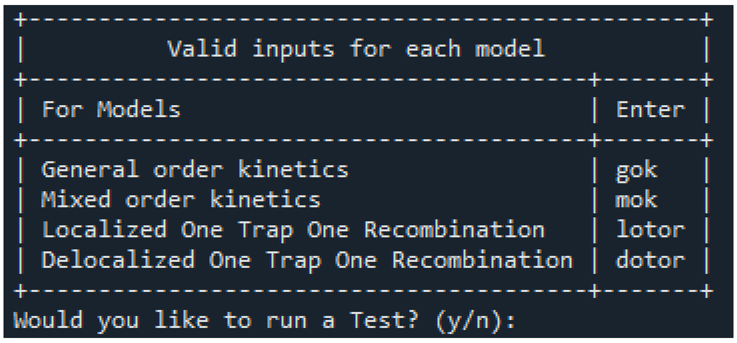
Main program execution

The main program consists of three separate libraries: **`PSM_functions`**, **`Tables`**, and **`main_program`**. Each library serves a distinct purpose. The **`PSM_functions`** library contains all the necessary relations to compute the Activation Energy for each model. The **`Tables`** library takes the results from the first library and presents them using the **`PrettyTable`** library. Finally, the **`main_program`** acts as the core, calling the other two libraries to compute the activation energy based on user inputs.

When running the main program, the user will encounter the following table, which lists the models along with some appropriate insertion keywords for each model.

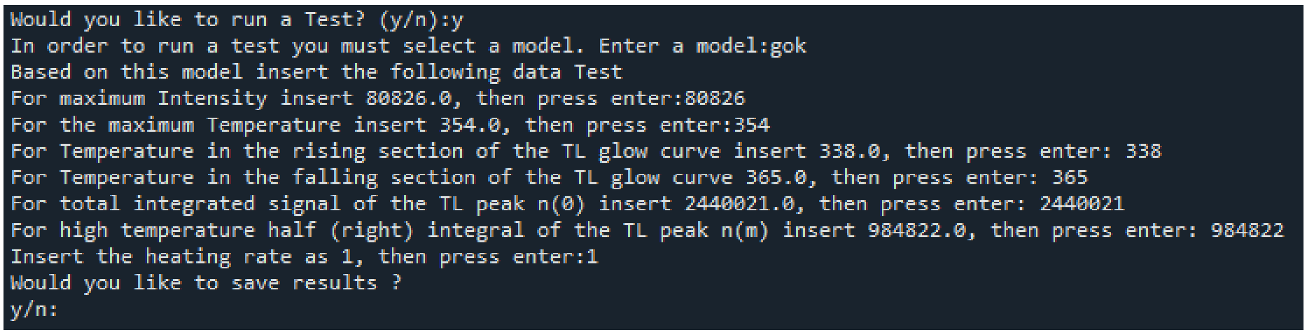


```
+-------------------------------------------------+
|           Valid inputs for each model           |
+---------------------------------------+---------+
| For Models                            | Enter   |
+---------------------------------------+---------+
| General order kinetics                | gok     |
| Mixed order kinetics                  | mok     |
| Localized One Trap One Recombination  | lotor   |
| Delocalized One Trap One Recombination | dotor  |
+---------------------------------------+---------+
Would you like to run a Test? (y/n):
```

To run a test, the user must enter “y” and press Enter to allow the program to recognize the test assignment. Otherwise, they can enter “n” to continue the computation based on experimental data. If the user enters “y”, they must select a model to examine the test data and then proceed with the following instructions. If the user enters “n”, the examination of experimental data begins, and they must select a model again to continue with the experimental data insertion.

<u>Test Run</u>

When a user runs a test, they must follow the instructions presented step by step to input the test data. Then, a message will appear: 'Would you like to save the results?', informing the user that they have the option to save the results in the directory where the program is operating.

