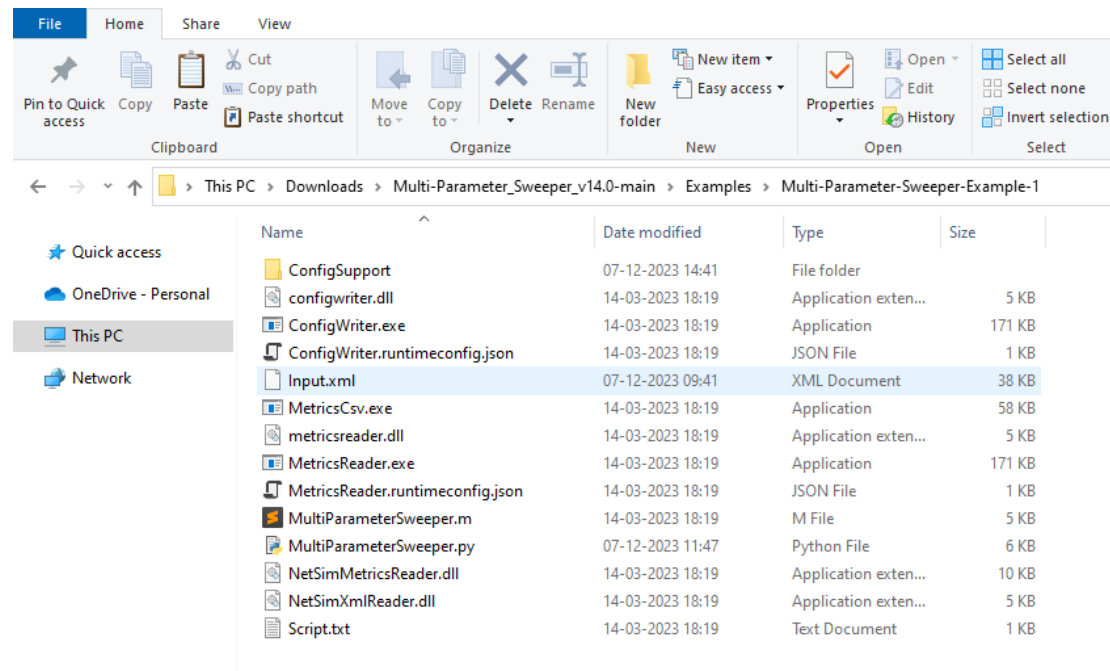


Figure 4-2: Renamed `Configuration.netsim` to `input.xml` and pasted in Multi-Parameter-Sweeper directory

- The value of the Y coordinate of Wireless Node that is to be modified during each simulation run is updated (`{0}`) in the configuration file as shown below:

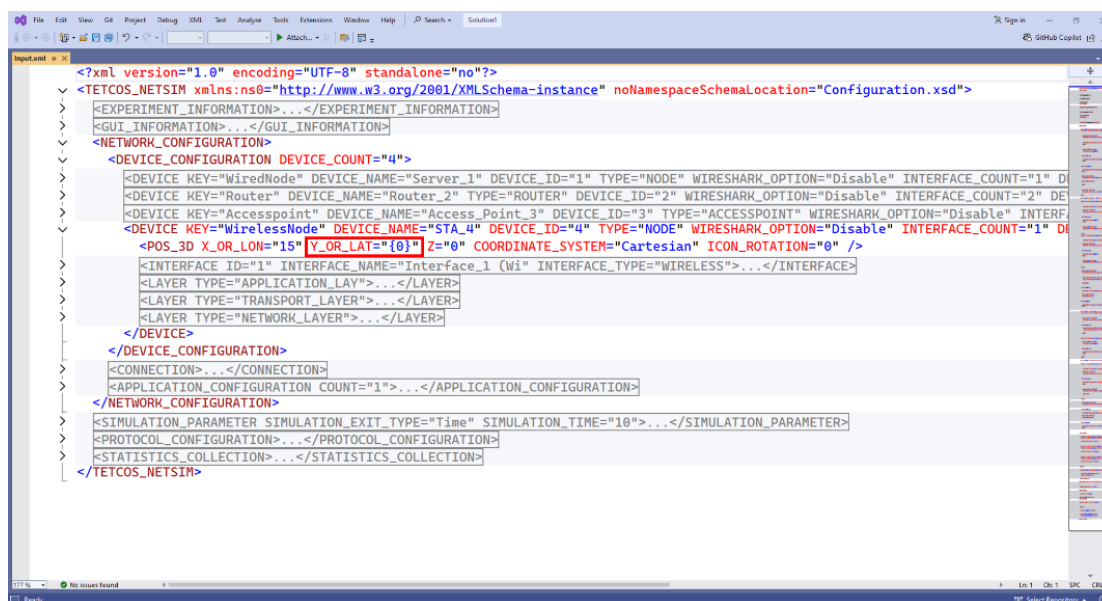


Figure 4-3: Modified Y coordinate of Wireless Node in input.xml

- 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.

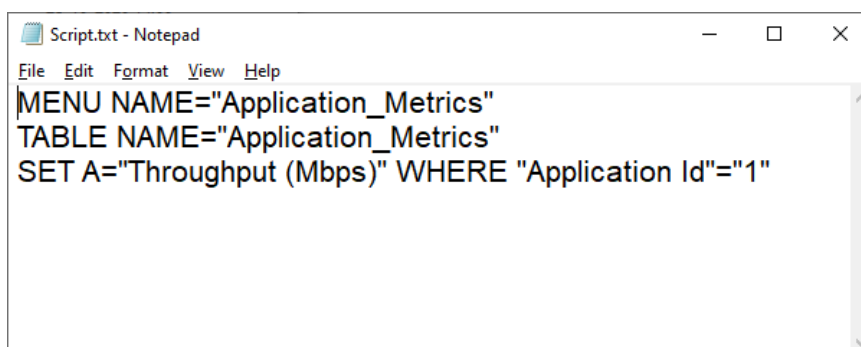


Figure 4-4: The application throughput is to be logged for each simulation modified in Script.txt

4.1 Python Script Changes

- `MultiParameterSweeper.py` is updated to pass the Y coordinate value during each iteration to generate Configuration file, run simulation, and update the result csv log.

The `MultiParameterSweeper.py` python script modified for running simulations for different values of Y coordinates starting from 20 up to 50 in steps of 10 is shown below:

- A `result.csv` file is created and added with headings Y and Throughput (Mbps).
- A `MetricsPrint.csv` is created inside every output folder.
- For loop is set to iteratively run simulations for values starting from 20 to 50 in steps of 10.

