



POLYTECHNIQUE
MONTRÉAL

STAGE E20

MULTI-DISCIPLINARY OPTIMIZATION
BUILDING A BRIDGE BETWEEN MDIDS-GT & NOMAD

Presented to:
John Kidikian, Ing.

By:
Amine Kchouk

On August 28th, 2020
in Montréal

Table of contents

Introduction	4
Setup.....	5
1. Install NOMAD	5
2. Install Python	5
Procedure #1	6
Procedure #2	6
3. Download MDIDS-NOMAD	6
4. Download MDIDS-GT	7
5. Unit tests.....	7
6. Running the example	8
Getting started.....	9
1. Generating an MDIDS master file	10
2. Writing the <i>config.ini</i> configuration file	11
3. Writing the <i>params.txt</i> configuration file	12
4. Running the optimization	13
Advanced usage	13
Scripts' role and usage.....	13
wrapper.py.....	13
postprocess.py.....	14
sensitivity.py	15
MDIDS-NOMAD.bat	15
Advanced NOMAD parameters	16
Input variables specifications.....	16
Stop criteria	18
Results handling.....	18
Optimization advices	19
Granularity	19
Problem specification	20
ANNEX A - Setup screenshots	23
NOMAD.....	23

Python.....	24
Procedure #1	24
Procedure #2	25
ANNEX B - Samples configuration files	26
References	27

Introduction

The objective of this paper is to explain how to use NOMAD to easily optimize any MDIDS-GT (*MDIDS* for short) problem defined by the user. NOMAD is an optimization software developed by professors of Polytechnique Montréal, and it is designed to solve any problem by trial and error and by only knowing the inputs and outputs of the said problem (aka a black box problem). The *problem* in the case of this report will be a particular step in the MDIDS-GT design process.

NOMAD itself can be used in conjunction with any piece of software, however some coding is required to adapt its usage to a specific software. This is where the code MDIDS-NOMAD comes into play. As its name implies, MDIDS-NOMAD is a piece of software that was designed specifically to make it easier to use NOMAD with MDIDS. The software is divided in 2 parts:

1. A wrapper, that serves to translate the messages sent between MDIDS and NOMAD
2. A post-processor, that creates tables and graphs with the results generated by MDIDS and NOMAD

In this report, you will find all the necessary information to learn how to use MDIDS-NOMAD (a MS Windows based software) and to start the optimization problem using MDIDS. We recommend that you read this paper in the order it is presented, as it presents the workflow a new user would have to execute to understand how to use MDIDS-NOMAD.

Setup

In this section, we will discuss all the preliminary steps required to be able to use MDIDS-NOMAD, namely:

1. **1.** Install NOMAD
2. **2.** Install Python
3. **3.** Download MDIDS-NOMAD
4. **4.** Download MDIDS-GT
5. **5.** Unit tests
6. **Running the example**

Unless otherwise specified, all the files and folders referenced are relative to the path where MDIDS-NOMAD has been extracted in the 3rd step. If you are prompted for admin privileges at any step, click **Yes** to allow admin privileges. Screenshots are available in Annex A.

1. Install NOMAD

MDIDS-NOMAD uses NOMAD in batch mode. This means that NOMAD needs to be installed as a standalone software. **NOTE:** If you have already installed NOMAD, you can skip to the next step.

Download

To download NOMAD, navigate to <https://www.gerad.ca/nomad/download.html>, and download the latest version of NOMAD for Windows (or use [this direct link](#) if it still works).

Install

Follow these instructions to install NOMAD correctly.

1. Launch the NOMAD setup executable that you just downloaded (it should be named *NOMAD_setup.exe*, in your downloads folder)
2. Click **Next**
3. Click **I Agree**
4. Click **Next**
5. Append “NOMAD” to the **Destination Folder** then click **Install**
6. Replace “Desktop” with “Documents” in the **Select Directory** then click **Next**
7. Click **Finish**

2. Install Python

Python is a free programming language which is very useful for writing simple and complex scripts. Since MDIDS-NOMAD has been coded using the Python 3 programming language, it is mandatory that it be installed for MDIDS-NOMAD to work. **NOTE:** If you have already installed Python 3, you can skip to the next step.

You may choose between 2 procedures to install Python 3 on your computer. Please choose only one of these procedures (if you want to try the other procedure, you must undo the changes made by the procedure you used the first time). Procedure #1 is the easiest, and procedure #2 is the most common (easier troubleshooting for advanced Python users).

Procedure #1

This method installs Python from the Microsoft Store. Follow these instructions to do so:

1. Open the Microsoft Store (search “Microsoft Store” in the Start Menu to find it)
2. Search for “Python 3.8” in the search bar of the Microsoft Store
3. Click **Get** (you can close the login prompt if you don’t have a Microsoft account or you don’t want to connect to your account)

Procedure #2

This method installs Python the *classic* way, using a setup executable file.

1. Navigate to <https://www.python.org/downloads/> and download the latest version of Python 3 for Windows (or use [this direct link](#) if it still works).
2. Launch the Python 3 setup executable that you just downloaded (it should be named *python-3.x.x.exe*, in your downloads folder)
3. Make sure **Add Python 3.X to PATH** is checked then click **Install Now**
4. Once it is done installing, click **Disable path length limit**, then click **Close**

3. Download MDIDS-NOMAD

MDIDS-NOMAD is the software developed to facilitate the use of NOMAD with MDIDS. Follow these instructions to download it:

1. Navigate to https://github.com/Tinynja/MDIDS-NOMAD_v1
2. Click **Code** (the green button with a download icon)
3. Click **Download ZIP** to download the MDIDS-NOMAD archive (or use [this direct link](#) if it still works)
4. Extract the MDIDS-NOMAD archive (it should be named *MDIDS-NOMAD_v1-master.zip*, in your downloads folder) to a folder where you have write access without admin privileges (e.g. a folder on your desktop)

Note: We will reference the directory containing the files *MDIDS-NOMAD.bat*, *wrapper.py*, *postprocess.py*, etc. as your **MDIDS-NOMAD home directory**

4. Download MDIDS-GT

Follow these steps to download MDIDS-GT and place the files in the correct directory:

1. Navigate to <https://sites.google.com/site/mdidsgtdownloadpage/> and download the latest version of MDIDS-GT
2. Open the MDIDS-GT archive you just downloaded (it should be named *Distribution-MDIDS-20XXXXXX.zip*, in your downloads folder)
3. Select the **MDIDS executable** that has “console” in its name and extract (aka copy) it into the folder you created when downloading MDIDS-NOMAD, alongside the *MDIDS-NOMAD.py* file.
4. If necessary, rename the MDIDS console executable (the one you just copied now) to “MDIDSGTconsole.exe”.