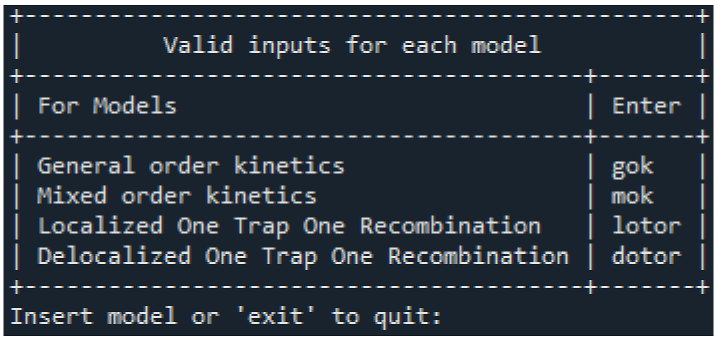


```
Would you like to run a Test? (y/n):y
In order to run a test you must select a model. Enter a model:gok
Based on this model insert the following data Test
For maximum Intensity insert 80826.0, then press enter:80826
For the maximum Temperature insert 354.0, then press enter:354
For Temperature in the rising section of the TL glow curve insert 338.0, then press enter: 338
For Temperature in the falling section of the TL glow curve 365.0, then press enter: 365
For total integrated signal of the TL peak n(0) insert 2440021.0, then press enter: 2440021
For high temperature half (right) integral of the TL peak n(m) insert 984822.0, then press enter: 984822
Insert the heating rate as 1, then press enter:1
Would you like to save results ?
y/n:
```

After this process the user may follow the exact same test for different models with the same inputs. If they don’t want to operate a test again they can select and enter “n” in order to proceed at the examination of experimental data.

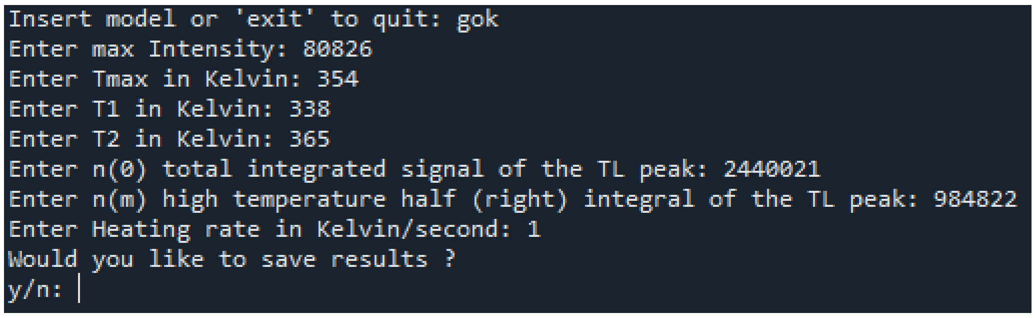
Experimental data examination

The process of the main program for experimental data examination is the same, but in this case the user must input already known and different data, after the model selection, for maximum Intensity, maximum Temperature, Temperature 1 and 2, the integrated parts such as the whole integrated signal of TL glow curve (n_0) and the right part of it (n_m) and finally the heating rate bita.



```
+-------------------------------------------------+
|           Valid inputs for each model           |
+-----------------------------------------+-------+
| For Models                              | Enter |
+-----------------------------------------+-------+
| General order kinetics                  | gok   |
| Mixed order kinetics                    | mok   |
| Localized One Trap One Recombination    | lotor |
| Delocalized One Trap One Recombination  | dotor |
+-----------------------------------------+-------+
Insert model or 'exit' to quit:
```

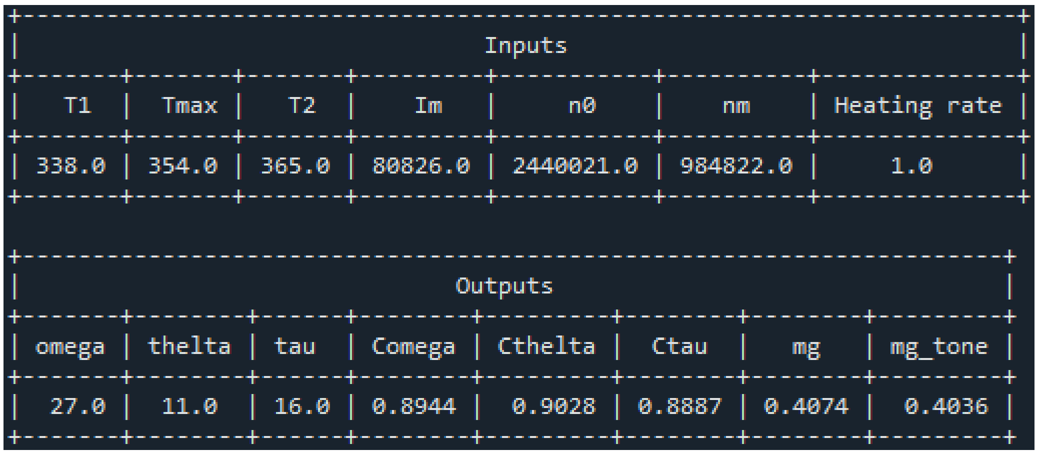
The following Table presents an example of how the process of insertion looks like with the data Test for General order kinetics model:



```
Insert model or 'exit' to quit: gok
Enter max Intensity: 80826
Enter Tmax in Kelvin: 354
Enter T1 in Kelvin: 338
Enter T2 in Kelvin: 365
Enter n(0) total integrated signal of the TL peak: 2440021
Enter n(m) high temperature half (right) integral of the TL peak: 984822
Enter Heating rate in Kelvin/second: 1
Would you like to save results ?
y/n: |
```

