World Journal of Information Systems 2(1) 57-69
DOI: https://doi.org/10.17013/wjis.v2i1.33

Article

From data to ODEs on the Web: how can it be done and shown for execution ?

Miguel Casquilho [1,*], Pedro Pacheco [2] , Rui Galhano [1], Ivo Paulo [3], João L. de Miranda [4] and João Bordado [1]

[1] Department of Chemical Engineering, Instituto Superior Técnico, University of Lisbon, and CERENA, Lisboa, Portugal

[2] Department of Computer Science and Engineering, Instituto Superior Técnico, University of Lisbon, Lisboa, Portugal

[3] Department of Chemical Engineering, Instituto Superior Técnico, University of Lisbon, Lisboa, Portugal

[4] Department of Technologies, Polytechnic Institute of Portalegre, Portalegre, Portugal

***** Correspondence: mcasquilho@tecnico.ulisboa.pt

(Article history: Received December 20, 2024; Received in revised form January 18, 2025; Accepted February 19, 2025; Available online March 15, 2025)

Abstract: Fitting systems of ordinary differential equations (ODEs) to experimental data is a common task in Engineering. In the case of Chemical Engineering, this underlies chemical kinetics, addressed in this work. The ODEs studied are systems of (two or more) explicit equations of a single variable, time. The problem is a regression to determine the values of the parameters in the system. Here, we select the 'function' in the assumed Python module, solve the problem, and provide a freely accessible web page where data can be inserted. Thus: we offer the computation, based on Python tools, in a web page; we select the 'function', circumventing 'curve_fit' in favor of 'minimize'; we use PHP on the web page to call Python, with the 'gnuplot' graphing utility; and we stress the Internet as a computing medium. The computation runs on the server side, avoiding, from the user, any software installation, special power or operating system match. This Web operation, which runs on a Linux platform, is also an illustration for many other problems. Generally, about web computing, we advocate (i) it use in web pages, where it employs programs similar to classical ones, the programs being the inevitable difficulty, and (ii) it uses in scientific publications. In our technological era, this seemingly little explored field also promotes the academia industry link and invites knowledge interchange.

Keywords: Ordinary differential equations; regression; Web based computing; Python; scientific computing; Engineering education

1. Introduction

In many branches of Engineering, fitting mathematical models to experimental data is a common operation. It is also frequent in Chemical Engineering and a typical need in the chemical kinetics domain. The term “kinetics” means “the rate of change in a physical or chemical system” (“rate” as velocity). In chemical kinetics, “change” leads to ordinary differential equations, with a single independent variable, time.

In any program built to solve a problem, data are naturally susceptible to vary and meant to be supplied by the user. On a web page , the underlying program, on the contrary, is typically fixed. Changing it is neither easy nor our objective. Here, the program is related to the chemical reactions adopted [1], shown in Eq. (1), the “Base Problem” (a “Complementary Problem” will also be briefly solved) and leads to a system of ODEs detailed in the next Section.

$$\begin{aligned} \mathrm{A} + \mathrm{B} &\underset{\mathrm{k_2}}{\overset{\mathrm{k_1}}{\rightleftharpoons}} \mathrm{C} \\ \mathrm{C} + \mathrm{C} &\underset{\mathrm{k_4}}{\overset{\mathrm{k_3}}{\rightleftharpoons}} \mathrm{D} \end{aligned} \qquad (1)$$

This problem in ODEs will be solved in Python, a language (far from perfect) that is popular, free, platform-neutral, and provisioned with numerous “modules” to solve a wide variety of problems. One of the objectives of this work is, precisely, to select a convenient ‘function’ in a pertinent ‘module’, ‘scipy.optimize’, wherein we find two plausible candidates, ‘curve_fit’ and ‘minimize’. (We are writing ‘function’, in quotes, in the technical meaning of being a part of a Python ‘module’.)

Web-based computing, in our view, characterizes itself by providing resources made accessible simply through a web page, just using a browser, to solve a certain type of problem, where the user supplies data and obtains results. We describe a web based program to obtain the optimum values of the parameters in a system of ODEs, these values being its only unknowns.

Our topic is an instance of ‘Computer Science Engineering’ or ‘Informatics’, a term we adopted some years ago and has since become common. As a web-based solver, we mean a program needing no installations by the user, a difference from the innumerable applications that (free or paid) depend on an installation, compatible operating system, power, maintenance. The adherence to the Internet makes the user action easiest, and invites cooperation, as claimed by us for two decades (e.g., [2]–[7]), often in earlier occurrences of CISTI.

