

NetSim Multi-Parameter Sweeper Python Script

Software: NetSim v14.2 (64 bit), DOT NET CORE SDK 3.1, Python 3.11.1 and later

Project Download Link:

https://github.com/NetSim-TETCOS/Multiparameter-Sweeper_Python_v14.2/archive/refs/heads/main.zip

Introduction:

NetSim multi-parameter sweeper automates the network simulation process by running NetSim through the command line interface (CLI). Instead of manually adjusting parameters and analyzing results in the NetSim GUI.

This program simplifies the task of sweeping through multiple input parameters. Users define the parameters to be varied, and the sweeper automates the process of adjusting values, running simulations, saving results, and organizing data. The goal is to save time and effort in the exploration of different parameter sets. The results are conveniently stored and can be easily compared using spreadsheet software like Microsoft Excel.

File Organization:

The project directory consists of several binaries which are responsible for different tasks during a multi-parameter sweep:

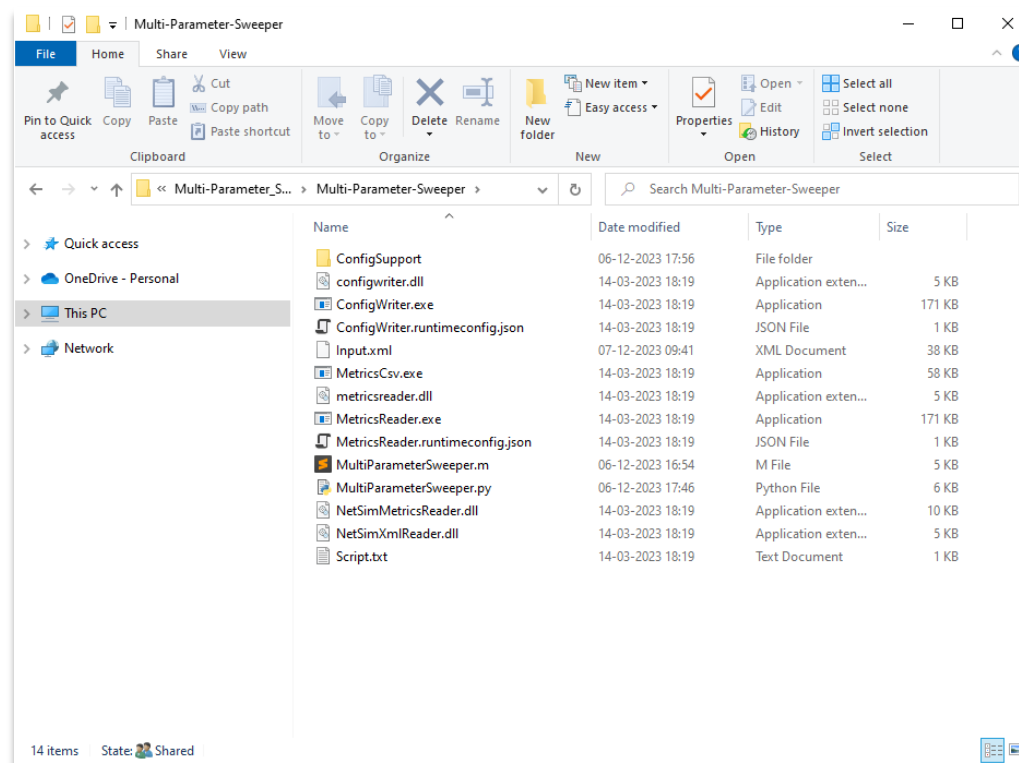


Figure 1: Project directory consists of several binaries

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.

Note: Copy the "Configsupport" folder along with the "input.xml" from the saved NetSim network.

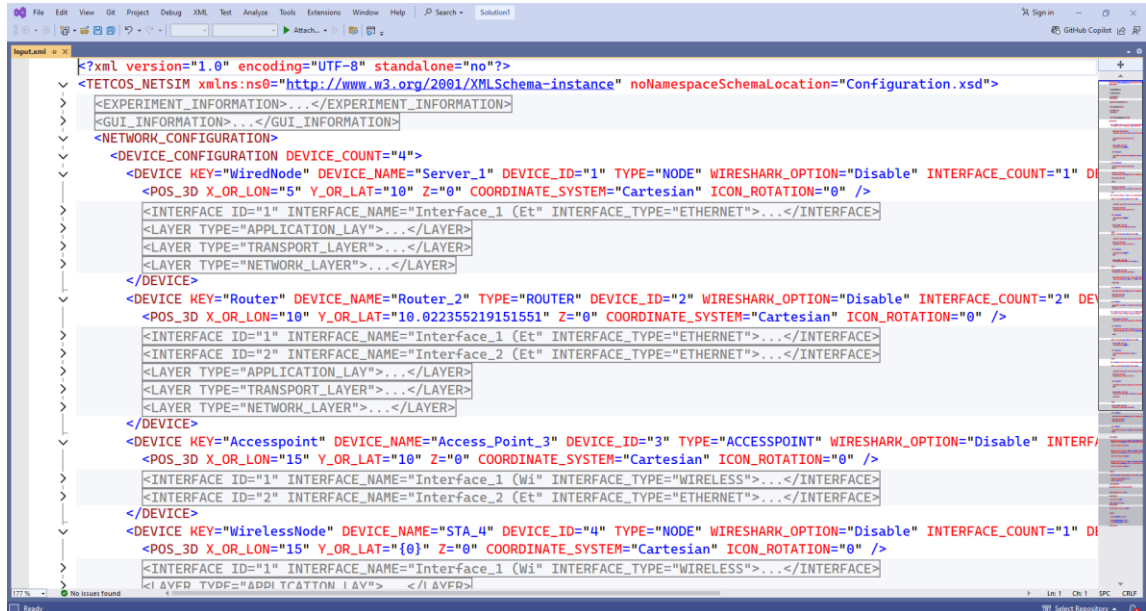


Figure 2: Input.xml file (NetSim Configuration file with the parameters to sweep)

- 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 Y coordinates of a device is to be varied the values can be modified In the input.xml file as shown below:

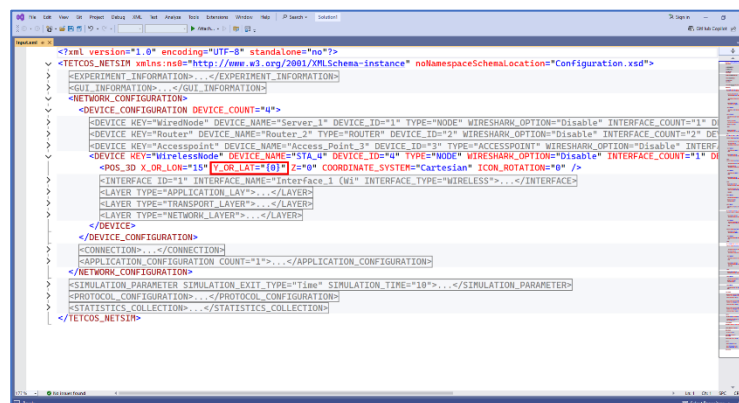


Figure 3: Modify X and Y coordinates in input input.xml file

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:

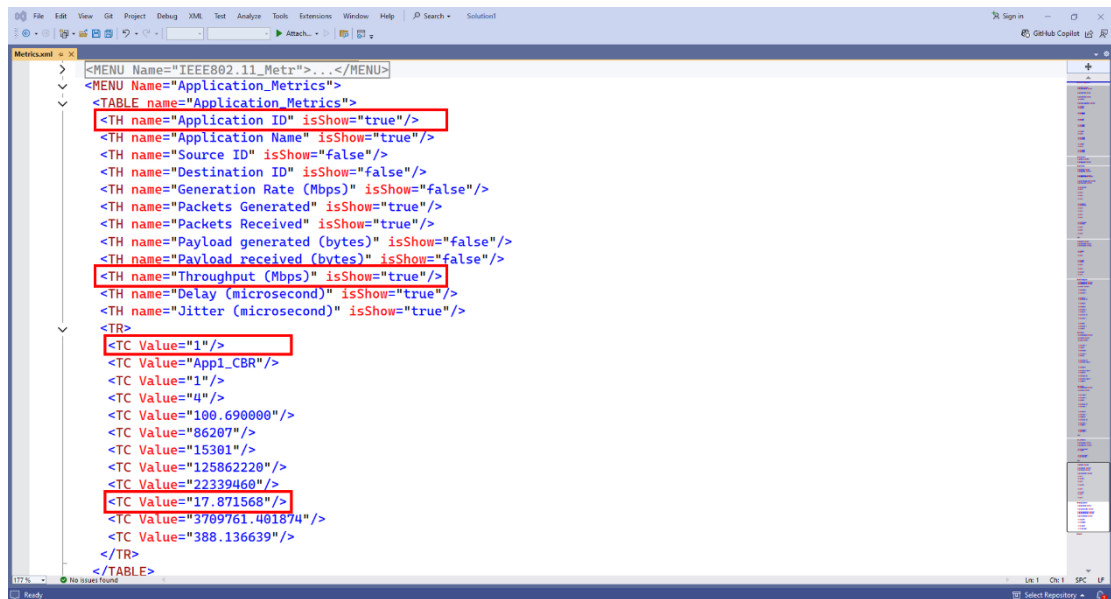


Figure 4: NetSim output Metrics.xml file

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

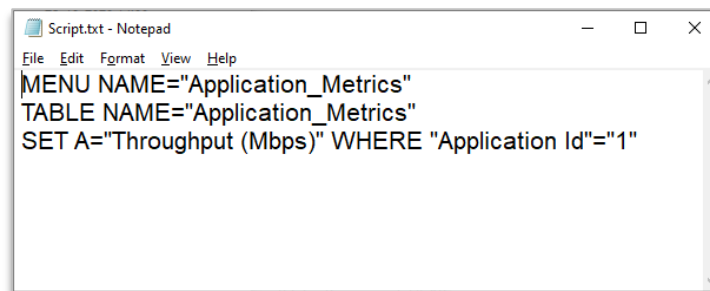


Figure 5: The application throughput is to be logged for each simulation modified in Script.txt

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

5. **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.