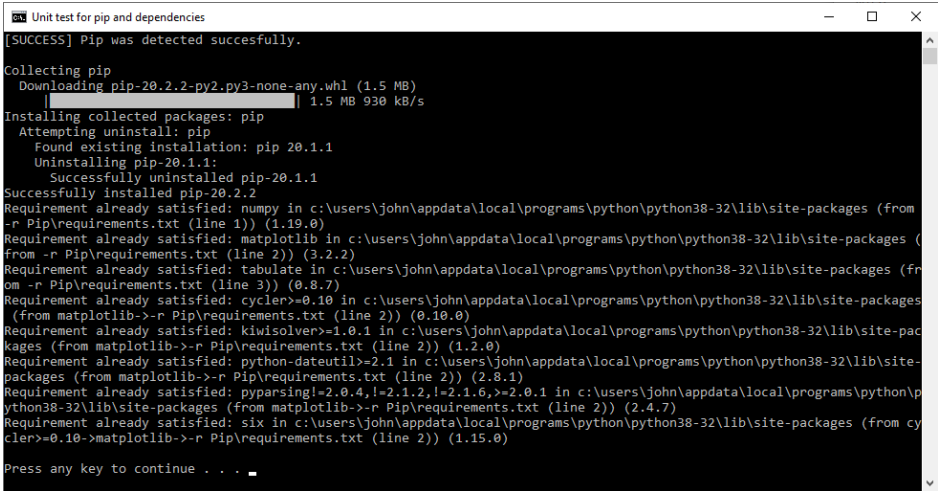
5. Unit tests

In the *Unit tests* folder, you will find four (4) unit test batch files. Each of these unit tests verify that the corresponding above software have been installed correctly and has been “detected”. To run a unit test, simply double click on the *.bat file and read the messages that appear on the black window. It is recommended to run these tests in the following order, for easier troubleshooting if you encounter any problems:

1. NOMAD.bat
2. Python.bat
3. Pip.bat
4. MDIDSGTconsole.bat

Note: If any one of these tests is not successful, MDIDS-NOMAD will not work.

Note: For 3. Pip.bat, the system will most likely download and install the required Pip file. Refer to the screenshot below of the possible outcome.



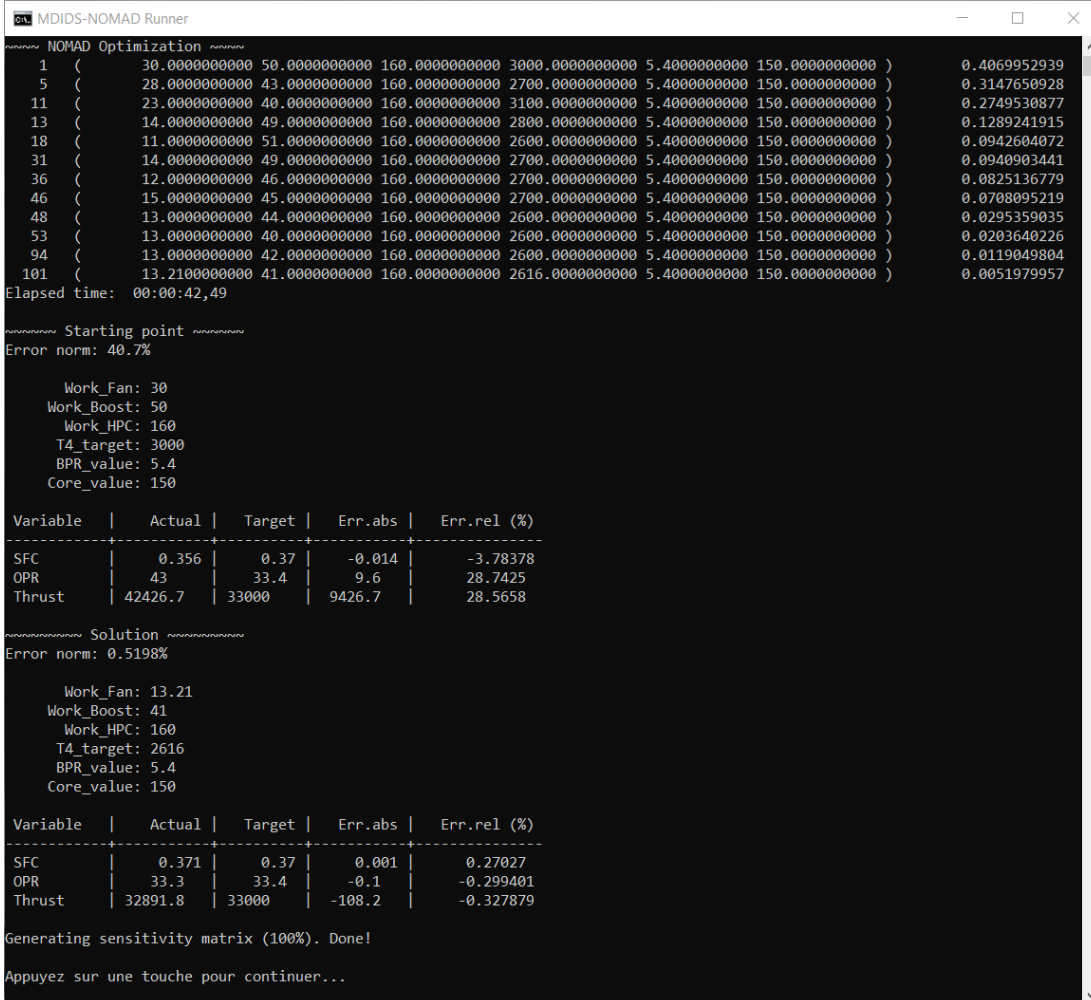
```
Unit test for pip and dependencies
[SUCCESS] Pip was detected successfully.
Collecting pip
  Downloading pip-20.2.2-py2.py3-none-any.whl (1.5 MB)
    | 1.5 MB 930 kB/s
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 20.1.1
    Uninstalling pip-20.1.1:
      Successfully uninstalled pip-20.1.1
  Successfully installed pip-20.2.2
Requirement already satisfied: numpy in c:\users\john\appdata\local\programs\python\python38-32\lib\site-packages (from
-r Pip\requirements.txt (line 1)) (1.19.0)
Requirement already satisfied: matplotlib in c:\users\john\appdata\local\programs\python\python38-32\lib\site-packages (
from -r Pip\requirements.txt (line 2)) (3.2.2)
Requirement already satisfied: tabulate in c:\users\john\appdata\local\programs\python\python38-32\lib\site-packages (fr
om -r Pip\requirements.txt (line 3)) (0.8.7)
Requirement already satisfied: cycler>=0.10 in c:\users\john\appdata\local\programs\python\python38-32\lib\site-packages
 (from matplotlib->-r Pip\requirements.txt (line 2)) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\john\appdata\local\programs\python\python38-32\lib\site-pac
kages (from matplotlib->-r Pip\requirements.txt (line 2)) (1.2.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\john\appdata\local\programs\python\python38-32\lib\site-
packages (from matplotlib->-r Pip\requirements.txt (line 2)) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!>2.1.2,!>2.1.6,>=2.0.1 in c:\users\john\appdata\local\programs\python\p
ython38-32\lib\site-packages (from matplotlib->-r Pip\requirements.txt (line 2)) (2.4.7)
Requirement already satisfied: six in c:\users\john\appdata\local\programs\python\python38-32\lib\site-packages (from cy
cler>=0.10->matplotlib->-r Pip\requirements.txt (line 2)) (1.15.0)
Press any key to continue . . .
```

Figure 1 - Pip.bat successful outcome

6. Running the example

A working example optimization problem is provided with MDIDS-NOMAD. This step is optional, but it helps to rule out setup errors if you encounter problems. To run the example problem, follow these steps:

1. Navigate to your MDIDS-NOMAD home directory
2. Go in *Samples/Example*, and copy the 3 files present (*config.ini*, *params.txt*, *EngineV2500-Master*)
3. Paste those 3 files into your MDIDS-NOMAD home directory.
4. Double click the file named *MDIDS-NOMAD.bat*
5. A black window should appear with a message similar to the one in **Figure 2** - Example optimization output.
6. A *Results* folder should be created, containing 9 files after a complete execution of the bat file.



```
MDIDS-NOMAD Runner
~~~~~ NOMAD Optimization ~~~~~
1 ( 30.000000000 50.000000000 160.000000000 3000.000000000 5.400000000 150.000000000 ) 0.4069952939
5 ( 28.000000000 43.000000000 160.000000000 2700.000000000 5.400000000 150.000000000 ) 0.3147650928
11 ( 23.000000000 40.000000000 160.000000000 3100.000000000 5.400000000 150.000000000 ) 0.2749530877
13 ( 14.000000000 49.000000000 160.000000000 2800.000000000 5.400000000 150.000000000 ) 0.1289241915
18 ( 11.000000000 51.000000000 160.000000000 2600.000000000 5.400000000 150.000000000 ) 0.0942604072
31 ( 14.000000000 49.000000000 160.000000000 2700.000000000 5.400000000 150.000000000 ) 0.0940903441
36 ( 12.000000000 46.000000000 160.000000000 2700.000000000 5.400000000 150.000000000 ) 0.0825136779
46 ( 15.000000000 45.000000000 160.000000000 2700.000000000 5.400000000 150.000000000 ) 0.0708095219
48 ( 13.000000000 44.000000000 160.000000000 2600.000000000 5.400000000 150.000000000 ) 0.0295359035
53 ( 13.000000000 40.000000000 160.000000000 2600.000000000 5.400000000 150.000000000 ) 0.0203640226
94 ( 13.000000000 42.000000000 160.000000000 2600.000000000 5.400000000 150.000000000 ) 0.0119049804
101 ( 13.210000000 41.000000000 160.000000000 2616.000000000 5.400000000 150.000000000 ) 0.0051979957
Elapsed time: 00:00:42,49

~~~~~ Starting point ~~~~~
Error norm: 40.7%

Work_Fan: 30
Work_Boost: 50
Work_HPC: 160
T4_target: 3000
BPR_value: 5.4
Core_value: 150

Variable | Actual | Target | Err.abs | Err.rel (%)
-----|-----|-----|-----|-----
SFC      | 0.356 | 0.37 | -0.014 | -3.78378
OPR      | 43    | 33.4 | 9.6    | 28.7425
Thrust   | 42426.7 | 33000 | 9426.7 | 28.5658

~~~~~ Solution ~~~~~
Error norm: 0.5198%

Work_Fan: 13.21
Work_Boost: 41
Work_HPC: 160
T4_target: 2616
BPR_value: 5.4
Core_value: 150

Variable | Actual | Target | Err.abs | Err.rel (%)
-----|-----|-----|-----|-----
SFC      | 0.371 | 0.37 | 0.001 | 0.27027
OPR      | 33.3  | 33.4 | -0.1  | -0.299401
Thrust   | 32891.8 | 33000 | -108.2 | -0.327879

