fortune Journals

Review Article

JOURNAL OF BIOINFORMATICS AND SYSTEMS BIOLOGY

ISSN: 2688-5107

Glophyt: a User-Friendly, General-Purpose Program for Nonlinear and Multidimensional Curve Fitting via a Hybrid Stochastic and Deterministic Approach

Georges Czaplicki* **and Serge Mazeres**

Abstract

Model validation depends on the agreement between the predicted and experimental data. However, finding solutions to problems, described by equations with many parameters, where even their orders of magnitude are not known, is a difficult task. This makes curve fitting very difficult in case of multidimensional and nonlinear data. This article presents a graphical user interface-based program employing a hybrid stochastic and deterministic approach, which allows for easy and reliable determination of model parameters by minimizing the differences between measured data and those calculated on the basis of a mathematical expression. The program has been extensively used in several laboratories and has proven to be efficient in determining model parameters in many different fields, such as pharmacological studies of ligand–receptor binding, entomological studies of populations, bacterial growth, photosynthesis, toxicology, differential scanning calorimetry, isothermal titration calorimetry and nuclear magnetic resonance spectroscopy. It is an effective solution for researchers facing the problem of estimating model parameters from multidimensional and nonlinear data where the orders of magnitude of parameters are not known.

Keywords: Multidimensional Nonlinear Curve Fitting; Stochastic and Deterministic Hybrid Algorithms; Global Optimization; Parameter Estimation;

Introduction

It is a common practice in experimental science to test different models of the studied phenomena by comparing measured results with those calculated on the basis of a mathematical expression. The purpose is to estimate the values of inherent parameters to gain a better understanding of the underlying processes. A good example of such a situation is provided by kinetic studies of enzyme-inhibitor reactions, which require time-consuming analyses, where fitting various conceptual models of interaction pathways to experimental data plays an important role. To find a set of parameters consistent with the observed data, nonlinear multidimensional global optimization techniques must be used (see [1] and references therein). There are a number of software solutions available to accomplish this task, which minimize differences between the predicted and experimental curves. The vast majority of them use gradient-based approaches, such as steepest descent, conjugate gradients or Newton–Raphson methods [2], which have the advantage of converging rapidly to the solution closest to the starting point. However, due to the complex profile of the minimized function, it is usually a local minimum rather

Affiliation:

Institut de Pharmacologie et de Biologie Structurale, Université de Toulouse 3, CNRS, Toulouse, France

***Corresponding author:**

Georges Czaplicki, Institut de Pharmacologie et de Biologie Structurale, Université de Toulouse 3, CNRS 205, route de Narbonne, 31077 Toulouse, France

Citation: Georges Czaplicki, Serge Mazeres. Glophyt: a User-Friendly, General-Purpose Program for Nonlinear and Multidimensional Curve Fitting via a Hybrid Stochastic and Deterministic Approach. Journal of Bioinformatics and Systems Biology. 7 (2024): 219-226.

Received: October 01, 2024
Accepted: October 08, 2024
Published: October 18, 2024

than a global one. The necessity to provide an initial guess for the searched parameters is the main drawback of these algorithms since they are not known a priori and may differ by orders of magnitude. When reasonable preliminary estimates of parameter values are known, deterministic minimization techniques can be used to refine them. However, if preliminary estimates are not available and the character of the newly found minimum is not known, the usual remedy consists of using multi-start methods, i.e. repeating minimization with different starting points and accepting the solution, which is closest to the experimental data [3]. By increasing the number of different starting points, we increase our confidence that the best fit will approach the global minimum (within the limits of a predetermined accuracy). Unfortunately, we can never be certain that the data set cannot be described more satisfactorily by a different combination of parameters. Finally, the overall time necessary to accomplish this task is significantly increased, which greatly reduces the appeal of this approach in a multidimensional case.