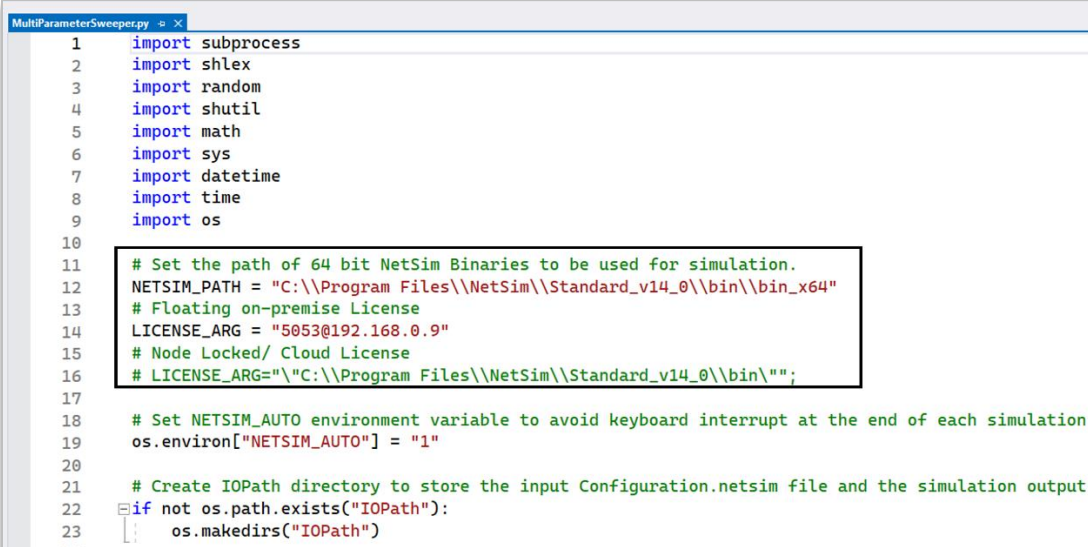
It uses the Script.txt file to determine which parameter to read from the Metrics file.

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.

6. **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.