We provide a web page to solve concretely the problem of a system of ODEs (in kinetics, anyway general) since we encountered no similar. We did find a hint at [8] (with no computational web page) and it induced us to consider, without a thorough search for others, the choice between the two Python ‘functions’ mentioned.

Nowadays, people erroneously say “technological” as a snobbish, if parochial, alternative to “informatic” or “related to computer science”. More consequential, most scientific articles, lauding their numeric achievements, give the reader no way out to verify them.   Sometimes their programs are indeed offered, namely, in ‘github’ [9], but perhaps in languages not matching our own or in different operating system.   In the ‘FAIR’ proposal, Wilkinson et al. [10] say: “There is an urgent need to improve the infrastructure supporting the reuse of scholarly data.” [our emphasis] In the literature, the case is more worrisome with the programs than with data.   In a small step forward, the editor of a statistics journal [11] encourages “data and code sharing associated with newly submitted papers”, a timid advice, because it is code and data together that resolve problems.   In a complementary view, Park et al. [12] mention the flood of publications with little or no added value, obeying the “publish or perish” rule, of which one further side effect seems to be a deficit of computing.

In the next Sections: “The kinetics problem” explains the problem chosen; “Web applications” are briefly mentioned, given their paucity;   and “Results and discussion” show the architecture utilized and the results. .   Some final “Conclusions” are added.

2. The kinetics problem

The Base Problem ([1]) is shortly described.   Substances (chemical components) A and B are put in contact and react, producing C, which in turn produces D in the same reaction medium (the so-called “reactor”), in both cases reversibly (hence, the double arrows in Eq. (1)).   (There will be an additional comment on a Complementary Problem at the end of the paper.)   Theory and practice propose the ODEs in Eq. (2), where C is the concentration (such as g/L, grams per liter) of component C, D of component D, as describing the evolution of the concentrations versus time, t, and a0 and b0 are given initial concentrations of A and B.

$$\begin{cases} \dfrac{\mathrm{d}C}{\mathrm{d}t} = k_1(a_0 - C - 2D)(b_0 - C - 2D) \\ \qquad\quad - k_2 C - 2k_3 C^2 + 2k_4 D \\ \dfrac{\mathrm{d}D}{\mathrm{d}t} = k_3 C^2 - k_4 D \end{cases} \tag{2}$$

In the reactions, the “rate” (i.e., velocity) constants are the parameters the values of which are to be found: k_i, i = 1..n, n = 4 in this case. (There are 2 reactions and 4 parameters.)   The word “constant” is often applied loosely in science, and certainly the rate constants can vary (out of our scope), namely, with temperature.

In some cases, equations as those in Eq. (2) of course have analytical solution (as will happen in the Complementary Problem in the Section on results), but we consider the more adverse situation where numerical methods are indispensable.

Generally, to solve a “curve fitting” in Python, preference goes, we presume, to the ‘curve_fit’ ([13]) ‘function’.   This choice entails a formal restriction: the objective function to be adjusted must be a single dependent (or univariate) function of a single independent variable. This Python ‘function’ is surely practical and advantageous, as seen in a mere example of ours in our website, on adjusting a “hat” parabola to the central part of a Gaussian curve [14] or to other user-given data.

In fact, ‘curve_fit’ was the frontal choice to solve the problem at hand. The multiple dependent variables, however, would have to be converted into a structure in a single dependent variable. This conversion from the structure in Table 1 to the one in Table 2 appears simple but has some programming drawbacks.

Table 1. Multivariate dependent variables

	Time		*C*	*D*
1	0		0	0
2	7		1.065	0.0058
3	14		1.383	0.2203
…	…		…	---
10	63		0.6149	0.7219
11	70		0.3369	0.7294

Table 2. Univariate dependent variable

	Time	*C* & *D*
1	0	0
2	7	1.065
3	14	1.383
…	…	…
10	63	0.6149
11	70	0.3369
12	0	0
13	7	0.0058
14	14	0.2203
…	…	---
21	63	0.7219
22	70	0.7294

We have seen a couple of scripts implementing this strategy to permit the use of ‘curve_fit’. However, we found its complexity not rewarding in this case of multiple dependent variables, since there is a simpler alternative.