- 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.

```

10
11 # Set the path of 64 bit NetSim Binaries to be used for simulation.
12 NETSIM_PATH = "C:\\Program Files\\NetSim\\Standard_v15_0\\bin\\bin_x64"
13 # Floating on-premise License
14 LICENSE_ARG = "5053@192.168.0.4"
15 # Node Locked/ Cloud License
16 # LICENSE_ARG="\"C:\\Program Files\\NetSim\\Standard_v15_0\\bin\\bin_x64\"";
17
18 # Set NETSIM_AUTO environment variable to avoid keyboard interrupt at the
19 os.environ["NETSIM_AUTO"] = "1"
20
21 # Create IOPath directory to store the input Configuration.netsim file and
22 if not os.path.exists("IOPath"):
23     os.makedirs("IOPath")
24
25 # Create Data directory to store the Configuration.netsim and the Metrics.
26 if not os.path.exists("Data"):
27     os.makedirs("Data")
28
29 # Clear the IOPath folder if it has any files created during previous mult
30 for root, dirs, files in os.walk("IOPath"):
31     for file in files:
32         os.remove(os.path.join(root, file))
33
34 # Delete result.csv file if it already exists
35 if os.path.isfile("result.csv"):
36     os.remove("result.csv")
37
38 # create a csv file to log the output metrics for analysis
39 csvfile = open("result.csv", "w")
40
41 # Add headings to the CSV file
42 csvfile.write("Y,THROUGHPUT(Mbps),")
43 csvfile.close()

```

Figure 4-5: To Create result.csv file, added with headings Y and Throughput (Mbps) and NetSim installation Path and License information

- NETSIM_PATH variable is set to the path of NetSim 64-bit binaries in the install directory or workspace in the system.
- LICENSE_ARG variable is set to the license server details.
- A result.csv file is created and added with headings Y and Throughput (Mbps).

```

42 csvfile.write("Y,THROUGHPUT(Mbps),")
43 csvfile.close()
44
45 # Create a folder with name as year-month-day-hour.minute.seconds inside the data folder
46 today = time.strftime("%Y-%m-%d-%H-%M-%S")
47 foldername = str(today)
48
49 # Iterate based on the number of times the simulation needs to be run and the input parameter range
50 for i in range(20, 51, 10):
51
52     if os.path.isfile("Configuration.netsim"):
53         os.remove("Configuration.netsim")
54
55     if os.path.isfile("IOPath\\Configuration.netsim"):
56         os.remove("IOPath\\Configuration.netsim")
57
58     if os.path.isfile("IOPath\\Metrics.xml"):
59         os.remove("IOPath\\Metrics.xml")
60
61     # Call ConfigWriter.exe with arguments as per the number of variable parameters in the input.xml file
62     cmd = "ConfigWriter.exe " + str(i)
63     print(cmd)
64     os.system(cmd)
65
66     # Copy the Configuration.netsim file generated by ConfigWriter.exe to IOPath directory
67     if os.path.isfile("Configuration.netsim"):
68         shutil.copy("Configuration.netsim", "IOPath\\Configuration.netsim")
69
70     # Copying the Additional Files folder
71     if os.path.exists('ConfigSupport'):
72         # Remove the existing 'ConfigSupport' directory in 'IOPath' if it exists
    
```

Figure 4-6: Varying Distance and set license server information

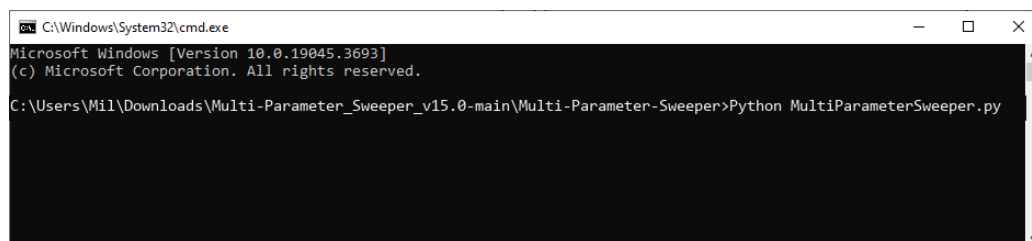
- For loop is set to iteratively run simulations for values starting from 20 to 51 in steps of 10.
- The value of the parameter Y in the current iteration is passed as input to ConfigWriter executable to generate Configuration.netsim file for each simulation.

```

100 # If more than one output parameter is to be read, multiple Script text
101 # ...,Scriptn.txt to be provided
102 OUTPUT_PARAM_COUNT = 1
103
104 if os.path.isfile("Metrics.xml"):
105     # Write the value of the variable parameters in the current iteration
106     csvfile = open("result.csv", "a")
107     csvfile.write("\n" + str(i) + ",")
108     csvfile.close()
109
110 if OUTPUT_PARAM_COUNT == 1:
111     # Call the MetricsReader.exe passing the name of the output log
112     os.system("MetricsReader.exe result.csv")
113 else:
114     for n in range(1, OUTPUT_PARAM_COUNT + 1, 1):
115         os.rename("Script" + str(n) + ".txt", "Script.txt")
116         os.system("MetricsReader.exe result.csv")
117         csvfile = open("result.csv", "a")
118         csvfile.write(",")
119         csvfile.close()
120         os.rename("Script.txt", "Script" + str(n) + ".txt")
121
122 else:
123     # Update the output Metric as crash if Metrics.xml file is missing
124     csvfile = open("result.csv", "a")
125     csvfile.write("\n" + str(i) + ", " + "crash" + ",")
126     csvfile.close()
127
128 # Name of the Output folder to which the results will be saved
129 OUTPUT_PATH = "Data\\" + str(foldername) + "\\Output_" + str(i)
130
    
```

Figure 4-7: Modify parameters in MultiParameterSweeper.py

- The value of the parameter Y in the current iteration is written to the result log file for analysis.
- Configuration file and Metrics file are copied and renamed appending the value of the parameter in the current iteration.
- 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:



```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.3693]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Mil\Downloads\Multi-Parameter_Sweeper_v15.0-main\Multi-Parameter-Sweeper>Python MultiParameterSweeper.py

```

Figure 4-8: Running Example 1 from Command Prompt

This starts the Multi-Parameter-Sweeping process which runs NetSim simulations iteratively for different values of Y parameter of Wireless Node.

At the end of the process the Multi-Parameter-Sweeping folder will have the following file and folders created:

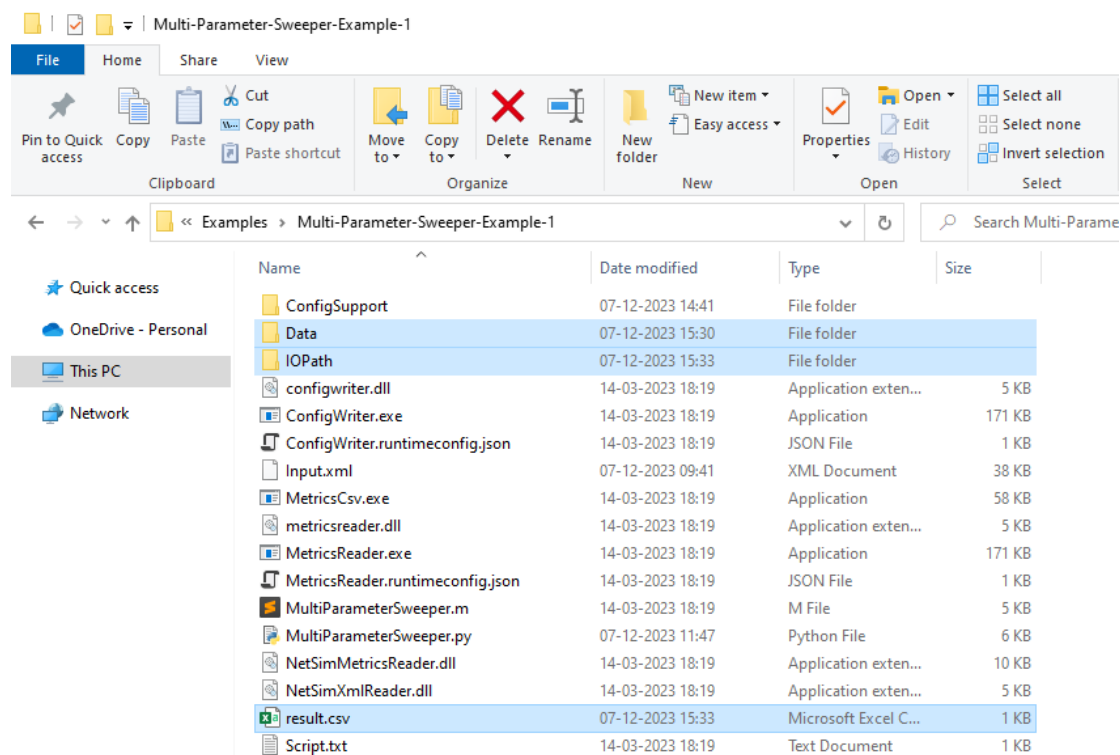


Figure 4-9: After Simulation Multi-Parameter-Sweeping folder contains output files like result.csv, Data etc

- **Data:** Contains multiple folders corresponding to each simulation run, with its name including the value of the parameters in that iteration. The output folders will be created inside folder with name in the format Year-Month-Day-Hours-Minutes-Seconds.

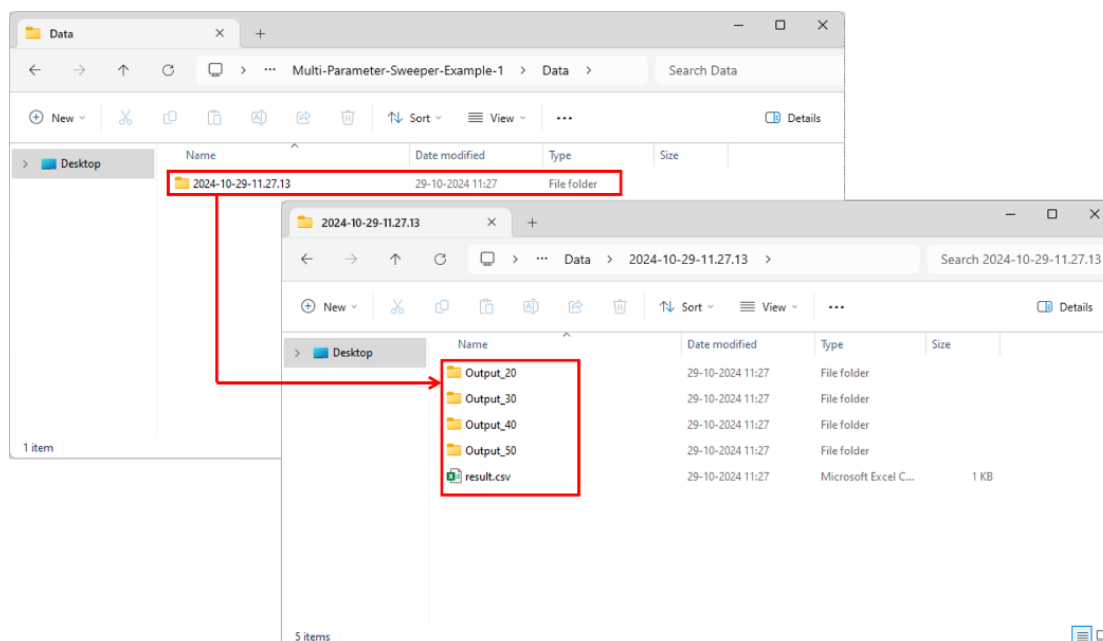


Figure 4-10: Based on distance Configuration.netsim files created in output folder

- Each folder contains the all the output files associated with the simulation run.

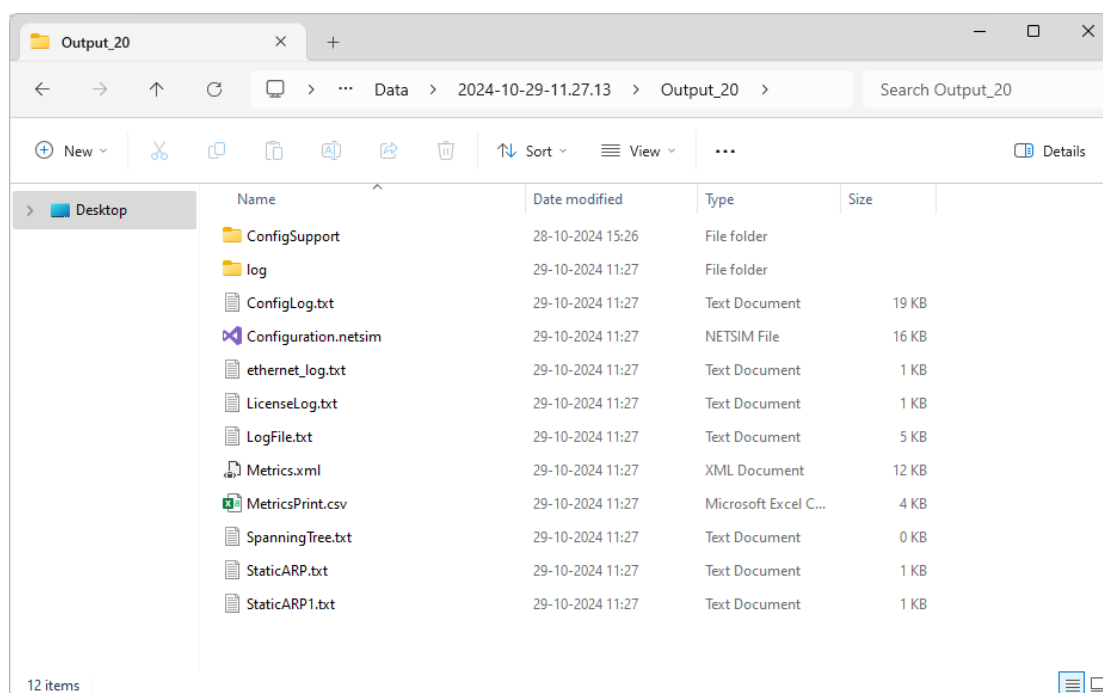


Figure 4-11: Each folder contains the all the output files

- **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.