Another option is to use stochastic approaches, such as genetic algorithms, simulated annealing or particle swarm optimization, which increase the probability of finding the global minimum [4]. There is no guarantee of success, but there is no need for a starting point, and the result depends on adequate sampling of the parameter space, which is user-controlled. Unfortunately, these algorithms are rarely available in widely accessible programs. Commercial software can be used, such as MATLAB [5], but it usually implies high cost and some programming skills.

Yet another option is to use deterministic global optimization algorithms [6] which use the branch-and-bound approach, to find the global minimum of a target function by constructing consecutive approximations to its upper and lower limits [7]. Some of them can handle mixed integer nonlinear programming (MINLP) problems, which represent a severe challenge because few software solutions are available. Most of them are commercial products such as ANTIGONE [8], BARON [9], or Octeract Engine [10] and only a few are open source, e.g., Couenne [11] or SCIP [12].

The major drawback is that most software packages require the studied problems to be coded in their proprietary languages. Since they are not specifically dedicated to model verification, in each case the user must code not only the function representing the tested model, but also the target function in optimization. In the simplest case it is a sum of squared differences between the calculated and experimental points. This implies the familiarity with coding loops and summation of vector/matrix elements, which in turn requires the knowledge of the syntax of a given language, such as R for MEIGOR [13], Python for SciPy [14] and PyPESTO [15], AMPL [16] for Couenne, or a user-selected language for the evolutionary approach DEEP [17]. For a non-programmer this may be a challenge and a time-consuming task. In Glophyt, the target function is already compiled in the program, and the user must only supply the expression defining the model, which has been deliberately made intuitive and user-friendly. Hence, the effort and time spent on preparing input data is reduced to a minimum.

The program presented in this article was born from the frustration of our colleagues, who wanted a simple tool for nonprogrammers, which would reliably analyze their data and help validate multiple models. Consequently, we have developed this program with the following objectives in mind: (i) simple operation requiring little to no programming techniques, (ii) user-friendly interface for data input and output, (iii) flexibility in defining models for fitting, (iv) customizable graphical visualization of fitting results, (v) comprehensible output of quantitative results, and (vi) cost-free use.

Materials and Methods

The program's graphical user interface (GUI) was written in C++, using the Qt library [18] with the plotting widget QCustomPlot [19]. For the numerical calculations, a library of Fortran routines was created and interfaced with the GUI. For its development, the Intel oneAPI Base and HPC toolkits [20] were used. The code was compiled and tested on Windows 10, as well as on several Linux platforms: Debian v.11 and v.12, openSUSE v.15, Ubuntu v.22, Fedora v.38 and Mint v.20.3 (unfortunately, we had no access to the MacOS platform). Two versions are available, either with or without the installer (the so-called "portable" version).

Two algorithms have been implemented: particle swarm optimization (PSO) [21] and simulated annealing (SA) [22]. The latter was combined with the conjugate gradient algorithm [23]. They have been modified in order to assure adequate sampling of the parameter space. The greatest problem from our point of view is the fact that ranges of parameters may differ by orders of magnitude and can thus lead to enormous computational efforts. In terms of efficiency (i.e. convergence), the best results are obtained when all unknown parameters are of the same order of magnitude. A rescaling of parameters can be embedded in the minimized functions, but it implies some a priori knowledge about the ranges of their variability and hence is not applicable in our case. Therefore we decided to contract the parameter space to decrease the searched volume. A simple linear transformation (e.g. division of all parameters by a constant factor) is not adequate, because it corresponds merely to a change in numeric precision. We chose a logarithmic transformation [24], which in principle also exhibits this problem, but to a much lesser degree. It is always possible to define a maximal range of variability v_i for each parameter (if it is to be unconstrained, the upper and lower bounds may be set to a very low and a very high value, respectively). If the lower and upper bounds are denoted by A and B, respectively, each point x from the range [A, B] can be transformed into y (and vice versa) in the following way:

$$y = \log(x - A + c)$$
$$x = e^{y} + A - c$$

where an arbitrary positive constant c is added to avoid calculating log(0) when $x = A$. As a result, the parameter space contracts (number wise), and sampling becomes more dense, which facilitates locating promising areas. It might be argued that logarithmically transformed uniformly generated random numbers sample the parameter space in a non-uniform, hence inadequate, way. However, exponential distribution of random points, implemented to sample the contracted parameter space uniformly, resulted in a marked loss of convergence, and hence in considerable increase of execution times.