The approach adopted, otherwise conventional, is to minimize the error (discrepancy) between experimental and calculated values. Inside the same Python ‘scipy.optimize’ module, an alternative is the simple ‘minimize’ ‘function’. To apply it, a matrix of ycalc (Eq. (3)) homologous to Table 1, is computed with “reasonable” values of k, the vector of the unknown parameters. Thus, an objective function, which will be textually designated “cost”, say, z(k), in Eq. (3), is defined to be minimized.

$$z(\mathbf{k}) = \sum_{i=1}^{T} \sum_{j=1}^{K} \left[y_{i,j}^{\text{calc}}(\mathbf{k}) - y_{i,j}^{\text{exp}} \right]^2 \qquad (1)$$

with K the number of components with experimental data (K = 2), T the number of instants (T = 11), and y_{exp} and y_{calc} the experimental and calculated values, respectively, for each instant, t_i. (The mention to reasonable **k**, initial guess, reminds that numerical optimizations are typically subject to not converging.) This was the objective function adopted; nevertheless (out of our scope), a number of other criteria might be used, with the same Python solver. For example, let Eq. (4) define a "distance", δ, of each calculated value to the corresponding experimental value. Then, among others, the so called minimax criterion would be the minimization of w, in Eq. (5). The diverse criteria obviously give different, but certainly approximated, results.

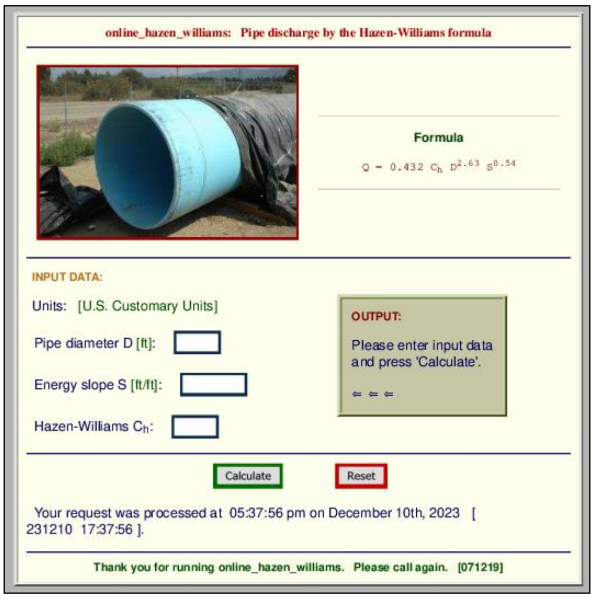
$$\delta_{i,j}(\mathbf{k}) = \left| y_{i,j}^{\text{calc}}(\mathbf{k}) - y_{i,j}^{\text{exp}} \right| \tag{2}$$

$$w = \max_{i,j} \delta_{i.j} \tag{3}$$

We detail in Section "Results and discussion" the computation, mainly in its setting on the Internet. This is readily accessible and, as the web page contains default data, immediately computable there, as detailed further.

3. Web applications

For this work, a web-based program does the determination of parameters in a system of ODEs. This points to the use by anyone, namely, students in STEM (Science, Technology, Engineering, Mathematics). A rare case of the Web as a computing medium we use to cite is V. M. Ponce's (below, [15]) in Hydraulics. Since 2000, his site is a "virtual laboratory", with a style similar to ours (since 1998). For a casual example of his, see Figure 1 ([15]).


online_hazen_williams: Pipe discharge by the Hazen-Williams formula

Formula

Q = 0.432 C_h $D^{2.63}$ $S^{0.54}$

INPUT DATA:

Units: [U.S. Customary Units]

Pipe diameter D [ft]:

Energy slope S [ft/ft]:

Hazen-Williams C_h:

OUTPUT:

Please enter input data and press 'Calculate'.

⇐ ⇐ ⇐

Your request was processed at 05:37:56 pm on December 10th, 2023 [231210 17:37:56].

Thank you for running online_hazen_williams. Please call again. [071219]

Figure 1 Ponce's Problem No. 151.

His web pages have no default data (his students have to know what to input…), whereas we always give default data for instant resolution, but our web pages and his do not discourage studying. From our experience, many Engineering students have quit programming, falling to poorly used Excel. Web pages to solve problems may persuade students to do their work and compare results. Other cases of ours appeared in previous communications, as said, and literature ([16]).

Here, we use Python with the ‘gnuplot’ graphing utility, under PHP. The preference for ‘gnuplot’ against the usual ‘matplotlib’ is an intention to generality: if the basic language has no graphical capabilities, for example, C, ‘gnuplot’ can be called with equal ease.

For Python, our version is 3.9.2, ‘gnuplot’ is 5.4. This runs on the public Linux system in our University, Debian 5.10.179-1 (2023-05-12) x86_64 GNU/Linux, 8 CPU’s, Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz. Python ([17]) might easily call C or Fortran for speed, if needed, because these are “compiled” (fast) languages, while Python is “interpreted”, although several of its elements are precisely built in compiled languages.

