DATTES: Data Analysis Tools for Tests on Energy Storage

Eduardo Redondo-Iglesias, Marwan Hassini, Pascal Venet, Serge Pelissier

DATTES: Data Analysis Tools for Tests on Energy Storage

Eduardo Redondo-Iglesias[1,3,*], Marwan Hassini[1,2,3], Pascal Venet[2,3], Serge Pelissier[1,3]

[1] Univ Lyon, Univ Eiffel, ENTPE, LICIT-ECO7 Lab, 69500 Bron, France
[2]Univ Lyon, Université Claude Bernard Lyon 1, INSA Lyon, Ecole Centrale de Lyon, CNRS, Ampère, UMR5005, 69622 Villeurbanne, France
[3]ERC GEST (Eco7/Ampère Joint Research Team for Energy Management and Storage for Transport

*corresponding author(s): (eduardo.redondo@univ-eiffel.fr)

Abstract

Experiments are essential to understand the behaviour and performance of energy storage systems. In this field, a considerable amount of experimental data is generated and data processing is a tedious task. To date, research teams working in the field of energy storage tend to focus on developing their own analysis tools rather than using existing open source software. This strategy can be detrimental to the quality and reproducibility of the research. This paper presents DATTES, a free and open source software for analysing experimental battery data. The software provides a comprehensive and customizable toolkit for extracting, analysing and visualizing experimental data. It also creates gateways to other open software and tools. In this way, DATTES enables users to get the most out of their experimental data and engage in open and reproducible science.

Keywords

Energy Storage , Experiments , Matlab , GNU Octave , Data analysis , Open Science , FAIR

Current code version

Nr.	Code metadata description	Please fill in this column
C1	Current code version	DATTES 23.05
C2	Permanent link to code/repository used for this code version	https://github.com/e-redondo/dattes
C3	Code Ocean compute capsule	
C4	Legal Code License	GNU GPL v3
C5	Code versioning system used	git
C6	Software code languages, tools, and services used	Matlab/GNU Octave
C7	Compilation requirements, operating environments & dependencies	Matlab or GNU Octave on Linux, OSX or Windows.
C8	If available Link to developer documentation/manual	https://dattes.gitlab.io/
C9	Support email for questions	dattes@univ-eiffel.fr

Table 1: Code metadata (mandatory)

1 Motivation and significance

Data processing plays an increasingly important role in scientific studies. Software such as MATLAB, Python, or R is at the heart of this activity, and according to a study by Hettrick et al., 70 % of scientists acknowledge that their research would not be possible without it [1]. This activity is also time-consuming for scientists, who typically spend 30 percent or more of their research time on it.

In the field of energy storage, a wide variety of tools and methodologies are used to generate experimental data. As a result, the experimental data produced is difficult to compare. In addition, most research teams have developed their own data processing tools adapted to their specific needs and data sets. Although this organisation can be explained by the need of researchers to quickly process the data of their projects, such a development strategy limits the reproducibility of the work and

tends to significantly increase the time spent by researchers on programming. Furthermore, since the vast majority of researchers are self-taught developers, they may lack experience with good software development practices. As a result, the quality of the research may be compromised. Comparison between studies from two different laboratories is also hampered because each team is likely to use its own methodology for generating and processing data. This lack of common data processing tools is a major barrier to sharing and comparing experimental results [2]. Open source software has recently gained increasing interest in the field as it allows for the sharing of programming efforts, best practices, and analysis methodologies. In the current landscape of open source software for energy storage systems, Python dominates. BEEP, cellpy, impedance.py or Pybamm are some examples of popular software in the field of batteries [3, 4, 5]. They are all written in Python. This monopoly can be explained by the qualities of this versatile programming language. Python is free and open source, it emphasizes the readability of the code and it comes with a very complete library of tools for data analysis, such as Scipy, Numpy or Matplotlib. However, although this language is slowly gaining interest in the energy storage community, MATLAB remains by far the most popular language. For example, Google Scholar has indexed more than 200,000 publications that mention the keywords "battery" and "MATLAB". This is three times more than for "Python". MATLAB's popularity is due to the variety of applications it offers. The use of matrices and arrays makes it easy to use in scientific problem solving. Finally, MATLAB is the basis for Simulink, a block diagram environment for modeling, simulation, and analysis of multiphysics dynamic systems.