In case of the SA algorithm we also modified the cooling schedule. Instead of simply multiplying each temperature by a factor < 1, we opted for a Gaussian-like scheme:

$$T_k = T_0 e^{-\left(\frac{k}{r}\right)^2}$$

where k is the index of the current temperature, and r corresponds to the number of iterations (i.e. temperature changes) which decrease T to 1/e of its original value T0. The advantage of this type of cooling lies in the fact that in the beginning the temperature values fall off slowly, permitting a thorough search of the parameter space. Cooling becomes faster when the algorithm has built up a rough idea about the function landscape, and slows down in the end, when refinement of results is necessary.

Applying the above modifications results in a relatively rapid convergence of the program to the (hopefully) global minimum. In view of the fact that there exist certain user-controlled parameters (not necessarily optimized on the first run), ideally the program should be run a few times along different trajectories, to gain certainty that the found optimum is indeed global.

The program minimizes the sum of squared residuals between the experimental points and those calculated on the basis of the user defined model. Each term in the sum can represent either an absolute deviation or a relative one, according to the user's choice. The examples given below were obtained using the absolute deviation option. The uncertainties of final parameter values are given in terms of 1, 2 or 3 standard deviations, according to the user's choice of the confidence interval.

Experimental data can be searched for outliers. For this purpose, robust regression is implemented, with median of squared deviations as the target function [25]. The outliers are displayed and listed, allowing the user to decide whether to include them in fitting or to exclude them from the analysis.

In case when it is difficult to obtain a mathematical expression describing the studied model, the program offers an option of automatically determining the degree of the polynomial which best fits the input data.

Calls to intrinsic functions are monitored to avoid passing arguments whose values might cause overflow or underflow of registers (e.g. in the *exp* function), which could corrupt the results of calculations.

The setup dialog allows full access to all parameters controlling their behavior. There are five tabs in the GUI that make it possible to do the following:

(i) Define the equation, variables, parameters and constants

To define an equation, users can type their own formulae. The parser is flexible and accepts multiline input, temporary variables, and user-defined functions, which can be nested. There are also a number of predefined, intrinsic functions that allow for even greater flexibility. There is also a possibility to work with implicit equations. The current limits on the number of independent variables and parameters are 25 and 100, respectively.

(ii) Enter or import experimental data

Experimental data can be entered manually, copied and pasted from external sources, or imported by reading data files (*.dat, *.txt, *.csv …).

(iii) Curve fitting with full user control of the relevant parameters

The progress of the fitting can be monitored on the screen. The results can be saved for pairwise model comparisons.

(iv) Create a customizable graph of the results and save it in different formats

Users can define symbols for points, line styles for curves and their colors. The title, axis labels and legend items are modifiable. Additionally, custom annotations can be added. The graph can be saved as a PDF file or as an image (svg, png, jpg, tiff or bmp). Moreover, additional curves can be generated with modified values of variables and/or parameters and generated tables of coordinates can be saved.

(v) Create a quantitative output from the fitting and save it in PDF or HTML format

The output includes a summary of the input data, target function and numerical results (the optimal parameters, correlation matrix, and quantitative data for model comparisons). The custom graph can be appended to this report.

Glophyt fits to a mathematical expression. It does not work with differential equations. Hence, in case of kinetic studies of biological systems the analysis can be performed when the analytical solution to the system of equations is known. Otherwise, a different software package should be used, such as COPASI [26]. The equation parser in Glophyt is flexible enough for the majority of problems to be entered as expressions in the equation field. However, it is not a programming language and does not allow certain constructs, such as loops. Moreover, it stores equations converted to internal code on a stack, which is interpreted during calculations. While the vast majority of equations encountered in the field of biology can be treated without any problems, we decided to offer an option for more advanced users, who have programming skills, to define equations of arbitrary complexity. Moreover, the compiled functions run faster than the interpreted functions. For this reason, we added the possibility of dynamically attaching an external library (*.dll for Windows, or *.so for Linux) to the program. Then, one can execute one's own code from the library. The speedup depends on the nature of the problem, but it may easily reach one order of magnitude.