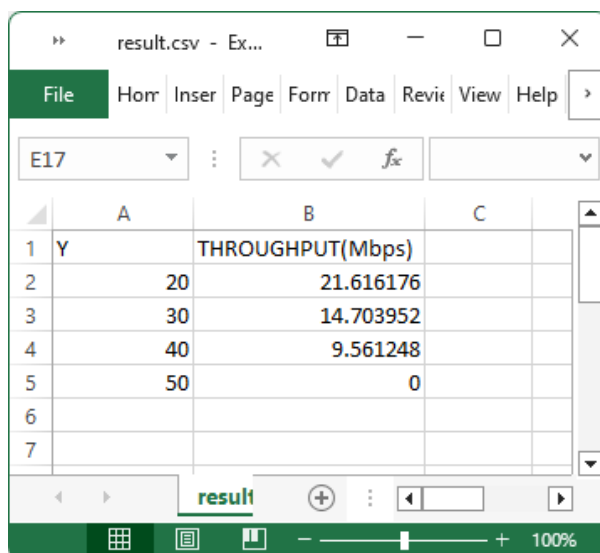


Figure 4-12: Distance Vs. Throughput obtained in result.csv

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, multiple script files can be used. If `n` output parameters are to be logged, then there can be `script1.txt`, `script2.txt`, `script.txt` in the sweeper folder. For example, there can be two Script files as shown below:

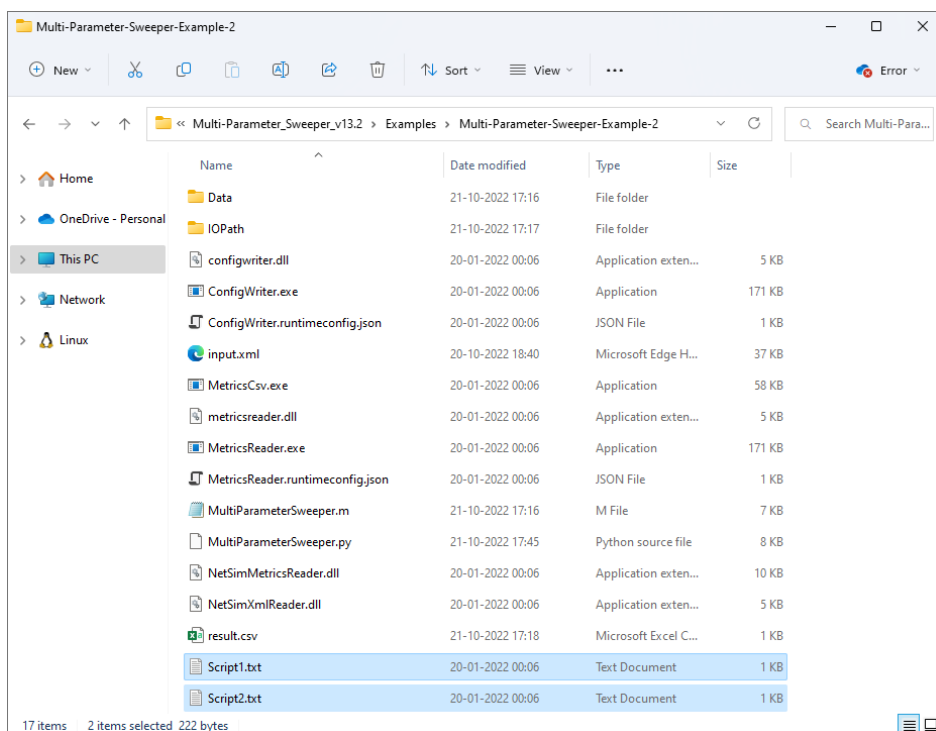


Figure 5-1: Multiple output parameter script files

5 Example 2: Multiple Input Parameters

Example 2 modifies multiple input parameters and logs multiple output parameter.

Consider the following Internetwork scenario in NetSim, comprising of a Wired Node, Router, Access Point and a Wireless Node.

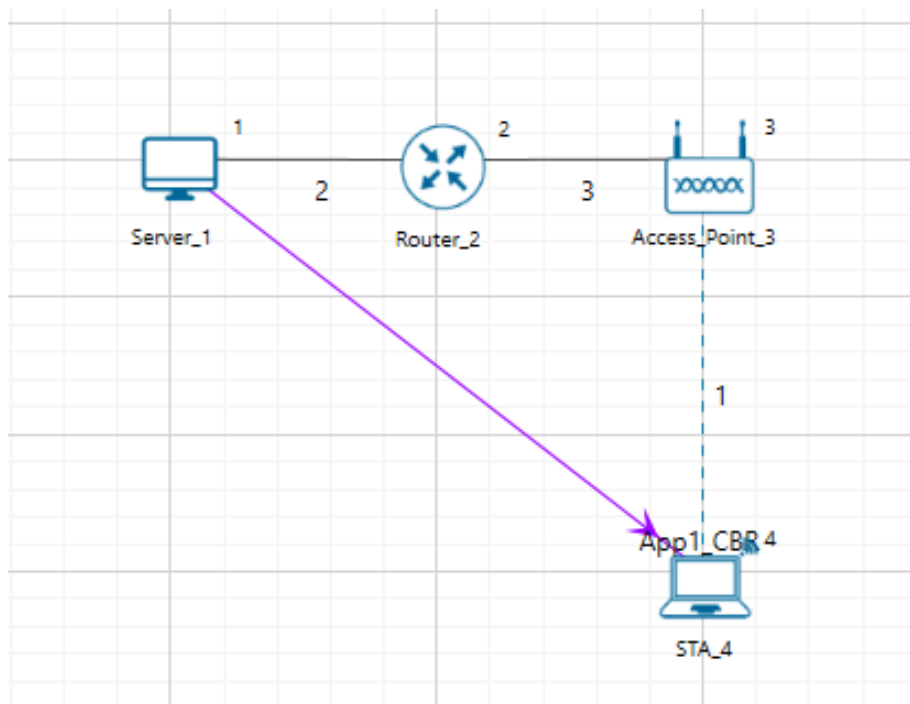


Figure 5-2: Network topology for the multiple-parameter sweep

Configure the scenario as follows:

- Set grid length to 60 m x 60 m grid setting property panel on the right. This needs to be done before the any device is placed on the grid.
- Distance between Access Point and the Wireless Node is set to 10 m.
- Set DCF as the medium access layer protocol under datalink layer properties of access point and wireless node. To configure any properties in the nodes, click on the node, expand the property panel on the right side, and change the properties as mentioned in the below steps.
- WLAN Standard is set to 802.11ac and No. of Tx and Rx Antenna is set to 1 in access point and No. of Tx is 1 and Rx Antenna is set to 1 in wireless node (Right-Click Access Point or Wireless Node > Properties > Interface Wireless > Transmitting Antennas and Receiving Antennas) and Bandwidth is set to 20 MHz in both Access-point and wireless-node Transmitter Power set to 100 mW in both Access-point and wireless-node.
- Wired Link speed is set to 1 Gbps and propagation delay to 10 μ s in wired links.
- Channel Characteristics: Path Loss Only, Path Loss Model: Log Distance, Path Loss Exponent: 3.5.
- Configure an application between any two nodes by selecting a CBR application from the Set Traffic tab in the ribbon on the top. Click on created application and expand the

application property panel on the right and set transport protocol to UDP, packet size to 1460 B and Inter arrival time to 116.8 μ s.

- Application Generation Rate: 100 Mbps (Packet Size: 1460, Inter Arrival Time: 116.8 μ s).
- Run the simulation for 10 s.