4. Results and discussion

On the Web, the computing is executable at

Ref. [18]

There, the user has the set of default data, and just clicks the “Execute” button or, previously, inserts other data (input) and gets the corresponding results (output). The results are built dynamically in HTML through the scripting language PHP, “native” for the Web [19].

The architecture is composed of: (i) a web based front-end, to show the user interface; and (ii) a computing back-end to execute the tasks. The front-end accepts the task’s arguments (data), processes them, schedules the task, and sends these data to the back-end computing system. The back-end finishes the task and replies to the front-end with the output in a dynamic new page. (The results might appear in the same page, through a PHP empty ‘action’, but that option seems less clear.) In the present type of problems, the output is simply alphanumeric with graphics, frequent in Engineering and usual in our problems, which are meant to solve them and make them available.

The problem is solved in Python, which has native graphical features, but we called the free graphical utility ‘gnuplot’ [20], which we prefer to Python modules like ‘matplotlib.pyplot’ as the ‘gnuplot’ general tool can be called with equal ease from non-graphical languages, such as C or Fortran.

The results of the problem are cast in HTML (essentially with the tag ‘pre’, i.e., pre-formatted text, intermixed with images), and shown to the user. The problem is run on the public system of CIIST, the university’s Computing Centre ([21]), on Linux (version already mentioned).

The architecture, with the names (bold, italics) for the particular case, is described:

a) The user addresses the problem's web page, P fitKinet.php, and uses default or inserts data in the fields of an HTML 'form' (data are sent to the next file through PHP POST variables) ("P" recalling a Python executable script).

b) Upon clicking the 'Execute' button, the data are sent to an intermediate PHP file, from 'action=FitKinet.php'.

c) This PHP file prepares the input received and sends it to the constructed executable script (next) as command line arguments. The PHP file takes the results from the execution and creates a (temporary, dynamic) PHP web page with the results.

d) The standalone executable, fitKinet.py, is run through PHP 'shell_exec'. This runs the calculations.

e) The output is sent to a temporary, dynamic, "Results" web page. This contains an HTML 'pre' tag with the text and graphics ('png') just generated (with a unique name) through a system call to Python 'base64' utility. Thus, no files are left on the server, needing no removal. This also avoids clashes between several users addressing the web page, an important aspect in public access.

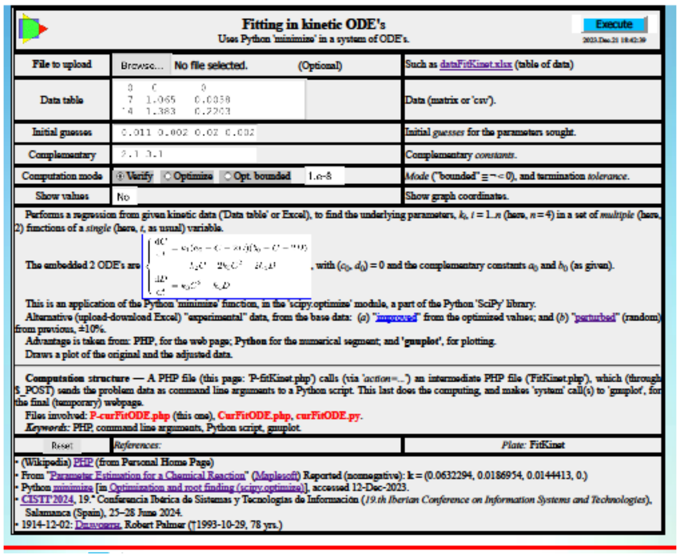
f) Auxiliary files: PHP environment files through 'include', i.e., cascading style sheet, and the images that characterize the website for consistency.

The Web environment adds little difficulty to the ordinary resolution of a scientific problem. Our setting permits to run the application also out of the Internet, facilitating the construction and debugging. In a technological problem, the definite hardship is to know the subject and methods to solve it.

The data for the problem in the web page are:

- (Optional) File to upload, similar to Table 1.
- Data table with default data (again as Table 1).
- Initial guesses for the sought parameters.
- Complementary constants for the computation.
- Computation mode: "verify", "optimize", "optimize bounded".
- Yes or No, "show values" of plot (also in an Excel file).

As default data exist, results appear with a click on the ‘Execute’ button. The input web page is depicted in Figure 2.



Fitting in kinetic ODE's
Uses Python 'minimize' in a system of ODE's.

File to upload	Browse... No file selected. (Optional)	Such as dataFitKinet.xlsx (table of data)

Data table	0 0 0
	7 1.065 0.0058
	14 1.383 0.2203
		Data (matrix or 'csv').