Results

To compare the performance of Glophyt with those of other programs, we performed calculations on several non-linear test problems from our own laboratory, using software often cited in the literature. In many cases, a paid license is needed, e.g., for MATLAB [5], OriginLab [27], CurveExpert [28], GraphPad Prism [29] and LAB Fit [30], although sometimes a free trial version is available. There are also free software packages for curve fitting, such as Octave [31], SciDAVis (32), Fityk [33], R/RStudio [34], Python/SciPy PyPESTO, MEIGOR and DEEP. They represent many different approaches to optimization, from gradient-based methods to metaheuristics. The program was executed on a PC equipped with an Intel Core i9 10980XE processor running at 3 GHz, both on Windows and on Linux.

We evaluated the performance of a given software by measuring its success rates, defined as the ratio of number of runs in which the final parameters were close to the best known solution to the total number of runs. In our case the latter was equal to 100. The conditions (specifically the lower and upper bounds for parameters) were identical in all cases. Parameters of all algorithms were left at their default values. The execution times were of less importance, since we prioritized correct final parameter values (in all cases the execution times ranged from seconds to minutes). To illustrate the results two specific examples were selected for this section. Figure 1 shows an example of a decaying periodic function of time, described by one independent variable and five parameters. Glophyt returns the correct solution rapidly with success rates of 100% for both the PSO and SA algorithms.

