DOCUMENTATION OF PYTHON CODES
A Complete Guide

Georgia Kioselaki
gkioselaki@auth.gr

Introduction

The present file gives information about codes that are used for the fitting of the Dose response and the resulted supralinearity index f(D) curves. Four scripts have been developed, two for the OTOR and two for the TTOR model. For each model, one code is suitable for the fitting of the dose response curves, and the second for the supralinearity index f(D) as a function of dose. To fit the curve of the experimental data with analytical expression of these models, a procedural programming approach within a Python environment is used. Python is under an open-source license, and it can run on Mac OS X, Windows and Linux.
In this file the examples are given using the TTOR model. The same procedure is also followed in the OTOR model.

User Guide

1) In order to run the codes, Python 3 and a suitable, free and open-source Scientific Python Development Environment, like Spyder, have to be installed in the user's computer. Install also the mandatory libraries through the command *pip install* in the command window.

a) Download Pyrhon3 https://www.python.org/downloads/

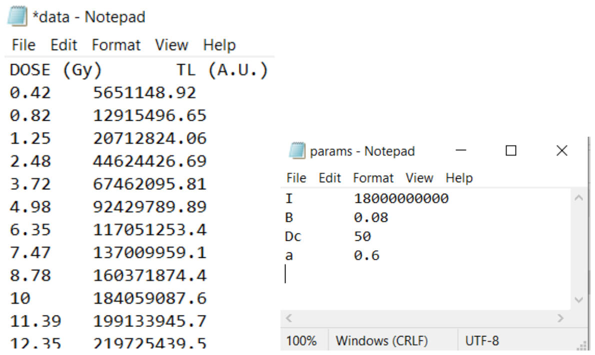
b) Download Spyder https://www.spyder-ide.org/

c) The necessary libraries are listed below:

- ***NumPy***: It is a numerical library, used for working with multi-dimensional arrays. NumPy also has functions for working in domain of linear algebra and matrices.

- ***SciPy***: Useful to manipulate the data and visualize the data using a wide range of high-level commands. This library contains all the algebraic functions, such as lambert W, some of which are there in NumPy to some extent and not in full-fledged form. In this project, SciPy has been used also for the optimization.
- ***Matplotlib***: It is a graphical plotting and data visualization library.
- ***Os***: provides functions for creating a directory (folder)
- ***tkinter***: provides a fast way to create GUI applications

2) Before running the program, the user should create a .txt file with name *data*, containing the experimental data as the fig. 1a shows. In the first row the name of the measured quantities with their units are written. The two columns must include the values of the dose and the intensity, respectively. The user should also create a .txt file, which is called *params* and includes the initial parameter values, based on the used model. As the figure 1b shows, in the first column the symbols of the required TTOR parameters are written, while in the second the user types a guess value for each one of them. Please, do not forget the bounds of the parameter values (0<R<1, Dc>0 for the OTOR model and B>0, Dc>0 and 0<a<1 when the TTOR model is used). These two files must be in the same folder with the python script that is to be used. The program is ready to run!

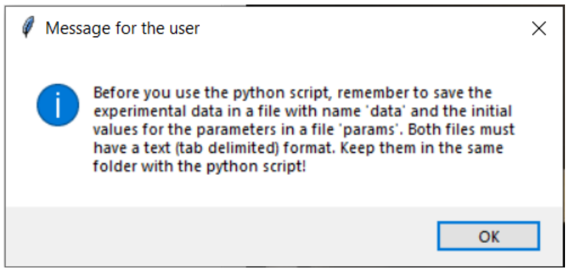

*data - Notepad
File Edit Format View Help
params - Notepad
File Edit Format View Help
100% Windows (CRLF) UTF-8

```
DOSE (Gy)       TL (A.U.)
0.42     5651148.92
0.82     12915496.65
1.25     20712824.06
2.48     44624426.69
3.72     67462095.81
4.98     92429789.89
6.35     117051253.4
7.47     137009959.1
8.78     160371874.4
10       184059087.6
11.39    199133945.7
12.35    219725439.5
```

```
I       18000000000
B       0.08
Dc      50
a       0.6
```

Fig. 1. *(a) Example of the .txt file contains the experimental data; (b) Example of the .txt file that includes some guess parameter values for the TTOR model.*

3) Open the code through the Scientific Python Development Environment (File-> Open-> TTOR). By running the program, a message for the user is appeared in the screen (Fig.2) and gives information about the creation of files mentioned in the second step. Click OK to continue.

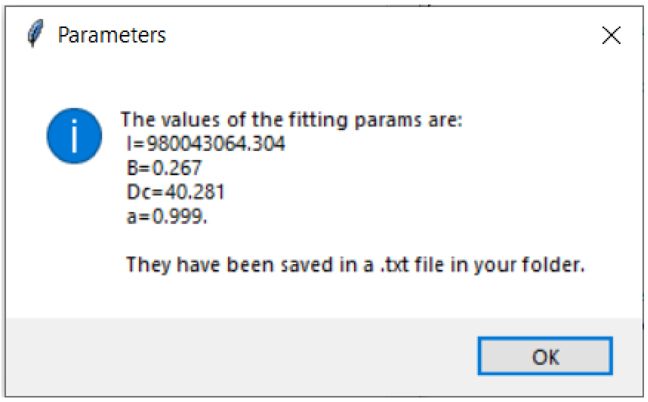


Before you use the python script, remember to save the experimental data in a file with name 'data' and the initial values for the parameters in a file 'params'. Both files must have a text (tab delimited) format. Keep them in the same folder with the python script!

Fig.2. Pop-up window that it is appeared in the user's screen.