Initial guesses	0.011 0.002 0.02 0.002	Initial *guesses* for the parameters sought.
Complementary	2.1 3.1	Complementary *constants*.
Computation mode	[✓ Verify] Optimize Opt. bounded 1.e-8	*Mode* ("bounded" ≡ ¬< 0), and termination *tolerance*.
Show values	No	Show graph coordinates.

Performs a regression from given kinetic data ('Data table' or Excel), to find the underlying parameters, k_i, $i = 1..n$ (here, $n = 4$) in a set of *multiple* (here, 2) functions of a *single* (here, *t*, as usual) variable.

The embedded 2 ODE's are $\begin{cases} \frac{dC}{dt} = k_1(a_0 - C - 2D)(b_0 - C - [illegible]) \\ \quad k_2 C \quad 2k_3 C^2 \quad 2k_4 D \\ \frac{dD}{dt} = k_3 C^2 \quad k_4 D \end{cases}$, with $(c_0, d_0) = 0$ and the complementary constants a_0 and b_0 (as given).

This is an application of the Python 'minimize' function, in the 'scipy.optimize' module, a part of the Python 'SciPy' library.
Alternative (upload-download Excel) "experimental" data, from the base data: (*a*) "improved" from the optimized values; and (*b*) "perturbed" (random) from previous, ±10%.
Advantage is taken from: PHP, for the web page; Python for the numerical segment; and '**gnuplot**', for plotting.
Draws a plot of the original and the adjusted data.

Computation structure — A PHP file (this page: 'P-fitKinet.php') calls (via *'action=...'*) an intermediate PHP file ('FitKinet.php'), which (through $_POST) sends the problem data as command line arguments to a Python script. This last does the computing, and makes 'system' call(s) to 'gnuplot', for the final (temporary) webpage.
Files involved: **P-curFitODE.php** (this one), **CurFitODE.php**, **curFitODE.py**.
Keywords: PHP, command line arguments, Python script, gnuplot.

Reset	*References:*	*Plate:* FitKinet

• (Wikipedia) PHP (from Personal Home Page)
• From "Parameter Estimation for a Chemical Reaction" (Maplesoft) Reported (nonnegative): k = (0.0632294, 0.0186954, 0.0144413, 0.)
• Python minimize [in Optimization and root finding (scipy.optimize)], accessed 12-Dec-2023.
• CISTI'2024, 19.ª Conferencia Ibérica de Sistemas y Tecnologías de Información (*19.th Iberian Conference on Information Systems and Technologies*), Salamanca (Spain), 25–28 June 2024.
• 1914-12-02: Dnuvoana, Robert Palmer (†1993-10-29, 78 yrs.)

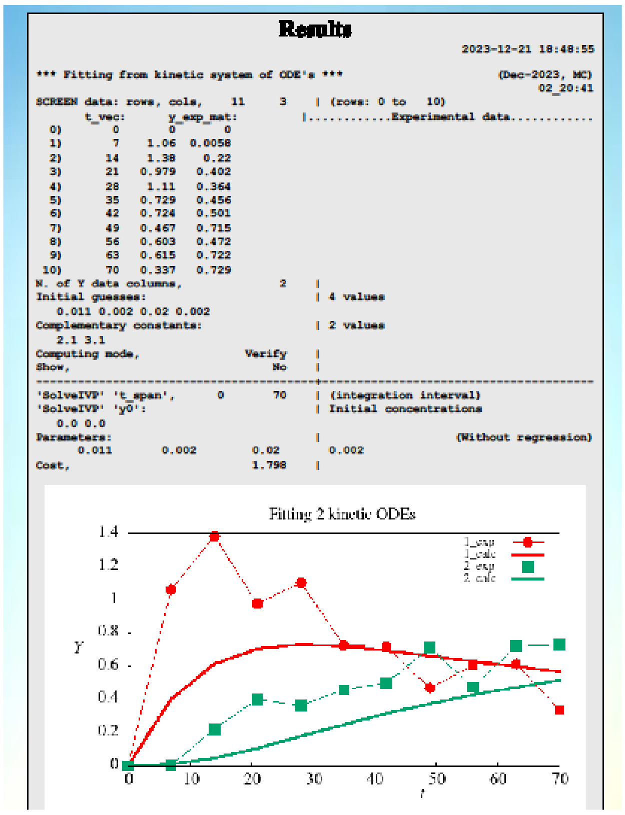
Figure 2 Web page (input) for the “fitting kinetic ODE’s” with the default options.

The default results appear in Figure 3. The computation mode is “verify”, without regression, i.e., the parameters are not calculated, their values being simply the initial guesses. Remark that, for clarity, the data used (from [1]) are “rough”, which is useful because in very friendly cases, the calculated values may visibly coincide with the experimental ones, confusing the two curves.