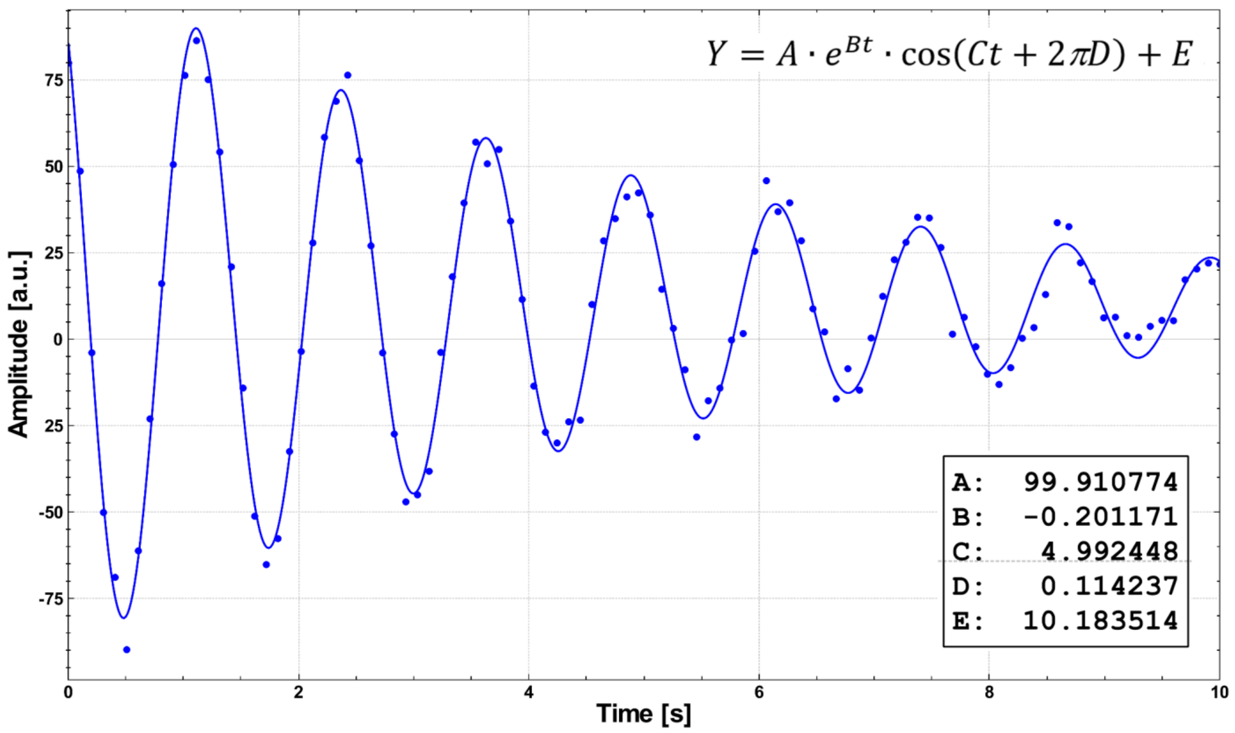

Figure 1: A synthetic data set with random noise representing an NMR free induction decay signal. There is one independent variable (time) and five parameters: A (amplitude), B (decay rate), C (frequency), D (phase) and E (baseline). The correct parameter values are: A=100, B=-0.2, C=5, D=0.1, E=10. The following bounds were used: (0, 1000) for A, (-100, 100) for B, (0, 2π) for C, (0, 1) for D and (-10^9, 10^9) for E. The curve shows the fitting results.

For the other programs the success depended on the optimization method used. In case of the gradient-based methods, the tested programs had a success rate of 100% when the starting point was close to the optimum. However, when the initial values were randomized, the success rate dropped to 0% and error messages appeared, such as "No convergence" or "Floating point error" (Prism, SciDAVis, Octave and Fityk). This is no surprise, in view of the well-known properties of the gradient-based methods. In the case of MEIGOR only the eSSR multi-start approach gave good results, in 18% of runs. In the case of PyPESTO the highest success rate was obtained with the Powell method (12%), then with the CMA-ES approach (8%), L-BFGS-B and multi-start were successful in 1% of runs. PSO had 0% of success. As for the deterministic global solvers, we tested Couenne within AMPL (free Student version). The program worked as expected, and it displayed correct parameter values in 100% runs.

The second test problem we selected for this paper is related to a kinetic study of enzymatic reaction pathways, which comes from the study conducted in our laboratory. Figure 2 presents the scheme used in the analysis, and Figure 3 shows the equations representing the analytical solution to the problem, describing the rate of product formation as a function of substrate (S) and inhibitor (I) concentrations. There are 14 parameters involved, of which 3 were kept fixed and the remaining 11 used in the fitting procedure.

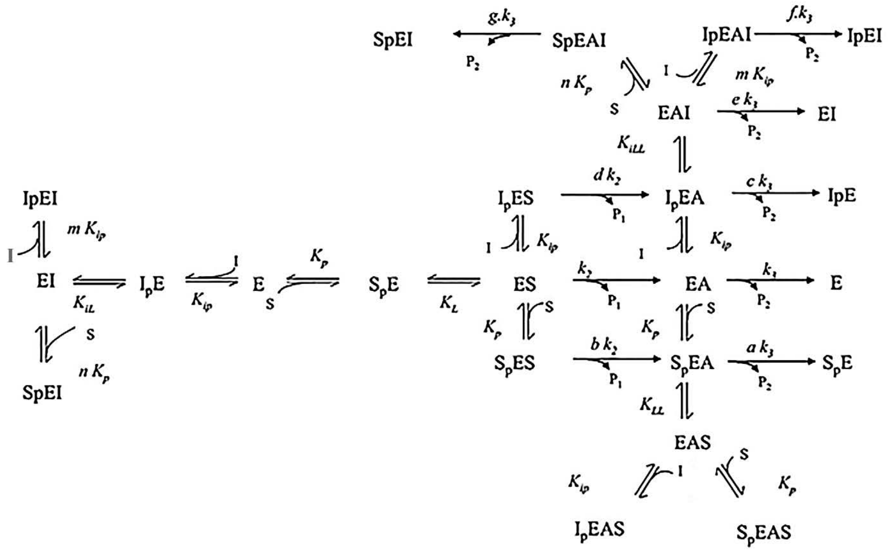

Figure 2: Kinetic study of enzymatic reaction pathways. The example shows the hydrolysis of acetylthiocholine by Drosophila acetylcholinesterase at pH 7 in the presence of edrophonium (courtesy of Prof. Didier Fournier).

