TERESA model for the radiometric dating of recent sediments using ^{210}Pb and time marks. A Python software package.

J.M. Abril-Hernández

Departamento de Física Aplicada I, E.T.S.I. Agronómica, University of Seville (Spain)
ORCID: https://orcid.org/0000-0003-2540-5576 email: jmabril@us.es

1. Background on ^{210}Pb-based radiometric dating of recent sediments

The reader can find a review on the ^{210}Pb-based dating models for recent sediments in Abril-Hernández, 2025; https://doi.org/10.1016/j.jenvrad.2025.107749 (Open access).

The Time Estimates from Random Entries of Sediments and Activities (TERESA) model was first presented in the paper: Abril, 2016. J. Environ. Radioact., 151, pp. 64-74, 10.1016/j.jenvrad.2015.09.018. It is based on empirical evidence of that in natural aquatic sediments, initial activity concentrations of unsupported ^{210}Pb and sedimentation rates show random and independent variability, which can be roughly described by normal or log-normal distributions and which results in $^{210}Pb_{exc}$ fluxes being statistically correlated with SAR. This empirical evidence was shown in the paper: Abril and Brunskill, 2014. J. Paleolimnol., 52, pp. 121-137, 10.1007/s10933-014-9782-6.

The multimodal version of TERESA, along with a software package, written in Quick-Basic and presented as supplementary material, was published in the paper: Abril, 2020. Quat. Geochronol., 55, Article 101032, 10.1016/j.quageo.2019.101032.

TERESA is based on the generation of a large number (of the order of 10^6) solvers, each one containing the necessary parameters for generating samples of size N (the number of sediment slices in the core with the $^{210}Pb_{exc}$ profile) representing (log)normally-distributed values of initial activity concentrations and sedimentation rates, which are randomly grouped in pairs. The code includes a sorting algorithm that resolves the best arrangement of such pairs so that the modelled $^{210}Pb_{exc}$ profile that results best fits the empirical one. The quality of the fit is measured using an adjusted (to the degrees of freedom) χ-function. This function is computed for the entire set of ($\sim 10^6$) solvers, so that the absolute minimum can be resolved and presented as the model-solution. This solution includes the chronology and history (temporal sequences) of initial activities, sedimentation rates, and fluxes.

The parameters required to define a solver are: $\bar{A}_0 . \bar{w} . s_A . s_w$. They correspond to the arithmetic mean value (computed for all the sediment slices in the core with the $^{210}Pb_{exc}$ profile) of the initial activity concentration ($\bar{A}_0$) and sedimentation rate ($\bar{w}$), and their respective relative standard deviations. The model uses a library with two sets of size N with numbers that follow the 'canonical' normal typified distribution and that have been randomly sorted and grouped into pairs. When combined with the above set of four model parameters, the sets of N initial activity concentrations and sedimentation rates with normally distributed values are generated.

The above strategy can be easily adapted to situations with a constant sedimentation rate (this is, $s_w = 0$). This special case is known as the CSAR model. Similarly, the initial activity concentration can be assumed as constant (s_A=0), resulting in the χ-CIC model. The code for TERESA can also be easily adapted for a constant-flux model, the χ-CF model.

These sets of models, which share the common method of mapping a χ-function through a very large number of solvers to find the absolute minimum as the best solution, are referred to as 'χ-mapping' models. The set can be enriched, including different possibilities to define normal or Log-normal distributions, and to use alternative 'attractors', such as objective functions that involve the χ function combined with time marks.

The reader can find details in the following set of publications from this author:

J.M. Abril. Pb-dating of sediments with models assuming a constant flux: CFCS, CRS, PLUM, and the novel χ-mapping. Review, performance tests, and guidelines. J. Environ. Radioact. 268–269 (2023), Article 107248, 10.1016/j.jenvrad.2023.107248.

J.M. Abril. ^{210}Pb-based dating of recent sediments with the χ-mapping version of the Constant Sediment Accumulation Rate (CSAR) model. J. Environ. Radioact. 268–269 (2023), Article 107247, 10.1016/j.jenvrad.2023.107247.

J.M. Abril. ^{210}Pb-based dating of recent sediments with χ-mapping versions of the CFCS, CIC, CF and TERESA models. Quat. Geochronol., 79 (2024)101484, 10.1016/j.quageo.2023.101484.

J.M. Abril. ^{210}Pb-dating of recent sediments with the χ-mapping CF and CSAR models. On the attractors. J. Environ. Radioact., 270 (2023), Article 107314, 10.1016/j.jenvrad.2023.107314.

2. The software package

The set of codes solves the TERESA model using normal distributions of initial activity concentrations and sedimentation rates. Optionally, they can include time marks and alternative definitions of the attractor. It is assumed that users already know how to process their empirical data on $^{210}Pb_{exc}$ and can use the mass depth scale. Also assumed is that they are familiar with simple analytical models such as the Constant Flux with Constant Sedimentation (CFCS) model, which involves an exponential fit. The input information for TERESA is relatively simple in content and format, as shown in the following. The software has been conceived as a series of codes that are applied in sequence. Thus, the task that each one is solving can be easily understood in the software. The sequence is also justified since it is advisable to have critical supervision by the user and to iterate and refine some tasks. The codes are written in Python.

The final model outputs are stored in files that can be easily used by graphical software. Here we recommend Gnuplot (http://gnuplot.info/), but the user can adopt any other tool, or just using the graphical tools by Python. This is a common task in scientific research, and the aim here is to segregate it from the pure application of the TERESA model.