7. **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.



```

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_v14_0\\bin\\bin_x64"
13  # Floating on-premise License
14  LICENSE_ARG = "5053@192.168.0.9"
15  # Node Locked/ Cloud License
16  # LICENSE_ARG="\"C:\\Program Files\\NetSim\\Standard_v14_0\\bin\\\"";
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 6: Set the NetSim Path where `NetSimCore.exe` is located.

For example,

NETSIM_PATH= "C:\\Program Files\\NetSim\\Standard_v14_0\\bin\\bin_x64"

Floating on-premise license (<port no>@<server ip address>):

LICENSE_ARG = "5053@192.168.0.9"

Node Locked or Cloud licenses (path of license file):

LICENSE_ARG="\"C:\\Program Files\\NetSim\\Standard_v14_0\\bin\\bin_x64\""

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

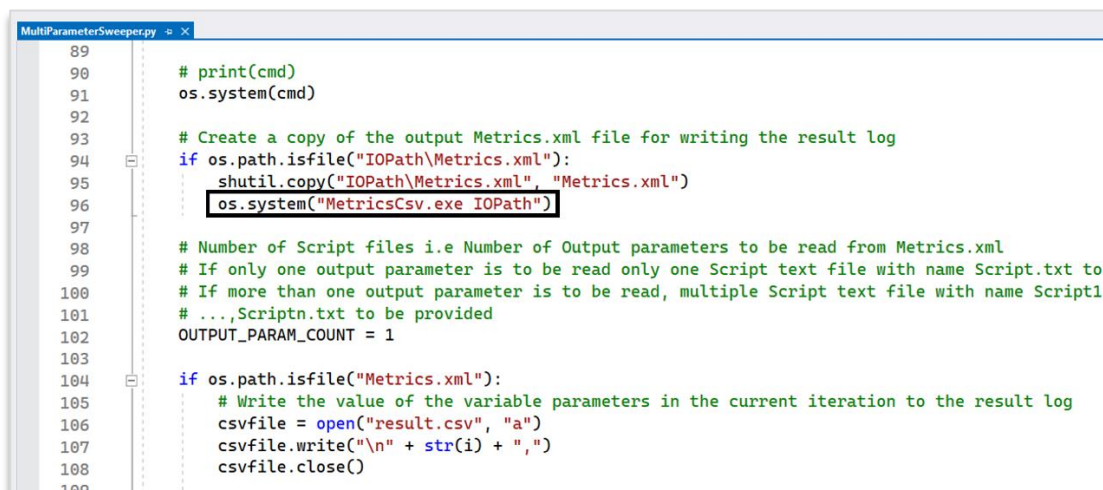


Figure 7: MetricsPrint.csv file will be created in the IOPath and then copied into the respective output folder

- 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:

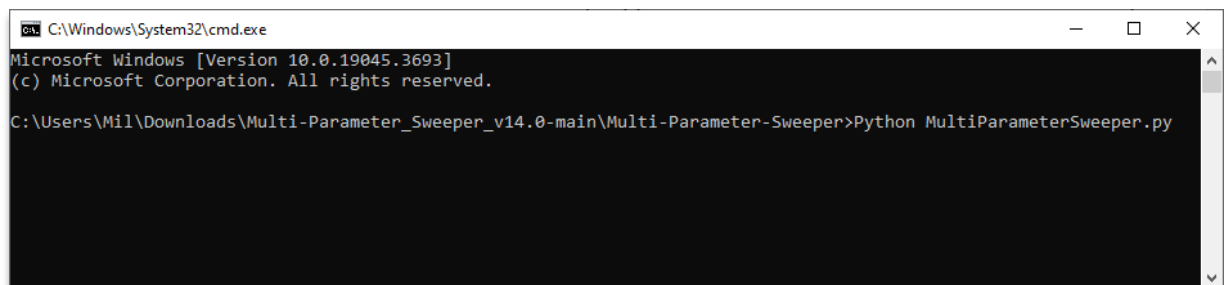


Figure 8: Running python script using command prompt

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

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

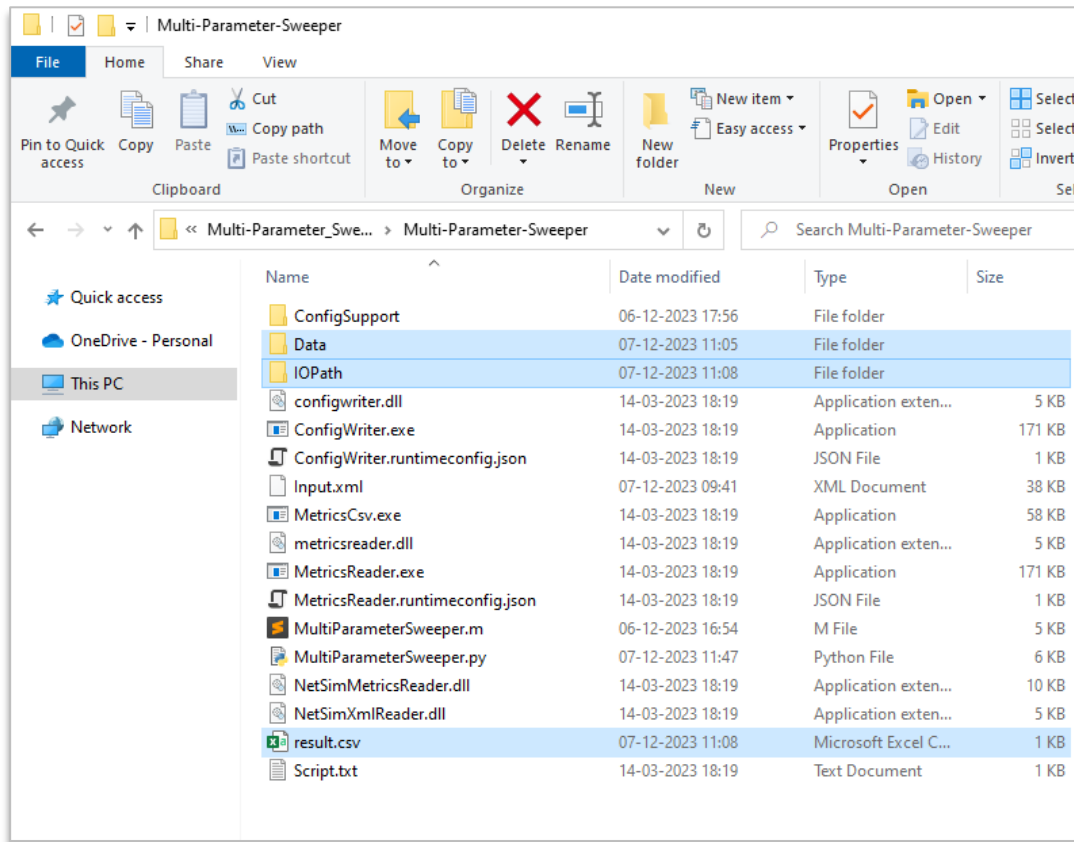


Figure 9: After Simulation Multi-Parameter-Sweeping folder contains output files like result.csv, Data etc

- **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.

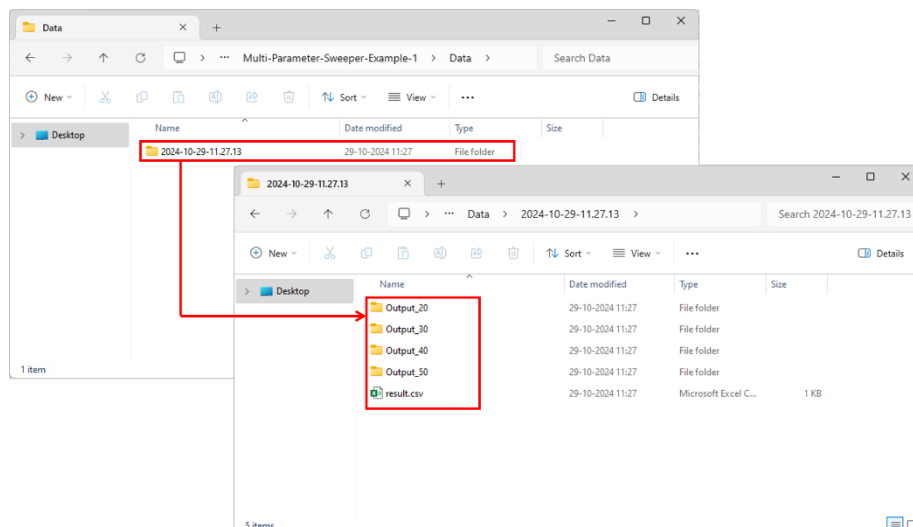


Figure 10: Based on values in the iteration, Output folder gets created in the Data directory inside a folder named as per date and time of simulation along with a copy of result.csv file.

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

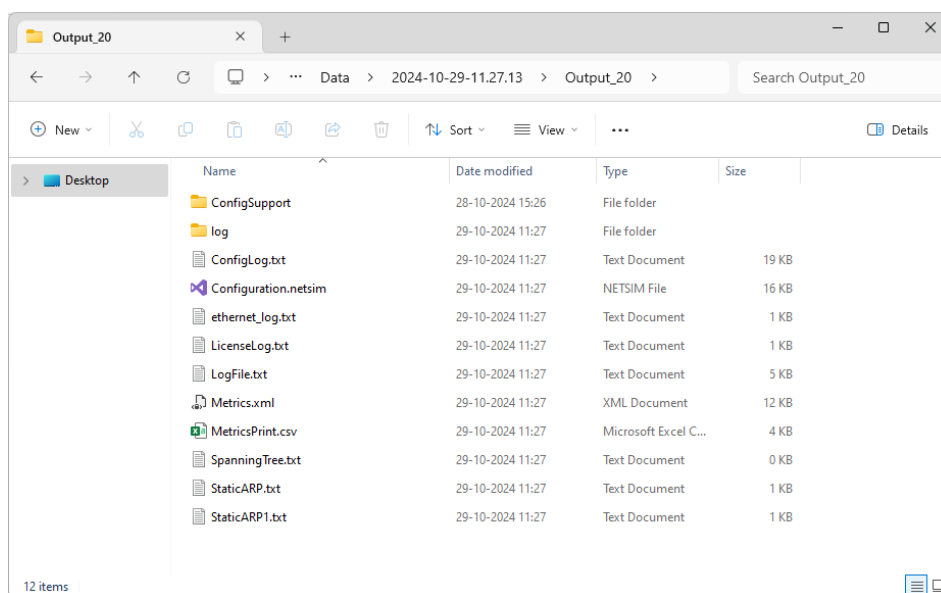


Figure 11: Each folder contains all the output files and the Metrics.xml file converted to MetricsPrint.csv file

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 12: Iterated value and Throughput obtained listed in result.csv

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

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

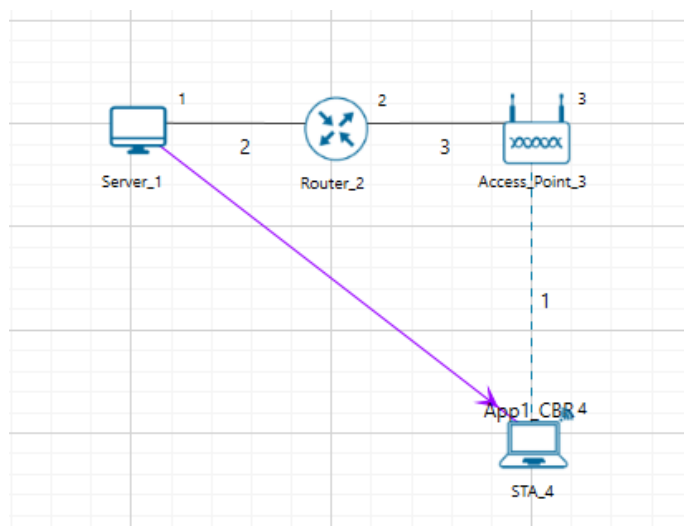


Figure 13: Network Topology

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 (Multi-Parameter-Sweeper_v14.2-main\Examples\Multi-Parameter-Sweeper-Example-1)

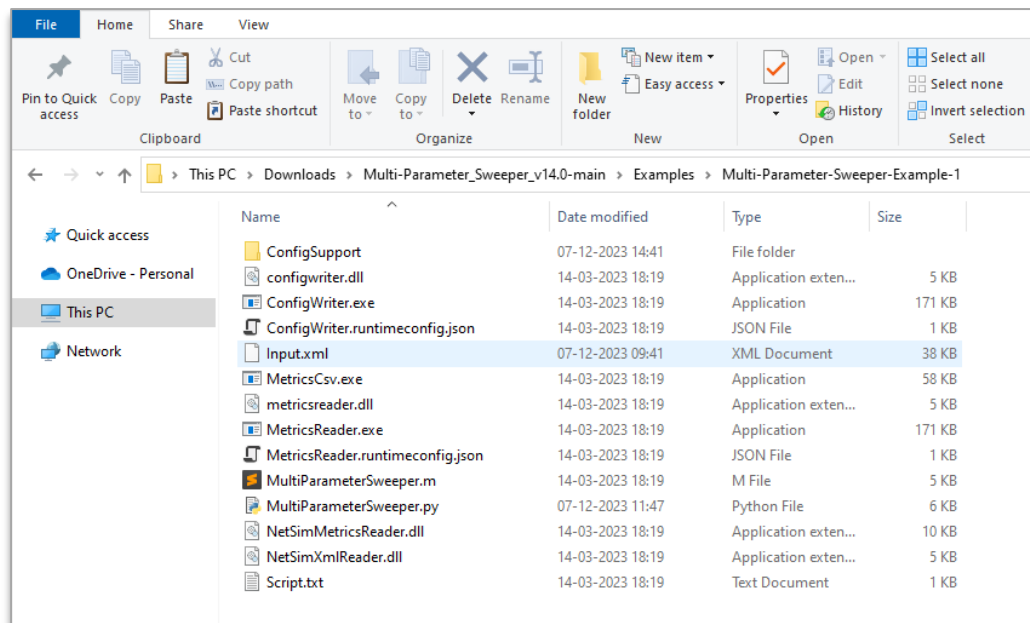


Figure 14: Renamed Configuration.netsim to input.xml and pasted in Multi-Parameter-Sweeper directory

1. 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:

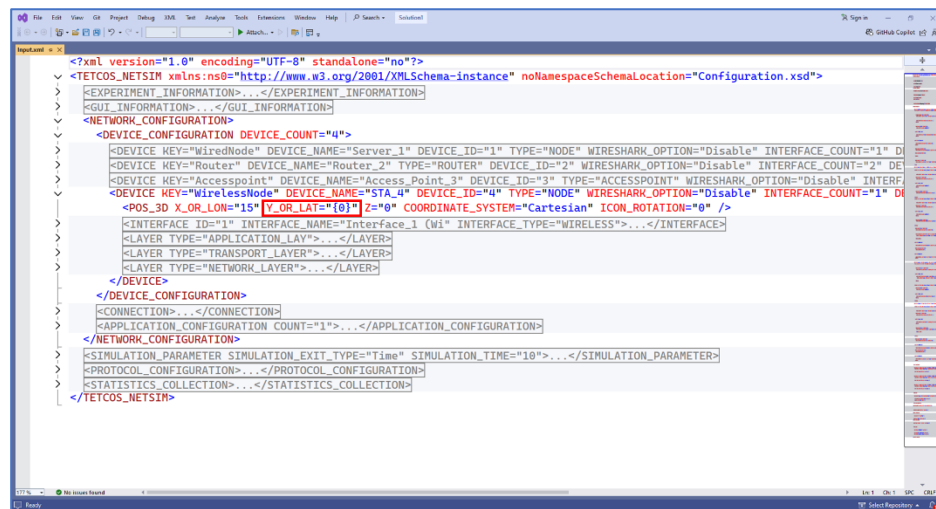


Figure 15: Modified Y coordinate of Wireless Node in input.xml

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.



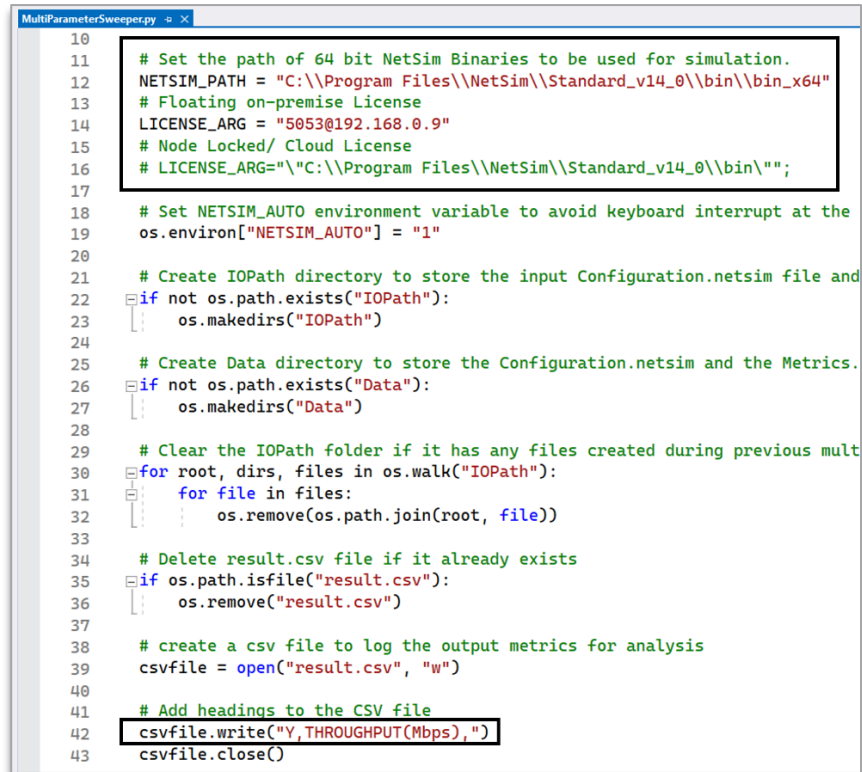
Figure 16: The application throughput is to be logged for each simulation modified in Script.txt

3. 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 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.

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:



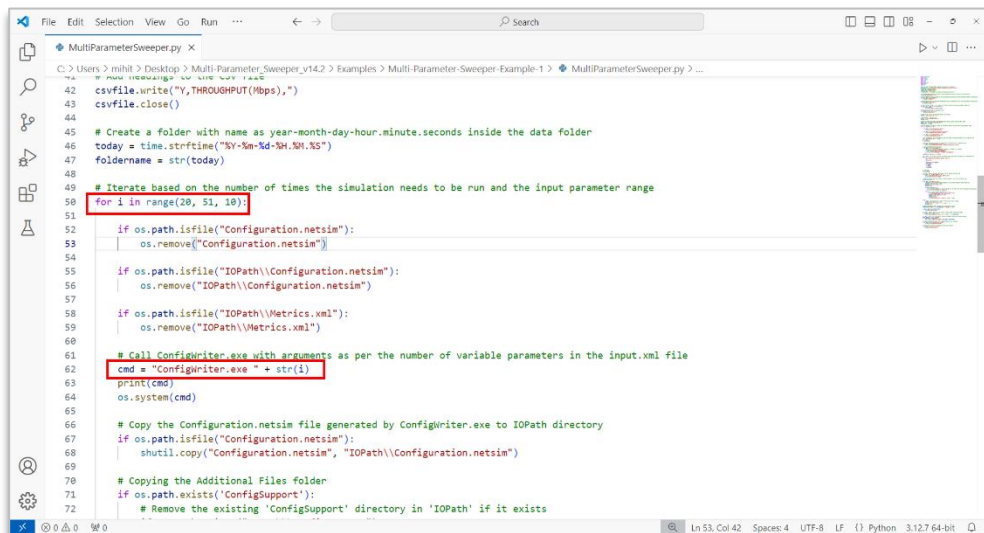
```