$$v = \frac{g1}{g2 + S \cdot g3}$$

$$g1 = k_3 k_2 S$$

$$g2 = k_3 K_p K_L \frac{1 + \frac{S}{K_p}\left(1 + \frac{1}{K_L} + \frac{S}{K_p K_L} + \frac{I}{K_{ip} K_L} + \frac{nI}{K_{ip} K_{iL}}\right) + \frac{I}{K_{ip}}\left(1 + \frac{1}{K_{iL}} + \frac{mI}{K_{ip} K_{iL}}\right)}{1 + b\frac{S}{K_p} + d\frac{I}{K_{ip}}}$$

$$g3 = k_2 \frac{1 + \frac{S}{K_p}\left(1 + \frac{1}{K_{LL}} + \frac{S}{K_p K_{LL}} + \frac{I}{K_{ip} K_{LL}} + \frac{nI}{K_{ip} K_{iLL}}\right) + \frac{I}{K_{ip}}\left(1 + \frac{1}{K_{iLL}} + \frac{mI}{K_{ip} K_{iLL}}\right)}{1 + a\frac{S}{K_p} + \frac{I}{K_{ip}}\left(c + \frac{e}{K_{iLL}} + \frac{fI}{K_{ip} K_{iLL}} + \frac{gS}{K_p K_{iLL}}\right)}$$

Figure 3: The rate of enzymatic reaction *v* as a function of substrate (S) and inhibitor (I) concentrations. The equations have been derived for the schema presented in Figure 2.

Although this is not the most complex problem we worked with, it presents a challenge for most programs since it has two independent variables and eleven fitting parameters. Figure 4 presents the Glophyt fitting results. The program found the solution in 21 s using the PSO algorithm and in 235 s using the SA algorithm when using the default settings. It is worth mentioning that when the number of cycles of the SA algorithm (which controls the sampling of the parameter space) is reduced tenfold, the algorithm still yields correct results in 24 s. For comparison, we tested the example from Figure 2 on identical but precompiled code contained in a dynamically loaded library, and the execution time decreased to 2 s for the PSO algorithm and to 18 s for the SA algorithm (3 s with a reduced number of cycles). The values of the fitted parameters span five orders of magnitude, which is sufficient to present a challenge to most of the tested programs. Although the optimal solution is not known, the above result is likely to be close to it, since nearly the same final parameter values were obtained on all 100 runs.