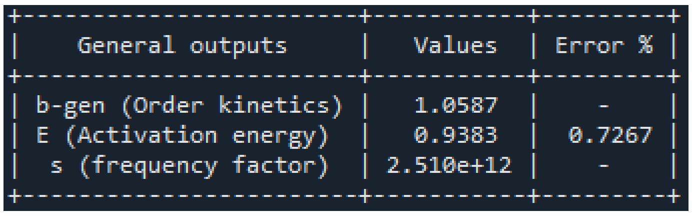
Results

The following Tables presents the user's Inputs data based on the gradually insertion process, alongside the Outputs that represents the Peak Shape Method coefficients such as the fwhm for each triangle assumptions the fwhm coefficients C_a with a=ω, δ, τ and the symmetry factors μ_g and μ_g'.



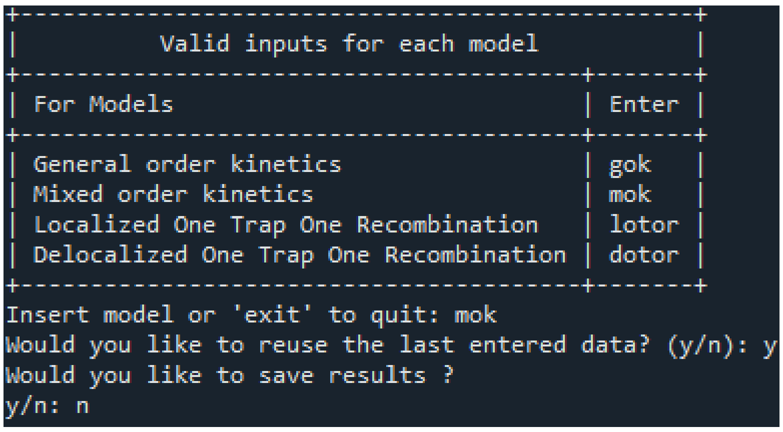
```
+-----------------------------------------------------------------------------+
|                                   Inputs                                    |
+-------+-------+-------+---------+-----------+----------+--------------+
|   T1  |  Tmax |   T2  |    Im   |     n0    |    nm    | Heating rate |
+-------+-------+-------+---------+-----------+----------+--------------+
| 338.0 | 354.0 | 365.0 | 80826.0 | 2440021.0 | 984822.0 |     1.0      |
+-------+-------+-------+---------+-----------+----------+--------------+


+------------------------------------------------------------------------+
|                                Outputs                                 |
+-------+--------+------+--------+---------+--------+--------+---------+
| omega | thelta | tau  | Comega | Cthelta |  Ctau  |   mg   | mg_tone |
+-------+--------+------+--------+---------+--------+--------+---------+
|  27.0 |  11.0  | 16.0 | 0.8944 |  0.9028 | 0.8887 | 0.4074 |  0.4036 |
+-------+--------+------+--------+---------+--------+--------+---------+
```

Below these Tables the results are presented for the activation energy and the evaluated kinetic parameters of each model.



```
+-----------------------+-----------+---------+
|    General outputs    |   Values  | Error % |
+-----------------------+-----------+---------+
| b-gen (Order kinetics) |   1.0587  |    -    |
| E (Activation energy)  |   0.9383  |  0.7267 |
|  s (frequency factor)  | 2.510e+12 |    -    |
+-----------------------+-----------+---------+
```

After examining a model, there is an option to reuse the previously inserted data for a different model to save time. This option will appear after selecting a new model. For example, after the initial data entry and computation using the GOK model, if the user selects the Mixed-Order Kinetics (MOK) model, the following message will appear:



```
+----------------------------------------------+
|          Valid inputs for each model         |
+--------------------------------------+-------+
| For Models                           | Enter |
+--------------------------------------+-------+
| General order kinetics               | gok   |
| Mixed order kinetics                 | mok   |
| Localized One Trap One Recombination | lotor |
| Delocalized One Trap One Recombination | dotor |
+--------------------------------------+-------+
Insert model or 'exit' to quit: mok
Would you like to reuse the last entered data? (y/n): y
Would you like to save results ?
y/n: n
```

By entering “y”, the user can examine results using the same inputs for different models, following the same saving process as before. If the user presses “n”, the data insertion process must be repeated step by step. This option is preferable when examining different experimental data.