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 2 directory which is part of the project folder:

Examples\Multi-Parameter-Sweeper-Example-2

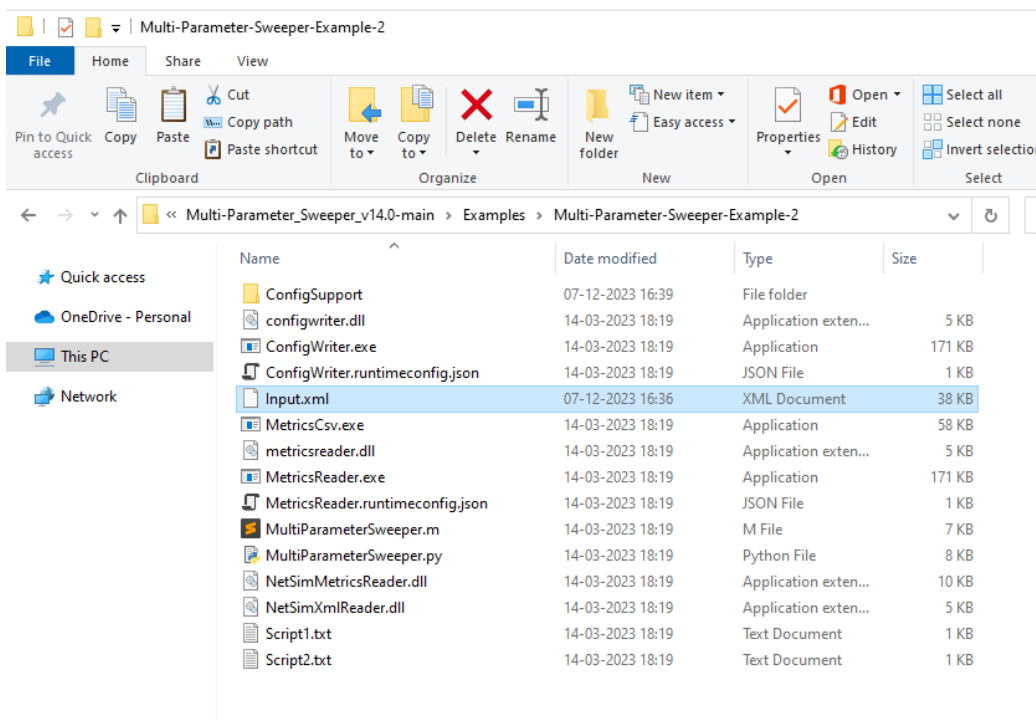


Figure 5-3: Renamed `Configuration.netsim` to `input.xml` and pasted in `Multi-Parameter-Sweeper` directory

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

Table 5-1: Variables modified in `Input.xml`

Input variable	Placeholder
Bandwidth (MHz)	{0}
Tx antenna count	{1}
Rx antenna count	{2}
Inter-arrival time (μ s)	{3}

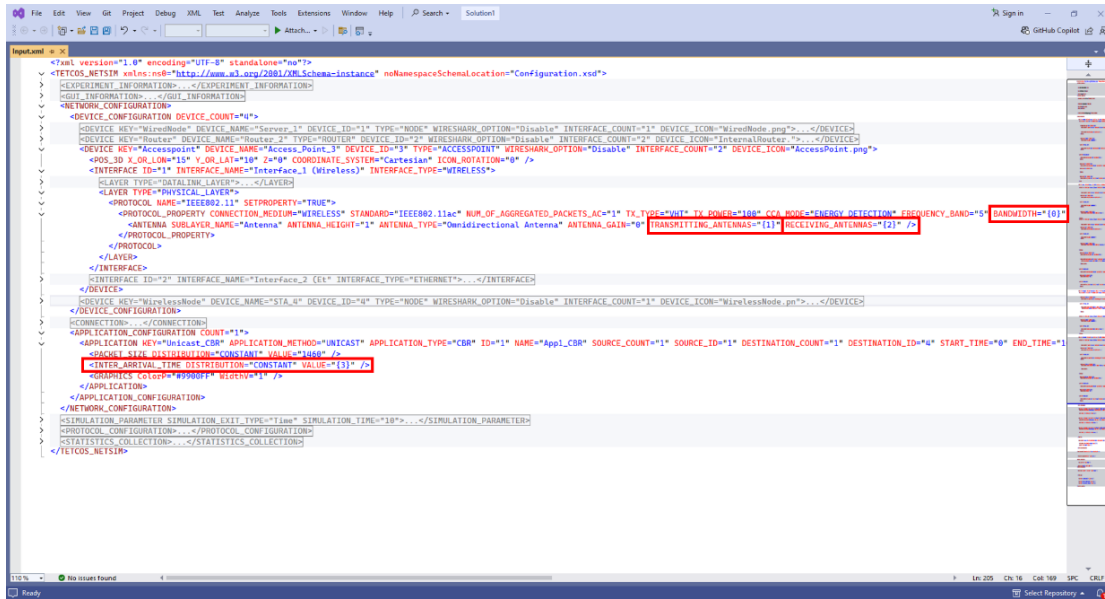


Figure 5-4: The above table Variables are modified in input.xml file

The python script MultiParameterSweeper.py is modified to run simulation for all possible combinations of Bandwidth and Tx Antenna Count and Rx Antenna Count with the respective values of IAT that is calculated.

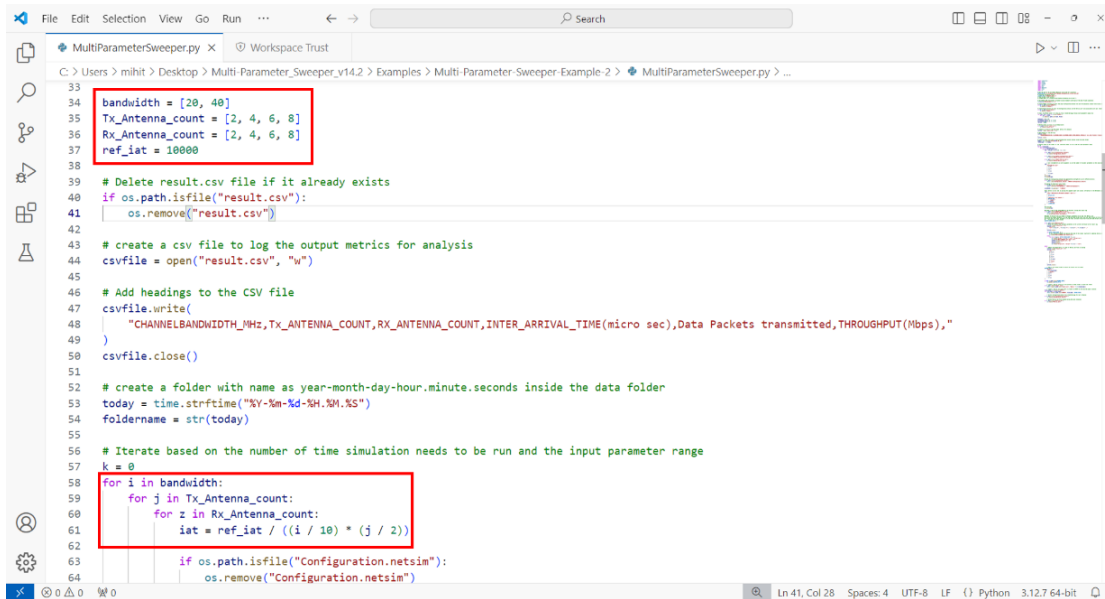


Figure 5-5: Modified MultiParameterSweeper.py based on input parameter

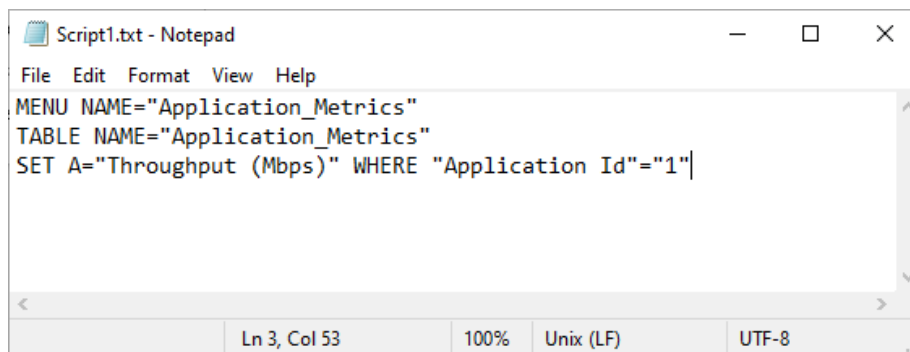
Multiple parameters are read from the Metrics.xml file and logged in the results.csv file along with the input parameters such as BANDWIDTH_MHz, TX_ANTENNA_COUNT, RX_ANTENNA_COUNT, INTER_ARRIVAL_TIME (micro sec).