Generating sensitivity matrix (100%). Done!
Appuyez sur une touche pour continuer...
```

Figure 2 - Example optimization output

Getting started

Before explaining how to use (or modify) MDIDS-NOMAD, let us first demonstrate through an actual MDIDS example on how it can be useful. If we want to know what combination of TurboFan values of fan work, boost work, HPC work and T4 generate a given SFC, OPR and thrust, we could study the mathematical equations that link these variables together and optimize this problem algebraically. Even though this approach has the potential of giving us an exact relationship between those variables, it can be very tricky to optimize a problem this way if the equations are very complex.

Another way of figuring out what values of fan work, boost work, HPC work and T4 generate a given SFC, OPR and thrust is to go through a trial-and-error approach. We try different values of fan work, and see what SFC, OPR and thrust we get. We do the same thing with the other variables, and by trial and error we come to a result that is close enough to the values of SFC, OPR and thrust we targeted. Note that we do not need to know the relationship between the input and output variables to optimize the problem with this approach. However, this approach takes a lot of time and is not very accurate.

An even better approach is to do trial-and-error, but intelligently. This is where MDIDS-NOMAD (more specifically NOMAD) comes into play. NOMAD was designed to try many input values intelligently by using the MADS algorithm (if you want, you can read more about the MADS algorithm in reference [1]) until it finds a solution that optimizes the output variables. Note that the problem can still be a black box (unknown input-output relationships) and this approach can still find a solution.

Under the hood, NOMAD can only minimize a scalar value, so it is unfortunately impossible to set the error of each output variable separately. To accommodate for this, MDIDS-NOMAD performs a few essential calculations to generate this scalar value(s). First, we set a target value for each output variable. Then, at each iteration, we calculate the Euclidian norm of the relative error of every output variable:

$$E = \sqrt{\sum e_i^2}$$

Let's show how this is actually done through an example:

Suppose we have 2 input variables (**a**, **b**) and 3 output variables (**x**, **y**, **z**), and we want to achieve the following output variable values: x=5, y=100, z=20.

Let's say for input values of a=10 and b=20, we get the following output variable values: x=7, y=90, z=25.

The relative error norm in this case would be calculated as such:

$$E = \sqrt{\left|\frac{x_1 - x_0}{x_0}\right|^2 + \left|\frac{y_1 - y_0}{y_0}\right|^2 + \left|\frac{z_1 - z_0}{z_0}\right|^2}$$

$$E = \sqrt{\left|\frac{7 - 5}{5}\right|^2 + \left|\frac{90 - 100}{100}\right|^2 + \left|\frac{25 - 20}{20}\right|^2} = 0,4822 \dots \approx 48,22\%$$

This way, we get a representation of how “good” the result of this evaluation is.

When an optimization problem is defined correctly, MDIDS-NOMAD will be able to:

1. Find the solution of an optimization problem with any precision
2. Generate a history (table and graph) of all the evaluations that lead to the solution
3. Generate a sensitivity matrix of the given optimization problem (table and graph)

We will now show you everything you need to do to define an optimization problem in MDIDS-NOMAD and to start optimizing on your own. For every problem, you need to do the 4 following steps.

1. Generating an MDIDS master file

Since MDIDS-NOMAD uses MDIDS-GT to optimize a given file, you first need to generate your specific *master* file. To do this, you need to design your gas turbine engine in MDIDS-GT (using the beautiful interface). Once you are done designing your gas turbine engine, it is mandatory that you save your MDIDS file using **version 6** (refer to the MDIDS user documentation to know the version number to save to). You can name the save file however you like, as it is shown in **Figure 3** - Saving the MDIDS-GT master file. You can place your master file wherever you want, however it is recommended to place it in your MDIDS-NOMAD home directory.

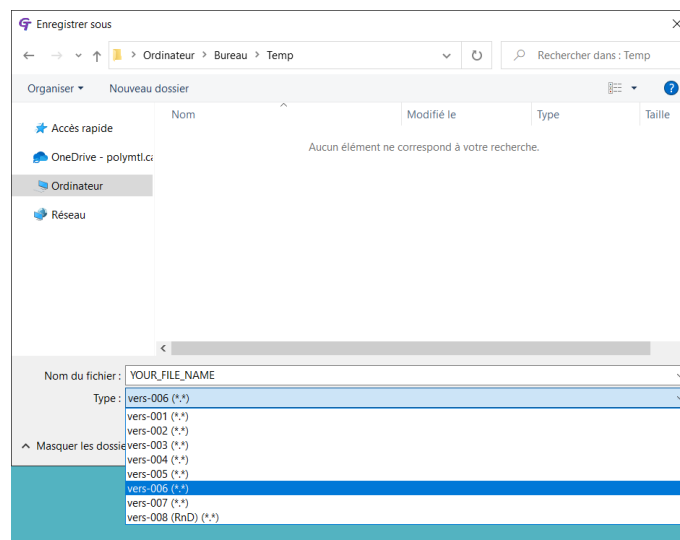


Figure 3 - Saving the MDIDS-GT master file

2. Writing the *config.ini* configuration file

The *config.ini* file contains some information that MDIDS-NOMAD uses when running the optimization and when generating the sensitivity matrix. This file should be named *config.ini* and it should be placed in your MDIDS-NOMAD home directory. A sample is available in **ANNEX B - Samples configuration files**, it is highly recommended you use this to follow these instructions. This configuration file is divided into 3 sections:

- Main
- Inputs
- Outputs

Inside the **main section**, 1 parameter is available:

- `master_file`
 - Defines the path to your MDIDS-GT master file you saved in the step before
 - Accepts relative and absolute paths, without quotes
 - Example: if you saved your MDIDS file as *Jet-Turbine_v1* inside your MDIDS-NOMAD home directory, then the value of this parameter should be *Jet-Turbine_v1*

Inside the **inputs section**, you define your input variable names as sub-sections. The order in which you define your input variables in this file must be the same as the order used by the MDIDS-GT console application (take a look at the MDIDSGTconsole readme). Each input variable sub-section has two (2) parameters available:

- `sensitivity_range`
 - Sets the lower and upper bounds, separated by a comma, when generating the sensitivity matrix
 - Example: 30,70
- `sensitivity_npoints`
 - Sets the number of points to evaluate inside the range specified range
 - Example: 10

Inside the **outputs section**, you define your output variable names as sub-sections. The order in which you define your output variables in this file must be the same as the order used by MDIDS-GT console (take a look at the MDIDSGTconsole readme). Each output variable sub-section has one (1) parameter available:

- `target`
 - Sets the target value of this variable when running the optimization
 - Example: 33000

3. Writing the *params.txt* configuration file

The *params.txt* file contains information used by NOMAD to execute the optimization. This file should be named *params.txt* and it should be placed in your MDIDS-NOMAD home directory. A sample is available in **ANNEX B - Samples configuration files** (and in the *Samples* folder of your MDIDS-NOMAD home directory), it is highly recommended you use this to follow along these instructions. This configuration file can be made very complex, but we will discuss here only the bare minimum parameters you must define in order to get you started with MDIDS-NOMAD. **Note:** on a given line, all text preceded by a number sign (#) is ignored (aka commented).

BB_OUTPUT_TYPE

- **NOTE:** MDIDS-NOMAD does not currently support constraints, so this parameter should be set to “OBJ”, unless you made modifications to the source code.