10
11 # Set the path of 64 bit NetSim Binaries to be used for simulation.
12 NETSIM_PATH = "C:\\Program Files\\NetSim\\Standard_v14_0\\bin\\bin_x64"
13 # Floating on-premise License
14 LICENSE_ARG = "5053@192.168.0.9"
15 # Node Locked/ Cloud License
16 # LICENSE_ARG="C:\\Program Files\\NetSim\\Standard_v14_0\\bin\\";
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 17: 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).



```

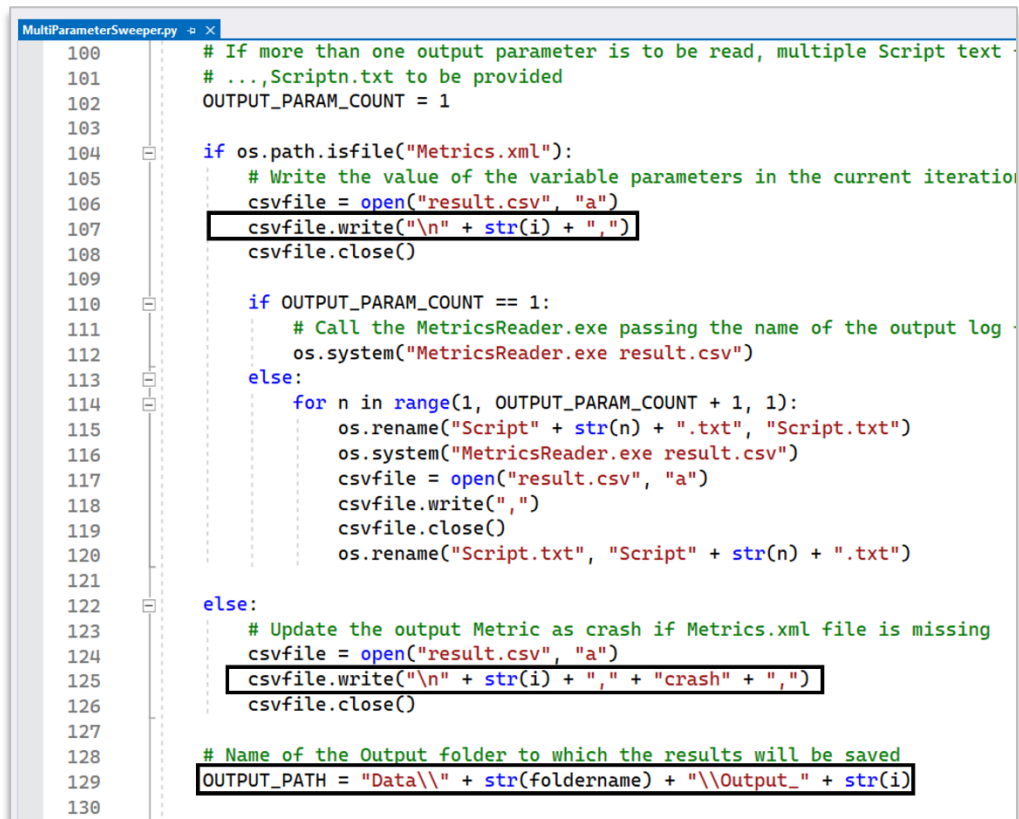
42 csvfile.write("Y,THROUGHPUT(Mbps),")
43 csvfile.close()
44
45 # Create a folder with name as year-month-day-hour.minute.seconds inside the data folder
46 today = time.strftime("%Y-%m-%d-%H-%M-%S")
47 folderName = str(today)
48
49 # Iterate based on the number of times the simulation needs to be run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