The other mentioned χ-mapping models and the use of log-normal distributions will be presented elsewhere as separated material. The present basic package includes:

Random_generator.py
TERESA_map.py
TERESA_clouds.py TERESA_clouds_ages.py
TERESA_plots.py TERESA_plot_ages.py
Data for an application case, as example: Core_C1.txt

The files can be placed in a new empty folder, and accessed with Visual Studio Code (VSC), or other working environments for Python.

The codes are not intended for commercial use, nor for any end user, so they do not prevent any possible malfunction. Instead, they use a simple programming strategy in Python and include many notes for guidance, so that users who are already familiar with the ^{210}Pb dating and have read some of the papers mentioned above on TERESA and χ-mapping models, can apply the model to their own dataset, following and understanding each step in the process. After that, users can customize the codes to adapt to their particular needs and uses.

No installation is required, but **Random_generator.py** must be run for the first time. It generates a library with the canonical representative samples of normal-typified distributions (mean = 0, sigma = 1) of size ranging from 5 up to 99 (or higher, if needed), labelled as z_0. It is copied twice in lists z_1 and z_2 and randomly sorted. The result is stored in a 2-column text files named "aleat_N.txt"

The output files will be created in the same folder where the code is actually placed. It is suggested to create then a sub-folder named "aleat" and store all the aleat_N files into it. Alternatively, the folder can be created first, placing and running **Random_generator.py** into it.

This will be your library from where the other codes will read the necessary information. Note that your library is "unique" since, although all the users will share the same z_0, the random rearrangement can be different from one user to another. Using a library ensure the repeatability of computations and separating the effect of random sorting from the variability in the distributions of physical magnitudes A_0 and SAR.

You can use the folder /aleat> for all your application cases without the need of running Random_generator again, or you can create different libraries if you are particularly interested in testing the effect in chronologies of the random sorting in z_1 and z_2.

Core_C1.txt is a text file containing the basic empirical data of the core consisting of: i) the mass depth of the slice, referred to its bottom, in units of g cm^{-2}; ii) the ^{210}Pb$_{exc}$ mass activity concentration (in units of Bq kg^{-1}, and typically found as the total ^{210}Pb minus the ^{226}Ra mass activity concentrations) and iii) the associated uncertainty (also in Bq kg^{-1}).

Note that TERESA does not need the complete recovery of the ^{210}Pb$_{exc}$ inventory, so the last measured slice may contain ^{10}Pb$_{exc}$ values well above zero.

TERESA needs a continuous record, so if some slices were not measured, you need estimate the missing values, typically through interpolations. This pre-treatment of the data is not included in the software, and the user can do it with Excel or by other means. The starting point for TERESA is this text file, with a 3-coloumn format separated by space or tabulation, as shown in the figure below.

Of course, you can use different names for your cores, but you need updating the information in the TERESA codes, as commented below. If you adopt the name **Core_C1.txt** for your data file, then you can run the models in sequence without need of paying attention to this point.

This simple format has the advantage of being directly read and plotted by graphical software such as Gnuplot.

Note that in ^{210}Pb-dating models we use two mass depth scales, one referring to the midpoint of each sediment sliced (e.g., used for plotting $^{210}Pb_{exc}$ versus mass depth profiles and applying the CF-CS model) and another referred to the bottom of the slice (e.g., used to estimate the ages with the CRS model). The software TERESA interprets the mass depth scale referred to the bottom of the slice, as above commented, and from it, the code generates the second mass depth scale referred to the midpoint of the slice, for using the appropriate one in each step of the calculations.

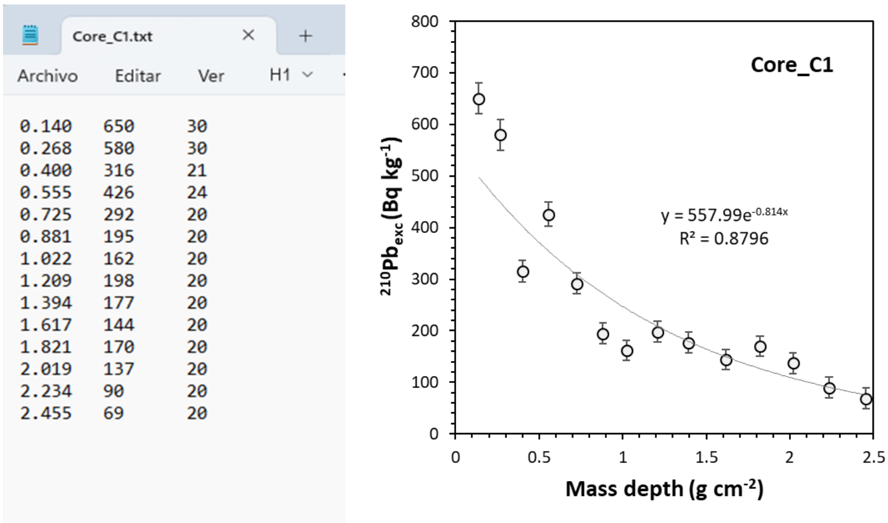


Figure 1: Data file (left panel; the first column is mass depth, referred to the bottom of the slices) and plot of the empirical $^{210}Pb_{exc}$ versus mass depth profile for Core_C1 (a varved sediment core, from original data by Tylman et al., 2013), referred to the midpoint of the slices (right panel). The exponential fit is the practical application of the CF-CS model.