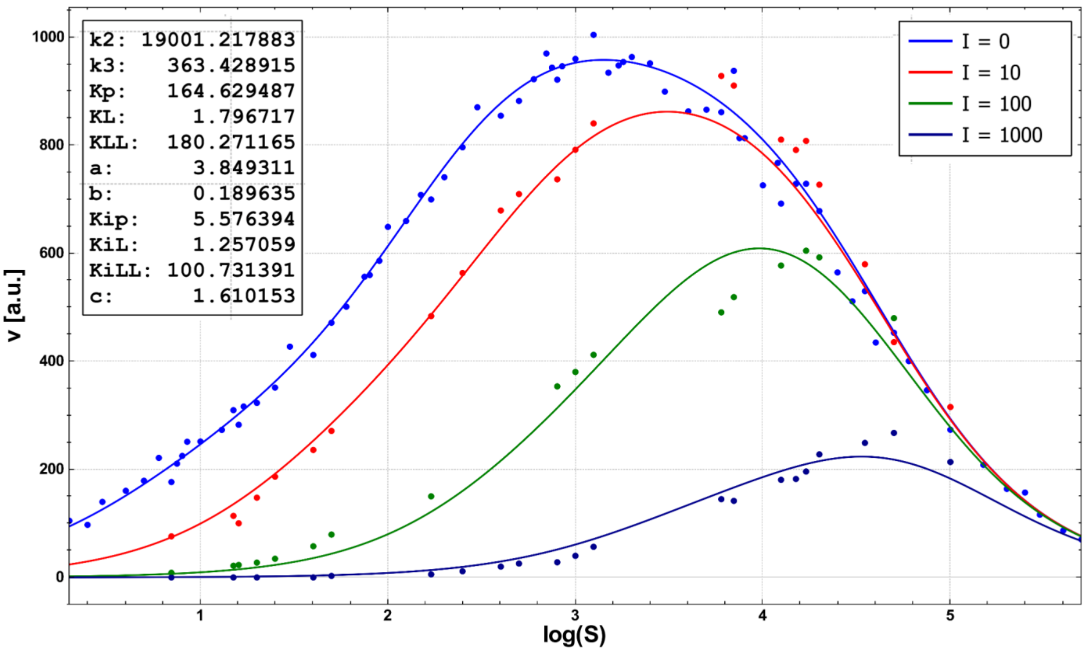

Figure 4: Curve fitting results for the example detailed in Figure 2. There are two independent variables, the concentrations of substrate (S) and inhibitor (I), as well as fourteen parameters, of which three were fixed (m = n = 1, e = 0). The remaining eleven parameters were used in fitting. The vectors of lower and upper bounds for the parameters had the following values: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] and [20000, 500, 300, 2, 200, 5, 1, 20000, 1, 1, 1].

The second example, presented in Figure 2, turned out to be a serious challenge for most of the tested software. Some of the fitting programs cannot be used because they are limited to one independent variable (Prism, Fityk). Other programs which accept multiple variables have their own limitations, for example, those imposed on the number of parameters to be used in fitting and/or on the length of the equation field, which contains a user-defined function (e.g., 10 parameters and 150 characters for LABFit). The latter is necessary because the function in Figure 3 cannot be found in libraries often included with the software and needs to be coded as a user-defined function. In our case, we have 11 fitting parameters, and the function requires nearly 300 characters.

As for the programs that could be tested on this example, all of the gradient-based variants had a success rate of 0%. The same result was obtained for MEIGOR's multi-start method. However, PyPESTO's CMA-ES evolutionary approach gave excellent results, with 100% success rate. All the other algorithms had the success rate of 0%. Among them, the PSO algorithm was the most promising, but the best target function value it provided was 10% higher than the one found by Glophyt. The deterministic global solver Couenne with AMPL was also tested. Unfortunately, in this case the program stops with an error message and does not find any solution.

To summarize the results, there are few programs which provide correct solutions to the problems discussed above with high success rate. The best of them are PyPESTO's CMA-ES method and MEIGOR's multi-start approach. Couenne works well only on some problems. Glophyt is more reliable and versatile, and it offers ease of use that can rarely be found elsewhere. It has already been mentioned before that stochastic algorithms do not guarantee success, but the probability of reaching the optimum is rather high. According to our estimates, based on the ratio of positive results to the overall number of runs, the success ratio of Glophyt is 80-90% in most of the studied cases (other examples are provided with the software distribution and discussed in detail in the accompanying manual). If the result of a given run is not satisfactory, it is advisable to try again because each time the program is launched, it follows a different trajectory through the parameter space; hence, the chances of finding the global minimum increase.

Discussion

The objective of this work was to provide researchers with an easy-to-use, reliable program capable of finding global minima of arbitrary functions. The results of test cases conducted on many levels prove that the goal has been achieved. Glophyt succeeds where many other programs fail. It is based on the combination of stochastic and deterministic algorithms and finds the global minimum of a specified function in a given range of variability of unknown parameters in reasonable times (in most cases, from seconds to minutes). For more demanding users, an option is provided to use one's own precompiled library of equations. The program is available on Windows and Linux platforms, both with an installer and as a portable version, and can be downloaded from the page glophyt.free.fr.