```

Figure 18: 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.



```

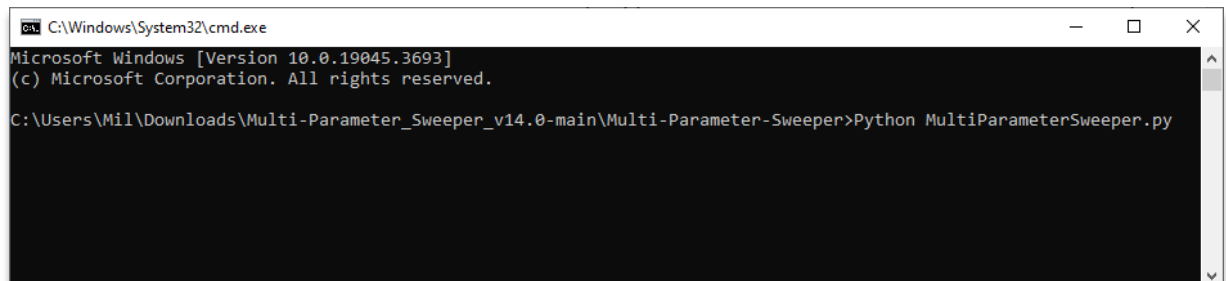
100 # If more than one output parameter is to be read, multiple Script text
101 # ...,Scriptn.txt to be provided
102 OUTPUT_PARAM_COUNT = 1
103
104 if os.path.isfile("Metrics.xml"):
105     # Write the value of the variable parameters in the current iteration
106     csvfile = open("result.csv", "a")
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 19: 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.

4. 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_v14.0-main\Multi-Parameter-Sweeper>Python MultiParameterSweeper.py

```

Figure 20: Running python script using 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:

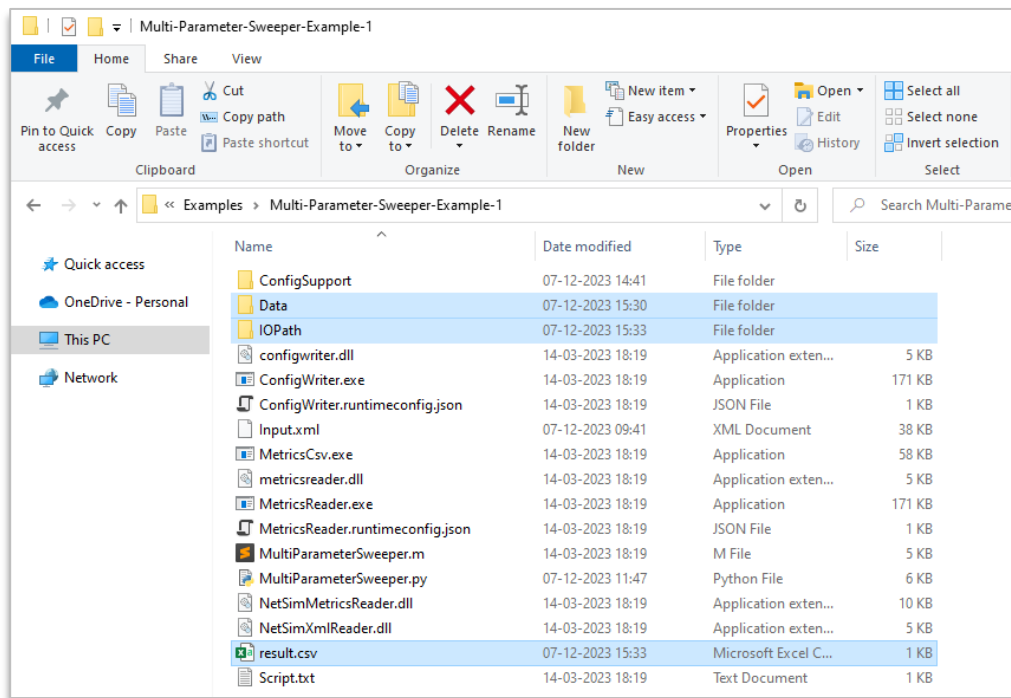


Figure 21: After Simulation Multi-Parameter-Sweeping folder contains output files like result.csv, Data etc

- **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.

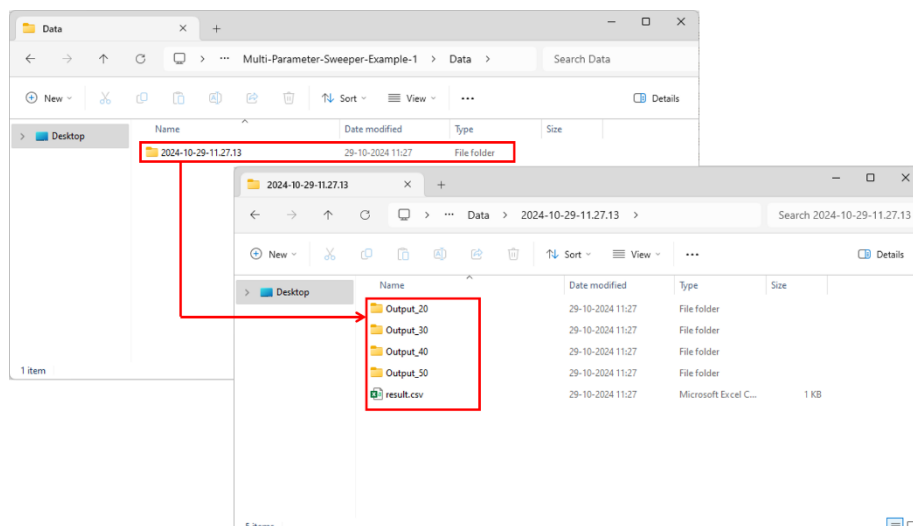


Figure 22: Based on distance Configuration.netsim files created in output folder

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

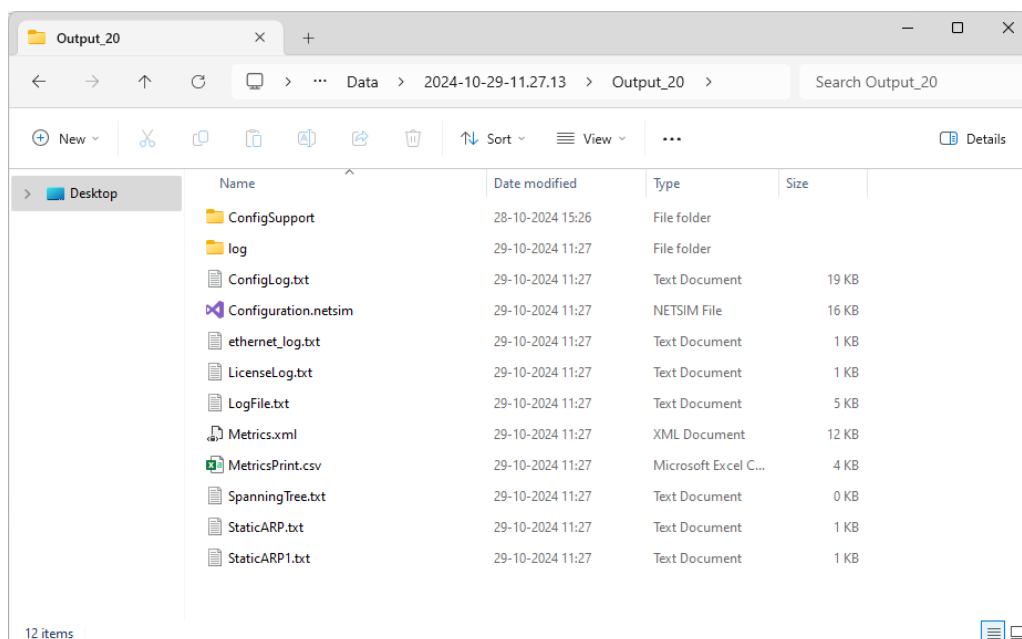


Figure 23: Each folder contains the all the output files

- **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.

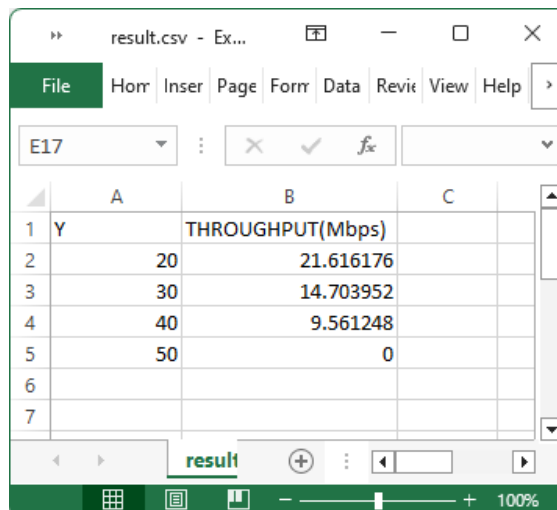


Figure 24: Distance Vs. Throughput obtained in result.csv

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, 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.