Table 5-2: *Output parameters configured in script files*

Output parameter	Script file
Throughput (Mbps)	Script1.txt
Data packets transmitted	Script2.txt

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. The variable `OUTPUT_PARAM_COUNT` is set to 2 as per the number of Script files.

Script1.txt



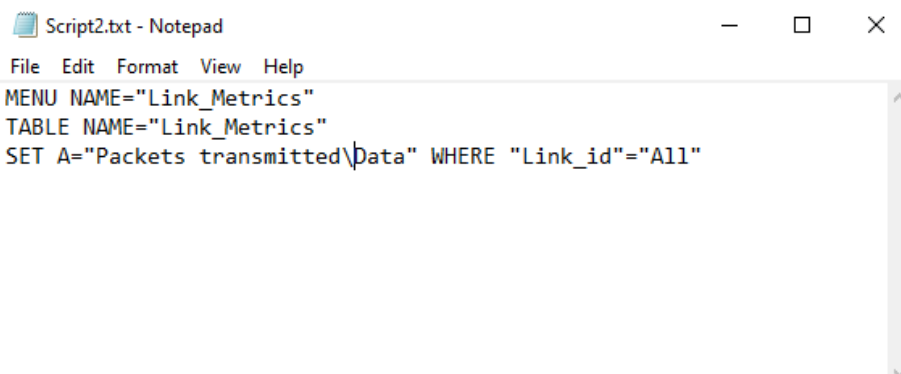
```

Script1.txt - Notepad
File Edit Format View Help
MENU NAME="Application_Metrics"
TABLE NAME="Application_Metrics"
SET A="Throughput (Mbps)" WHERE "Application Id"="1"
Ln 3, Col 53 100% Unix (LF) UTF-8

```

Figure 5-6: *The application throughput is to be logged for each simulation modified in Script1.txt*

Script2.txt



```

Script2.txt - Notepad
File Edit Format View Help
MENU NAME="Link_Metrics"
TABLE NAME="Link_Metrics"
SET A="Packets transmitted\Data" WHERE "Link_id"="All"

```

Figure 5-7: *The Data Packets transmitted is to be logged for each simulation modified in Script2.txt*

In the python script `MultiParameterSweeper.py`, `MetricsReader.exe` is called to log each parameter specified in the script text files separating the entries with a comma (","). If simulation crashes, without generating the output `Metrics.xml`, then "crash" message is written to the log for each output parameter. The input parameters that were varied during each simulation run are also logged in the `results.csv` file.

```

127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165

```

```

if os.path.isfile("Metrics.xml"):
    # Write the value of the variable parameters in the current iteration to the result log
    csvfile = open("result.csv", "a")
    csvfile.write(
        "\n" + str(i) + "," + str(j) + "," + str(z) + "," + str(iat) + ","
    )
    csvfile.close()

if OUTPUT_PARAM_COUNT == 1:
    # Call the MetricsReader.exe passing the name of the output log file for updating the log based on script.txt
    os.system("MetricsReader.exe result.csv")
else:
    for n in range(1, OUTPUT_PARAM_COUNT + 1, 1):
        os.rename("Script" + str(n) + ".txt", "Script.txt")
        os.system("MetricsReader.exe result.csv")
        csvfile = open("result.csv", "a")
        csvfile.write(",")
        csvfile.close()
        os.rename("Script.txt", "Script" + str(n) + ".txt")

else:
    # Update the output Metric as crash if Metrics.xml file is missing
    csvfile = open("result.csv", "a")
    csvfile.write(
        "\n"
        + str(i)
        + ","
        + str(j)
        + ","
        + str(z)
        + ","
        + str(iat)
        + ","
        + "crash"
        + ","
    )
    csvfile.close()

```

Figure 5-8: Modify python script *MultiParameterSweeper.py* file

The simulation Configuration file and all the output files associated with each simulation run is saved to folders with name including the bandwidth and DL MIMO count values that were used during each simulation run.

```

166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197

```

```

# Name of the Output folder to which the results will be saved
OUTPUT_PATH = (
    "Data\\"
    + str(foldername)
    + "\\Output_"
    + str(i)
    + "_"
    + str(j)
    + "_"
    + str(z)
)

if not os.path.exists(OUTPUT_PATH):
    os.makedirs(OUTPUT_PATH)

# create a copy of result.csv file present in sweep folder to date-time folder
if os.path.isfile("result.csv"):
    shutil.copy(os.path.join("result.csv"), "Data\\" + str(foldername))

# Create a copy of all files that is present in IOPATH to the desired output location
files_names = os.listdir("IOPATH")
for file_name in files_names:
    shutil.move(os.path.join("IOPATH", file_name), OUTPUT_PATH)

# Delete Configuration.netsim file created during the last iteration
if os.path.isfile("Configuration.netsim"):
    os.remove("Configuration.netsim")

# Delete Metrics.xml file created during the last iteration
if os.path.isfile("Metrics.xml"):
    os.remove("Metrics.xml")

```

Figure 5-9: Modify python script *MultiParameterSweeper.py* file

- 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:

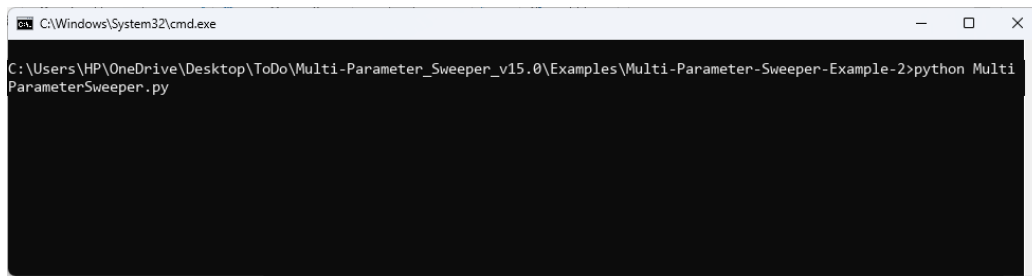


Figure 5-10: *Running Example 2 from Command Prompt*

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:

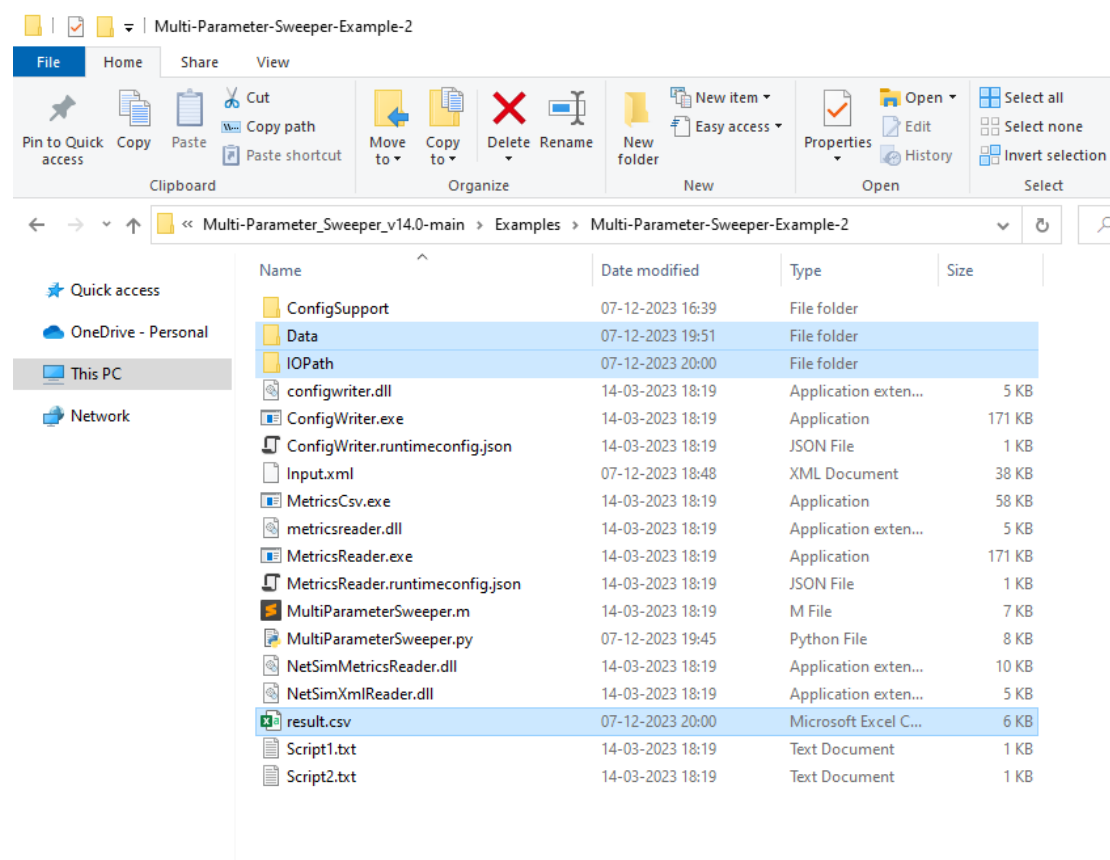


Figure 5-11: *After Simulation Multi-Parameter-Sweeping folder contains output files like result.csv, Data etc*

- **Data:** The Data directory contains multiple output folders with the output files associated with each simulation run.

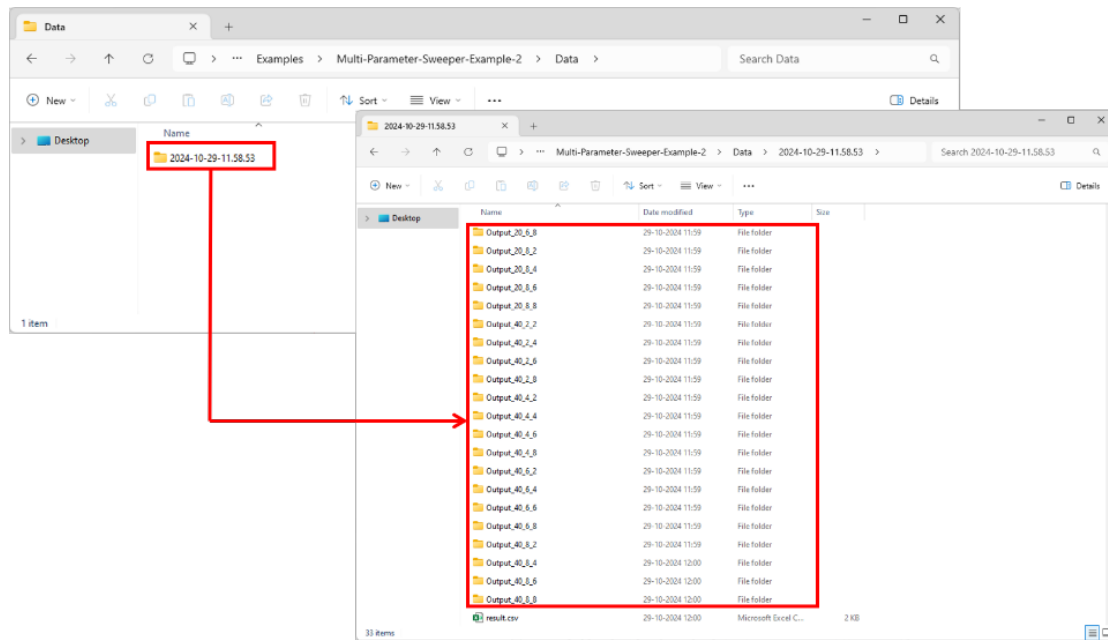


Figure 5-12: Based on input parameter Configuration.netsim and other files created in output folder

- **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.