Despite the popularity of MATLAB, no open software for processing experimental data has been offered in this language. This article introduces DATTES (Data Analysis Tools for Tests on Energy Storage), the first open software , as far as we know, written in MATLAB for processing experimental battery data.

2 Software description

DATTES is an open source software written in MATLAB code and compatible with GNU Octave that aims to facilitate data analysis for energy storage systems. Since these programming languages are very popular in the field, the software can enable a large part of the energy storage community to use an open data processing tool.

2.1 DATTES workflow

DATTES aims to transform the raw experimental data into valuable processed results that can be used for visualisation, characterisation and modelling. DATTES have been designed as a modular framework which architecture can be divided into four steps: *dattes_import*, *dattes_structure*, *dattes_configure* and *dattes_analyse*.

2.1.1 dattes import

Energy storage research suffers from data quality and format heterogeneity. The first step in this process is to standardize the experimental data. The function *dattes_import* is used to convert the raw data from a battery cycler into a standard format, the XML format following the structure defined in the open source VEHLIB [6] library.

2.1.2 dattes structure

The second step is to enrich the experimental data with metadata and to structure the information. Metadata allow the enrichment of databases, for example during ageing campaigns, with data about each experiment (experimenter, equipment used, etc.) or about the battery (nominal characteristics, cell identifier, etc.). During this step, the function *dattes_structure* allows to divide the experiment into characteristic phases according to the different modes of battery stress: rest, constant current (CC), constant voltage (CV), etc. This function also makes it possible to determine the state of charge (SoC) of the battery at any moment of the experiment.

2.1.3 dattes configure

In the third step of data processing, the user can adapt the analysis methodology to his needs. The function *dattes_configure* allows to modify the analysis configuration defined by default in DATTES. The default configuration consists in identifying and determining the characteristics of the batteries wherever possible. Configuring the impedance analysis method allows the user to choose to work only with current pulses of a desired duration or to select the type of equivalent circuit model to identify.

2.1.4 dattes analyse

Finally, the fourth step is the data analysis step. By using the function *dattes_analysis*, the user can analyse the characteristic magnitudes of a battery. The capacity can be determined in charge or discharge on a constant current phase associated or not with a constant voltage phase. The resistance of the battery can be calculated on charging or discharging current pulses with different times (R_{2sec}, R_{10sec}, etc.). DATTES enable impedance analysis by a current pulse (time-domain model identification) or by the frequency method of Electrochemical Impedance Spectroscopy (EIS). It also makes possible to analyse the behaviour of the battery at low current, either by identifying the Open Circuit Voltage (OCV) through partial charge/discharge (OCV by points) or through low current cycles (pseudo-OCV). The low current cycles can also be used for Incremental Capacity Analysis (ICA) and Differential Voltage Analysis (DVA).

2.2 DATTES auxiliary tools

2.2.1 dattes save and dattes load

The results of the DATTES workflow can be saved to **.mat* files using the *dattes_save* function, or by adding the *'s'* execution parameter to previous DATTES operations (*dattes_structure*, *dattes_configure*, *dattes_analysis*). This allows the user to interrupt the DATTES workflow at any time and resume it later by loading the result stored in the **.mat* file using the *dattes_load* function.

2.2.2 dattes plot

The function *dattes_plot* allows user to visualize the results of the different stages of the DATTES workflow.

2.2.3 dattes export

The function *dattes_export* allows to store the analysis results into various files formats. This facilitate the DATTES interoperability with other existing software in the field. By default, the DATTES structure is saved as **.mat* file but other standard file formats are possible such as **.csv* or **.json* formats.