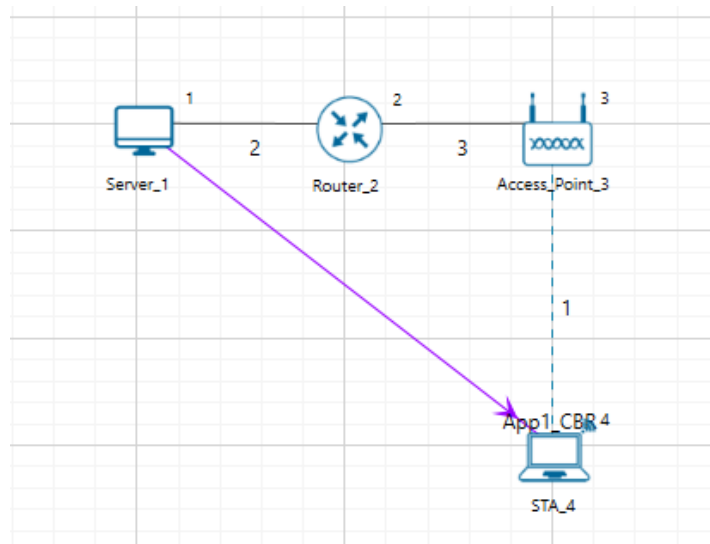
For Example, there can be two Script files as shown below:

Name	Date modified	Type	Size
Data	21-10-2022 17:16	File folder	
IOPath	21-10-2022 17:17	File folder	
configwriter.dll	20-01-2022 00:06	Application exten...	5 KB
ConfigWriter.exe	20-01-2022 00:06	Application	171 KB
ConfigWriter.runtimeconfig.json	20-01-2022 00:06	JSON File	1 KB
input.xml	20-10-2022 18:40	Microsoft Edge H...	37 KB
MetricsCsv.exe	20-01-2022 00:06	Application	58 KB
metricsreader.dll	20-01-2022 00:06	Application exten...	5 KB
MetricsReader.exe	20-01-2022 00:06	Application	171 KB
MetricsReader.runtimeconfig.json	20-01-2022 00:06	JSON File	1 KB
MultiParameterSweeper.m	21-10-2022 17:16	M File	7 KB
MultiParameterSweeper.py	21-10-2022 17:45	Python source file	8 KB
NetSimMetricsReader.dll	20-01-2022 00:06	Application exten...	10 KB
NetSimXmlReader.dll	20-01-2022 00:06	Application exten...	5 KB
result.csv	21-10-2022 17:18	Microsoft Excel C...	1 KB
Script1.txt	20-01-2022 00:06	Text Document	1 KB
Script2.txt	20-01-2022 00:06	Text Document	1 KB

Figure 25: Multiple output parameters

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

1. Consider the following interwork scenario in NetSim, comprising of a Wired Node, Router, Access point and a Wireless Node.



2. Set grid length to 60m x 60m grid setting property panel on the right. This needs to be done before the any device is placed on the grid.
3. Distance between Access Point and the Wireless Node is set to 10m

4. 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.
5. 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 100mW in both Access-point and wireless-node.
6. Wired Link speed is set to 1Gbps and propagation delay to 10 μ s in wired links.
7. Channel Characteristics: Path Loss Only, Path Loss Model: Log Distance, Path Loss Exponent: 3.5.
8. 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.
9. Application Generation Rate: 100 Mbps (Packet Size: 1460, Inter Arrival Time: 116.8 μ s)
10. Run the simulation for 10s.
11. 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.
12. Refer to the Example 2 directory which is part of the project folder (Multi-Parameter-Sweeper_v14.2\Examples\Multi-Parameter-Sweeper-Example-2).

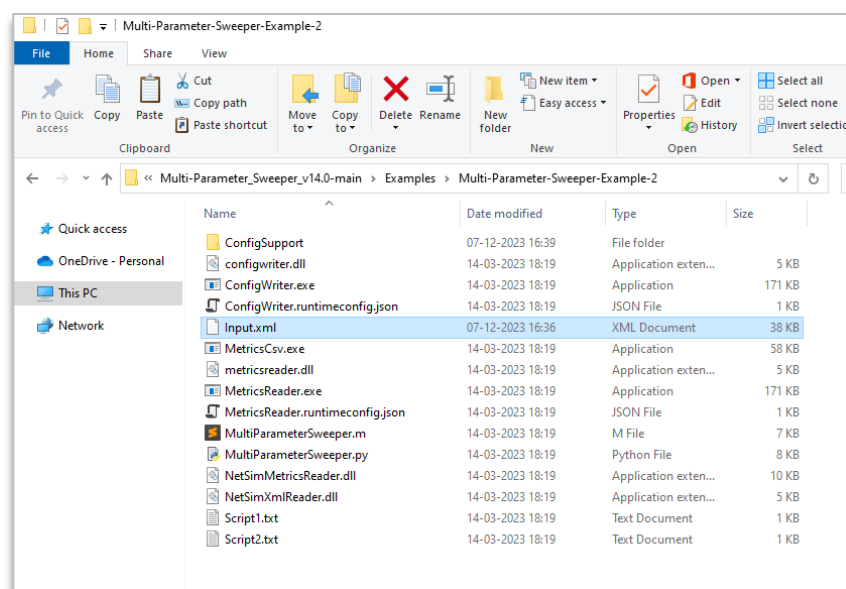


Figure 26: Renamed Configuration.netsim to input.xml and pasted in Multi-Parameter-Sweeper directory

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

Input Variables	
Bandwidth (MHz)	{0}
Tx Antenna Count	{1}
Rx Antenna Count	{2}
Inter Arrival Time (Microseconds)	{3}

Table 1: Variables are modified to the input.xml file

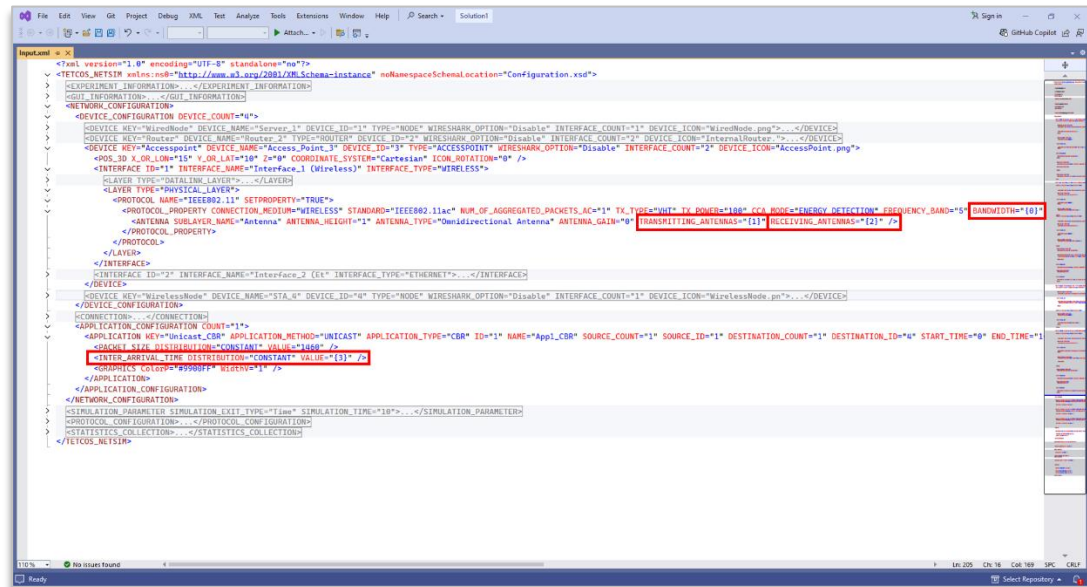


Figure 27: The above table 3 Variables are modified in input.xml file

14. 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.