Figure 1 shows the empirical data set used for illustrating the functioning of the TERESA software. They provide a basis for a first estimate of the entry parameters: $\bar{A}_0 . \bar{w} . s_A . s_w$. Thus, $\bar{A}_0 \sim 550$ Bq/kg, as seen from the exponential fit, or just the empirical value in the first slice. It is not necessary higher precision, since latter it will be defined a wide interval around this value, e.g. (440, 660) for generating solvers. From the exponential fit, $\bar{w} \sim \frac{\lambda}{0.814} \sim 0.038$ g cm^{-2}y^{-1}. As before, a wide interval will be defined around this value. This gross estimate is enough to define the intervals for searching for the absolute minimum. The dispersion of empirical data in the plot is related with $s_A . s_w$. In this case it seems moderate. Anyhow, if you are not sure, use 0.25 for these two values and explore a wide range in your first attempt.

Now you are ready to use the codes. The flow chart in Figure 2 can be useful for following the process.

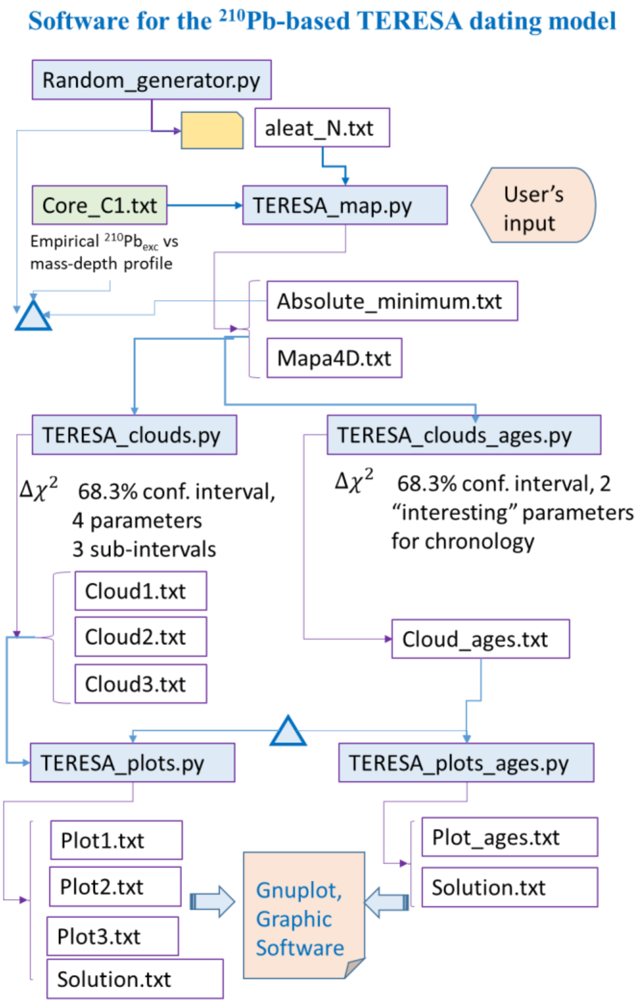


Figure 2. Flow chart that illustrates the set of codes (blue boxes) and the sequence to apply them. The blue arrows indicate inputs and the cyan arrows are outputs. The triangle merges several inputs used to generate plots. For routine applications you can use only the branch of TERESA_clouds_ages.py.

TERESA_map.py

This code reads a file that contains experimental data on $^{210}Pb_{exc}$ in sediment core layers. You must specify the name and path of the file. It also reads a file from the '/aleat' folder. with the same number of entries.

Required input parameters:

- Initial estimates of of $\bar{A}_0.\bar{w}.s_A.s_w$.
- Definition of the intervals around these values to generate solvers.
- The number NR of divisions per interval. By default, the same NR is used for all four parameters, resulting in NR^4 solvers.
- Optional: A time-mark with information to define an objective function.

The code defines a sorting function to find the best arrangement of (A_{0i}, w_i) pairs for each solver. This means the sorting that minimises the quadratic distance to the experimental profile.

The adjusted chi-value (to degrees of freedom), χ_ν, is calculated for the best sorting of each solver.

Outputs:
-Mapa4D.txt: contains the four input values and the corresponding χ_{gl} for each solver.
- Absolute_min.txt: contains the absolute minimum found across all solvers.

The parts of the code that need action (user's input) are bounded by lines +++++++. They are commented on below:

```
# +++++++++++++ INFORMATION FOR USING A TIME-MARK ++++++++++++++
        # Please, ignore this part if you are not using a time mark.
kr = 9  # index for the sediment slice that contains the time mark (note that in Python the index
     stars at zero). It must be an integer >= 0 and < N (the number of slices in the core).
Tmr = 43 # 'The known age of the slice with the time mark (yr)
sgt = 1.5 # 1-sigma error of Tmr. By default, use 1.0
peso = 0* 1 / 2 # Weight to be used in the Objective function. "0" means "no time mark"
           and 1/2 is a recommended value that you can tune.
# +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

This first section to input information will be revisited in the latter section. In the first example of application, no time-marks will be used. Thus, here we check that *peso* = 0, which confirms our choice. The values for the other magnitudes have no effect in computations with these settings, but they are prepared to be used later as an example of time mark.

```
# ++++++++ UPDATE HERE ONLY THE NAME OF THE TEXT FILE CONTAINING
EXPERIMETAL DATA ++++++
with open('Core_C1.txt', 'r') as file:
```

In this line, we must check that the name (and path) of the input file is correct.