Acknowledgments

The authors thank Didier Fournier for providing examples of kinetic studies used in testing the program.

Conflict of interest: The authors declare that they have no competing interests.

References

1. Locatelli M, Schoen F. (Global) Optimization: Historical notes and recent developments. EJCO 9 (2021): 100012.
2. Luenberger DG, Ye Y. Linear and Nonlinear Programming. New York: Springer 228 (2016).
3. Marti R, Lozano JA, Mendiburu A, et al. Multi-start Methods. In: Handbook of Heuristics. Springer (2018): 155–175.
4. Li C, Grossmann IE. A Review of Stochastic Programming Methods for Optimization of Process Systems Under Uncertainty. Front Chem Eng 2 (2021): 622241.
5. Matlab (2024). https://fr.mathworks.com/products/matlab.html
6. Floudas CA. Deterministic Global Optimization. Theory, Methods and Applications. Springer 37 (2000).
7. Stripinis L, Paulavicius R. An extensive numerical benchmark study of deterministic vs. stochastic derivative-free global optimization algorithms 10 (2022).
8. Floudas CA. ANTIGONE (2013).
9. The Optimization Firm. BARON (2024). https://www.minlp.com/baron-solver
10. Octeract Engine (2024). https://octeract.gg/octeract-engine/.
11. Couenne (2024). https://github.com/coin-or/Couenne.
12. SCIP (2024). https://www.scipopt.org/
13. Egea JA, Marti R, Banga JR. An evolutionary method for complex-process optimization. COR 37 (2010): 315–324.
14. Virtanen P, Gommers R, Oliphant TE. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nat Methods 17 (2020): 261–72.
15. Schälte Y, Fröhlich F, Jost PJ. pyPESTO: a modular and scalable tool for parameter estimation for dynamic models. Bioinformatics 39 (2021).
16. AMPL (2024). https://ampl.com/
17. Kozlov K, Samsonov AM, Samsonova M. A software for parameter optimization with Differential Evolution Entirely Parallel method. PeerJ Comput Sci 2 (2016): e74.
18. Qt (2024). https://www.qt.io/.
19. Eichhammer E. QCustomPlot (2024). https://www.qcustomplot.com/.
20. Intel oneAPI Toolkits (2024). https://www.intel.com/content/www/us/en/developer/tools/oneapi/toolkits.html.
21. Mishra SK. Global Optimization By Particle Swarm Method: A Fortran Program (2024). https://papers.ssrn.com/sol3/papers.cfm?abstract_id=921504.
22. Goffe WL, Ferrier GD, Rogers J. Global optimization of statistical functions with simulated annealing. J Econometrics 60 (1994): 65–99.
23. Morales JL, Nocedal J. Remark on “algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization.” ACM Transactions on Mathematical Software 38 (2011): 1–4.
24. Villaverde AF, Fröhlich F, Weindl D, et al. Benchmarking optimization methods for parameter estimation in large kinetic models. Bioinformatics 35 (2019): 830–838.
25. Riazoshams H, Midi H, Ghilagaber G. Robust Nonlinear Regression: With Applications Using R. [Internet]. Wiley (2019).
26. Hoops S, Sahle S, Gauges E, Lee. COPASI: a COmplex PAthway SImulator. Bioinformatics 22 (2006): 3067–3074.
27. OriginLab (2024). https://www.originlab.com/.
28. CurveExpert (2024). https://www.curveexpert.net/.

29. Prism (2024). https://www.graphpad.com/features.

30. LAB Fit (2024). https://www.labfit.net/.

31. GNU Octave (2024). https://octave.org/.

32. SciDAVis (2024). https://scidavis.sourceforge.net/.

33. Fityk (2024). https://fityk.nieto.pl/.

34. R/RStudio (2024). https://posit.co/download/rstudio-desktop/.