Running in (unconstrained) “optimize” mode (i.e., with regression), the results (graph only) are shown in Figure 4 with the calculated points adjusted to the (rough) experimental ones, and a cost of 0.21.

These rough data ([1]), we remark, even yield some physic–ally inconsistent parameter values: one of the rate constants is slightly negative. The accepted practice is to impose some kind of bounds to the values. To this end, another computing mode is available, the “optimization with bounds”, where the bounds are simply nonnegativity. The new results lead, of course, to a slightly worse adjustment, reflected in cost going from 0.21 to 0.23. The need for bounds prevents the use of the classical Nelder Mead ([22]) optimization method. So, the alternative SLSQP method [23], having this feature, was adopted.

Now, as noted in Section II, we use the “well behaved” data of Li et al. [24] as the Complementary Problem. The scheme is given in Eq. (6), where A is hemicellulose, B is furfural , and C a mix of degradation products, with the ODEs in Eq. (7).



Results

```
                                                    2023-12-21 18:48:55

*** Fitting from kinetic system of ODE's ***               (Dec-2023, MC)
                                                               02_20:41
SCREEN data: rows, cols,    11     3    | (rows: 0 to   10)
       t_vec:       y_exp_mat:         |............Experimental data............
  0)        0        0        0
  1)        7     1.06   0.0058
  2)       14     1.38     0.22
  3)       21    0.979    0.402
  4)       28     1.11    0.364
  5)       35    0.729    0.456
  6)       42    0.724    0.501
  7)       49    0.467    0.715
  8)       56    0.603    0.472
  9)       63    0.615    0.722
 10)       70    0.337    0.729
N. of Y data columns,              2    |
Initial guesses:                        | 4 values
   0.011 0.002 0.02 0.002
Complementary constants:                | 2 values
   2.1 3.1
Computing mode,               Verify    |
Show,                             No    |
----------------------------------------+----------------------------------------
'SolveIVP' 't_span',      0       70    | (integration interval)
'SolveIVP' 'y0':                        | Initial concentrations
   0.0 0.0
Parameters:                             |                     (Without regression)
      0.011       0.002        0.02       0.002
Cost,                          1.798    |
```

Figure 3 Results for the Base Problem, with the "verify" mode (i.e., without regression), giving a cost of 1.8.

This case has K = 3 chemical components (A, B, C) and n = 2 rate constants (k1, k2) that are to be determined. The problem is solved at another web page,

Ref. [25]

The results are shown in Figure 5 in "verify" mode (without regression), giving a cost of 4.5, and in Figure 6 in "optimize" mode (with regression), giving a cost near zero, 2e-5. In this case, by contrast with the Base Problem, the agreement is so complete that the experimental and the calculated curves are practically indistinguishable. This fact can be further evaluated in Figure 6 showing the residuals (calculated minus experimental).

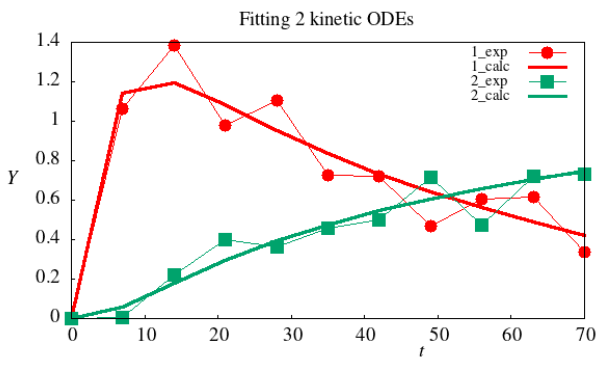


Figure 4 Results of the Base Problem, with the “optimize” mode (i.e., with regression), giving a cost of 0.21.

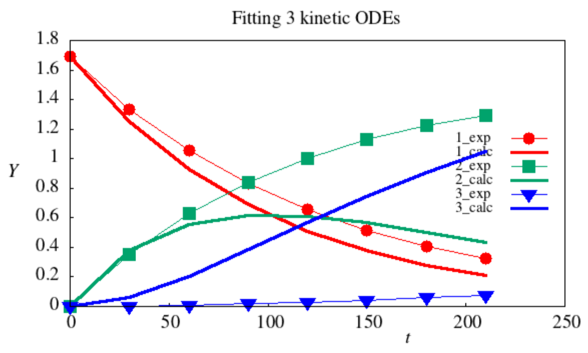


Figure 5 Results of the Complementary Problem, “verify” mode, with initial guesses 0.01, 0.01, giving a cost of 4.5.