```
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#        This requires action: INPUT PARAMETERS
# The initial activity concentration Am0 can be estimated from the value of the first slices
# and/or you can use the CFCS model (exponential fit) to estimate this value, and the mean SAR
(wm0)
# For relative deviations (sgA0 and sgw0), you can start with recommended values in the range.
# 0.1 - 0.3, depending on how "noisy" the experimental profile is.  For complex cases requiring
values higher than 0.35 -0.4, it is better to use log_normal distributions (see more in publications).
# If the LN[A(m)] plot shows discontinuities, think about using Multimodal-TERESA (see
publications).
# Recommendation: if not sure, star with sgA0 = 0.25, sgw0= 0.25

Am0 = 550.0  # Bq/kg
wm0 = 0.038  # g/(cm^2 yr).
```

```
sgA0 = 0.25
sgw0 = 0.25

# A factora defines the interval from Am0*(1-factora) to Am0*(1+factora)
# For example, for Am0 =100 and factora = 0.5, the interval is [50, 150)
# The same for the other factors
# It is recommended to run this code at least twice. In the first run, you can
# define wide intervals, particularly for those magnitudes with high uncertainty.
# The model output identifies the region for the absolute minimum.
# In a second run, you can use the above values at the minimum as input parameters.
# and using factors now defining narrower intervals with a higher resolution.

factora = 0.25
factorw = 0.25 # For very low wm0 you can try alternative definitions of the interval.
factorsa = 0.5
factorsw = 0.5

# The number NR of divisions of each interval NR
# Note that NR = 10 leads to 10^4 solvers, NR = 50 leads 6.25·10^6
# The CPU time increases with NR. You can find a reasonable compromise between resolution
# and CPU time using NR < 50 and repeated runs.

NR = 20
# +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

This section is self-explicative. Here the name of variables *Am0*, *wm0*, *sgA0* and *sgw0*, refer to the initial estimates of of $\bar{A}_0.\bar{w}.s_A.s_w$. The origin of numerical values ascribed has been commented above, from the analysis of Fig. 1. The set of values for *factor_x* defines the following 4D domain in the parametric space:

$\{[412.5, 687.5)_A$, $[0.0285, 0.0475)_w$, $[0.125, 0.375)_{sA}$, $[0.125, 0.375)_{sw}\}$

In this example, each interval is divided into NR = 20 equal parts (note that this defines a given resolution for each parameter). The regular mesh so created defines a solver at each grid point. In this case, the number of solvers is 160000. It is advisable to start with moderate numbers to test the performance of your computer and the required CPU time.

Do not forget to <u>save the changes</u> (user's input). During execution, the code outputs to the screen a counter that ranges from 0 up to NR, which is merely informative.

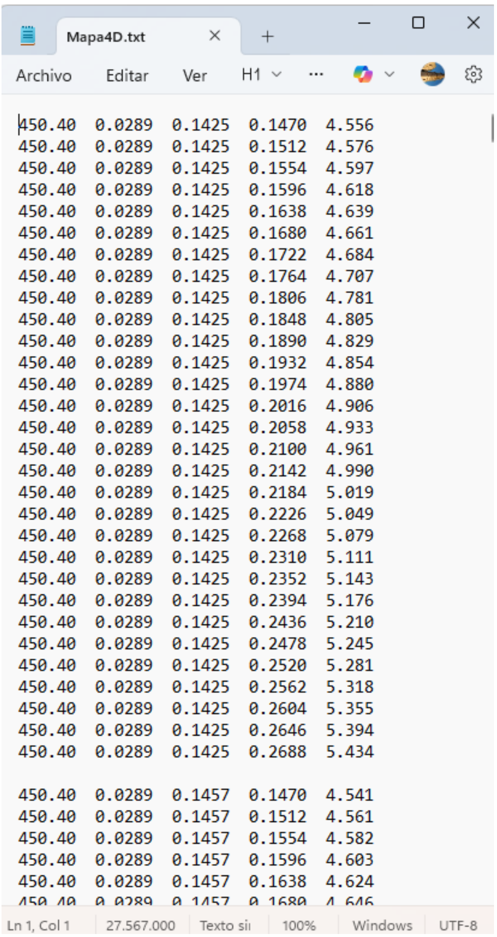
The code finishes creating the output files. The '**Absolute_minimum.txt**' file summarises the used setup and provides the solution for the absolute minimum.

You will need running TERESA_map.py several times. Please, take into account that when TERESA finds the absolute minimum very close to the boundary of any stated interval, you need a new run expanding such interval. However, when TERESA demands $s_A.s_w$ values larger than 0.35 you must think about using another version of the model with log-normal distributions. Also, the user must seek centring the domain around the absolute minimum and achieving a reasonable compromise between resolution (which increase when reducing the width of the intervals or when increasing NR), but with intervals being wide enough as to include the whole region with solvers within 1-σ (or 68.3% confidence intervals, as seen further below), and with the required CPU time. Some of these topics will be better understood after presenting the other codes.

In the sample, after several attempts, this is the '**Absolute_minimum.txt**' file which summarises the result for the absolute minimum and a summary of the input information and settings used in the model.

563.00	0.0385	0.1932	0.2142	0.556	14
3.75	0.00032	0.00158	0.00210		
563.00	0.03850	0.19000	0.21000		
30	0.200	0.250	0.250	0.300	
9	43.0	1.50	0.000		