PROGRAMS FUNCTION FOR WINDOWS

In the following section, the appropriate preparation for executing the program in different user environments will be presented. The program was developed using **Python 3.12.3**, installed through the conda-forge channel, which made setup and dependency management straightforward with Conda. This version was built using the **MSC v.1938** Microsoft C++ compiler for 64-bit systems, ensuring compatibility with modern operating systems. For a more interactive and user-friendly coding experience, I used **IPython 8.27.0**, which provided helpful features like syntax highlighting, auto-completion, and magic commands, making coding and debugging much easier. The entire coding process was done in the **Spyder IDE**, using the **miniconda3.1** package, and libraries were added either through the package manager or directly within Spyder.

All the libraries have been installed through the package manager with the following simple syntax: `conda install library` or with the syntax `!pip install library` through the Spyder’s console. If it must be installed a downgraded, an upgraded or a specific version of the libraries, this can be applied through the exact same environments with the syntax `conda update library` or `conda install library=version` and using the Spyder’s console through the syntax `!pip install --upgrade` `library` or `!pip install library==version`. The versions for each library will be presented in the Appendix section.

How to Install Modules in Spyder's Default Interpreter

If the user uses the default Spyder interpreter, some modules may not be included in certain versions. To install modules in this environment, the procedure requires more steps than the simple `!pip install` method. The user needs to find the path to the Isolated Virtual Environment, along with Spyder's default interpreter. This can be done using the `sys` module with the code `print(sys.executable)`. Once the path is located, the user can install the desired module by running the following command: `!C:\Spyder’s_environment_path -m pip install module`.

Modules and functions

In this section, the modules and functions used for each model will be presented, along with the final program and the versions of each library.

General order kinetics (Linear heating rate):

1 **scipy.special**:

- **expi**: This function computes the exponential integral, a special function used in mathematical modeling and analysis, particularly for solving integrals that involve exponential terms. It is useful in thermoluminescence calculations.

2 **numpy**:

- *** (all functions)**: Importing all functions from numpy allows for a wide range of numerical operations, including array handling, mathematical computations (e.g., exp, log), and linear algebra operations. numpy is essential for efficient data manipulation and numerical analysis.

3 **matplotlib.pyplot**:

- *** (all functions)**: Importing all functions from matplotlib.pyplot provides tools for creating static, animated, and interactive plots in Python. It is widely used for visualizing data, such as plotting thermoluminescence glow curves and other scientific data representations.

4 **prettytable**:

- **PrettyTable**: This class is used to create formatted tables in Python. It allows for easy presentation of data in tabular form, which is useful for displaying calculated values, results, and comparisons in a structured and readable way.

General order kinetics (EHF):

1. **`newton`** **from** **`scipy.optimize`**:

The newton function is used to find the root (i.e., zero) of a function. It applies a numerical method to iteratively approximate the value where a given function equals zero. The Newton-Raphson method is the most common approach for this, though it can also use secant methods when the derivative is not available.

2. **`trapezoid`** **from** **`scipy.integrate`**:

The trapezoid function performs numerical integration using the trapezoidal rule. The trapezoidal rule is a simple technique for estimating the area under a curve. It approximates the area by dividing the region under the curve into trapezoidal segments, rather than rectangles (as in the simpler Riemann sum approach).

Delocalized One Trap One Recombination

`wrightomega` from `scipy.special`:

The wrightomega function in the scipy.special module computes the **Wright Omega function**, which is a special function related to the Lambert W function. The Wright Omega function is used to solve equations of the form $y = xe^{x}$, and it is especially useful in mathematical physics, control theory, and other areas where transcendental equations appear. The Wright Omega function is used when solving equations where an unknown is both inside and outside of an exponential function, similar to the Lambert W function but with additional complexity.

Localized One Trap One Recombination

`minimize_scalar` from `scipy.optimize`:

The minimize_scalar function from scipy.optimize is used to minimize a scalar function of one variable. It provides several optimization methods (e.g., Brent's method, golden-section search) for finding the minimum value of a function within a given interval or with a specific initial guess.

Simulations

The simulations are consist of constants and initial parameters, which can be changed in order to simulate different TL glow curves for different initial values such as activation energy, frequency factor, order kinetics retrapping possibilities or heating rate.

Version of libraries

scipy version: 1.14.1

numpy version: 2.1.0

matplotlib version: 3.9.2

prettytable version: 3.12.0