The ODEs in this type of A-B-C reaction in Eq. (6) (incidentally, frequent in many contexts) are shown in Eq. (7), with the analytical solution in Eq.(8). In this situation, it is never k1 = k2, which would make components B and C identical. Anyway, when k2 □ k1, the mathematics is simple: remember L’Hôpital’s rule.

$$\mathrm{A} \rightarrow \mathrm{B} \rightarrow \mathrm{C} \qquad (4)$$

$$\begin{cases} \dfrac{\mathrm{d}A}{\mathrm{d}t} = -k_1 A \\ \dfrac{\mathrm{d}B}{\mathrm{d}t} = k_1 A - k_2 B \\ \dfrac{\mathrm{d}C}{\mathrm{d}t} = k_2 B \end{cases} \qquad (5)$$

$$
\begin{aligned}
A(t) &= \exp(-k_1 t) \\
B(t) &= \frac{k_1}{k_2 - k_1}\left[\exp(-k_2 t) - \exp(-k_1 t)\right] \\
C(t) &= \frac{1}{k_2 - k_1}\left\{\left[\exp(-k_1 t) - 1\right]k_2 - \left[\exp(-k_2 t) - 1\right]k_1\right\}
\end{aligned}
\qquad (6)
$$

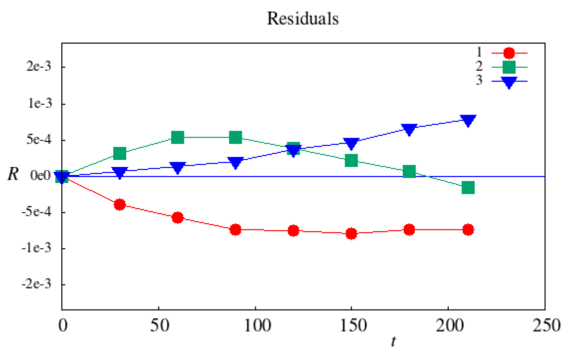


Figure 6 Regression residuals (very small, ~ -1e-3 to ~ +1e-3) for the Complementary Problem.

5. Conclusion

There is, we think, a little-explored road to use the Internet medium for scientific computing and in education. Another author's such computing is mentioned, and an adverse reality is the generality of numerical publications offering no web-based verification. This and other applications of ours just need a browser, and no software installation by the user. The illustrative problem is in kinetics, to fit calculated curves to data in a system of ODEs. It is set in PHP calling Python, with 'gnuplot', and is available for execution on the authors' website. We sought: to show this type of computing based on the client server model; to select a suitable but nonobvious Python 'function' to adjust the system to data; and to draw attention of students and authors to computing on the Internet environment. We intend soon to propose this tool to students. The limitations are the difficulty of certain kinetics, namely for stiff ODEs.

The problem discussed stresses the usefulness of the Web as a medium for scientific computing. The architecture proposed serves innumerable technical problems, for which the inevitable difficulty is their intrinsic mathematical nature, and the language used can be other, if effective on the Web, with adequate speed, security and licensing.

The programs and scripts in this approach are essentially the same as the ones used if run in a local computer, lending themselves to previous building and debugging. As we have verified in our academic and industrial practice, Web computing is recommendable for both purposes, favoring the connection and technology transfer between academia and industry.

Acknowledgments: MC does research at the Department of Chemical Engineering, IST, University of Lisbon, Lisbon, Portugal, and CERENA, "Centro de Recursos Naturais e Ambiente" (Centre for Natural Resources and the Environment), under Project UID/04028/2020, funded by FCT, "Fundação para a Ciência e a Tecnologia" (Portuguese National Science Foundation); PP is MSc student at the Dept. of Computer Science and Engineering, IST; RG teaches and does research at the Dept. of Chemical Engineering, IST, and CERENA; IP is PhD student at the Dept. of Chemical Engineering, IST, and CERENA; JLM teaches and does research at the Dept. of Technologies, ESTG, Polytechnic Institute of Portalegre, Portalegre, Portugal, and CERENA; and JB does research at the Dept. of Chemical Engineering, IST, and CERENA. CIIST, "Centro de Informática do IST" (Informatics Centre of IST) supplies the computing system.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] Maplesoft (2023), "Parameter Estimation for a Chemical Reaction", https://de.maplesoft.com/support/help/maple/view.aspx?path=applications%2FChemicalKineticsParameterEstimation , accessed 05-Jan-2024.

[2] Franco, B., Casquilho, M. (2011), "A Web application for scientific computing: combining several tools and languages to solve a statistical problem", CISTI'2011, 6.ª Conf. Ibérica de Sistemas e Tecnologias de Informação (6.th Iberian Conference on Information Systems and Technologies), Chaves (Vila Real), Portugal.