\# Line 1: Aomin, wmin, sAmin, swmin, χ_v, N slices
\# Line 2: half resolution intervals for A_o, w, s_A, s_w
\# Line 3: Input parameters Am0, wm0, sgA0, sgw0
\# Line 4: NR, *factora*, *factorw*, *factorsa*, *factorsw*
\# Line 5: Information for time mark: kr, Tmr, sgt, *peso*




```
450.40  0.0289  0.1425  0.1470  4.556
450.40  0.0289  0.1425  0.1512  4.576
450.40  0.0289  0.1425  0.1554  4.597
450.40  0.0289  0.1425  0.1596  4.618
450.40  0.0289  0.1425  0.1638  4.639
450.40  0.0289  0.1425  0.1680  4.661
450.40  0.0289  0.1425  0.1722  4.684
450.40  0.0289  0.1425  0.1764  4.707
450.40  0.0289  0.1425  0.1806  4.781
450.40  0.0289  0.1425  0.1848  4.805
450.40  0.0289  0.1425  0.1890  4.829
450.40  0.0289  0.1425  0.1932  4.854
450.40  0.0289  0.1425  0.1974  4.880
450.40  0.0289  0.1425  0.2016  4.906
450.40  0.0289  0.1425  0.2058  4.933
450.40  0.0289  0.1425  0.2100  4.961
450.40  0.0289  0.1425  0.2142  4.990
450.40  0.0289  0.1425  0.2184  5.019
450.40  0.0289  0.1425  0.2226  5.049
450.40  0.0289  0.1425  0.2268  5.079
450.40  0.0289  0.1425  0.2310  5.111
450.40  0.0289  0.1425  0.2352  5.143
450.40  0.0289  0.1425  0.2394  5.176
450.40  0.0289  0.1425  0.2436  5.210
450.40  0.0289  0.1425  0.2478  5.245
450.40  0.0289  0.1425  0.2520  5.281
450.40  0.0289  0.1425  0.2562  5.318
450.40  0.0289  0.1425  0.2604  5.355
450.40  0.0289  0.1425  0.2646  5.394
450.40  0.0289  0.1425  0.2688  5.434

450.40  0.0289  0.1457  0.1470  4.541
450.40  0.0289  0.1457  0.1512  4.561
450.40  0.0289  0.1457  0.1554  4.582
450.40  0.0289  0.1457  0.1596  4.603
450.40  0.0289  0.1457  0.1638  4.624
450.40  0.0289  0.1457  0.1680  4.646
```

Figure 3. A view of the Mapa4D.txt output file of 27.8 Mb size.

TERESA_clouds.py

This code reads the output files generated by TERESA_map.py: Absolute_min.txt and 'Mapa4D.txt'.

- From the absolute minimum of χ_v, it estimates the $\Delta\chi^2_\sigma$ interval corresponding to a 68.3% confidence level (equivalent to 1-σ), allowing optimization of the four model parameters. (see G. Cowan, 1998: Statistical Data Analysis).

- This interval is divided into three equal parts, defining three threshold levels of χ_v. This is for visualization purposes (e.g., displaying the topology of the attractor in a 2D or 3D sub-spaces).

- The code reads **'Mapa4D.txt'** and selects the solvers with χ_v values within each threshold group, storing them in three files: **'Cloud1.txt'**, **'Cloud2.txt'**, and **'Cloud3.txt'**. These three files together represent the 1-σ confidence region.

Note: For complex χ-hypersurfaces, multiple relative minima may exist around the absolute minimum. This method samples solvers from the entire domain that fall within the threshold values. The three output files can be used by TERESA_plots.py to visualise solution clouds or study the topology of attractors.

You can inspect the extreme parameter values in the third cloud to verify that they are within the domain. If not, revisit TERESA_map.py to redefine the boundaries.

If you keep the default names for the input and output files, you can run this code without any further action. After familiarising yourself, you can customise as needed.

In the example, these are the three threshold levels for χ_v: 0.665, 0.775, and 0.884. The number of *solvers* found within each level was: (1) 4077, (2) 7222, (3) 8961. The total number of *solvers* processed was 810000.

The output files are subsets of **'Mapa4D.txt'**, and they keep the same format, but without the empty lines associated to the regular mesh.

The maximum and minimum values for each entry parameter can be found in the **Cloud3.txt** file (you can use Excel or a toy code for this task), and it has been observed that they fall within the 4D- parameters' domain.

TERESA_plots.py

This code computes the theoretical profiles of activity concentrations, chronology, and related quantities for each solver included in the output files generated by TERESA_clouds.py: 'Cloud1.txt', 'Cloud2.txt', 'Cloud3.txt', as well as the solver from 'Absolute_min.txt'.

It requires reading the empirical profile (stored in 'Core_C1.txt' in this example - please update as needed), and the corresponding 'aleat_N.txt' file from the '.\aleat>' folder.

The function *profile* is defined, similar to '*sorting*', but returns the full theoretical profile generated from each solver.

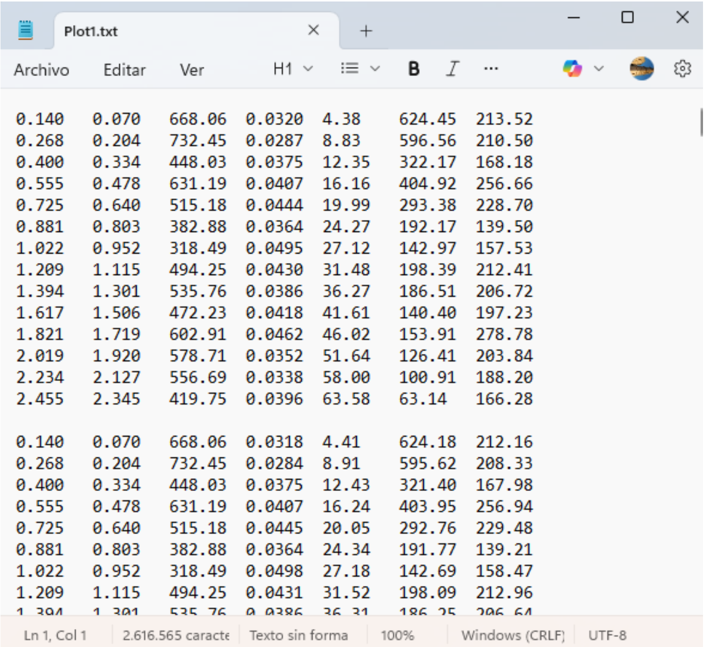
The computed solutions are stored in the output files: **'Plot1.txt'**, **'Plot2.txt'**, **'Plot3.txt'**, and **'Solution.txt'**. These are 7-column files, with an empty line separating each solver:

m_i, mi_m, SolA, Solw, Time_sol, Sol_Th, Sol_flux

Column descriptions:
- m_i: Mass depth at the bottom of the sediment slice.
- mi_m: Mass depth at the midpoint of the slice.
- SolA: Initial activity concentration at mi_m (in Bq/kg).
- Solw: Mean sedimentation rate in the slice (at mi_m), in g/(cm^2·yr).
- Time_sol: Age refers to the bottom of the slice (in years).
- Sol_Th: Theoretical ^{210}Pb$_{ex}$c profile at mi_m, comparable to the empirical profile (in Bq/kg).
- Sol_flux: Mean ^{210}Pb$_{ex}$c flux captured in the slice, in Bq/(m^2·yr).

The output files can be directly read and plotted using graphic software such as Gnuplot, or any other preferred by the user.

If you keep the default names for the input and output files, you can run this code without any further action. After getting familiar, you can customise as needed.



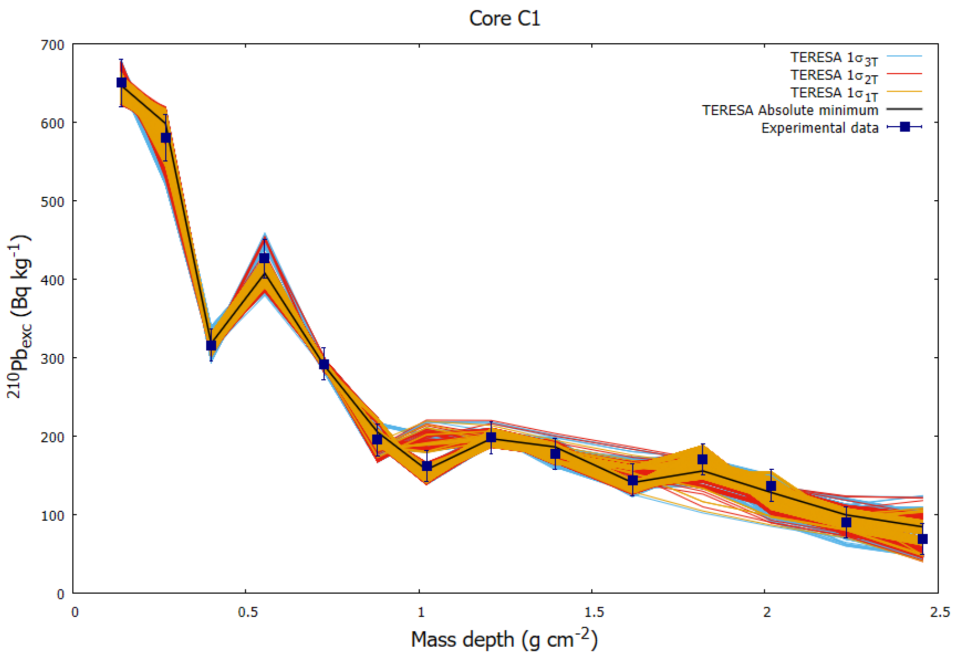

```
0.140   0.070   668.06  0.0320  4.38    624.45  213.52
0.268   0.204   732.45  0.0287  8.83    596.56  210.50
0.400   0.334   448.03  0.0375  12.35   322.17  168.18
0.555   0.478   631.19  0.0407  16.16   404.92  256.66
0.725   0.640   515.18  0.0444  19.99   293.38  228.70
0.881   0.803   382.88  0.0364  24.27   192.17  139.50
1.022   0.952   318.49  0.0495  27.12   142.97  157.53
1.209   1.115   494.25  0.0430  31.48   198.39  212.41
1.394   1.301   535.76  0.0386  36.27   186.51  206.72
1.617   1.506   472.23  0.0418  41.61   140.40  197.23
1.821   1.719   602.91  0.0462  46.02   153.91  278.78
2.019   1.920   578.71  0.0352  51.64   126.41  203.84
2.234   2.127   556.69  0.0338  58.00   100.91  188.20
2.455   2.345   419.75  0.0396  63.58   63.14   166.28

0.140   0.070   668.06  0.0318  4.41    624.18  212.16
0.268   0.204   732.45  0.0284  8.91    595.62  208.33
0.400   0.334   448.03  0.0375  12.43   321.40  167.98
0.555   0.478   631.19  0.0407  16.24   403.95  256.94
0.725   0.640   515.18  0.0445  20.05   292.76  229.48
0.881   0.803   382.88  0.0364  24.34   191.77  139.21
1.022   0.952   318.49  0.0498  27.18   142.69  158.47
1.209   1.115   494.25  0.0431  31.52   198.09  212.96
```

Figure 4. A view of the output file '**Plot1.txt**' with the example of Core_C1.txt. The solution for each solver is separated by empty lines.

These files can be read and plotted with Gnuplot, as shown in Figure 5. The very basic command for plotting the theoretical versus empirical ^{210}Pb$_{exc}$ profiles is:

plot 'Plot3.txt' us 1:6 w l, 'Plot2.txt' us 1:6 w l, 'Plot1.txt' us 1:6 w l,'Solution.txt' us 1:6 w l, 'Core_C1.txt' us 1:2:3 w err

For the chronology, the sequence is similar but using columns 1:5. As the core in the example comes from a varved sediment, there is a varve chronology (see Tylman et al., 2013), which has been used for comparison purposes. Note that this is an external validation, since information on varve ages has not been used at any stage of the model.