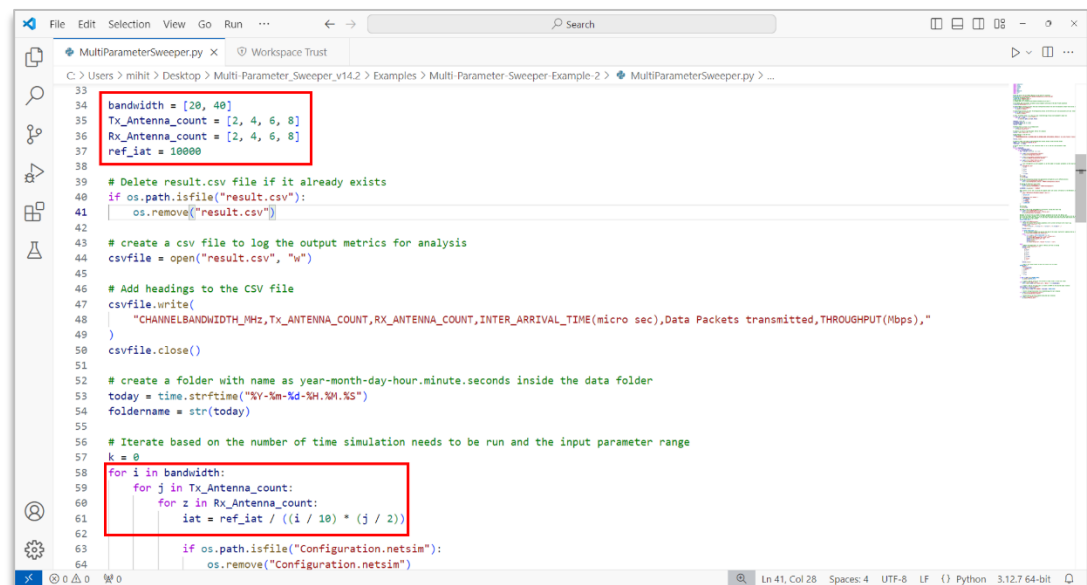


Figure 28: Modified MultiParameterSweeper.py based on input parameter

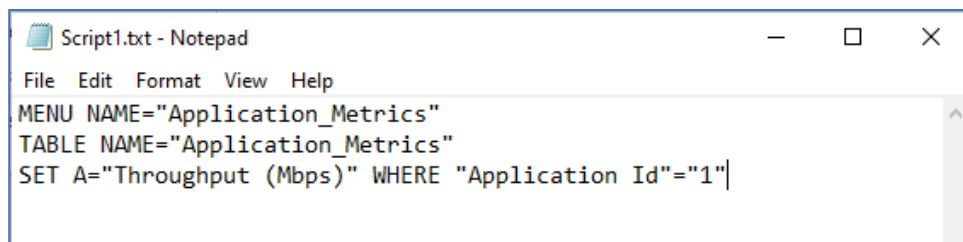
15. Multiple parameters are read from the Metrics.xml file and logged in the results.csv file along with the input parameters such as BANDWIDTH_MHz, TX_ANTENNA_COUNT, RX_ANTENNA_COUNT, INTER_ARRIVAL_TIME (micro sec).

Output Parameters
Throughput (Mbps)
Data Packets transmitted

Table 2: User need to modify these two parameters in Script files

16. 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.

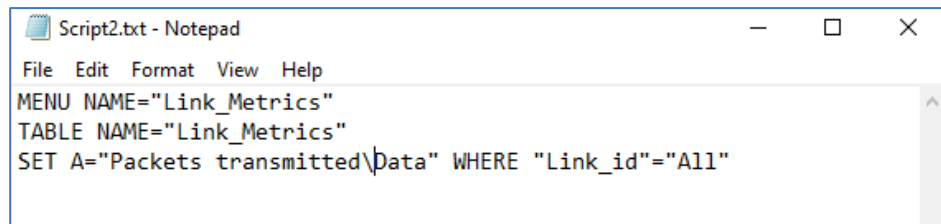
Script1.txt



```
Script1.txt - Notepad
File Edit Format View Help
MENU NAME="Application_Metrics"
TABLE NAME="Application_Metrics"
SET A="Throughput (Mbps)" WHERE "Application Id"="1"
```

Figure 29: The application throughput is to be logged for each simulation modified in Script1.txt