- Specifies the **type** and the **order** of the BB program output
- Most basic types are OBJ (objective function) and CSTR (constraint). More output types are defined in reference [2].
- Example definitions:
 - 1 objective function: `BB_OUTPUT_TYPE OBJ`
 - 1 objective function and 2 constraints: `BB_OUTPUT_TYPE OBJ CSTR CSTR`
 - 1 objective function and 2 constraints: `BB_OUTPUT_TYPE CSTR OBJ CSTR`

DIMENSION

- Specifies the number of dimensions of the problem, aka the number of input variables

X0

- Specifies the starting point (aka initial value) of each variable
- The order of the values must be the same as the order you defined your input variables in *config.ini*
- Syntax: `(Var1, Var2, Var3, ...)`
- Example: For a problem with 2 variables, **a** and **b**, a starting point of **a=0** and **b=5** would be defined as `X0 (0 5)`

F_TARGET

- Specifies the relative error norm below which the optimization should stop.
- The relative error norm is calculated as a Euclidian norm of the relative error of every output variable: $L_2 = \sqrt{\sum e_i^2}$
- Example: We want to solve a given optimization problem with a relative error norm of at most 1%. This translates to `F_TARGET 0.01`

4. Running the optimization

Once you have completed the 3 above steps, you are now ready to execute MDIDS-NOMAD. To do this, go in your MDIDS-NOMAD home directory and double click on the file named *MDIDS-NOMAD.bat*.

A black window similar to the one in **Figure 2** - Example optimization output should appear and, once the execution is done, all the results will be saved under the *Results* folder. In there, you will find a history of every evaluation that was done by NOMAD during the optimization, and the sensitivity matrix (both in table and graph form). Additionally, you will find the solution in the text file *solution.txt*.

Advanced usage

In this section, we go into more details on the inner workings of MDIDS-NOMAD, we take a brief look at some of the more advanced NOMAD parameters and give you some advices on how to optimize a problem with better success.

Scripts' role and usage

MDIDS-NOMAD is comprised of 4 major scripts (excluding libraries), each having a specific purpose. Here is a quick summary of each script's role:

- **wrapper.py**: translates the messages between NOMAD and MDIDS
- **postprocess.py**: generate the evaluations history table and graph and save the solution
- **sensitivity.py**: generate the sensitivity matrix tables and graphs
- **MDIDS-NOMAD.bat**: convenience script that automatically executes NOMAD, *postprocess.py* and *sensitivity.py*

Note: to use any python script manually (files that end with *.py*), you must do so from a command prompt. To open a command prompt, simply execute the *cmd_here.bat* script by double clicking on it.

wrapper.py

This script is the wrapper program that is responsible of translating the messages sent between MDIDS and NOMAD. More specifically, this is what happens under the hood:

- Serve as a “dummy” BB program for NOMAD
- Read the **master_file** and **target** of each output variable of the MDIDS-NOMAD configuration file
- Execute MDIDSGTconsole in optimizer mode by passing to it the master_file and the NOMAD input file

- Append the results to the log file for future use by **postprocess.py** (by default, the log file is *Logs/MDIDS-NOMAD_results.log*)
- Calculate the Euclidian norm of the relative errors of the results generated by MDIDS
- Display the relative error norm to NOMAD

This script should not really be used manually, because it was designed in the first place to be used by NOMAD. However, it is still possible to do so for testing purposes, and here are some guidelines on how to use this script:

- The basic syntax is: `python wrapper.py INPUT_FILE`
 - The input file has the same structure as the NOMAD input files
- A few flags can be added to modify the behaviour of the script:
 - `-ns` or `--no-save`: Do not save the results in the log file
 - `-log LOG`: Path of the log file where to append results

If MDIDS-GT console ever comes to be updated, it is in this file on line #38 you should modify the MDIDSGTconsole execution command.

postprocess.py

This script is automatically run by **MDIDS-NOMAD.bat** after the optimization run, and since the results are saved during the execution of the script, it isn't strictly necessary to execute this script. However, it can be useful to run this program to regenerate the results of a previously archived optimization run. Here are some guidelines on how to use this script:

- Assuming all the necessary files are in the default location, the basic syntax is simply: `python postprocess.py`
- The default file locations are the following:

NOMAD's SOLUTION_FILE	<i>Logs/NOMAD_solution.0.log</i>
NOMAD's HISTORY_FILE	<i>Logs/NOMAD_history.0.log</i>
wrapper's results log file	<i>Logs/MDIDS-NOMAD_results.log</i>
MDIDS-NOMAD config file	<i>config.ini</i>
Export directory	<i>Results</i>

- A few optional flags can be added to modify the behaviour of the script:
 - `-ns` or `--no-save`: Do not export any results
 - `-sg` or `--show-graph`: Show the graphs at the end of the script
 - `-e` or `--read-exported`: Read the exported results instead of log results
 - `--config [PATH]`: path of the MDIDS-NOMAD configuration file
 - `--solution [PATH]`: path of NOMAD's SOLUTION_FILE
 - `--history [PATH]`: path of NOMAD's HISTORY_FILE
 - `--mdids-results [PATH]`: path of the wrapper's results log file
 - `--export [PATH]`: path of the directory where the results will be exported
 - `-d` or `--delimiter [CHAR]`: delimiter used when generating csv file (default: ";")
 - `-size` or `--figure-size [SIZE]`: size of the figures (unknown unit, try it yourself)

sensitivity.py

This script reads the information in the configuration file and generates a sensitivity matrix of the given problem as a list of graphs. Here's what happens under the hood:

- Read the **master_file**, the **sensitivity_range** and **sensitivity_npoints** of each input variable and the names of all variables in the MDIDS-NOMAD configuration file
- Execute MDIDSGTconsole in optimizer mode for every combination of input values defined in the configuration file
- Read the *MDIDS-NOMAD_sensitivity.log* file generated by [wrapper.py](#)
- Generate the sensitivity matrix as a table and graph for each output variable

This script is also automatically run by [MDIDS-NOMAD.bat](#) after the optimization run. However, it can be useful to only generate the sensitivity matrix without having to go through the whole optimization process. Here are some guidelines on how to use this script manually:

- Assuming all the necessary files are in the default location, the basic syntax is simply:
`python sensitivity.py`
- The default file locations are the following:

MDIDS-NOMAD config file	<i>config.ini</i>
Export directory	<i>Results</i>

- A few optional flags can be added to modify the behaviour of the script:
 - `-ns` or `--no-save`: Do not export any results
 - `-sg` or `--show-graph`: Show the graphs at the end of the script