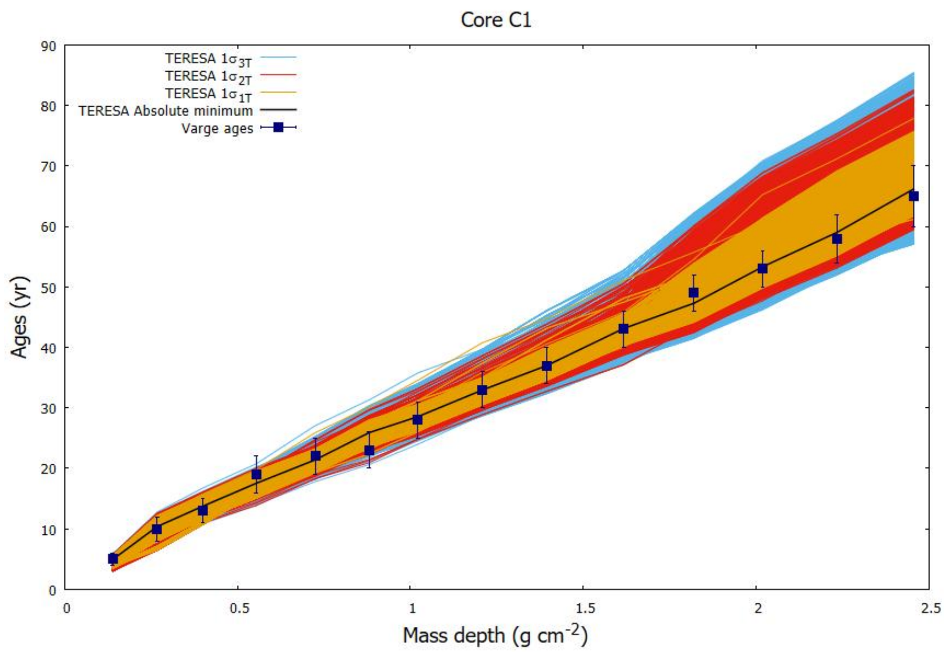


Figure 5: Plots with the cloud of model solutions for the $^{210}Pb_{exc}$ vs. mass depth profile (upper panel) and chronology (lower panel) within a 68.3% confidence interval, using different colours for the three χ_{ν} threshold levels. The solution for the absolute minimum is also plotted. They are compared with empirical data for $^{210}Pb_{exc}$ and for the known varve chronology (Tylman et al., 2013).

In the plot, you can customise colours and add titles and labels (your IA can help you with these details). Note that for fast plots, you can use column 1 (the mass depth at the bottom of the slice) and the input file containing the empirical data. However, for final plots, you should use column 2 and an adapted 'Core_C1_m.txt' file using mass-depth scale at the midpoint of the slice.

TERESA_clouds_ages.py

This code is based on TERESA_clouds.py, but defines a different threshold for χ_ν using the concept of "interesting parameters" (see G. Cowan, 1998: 'Statistical Data Analysis'). The interesting parameters are *wm0* and *sgw0*, since they strictly determine the sequence of SARs and, consequently, the chronology. The method keeps the values of the other two parameters fixed at their absolute minimum: *minA* and *minw*.

The code reads from **'Mapa4D.txt'** only those solvers that contain *minA* and *minw* (their total number is NR^2), and stores in a single output file (**'Cloud_ages.txt'**) those that fall within the new threshold value of χ_ν, corresponding to the 68.3% confidence level (equivalent to 1σ) under these settings.

If you keep the default names for the input and output files, you can run this code without any further action.

In the example of Core_C1.txt, the number of *solvers* within the threshold value of χ_ν were 121, from a total number of 900 s*olvers* processed.

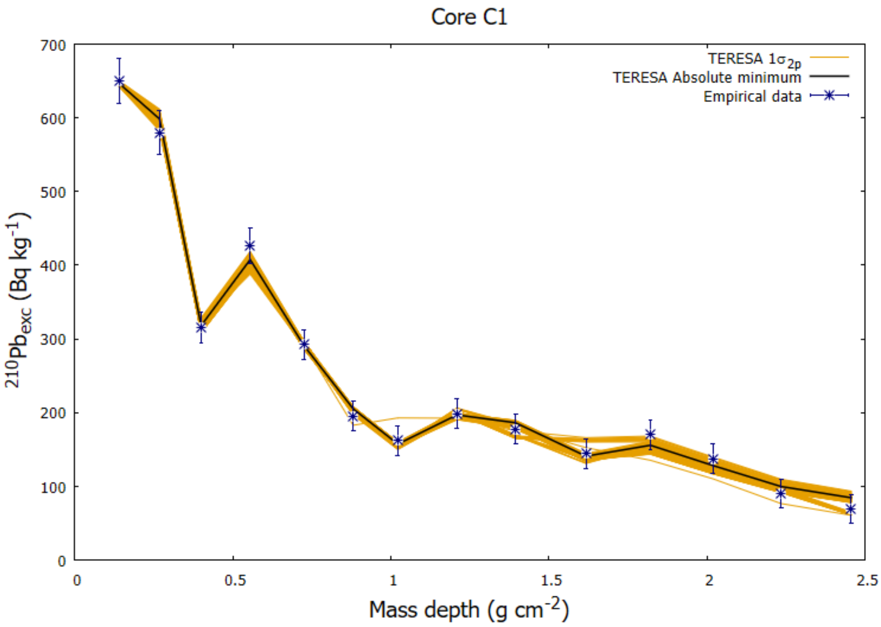
TERESA_plot_ages.py

This code is based on TERESA_plots.py (see previous reference), but it computes the theoretical profiles only for the set of solvers stored in the file **'Cloud_ages.txt'**. It stores the results in the output file **'Plot_ages.txt'** and also generates the same **'Solution.txt'** file, useful if you are only concerned with chronology, allowing you to skip both TERESA_clouds.py and TERESA_plots.py.