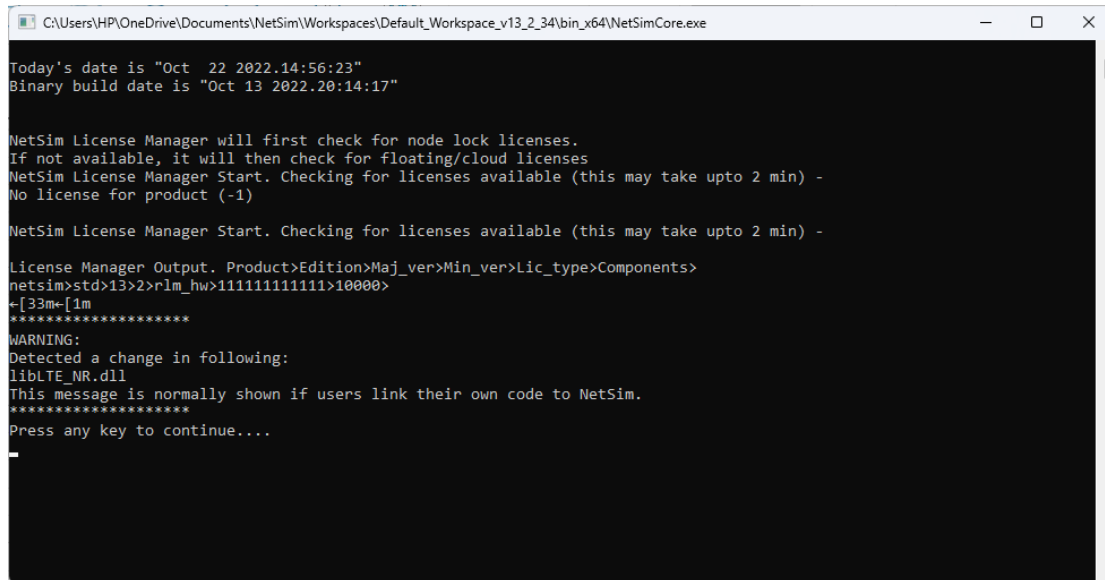
	A	B	C	D	E	F	G	H
	CHANNELBANDWIDTH_MHz	Tx_ANTENNA_COUNT	RX_ANTENNA_COUNT	INTER_ARRIVAL_TIME(micro sec)	Data Packets transmitted	THROUGHPUT(Mbps)		
2	20	2	2	5000	5990	2.327824		
3	20	2	4	5000	5990	2.327824		
4	20	2	6	5000	5990	2.327824		
5	20	2	8	5000	5990	2.327824		
6	20	4	2	2500	11982	4.659152		
7	20	4	4	2500	11982	4.659152		
8	20	4	6	2500	11982	4.659152		
9	20	4	8	2500	11982	4.659152		
10	20	6	2	1666.666667	17973	6.988144		
11	20	6	4	1666.666667	17973	6.988144		
12	20	6	6	1666.666667	17973	6.988144		
13	20	6	8	1666.666667	17973	6.988144		
14	20	8	2	1250	23959	9.3148		
15	20	8	4	1250	23959	9.3148		
16	20	8	6	1250	23959	9.3148		
17	20	8	8	1250	23959	9.3148		
18	40	2	2	2500	11982	4.659152		
19	40	2	4	2500	11982	4.659152		
20	40	2	6	2500	11982	4.659152		
21	40	2	8	2500	11982	4.659152		
22	40	4	2	1250	23959	9.3148		
23	40	4	4	1250	23959	9.3148		
24	40	4	6	1250	23959	9.3148		
25	40	4	8	1250	23959	9.3148		
26	40	6	2	833.333333	35943	13.973952		
27	40	6	4	833.333333	35943	13.973952		
28	40	6	6	833.333333	35943	13.973952		
29	40	6	8	833.333333	35943	13.973952		
30	40	8	2	625	47345	17.950992		
31	40	8	4	625	47929	18.63544		
32	40	8	6	625	47929	18.63544		
33	40	8	8	625	47929	18.63544		

Figure 5-13: Based on script result stored in result.csv file

6 Advanced Use Cases

6.1 Running the Sweeper with Modified Workspace Binaries

When source code in a workspace is modified, NetSim may print a checksum warning in the simulation console and wait for user input.

A screenshot of a Windows command prompt window titled "C:\Users\HP\OneDrive\Documents\NetSim\Workspaces\Default_Workspace_v13_2_34\bin_x64\NetSimCore.exe". The window shows the following text:

```
Today's date is "Oct 22 2022.14:56:23"  
Binary build date is "Oct 13 2022.20:14:17"  
  
NetSim License Manager will first check for node lock licenses.  
If not available, it will then check for floating/cloud licenses  
NetSim License Manager Start. Checking for licenses available (this may take upto 2 min) -  
No license for product (-1)  
  
NetSim License Manager Start. Checking for licenses available (this may take upto 2 min) -  
  
License Manager Output. Product>Edition>Maj_ver>Min_ver>Lic_type>Components>  
netsim>std>13>2>rlm_hw>111111111111>10000>  
+-[33m+ [1m  
*****  
WARNING:  
Detected a change in following:  
libLTE_NR.dll  
This message is normally shown if users link their own code to NetSim.  
*****  
Press any key to continue....  
-
```

Figure 6-1: *Checksum warning shown while running with modified workspace binaries*

To suppress the prompt, open the current workspace location from **Your Work > Workspaces > Open Workspace Location**. Go to **bin_x64** and rename the checksum file.

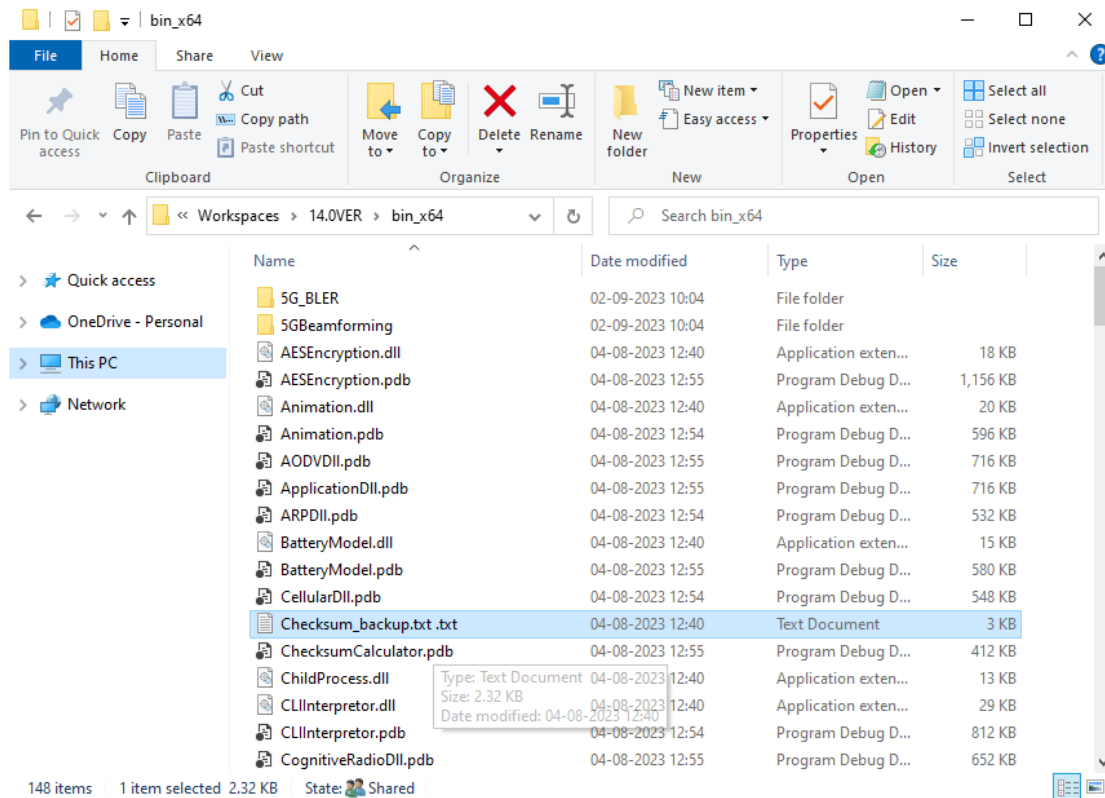


Figure 6-2: *Checksum.txt renamed in the workspace bin_x64 folder*

6.2 Configurations with Supporting Files

Some NetSim configurations use additional files such as mobility files, static route files, ACL input files, or SUMO configuration files for VANET studies. Place these files in the sweeper folder and copy them into IOPath with Configuration.netsim before each run.

```
if os.path.exists("ConfigSupport"):
    if os.path.exists("IOPath\\ConfigSupport"):
        shutil.rmtree("IOPath\\ConfigSupport")
    shutil.copytree("ConfigSupport", "IOPath\\ConfigSupport")
```

Figure 1: *Copying supporting configuration files into IOPath*