 - `--config [PATH]`: path of the MDIDS-NOMAD configuration file
 - `--export [PATH]`: path of the directory where the results will be exported
 - `-d` or `--delimiter [CHAR]`: delimiter used when generating csv file (default: ";")
 - `-size` or `--figure-size [SIZE]`: size of the figures (unknown unit, try it yourself)

MDIDS-NOMAD.bat

This is the main script which automatically makes all the necessary calls to the right scripts and programs. It helps speed up the process of running an MDIDS-NOMAD optimization problem by avoiding typing each command manually. Here's what this script does:

- Delete the *MDIDS-NOMAD_results.log* file to log a new optimization run
- Execute NOMAD with *params.txt* as an argument and measure the optimization time
- Execute [postprocess.py](#) to process the raw results
- Execute [sensitivity.py](#) to generate the sensitivity matrix

The *MDIDS-NOMAD.bat* script must be executed manually by the user by simply double-clicking on it.

Here are some guidelines on how you can modify this script:

- You can modify the behavior of *postprocess.py* and *sensitivity.py* by adding the flags you want to the end of lines 22 and 26
- You can skip the execution of *postprocess.py* by commenting out line 22 (add `::` to the beginning of the line)
- You can skip the execution of *sensitivity.py* by commenting out line 26 (add `::` to the beginning of the line)

Advanced NOMAD parameters

In this section we will discuss how to use certain interesting NOMAD parameters (the ones in *params.txt*). If you want to read on the full list of the available NOMAD parameters, see reference [2].

Input variables specifications

NOMAD uses the same general syntax for defining input variable parameters. Multiple statements of the same parameter can be written, but if one variable's parameter has already been set by a previous statement, it will be overwritten. Some optimization problems have physical constraints that must be specified via these parameters to help NOMAD find an optimal solution. However, **when possible**, the user should specify as many parameters as he can to speed up the optimization. Parameters can be defined using the vector notation or the range notation. Alternatively, some parameters allow the path of a text file as an argument, where the parameter definitions are written.

Vector notation: Suppose we have a problem with 5 variables. In that case the syntax for the vector notation to define a given parameter *PARAMETER* for each variable individually would be `PARAMETER (Param1 Param2 Param3 Param4 Param5)`

Range notation: Suppose we have a problem with 5 variables. In that case the syntax for the range notation to define a given parameter *PARAMETER* for the variables 3, 4 and 5 would be `PARAMETER 2-4 Param` (note that the **1st** variable has **index 0**). Here, variables 1 and 2 would keep their default values for this parameter, unless another statement involving them is given. An asterisk (*) can be used as range to define **ALL** variables.

File path notation: The syntax for the file path notation is very simple: `PARAMETER file_path`. The given parameter *PARAMETER* then must be defined for each variable individually in the text file in a space separated list, such as `Param1 Param2 Param3 Param4 Param5` (or line-break separated list)

Setting a parameter to `-` is equivalent to “unchanged”.

LOWER_BOUND

- Specifies the variables' smallest value that NOMAD is allowed to evaluate

- Treated as greater than or equal to (\geq)
- Default value is `-inf`
- Some problems have physical bounds, but bounds can also be used to speed up the convergence.
- Allows file path notation

UPPER_BOUND

- Specifies the variables' largest value that NOMAD is allowed to evaluate
- Treated as less than or equal to (\leq)
- Default value is `inf` or `+inf`
- Some problems have physical bounds, but bounds can also be used to speed up the convergence.
- Allows file path notation

BB_INPUT_TYPE

- Defines the type of the input variable. Basic types are
 - B: Binary
 - I: Integer
 - R: Real
- A forth type, categorical (C), exists, but hasn't been studied yet

FIXED_VARIABLE

- Fixes the value of a variable to some specific value
- Useful when dealing with a problem that has many variables. By fixing the value of some of the variables it is easier to analyse the results
- Especially useful for detecting an under-specified problem (see [Problem specification](#))
- A variable's value can only be defined once. For example, if we define `X0 (0 1 2)` and `FIXED_VARIABLE 0 2`, in this case the first variable's value has been defined twice and NOMAD will throw an error. **Either** replace X0 by `X0 (- 1 2)` **or** `FIXED_VARIABLE` by `FIXED_VARIABLE 0`
- Allows file path notation

GRANULARITY

- Defines the last digit precision of a given variable (not the number of significant digits)
- Even though the exact solution could be more precise than the given granularity, NOMAD will not bother looking for a more precise solution
- Examples:
 - `GRANULARITY 1`: only integers will be allowed
 - `GRANULARITY 0.001`: only numbers with at most 3 decimal places will be allowed
 - `GRANULARITY 1000`: only thousands will be allowed

Stop criteria

NOMAD needs a way to know when to stop optimizing. If no stop criterion is given or if NOMAD can't meet the given criteria, then NOMAD will simply stop when it can no longer refine the mesh of the MADS algorithm.

F_TARGET

- Specifies the value below which the objective function is considered “optimized” and NOMAD should stop.

MAX_BB_EVAL

- Specifies the maximum amount of BB evaluation to be done

MAX_TIME

- Specifies the maximum wall-clock time to run the optimization, in seconds.

Results handling

The following parameters determine how NOMAD presents the results of the optimization run.

DISPLAY_STATS

- Formats the outputs of every displayed BB evaluation
- Most useful arguments are (complete list available in reference [2]):
 - BBE: Number of black box evaluations
 - BBO: Black box output
 - OBJ: Objective function value
 - SOL: Current iteration variable values
 - TIME: Elapsed time in seconds
 - VARI: Value of variable i (starting at 0)
- All outputs may be formatted using C style, but it is not mandatory:
 - Available specifiers are: %f, %e, %E, %g, %G, %i and %d
 - The “%” character may be explicitly indicated with “\%”
 - Example: `DISPLAY_STATS BBE: f(%.3fSOL)=%.3fOBJ`
 - More information available in reference [3]
- Default: `DISPLAY_STATS BBE OBJ`

DISPLAY_ALL_EVAL

- Force NOMAD to display all evaluations, regardless of if it improves the current best solution or not (by default NOMAD only displays the improving evaluations)

DISPLAY_DEGREE

- Determines how much information will be displayed on the screen during the optimization run
- 4 degrees of display are available:
 - 0: No display

- 1: Display stats only
- 2: Normal mode
- 3: Debugger mode
- Note that the more information is displayed, the slower the optimization will be, so this parameter should be set to 0 or 1 once the optimization problem has been well defined
- Default: 2