The output can be used to plot (with Gnuplot or other graphic packages) the chronological line derived from the solver at the absolute minimum, along with the cloud of solvers corresponding to a 68.3% confidence interval by this method.

If you keep the default names for the input and output files, you can run this code without any further action.

The results for the sample studied are shown in Figure 6.



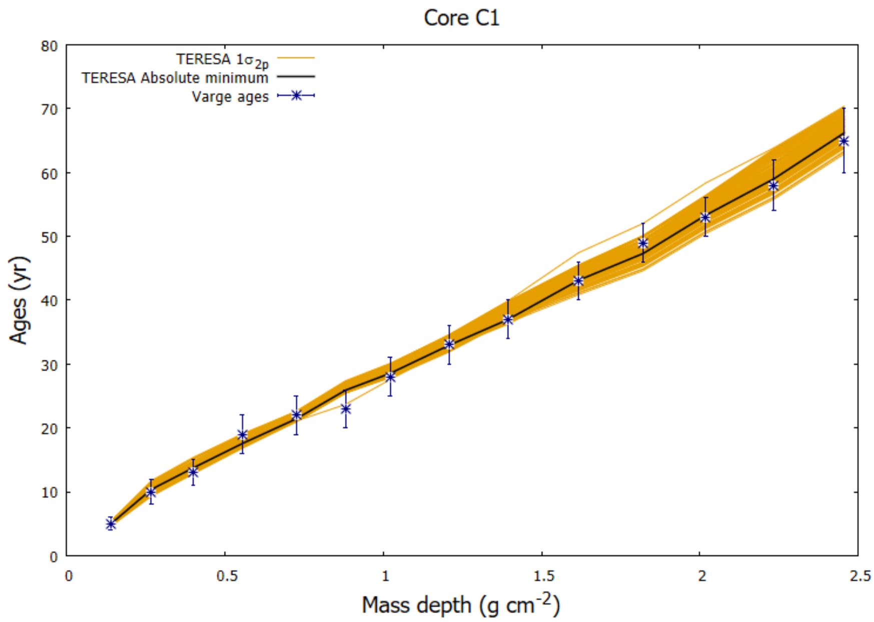


Figure 6: As in Figure 5, this panel plots the chronologies from the *solvers* within a 68.3% confidence interval using the method of two 'interesting parameters'.

Using time marks with an objective function

This is only an introductory note to illustrate this option. In the main code TERSA_map.py introduce the value *peso* = 1 / 2 instead of zero, keeping the rest of the values, which define a time mark of age 42 yr at the bottom of the slice of index kr = 9 (mass depth 1.617 g cm^{-2}). The default objective function is:

$$\Theta^2 = \chi^2 + peso\ N\left[(T_r - T_{model})/\sigma_{T_r}\right]^2 ,$$

where T_{model} is the age given by the solver at the position of the time mark.

The execution in sequence of the other codes follows as before. A deep discussion of the confidence intervals is not faced here. However, it is worth noting that an alternative definition of the objective function as $\Theta^2 = \chi^2 + \left[(T_r - T_{model})/\sigma_{T_r}\right]^2$ keeps the form of a χ^2 with an additional degree of freedom, but still with four parameters.

In this examplethe model chronology already fitted pretty well the varve ages, so the introduction of a time mark does not improve the new chronology. However, the confidence intervals become narrower, particularly around the time mark, as shown in Figure 7 (obtained by using the default formulation for Θ^2 with *peso* = 1 / 2).

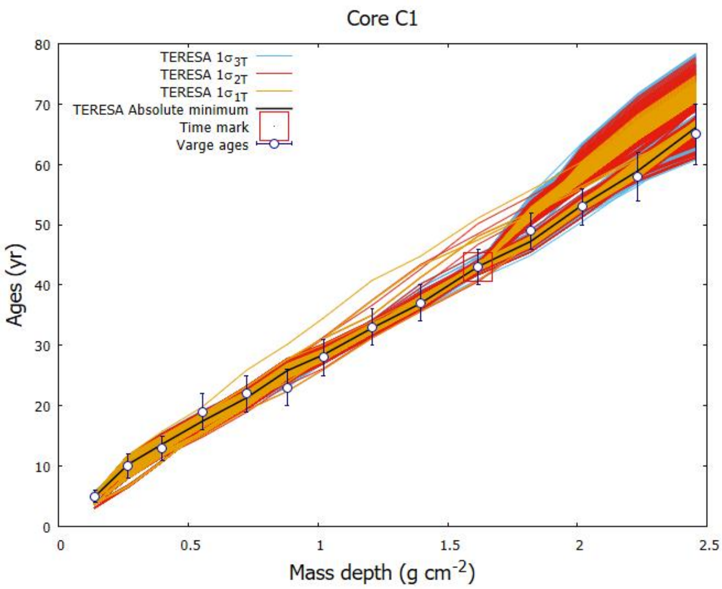


Figure 7. This is as panel 2 in Figure 5, but using a time mark with the default formulation for Θ^2 with *peso* = 1 / 2.