4) The program optimizes the data with the defined scientific equation, using the command curve_fit from the SciPy library. The bounds for the dependent parameters are already set in the package as the theory predicts, and no user's action is required in this step.

5) Following this, in the user's screen it is appeared a new window with the values of the fitting parameters (Fig.3). Two new .txt files with names *Fitting Params* and *Theoretical curve* are created in the folder. Click OK to continue.



```
The values of the fitting params are:
 I=980043064.304
 B=0.267
 Dc=40.281
 a=0.999.

 They have been saved in a .txt file in your folder.
```

Fig.3. Pop-up window with the values of the fitting parameters.

6) The plot which illustrates the fitted experimental with the theoretical curve, and the FOM is follow (Fig.4). A new .png file with name *plot* is created in the folder. Click 'x'.

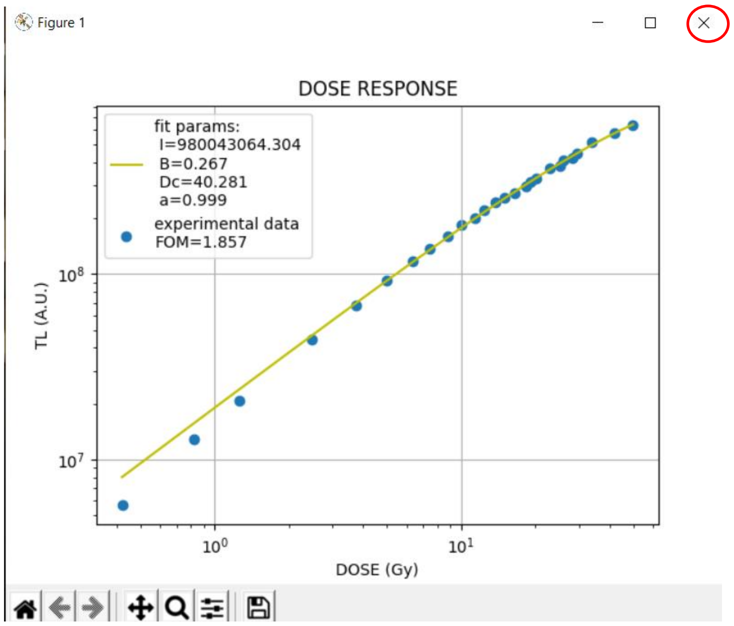
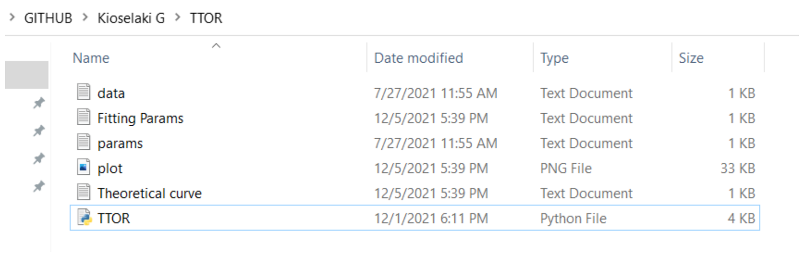


Fig.4. Pop-up window with the fitted plot.

7) The code automatically stores the data from the theoretical curve and the fitting parameters after the optimization in a .txt output file, as well as the figure in a .png file, in the same folder with the used Python script (Fig. 5).



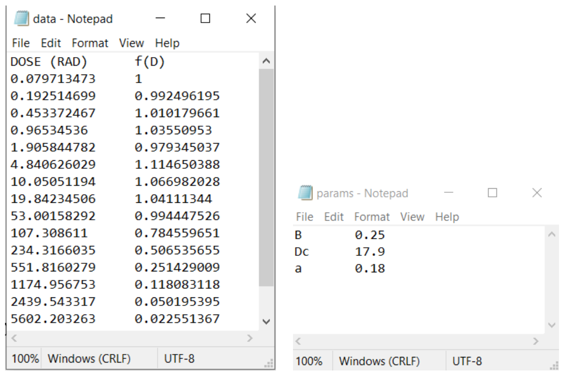
› GITHUB › Kioselaki G › TTOR

Name	Date modified	Type	Size
data	7/27/2021 11:55 AM	Text Document	1 KB
Fitting Params	12/5/2021 5:39 PM	Text Document	1 KB
params	7/27/2021 11:55 AM	Text Document	1 KB
plot	12/5/2021 5:39 PM	PNG File	33 KB
Theoretical curve	12/5/2021 5:39 PM	Text Document	1 KB
TTOR	12/1/2021 6:11 PM	Python File	4 KB

Fig.5. Output files

<u>Notes:</u>

1. Regarding the fitting of the supralinearity index f(D) curves, the structure of the input files (Step 2) is shown in figure 6. The other steps remain the same.


```
DOSE (RAD)        f(D)
0.079713473       1
0.192514699       0.992496195
0.453372467       1.010179661
0.96534536        1.03550953
1.905844782       0.979345037
4.840626029       1.114650388
10.05051194       1.066982028
19.84234506       1.04111344
53.00158292       0.994447526
107.308611        0.784559651
234.3166035       0.506535655
551.8160279       0.251429009
1174.956753       0.118083118
2439.543317       0.050195395
5602.203263       0.022551367
```

```
B        0.25
Dc       17.9
a        0.18
```
data - Notepad
File Edit Format View Help
100% Windows (CRLF) UTF-8
params - Notepad
File Edit Format View Help
100% Windows (CRLF) UTF-8

Fig. 6. *(a) Example of the .txt file contains the experimental data; (b) Example of the .txt file that includes some guess parameter values for the TTOR model.*

2. Use a different folder for each Python code and for every new fitting. Otherwise, every new output file from the code will be written on the previous one, with the same name.