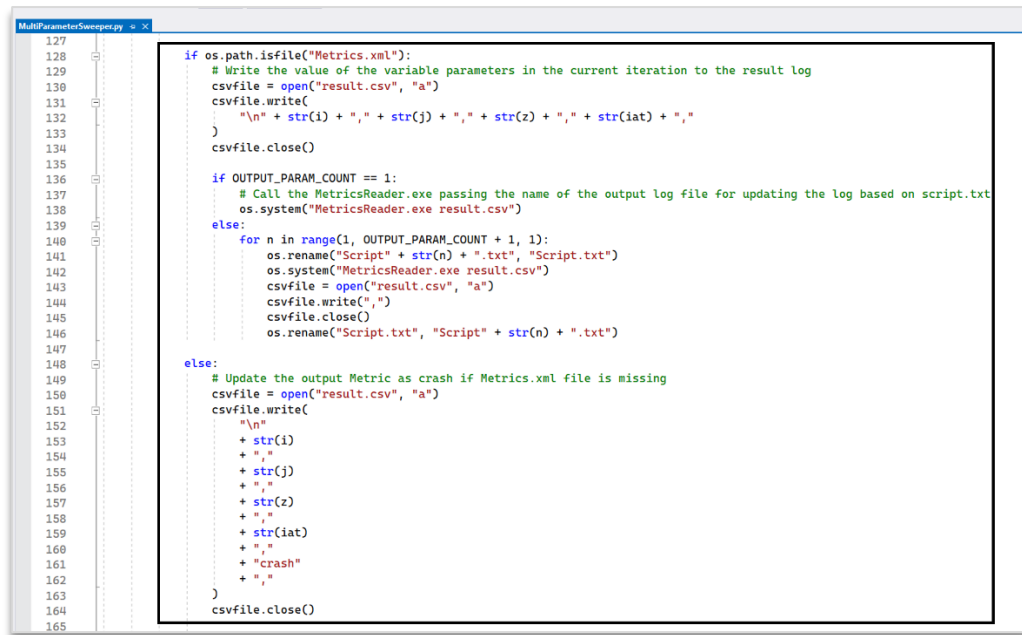
Script2.txt



```
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 30: The Data Packets transmitted is to be logged for each simulation modified in Script2.txt

17. In the python script MultiParameterSweeper.py, MetricsReader 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()

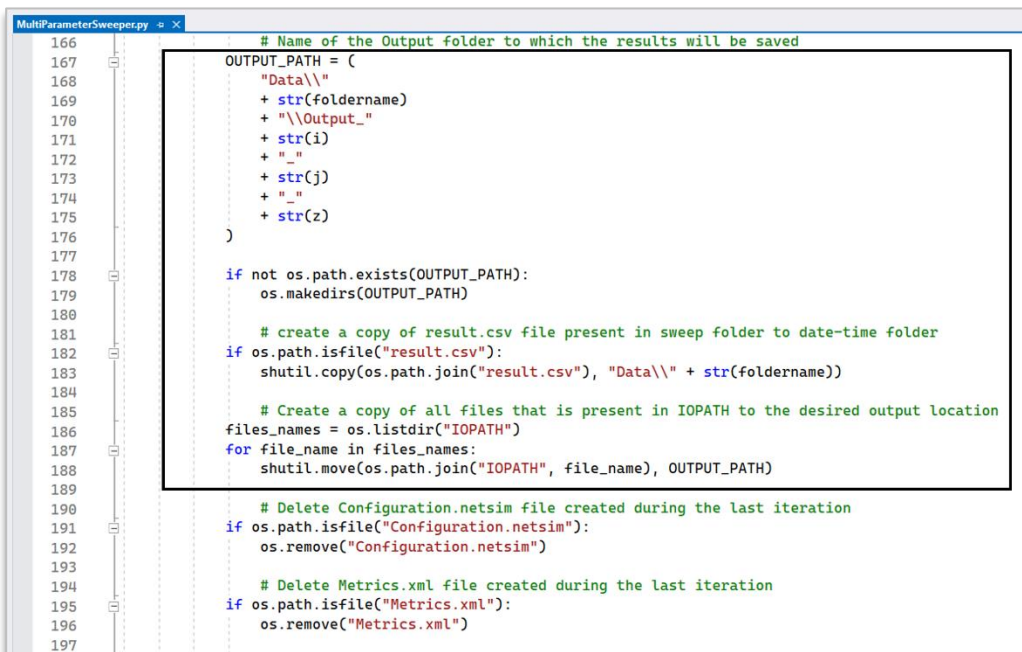
    if OUTPUT_PARAM_COUNT == 1:
        # Call the MetricsReader.exe passing the name of the output log file for updating the log based on script.txt
        os.system("MetricsReader.exe result.csv")
    else:
        for n in range(1, OUTPUT_PARAM_COUNT + 1, 1):
            os.rename("Script" + str(n) +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

    # 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 31: Modify python script MultiParameterSweeper.py file

18. 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 32: Modify python script MultiParameterSweeper.py file

19. 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:

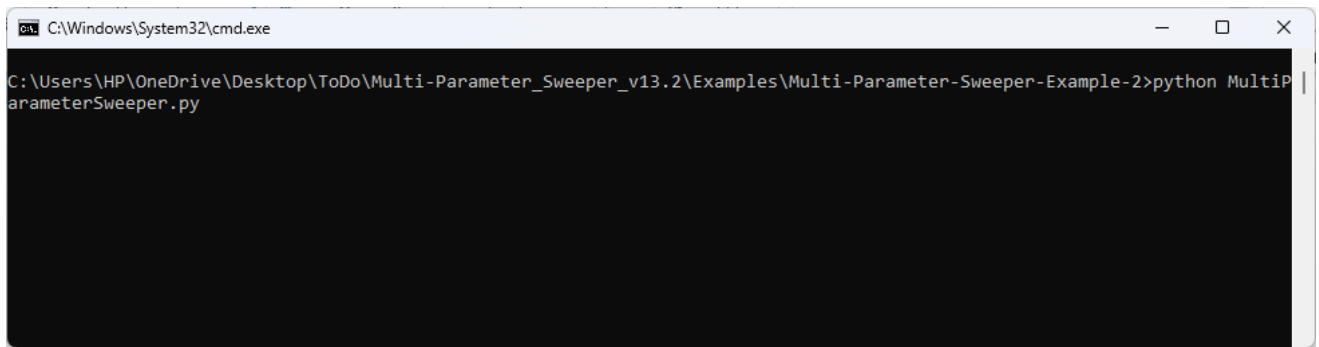


Figure 33: Running Python Script using command Prompt

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:

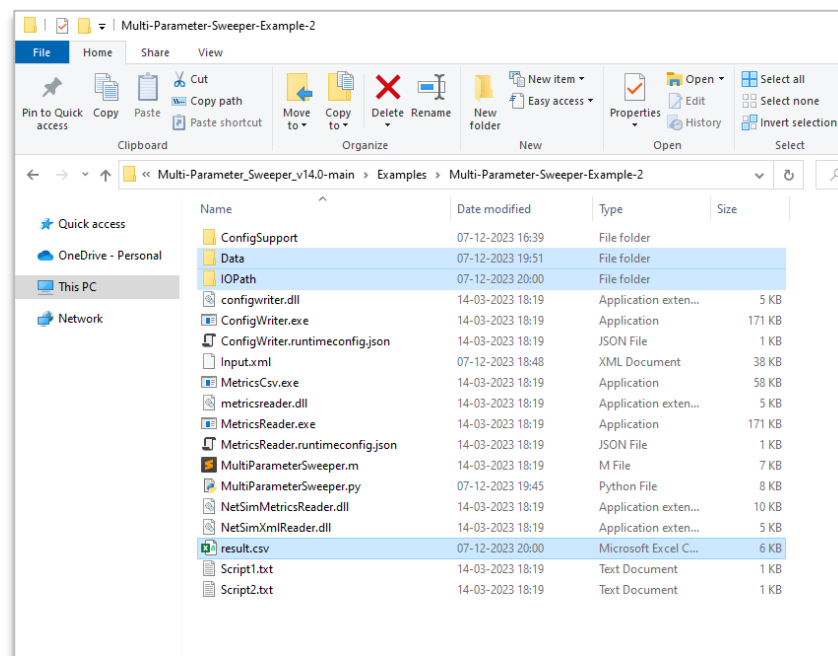


Figure 34: After Simulation Multi-Parameter-Sweeping folder contains output files like result.csv, Data etc

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

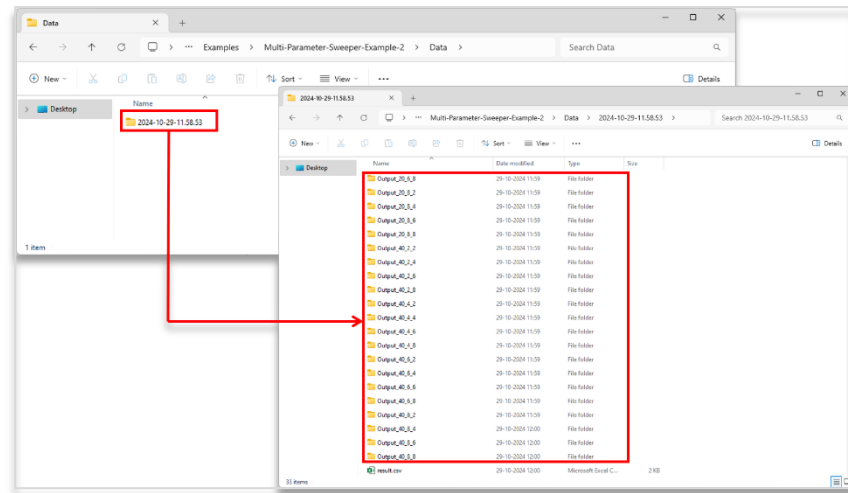


Figure 35: Based on input parameter Configuration.netsim and other files created in output folder

21. **IOPath**: Used for storing the Configuration.netsim file and the simulation files generated during each simulation run.
22. **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)		
1	20	2	2	5000	5990	2.327824		
2	20	2	4	5000	5990	2.327824		
3	20	2	6	5000	5990	2.327824		
4	20	2	8	5000	5990	2.327824		
5	20	4	2	2500	11982	4.659152		
6	20	4	4	2500	11982	4.659152		
7	20	4	6	2500	11982	4.659152		
8	20	4	8	2500	11982	4.659152		
9	20	6	2	1666.666667	17973	6.988144		
10	20	6	4	1666.666667	17973	6.988144		
11	20	6	6	1666.666667	17973	6.988144		
12	20	6	8	1666.666667	17973	6.988144		
13	20	8	2	1250	23959	9.3148		
14	20	8	4	1250	23959	9.3148		
15	20	8	6	1250	23959	9.3148		
16	20	8	8	1250	23959	9.3148		
17	40	2	2	2500	11982	4.659152		
18	40	2	4	2500	11982	4.659152		
19	40	2	6	2500	11982	4.659152		
20	40	2	8	2500	11982	4.659152		
21	40	4	2	1250	23959	9.3148		
22	40	4	4	1250	23959	9.3148		
23	40	4	6	1250	23959	9.3148		
24	40	4	8	1250	23959	9.3148		
25	40	6	2	833.333333	35943	13.973952		
26	40	6	4	833.333333	35943	13.973952		
27	40	6	6	833.333333	35943	13.973952		
28	40	6	8	833.333333	35943	13.973952		
29	40	8	2	625	47345	17.950992		
30	40	8	4	625	47929	18.63544		
31	40	8	6	625	47929	18.63544		
32	40	8	8	625	47929	18.63544		
33	40							
34								

Figure 36: Based on script result stored in result.csv file

Advanced use cases:

1. Running the sweeper with binaries of a modified workspace.
When the source codes of a workspace are modified, a warning message will be displayed in the simulation console waiting for user interrupt, as shown below:

To suppress this message and user interrupt, so that the sweeper runs without requiring any intervention, follow the steps given below:

-
- The screenshot shows a Windows File Explorer window with the address bar set to 'bin_x64'. The left sidebar shows the navigation pane with 'This PC' selected. The main area displays a list of files and folders. The file 'Checksum_backup.txt.txt' is selected, and its context menu is open, showing the following details:
- Type: Text Document
 - Size: 2.32 KB
 - Date modified: 04-08-2023 12:40
- The file list includes the following items:
- | Name | Date modified | Type | Size |
|-------------------------|------------------|----------------------|----------|
| 5G_BLER | 02-09-2023 10:04 | File folder | |
| 5GBEarming | 02-09-2023 10:04 | File folder | |
| AEESEncryption.dll | 04-08-2023 12:40 | Application exten... | 18 KB |
| AEESEncryption.pdb | 04-08-2023 12:55 | Program Debug D... | 1,156 KB |
| Animation.dll | 04-08-2023 12:40 | Application exten... | 20 KB |
| Animation.pdb | 04-08-2023 12:54 | Program Debug D... | 596 KB |
| AODVDLL.pdb | 04-08-2023 12:55 | Program Debug D... | 716 KB |
| ApplicationDll.pdb | 04-08-2023 12:55 | Program Debug D... | 716 KB |
| ARPDLL.pdb | 04-08-2023 12:54 | Program Debug D... | 532 KB |
| BatteryModel.dll | 04-08-2023 12:40 | Application exten... | 15 KB |
| BatteryModel.pdb | 04-08-2023 12:55 | Program Debug D... | 580 KB |
| CellularDll.pdb | 04-08-2023 12:54 | Program Debug D... | 548 KB |
| Checksum_backup.txt.txt | 04-08-2023 12:40 | Text Document | 3 KB |
| ChecksumCalculator.pdb | 04-08-2023 12:55 | Program Debug D... | 412 KB |
| ChildProcess.dll | 04-08-2023 12:40 | Application exten... | 13 KB |
| CLInterpretor.dll | 04-08-2023 12:40 | Application exten... | 29 KB |
| CLInterpretor.pdb | 04-08-2023 12:54 | Program Debug D... | 812 KB |
| CognitiveRadioDll.pdb | 04-08-2023 12:55 | Program Debug D... | 652 KB |

Now upon running any further simulations, warning messages will not be printed in the simulation console and no user intervention will be required.

- Page 22 of 23

- ACL input
- SUMO configuration files in case of VANET, etc

In such cases, the associated files can also be placed in the sweeper folder which contains the input.xml file and code can be slightly modified to copy all associated files when copying the Configuration file to the IOPath.