[3] Casquilho, M. (2013), "Computação científica, Internet, Indústria" (Scientific computing, Internet, Industry), 1.st Portuguese Meeting on Mathematics for Industry, FCUP, University of Porto, Porto (Portugal).

[4] Barros, M., Casquilho, M. (2019), "Linear Programming with CPLEX: an illustrative application over the Internet", CISTI'2019, Coimbra (Portugal).

[5] Casquilho, M., Buescu, J. (2022), "From unequal size sample sums to estimating the standard deviation", ICMASC 2022, International Conference on Mathematical Analysis and Applications in Science and Engineering, Porto (Portugal).

[6] Casquilho, M., Buescu, J. (2022), "Mean of Exponential distribution: estimation from sums of unequal size samples", COMPSTAT 2022, 24.th International Conference on Computational Statistics, Bologna (Italy), 23–26 Aug..

[7] Carolino, E., Casquilho, M. Ramos M. R., Barão, I. (2016), "Applied scientific computing over the Web: robust methods in Acceptance Sampling for Weibull variables", ISBIS 2016 Meeting on Statistics in Business and Industry, 08–10 Jun., Barcelona (Spain).

[8] Github, A. Ford Versypt, L13 "Parameter estimation in Python", https://github.com/ashleefv/ApplNumComp/ , accessed 05-Jan-2024.

[9] Github, https://github.com/ , accessed 05-Jan-2024.

[10] M. D. Wilkinson (2016), "Comment: The FAIR Guiding Principles for scientific data management and stewardship", Scientific Data, 3:160018, 9 pp.

[11] Colosimo, B. M., ed. (2019), Message, J. of Quality Technology, 51 (1), 1–2 or https://www.tandfonline.com/doi/full/10.1080/00224065.2019.1569896., accessed 05-Jan-2024.

[12] Park, M., Leahey, E., Funk, R. (2022), "Papers and patents are becoming less disruptive over time", Nature, 613, 5 Jan 2023, pp 138–144 (doi: https://www.nature.com/articles/s41586-022-05543-x).

[13] Python, 'curve_fit', "scipy.optimize.curve_fit", https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html , accessed 05-Jan-2024.

[14] Casquilho, M., "From Gaussian to a parabola", http://web.tecnico.ulisboa.pt/~mcasquilho/compute/explore/GaussToParab/P-GaussToParab.php .

[15] Ponce, V. M., Problem 151, https://ponce.sdsu.edu/onlinehazenwilliams.php , accessed 05-Jan-2024.

[16] Casquilho, M., Buescu, J. (2022), "Standard deviation estimation from sums of unequal size samples", Monte Carlo Methods and Applications (doi: 10.1515/mcma-2022-2118).

[17] The SciPy community, "Using Python as glue", https://numpy.org/doc/stable/user/c-info.python-as-glue.html , accessed 05-Jan-2024.

[18] Casquilho, M., "Fitting in kinetic ODEs", http://web.tecnico.ulisboa.pt/~mcasquilho/compute/explore/fitKinet/P-fitKinet.php .

[19] PHP, The PHP Group, https://www.php.net/ , accessed 05-Jan-2024.

[20] Gnuplot, http://www.gnuplot.info/ , accessed 05-Jan-2024.

[21] CIIST, "Centro de Informática do IST" (Informatics Centre of Instituto Superior Técnico), https://ciist.ist.utl.pt/eng.php , accessed 05-Jan-2024.

[22] Nelder, J., Mead, R. (1965), "A simplex method for function minimization". Computer Journal. 7(4), 308–313 (doi: https://doi.org/10.1093%2Fcomjnl%2F7.4.308)

[23] Kraft, D. (1988), "A software package for sequential quadratic programming", http://yetanothermathprogrammingconsultant.blogspot.com/2022/02/slsqp-original-paper.html , accessed 05-Jan-2024.

[24] Li, X., Yang, J., Xu, R., Lu, L., Kong, F., Liang, M., Jiang, L-. Nie, S., Si, C. (2019), "Kinetic study of furfural production from Eucalyptus sawdust using H-SAPO-34 as solid Brønsted acid and Lewis acid catalysts in biomass-derived solvents", Industrial Crops & Products, 135, 196–205 (doi: 10.1016/j.indcrop.2019.04.047).

[25] Casquilho, M., "Furfural: fitting in kinetic ODE's", http://web.tecnico.ulisboa.pt/~mcasquilho/compute/explore/Furfural/P-furfural.php.