SOLUTION_FILE

- Specifies the path to a text file where NOMAD will save the optimal solution
- Each variable's value will be written on a separate line
- If you modify this parameter, you need to specify the new path to `postprocess.py` by using the `--solution` flag

HISTORY_FILE

- Specifies the path to a text file where NOMAD will save a history of all trial points
- The saved data includes the values of:
 - Variable(s)
 - Objective function(s)
 - Constraint(s)
- If you modify this parameter, you need to specify the new path to `postprocess.py` by using the `--history` flag

STATS_FILE

- Very similar to `DISPLAY_STATS`, but the stats will be saved to a file instead of being displayed on the terminal (note that `DISPLAY_STATS` doesn't affect `STATS_FILE`'s output format)
- The 1st argument is the path to a text file where NOMAD will save the stats
- The 2nd argument is a string following the same logic as `DISPLAY_STATS`
- If only the path is given, then the stats will be saved using the default format `BBE OBJ`
- Example: `DISPLAY_STATS stats.txt BBE: f(%.3fSOL)=%.3fOBJ`

Optimization advices

MDIDS-NOMAD *should* be able to solve any well defined optimization problem simply by following the [Getting started](#) instructions. However, here are some advices on things to look for when solving an optimization problem with MDIDS-NOMAD and on how to speed up the optimization.

Granularity

The `GRANULARITY` parameter is very useful to force NOMAD into exploring the effects of all the input variables, rather than optimizing a single variable at a time. By setting the granularity of the input variables to a decent value (e.g. 3 significant digits), this will prevent NOMAD from evaluating values with too little of a difference from each other. Furthermore, it is sometimes useless to have

a solution down to the 16th decimal place, as it greatly decreases performance. This parameter is greatly recommended.

Problem specification

As you may already know, a problem that is under-specified has an infinite number of solutions, and in that case NOMAD may return unexpected results. Therefore, it is very important to first determine if the problem is fully specified or not. We will show how to determine if a problem is under-specified or not via an example.

In this example, we will be using the sample file *EngineV2500-Master*. We will try to optimize (SFC, OPR and thrust) for a given combination of (Work_Fan, Work_Boost, Work_HPC and T4_Target). We have no idea if this optimization problem is fully specified or not, but we will find out with the following example. First, we set the NOMAD starting point to the following arbitrary value:

Variable	Value
Work_Fan	18
Work_Boost	30
Work_HPC	150
T4_Target	2250

The execution of MDIDS-NOMAD returns the following solution (less than 0.0001% relative error norm):

Input		Output			
Variable	Value	Variable	Target	Value	Error (%)
Work_Fan	13.3752	SFC	0.37	0.37	0
Work_Boost	30.6627	OPR	33.4	33.4	0
Work_HPC	170.61	Thrust	33000	33000	0
T4_Target	2618.03				

Then, we execute MDIDS-NOMAD again, but this time with another arbitrary starting point:

Variable	Value
Work_Fan	30
Work_Boost	50
Work_HPC	100
T4_Target	3000

The execution of MDIDS-NOMAD returns the following solution:

Input		Output			
Variable	Value	Variable	Target	Value	Error (%)
Work_Fan	13.3709	SFC	0.37	0.37	0
Work_Boost	50.6204	OPR	33.4	33.4	0
Work_HPC	150.673	Thrust	33000	33000	0
T4_Target	2618.52				

If there was a unique solution (aka the problem was fully specified), then the starting point should not matter. In this case, we see that by modifying the starting point, the **fan work** and the **T4 target** converge towards a similar value (less than 0.03% difference), but the **boost work** and the **HPC work** resulted in totally different values (more than 13.2% difference). This leads to think that the optimization problem is underspecified regarding the **boost work** and the **HPC work**. Let us prove this.

By fixing the value of the boost work, we can run MDIDS-NOMAD to calculate the HPC work. If the problem is fully specified, then we should be able to fix the calculated value of the HPC work and let MDIDS-NOMAD find the value of the boost work we fixed earlier. Let's try with the following starting point:

Variable	Value
Work_Fan	30
(fixed) Work_Boost	40
Work_HPC	100
T4_Target	3000

The execution of MDIDS-NOMAD gives the following results:

Input		Output			
Variable	Value	Variable	Target	Value	Error (%)
Work_Fan	13.3721	SFC	0.37	0.37	0
Work_Boost	40	OPR	33.4	33.4	0
Work_HPC	161.26895	Thrust	33000	33000	0
T4_Target	2618.3525				

Now we set the following **different starting point**, but where the **HPC work** is the same as the last result:

Variable	Value
Work_Fan	10
Work_Boost	80
(fixed) Work_HPC	161.26895
T4_Target	2250

The execution of MDIDS-NOMAD gives the following results:

Input		Output			
Variable	Value	Variable	Target	Value	Error (%)
Work_Fan	13.378462	SFC	0.37	0.37	0
Work_Boost	39.984392	OPR	33.4	33.4	0
Work_HPC	161.26895	Thrust	33000	33000	0
T4_Target	2617.6633				

As we can see, both results are very similar, with an error of less than 0.05%. This error can probably be explained by the fact that MDIDS-GT console currently returns values with 3 significant digits for SFC and OPR, so this will be something that will need improving. Therefore, in this example, the problem is under-specified, and we need to fix either one of the boost work or the HPC work value to fully specify the problem.

It should be noted that it is unknown how MDIDS-NOMAD would react if it was presented with a problem that was over-specified.

ANNEX A - Setup screenshots

NOMAD

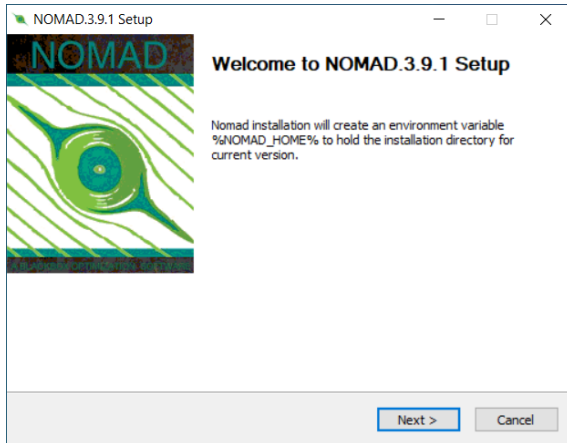


Figure 4 - NOMAD Setup (Step 1)

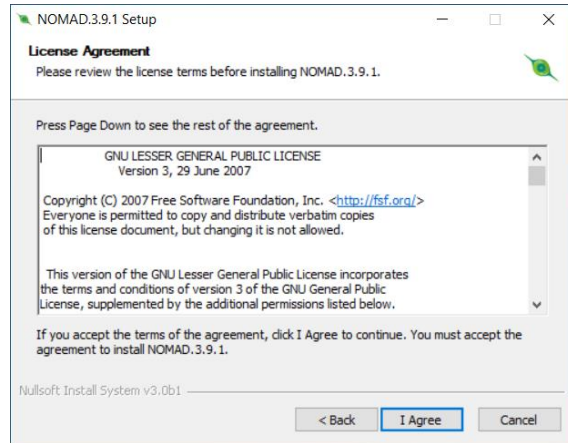


Figure 5 - NOMAD Setup (Step 2)

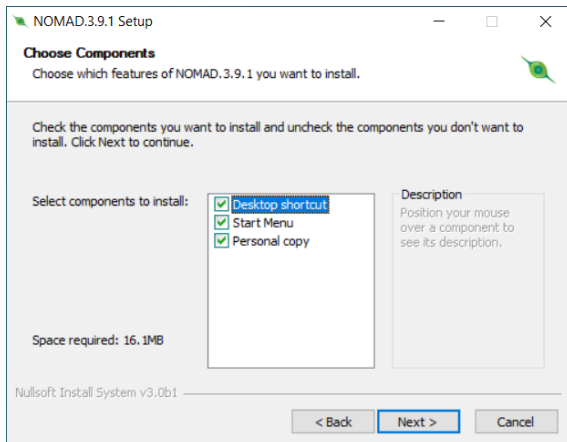


Figure 6 - NOMAD Setup (Step 3)

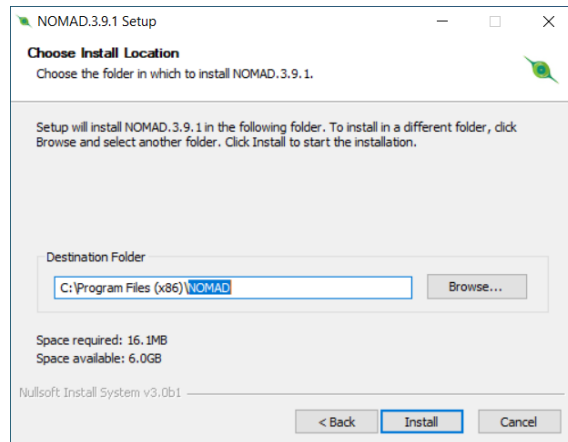


Figure 7 - NOMAD Setup (Step 4)

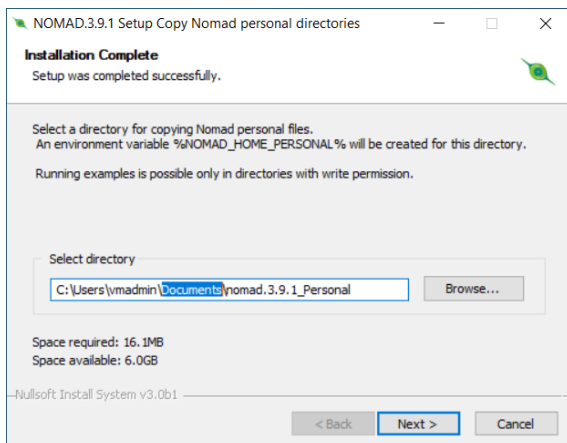


Figure 8 - NOMAD Setup (Step 5)

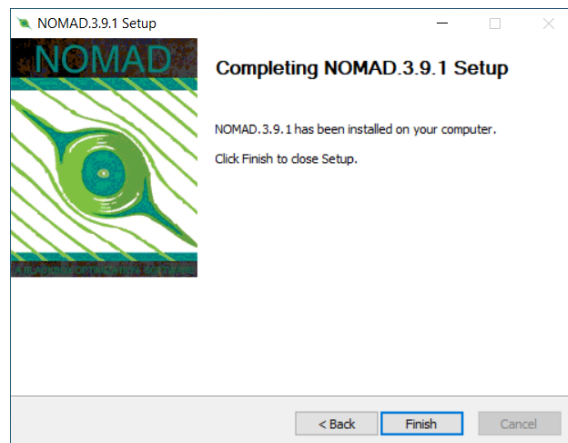


Figure 9 - NOMAD Setup (Step 6)

Python

Procedure #1

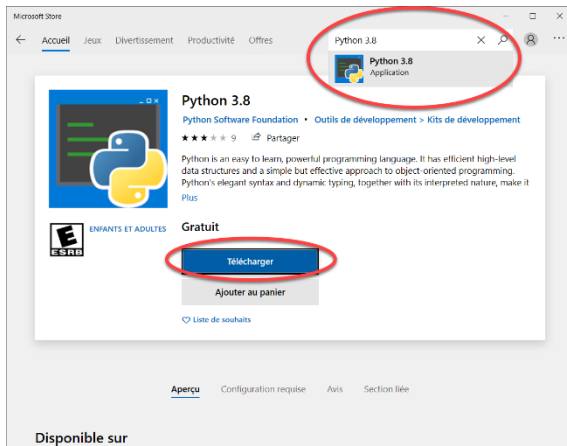


Figure 10 - Python Setup #1 (Step 2)

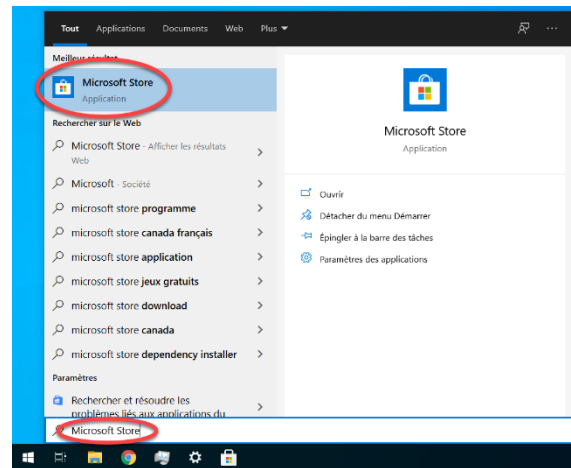


Figure 11 - Python Setup #1 (Step 1)

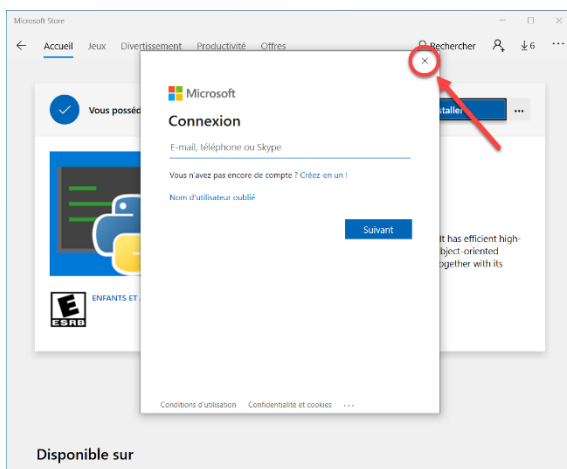


Figure 12 - Python Setup #1 (Step 3)

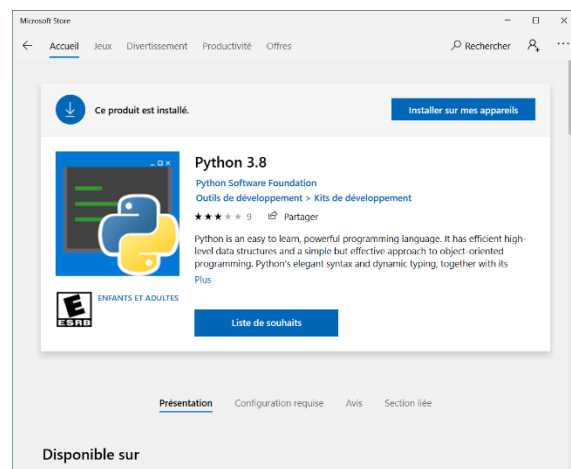


Figure 13 - Python Setup #1 (Step 4)

Procedure #2

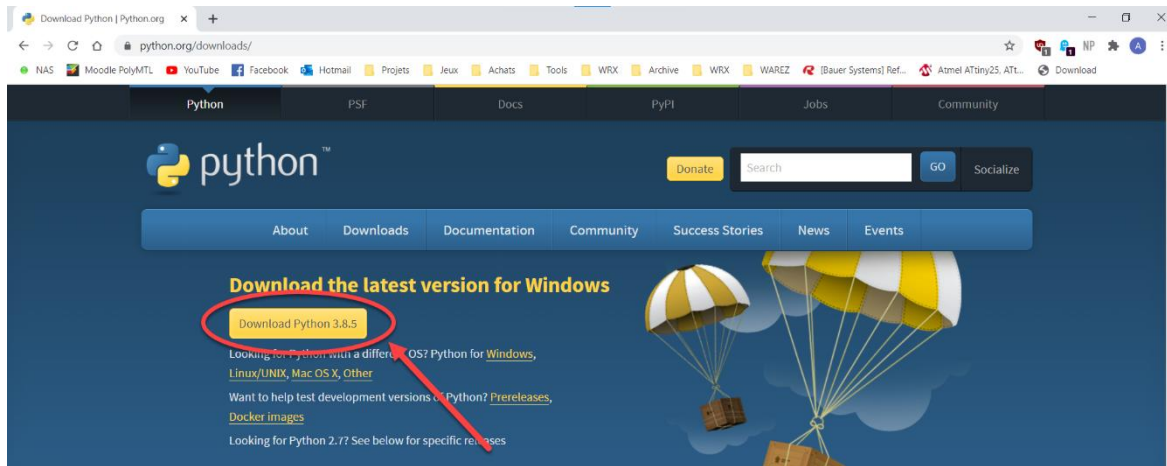


Figure 14 - Python Setup #2 (Step 1)

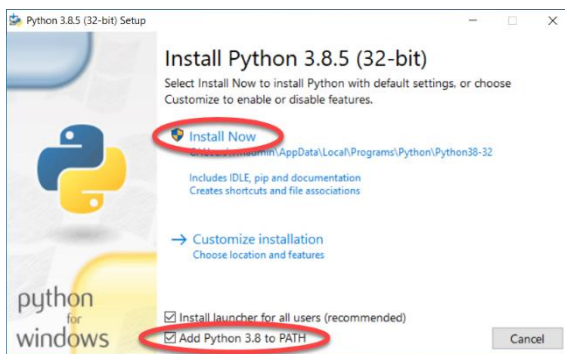


Figure 15 - Python Setup #2 (Step 2)

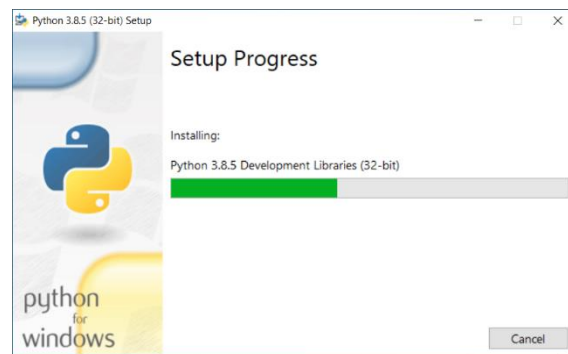


Figure 16 - Python Setup #2 (Step 3)

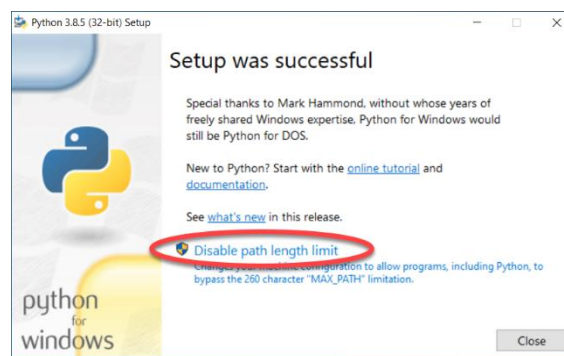


Figure 17 - Python Setup #2 (Step 4)

ANNEX B - Samples configuration files

These sample configuration files are also available in the *Samples* folder of your MDIDS-NOMAD home directory. The values in bold are those you should modify according to your specific problem.

config.ini - MDIDS-NOMAD configuration file

```
[main]
master_file = PATH_TO_MDIDS_SAVE_FILE

[inputs]
[NAME_OF_INPUT_VARIABLE]
sensitivity_range = LOWER_BOUND, UPPER_BOUND
sensitivity_npoints = NUMBER_OF_POINTS

[outputs]
[NAME_OF_OUTPUT_VARIABLE]
target = TARGET_VALUE
```

params.txt - NOMAD configuration file

```
# Mandatory parameters
BB_OUTPUT_TYPE          OBJ
DIMENSION                NUMBER_OF_INPUT_VARIABLES
X0                       ( Var0 Var1 Var2 ... )
F_TARGET                 RELATIVE_ERROR_NORM

BB_EXE                   "$python.exe wrapper.py"
DISPLAY_STATS            BBE ( SOL ) OBJ
DISPLAY_DEGREE           1
HISTORY_FILE             Logs/NOMAD_history.log
SOLUTION_FILE            Logs/NOMAD_solution.log

# Incomplete list of optional parameters:
#LOWER_BOUND             * 0
#UPPER_BOUND             * 1000
#BB_INPUT_TYPE           * R
#FIXED_VARIABLE          1
#GRANULARITY             * 0.001

#MAX_BB_EVAL             200
#MAX_TIME                60

#DISPLAY_ALL_EVAL        1
#STATS_FILE              Logs/NOMAD_stats.log
```

References

- [1] S. Le Digabel, "NOMAD: Nonlinear optimization with the MADS algorithm," *ACM Trans. Math. Softw.*, vol. 37, 44, 2011.
- [2] C. T. Sébastien Le Digabel, Viviane Rochon Montplaisir, Charles Audet, "NOMAD User Guide Version 3.9.1," July 18, 2018.
- [3] "printf." <http://www.cplusplus.com/reference/cstdio/printf/> (accessed August 29th, 2020).