3 Illustrative Examples

This section illustrates the DATTES workflow over a sample experimental dataset. The code corresponding to this example is available by running the function *demo_dattes*. Data in this dataset correspond to a characterisation experiment described in preceding works [7, 8, 9].

The experimental raw data is output from a Biologic cycler. The function *dattes_import* allows to convert these raw data into a standard format, i.e. XML.

Then the function *dattes_structure* allows to extract the characteristic values of the battery (time, current, voltage, etc.) and to structure the experiment in phases depending on the cycler mode (CC, CV, rest, etc.). At this level, metadata can be included in the experimental data using JSON files (**.meta* extension). Figure 1 shows each phase number during the experiment from the previously identified modes. Dividing the experiment into phases allows the user to adapt the analysis method(s) to each part of the experiment.

In this characterisation experiment, five parts were defined. Different analysis methods were applied to each part. The first part of the experiment includes phases 1 to 6. It consists of several partial discharges to ensure that the following part starts at a state of charge of 0%. The second part (phases 7 to 21) corresponds to several charge/discharge cycles to measure the battery capacity. The third part covers phases 22 to 182 and consists of a sequence of current pulses at different SoC levels (10, 20, 30,..., 90%). This part can be used to measure the battery impedance across the different SoC levels. The fourth part (phases 183 to 185) is a cycle at a low current rate. Finally, the fifth part (phases 186 and 187) is a battery partial discharge to avoid storing the battery at full charge after the experiment.

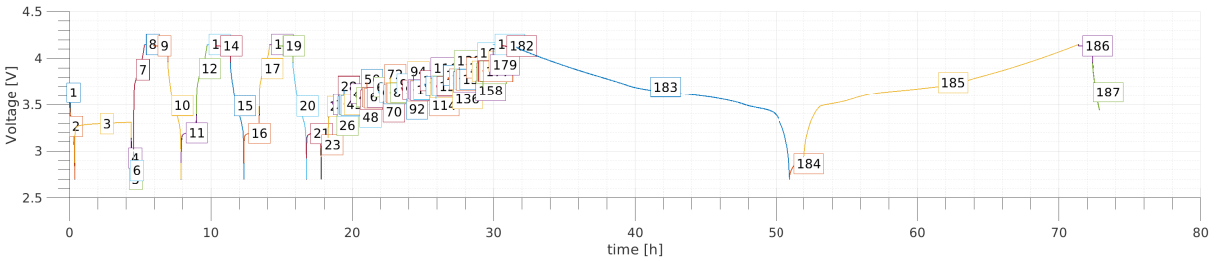

Figure 1: Experiment decomposition into phases according to the cycler mode.

In this example, full charges are performed with the standard CC+CV protocol and full discharges are performed with CC to the minimum cut-off voltage U_{min}. Phases 7 to 20 and 183 to 185 can be used to measure battery capacity at 1C and C/20. DATTES will detect these phases in the step *dattes_configure* and this will be used in the next step *dattes_analyse* with the parameter 'C' (analyse capacity).

DATTES also allows the analysis of the equivalent resistance over a constant current pulse, when providing 'R' parameter to *dattes_analyse*. The equation 1 shows that the equivalent resistance ($R_{\Delta t}$) is calculated as the ratio of the voltage drop and the current pulse amplitude at a given time after the pulse start (Δt). By default, the resistance is calculated with Δt equal to 2 and 10 seconds. If necessary, the function *dattes_configure* allows the user to define the value(s) of Δt of his choice, default configuration sets two values .

$$R_{\Delta t} = \frac{V(t_{pulse} + \Delta t) - V(t_{pulse})}{I_{pulse}} \tag{1}$$

Figure 2 shows the result of the resistance analysis. The blue and red markers show the 2-second and 10-second resistances, respectively, while the up and down triangles indicate the resistance measured during the charge and discharge pulses, respectively. Figure 1 shows that some parts of the experiment can be used to measure resistance. Phases 4 to 20 are examples of phases where

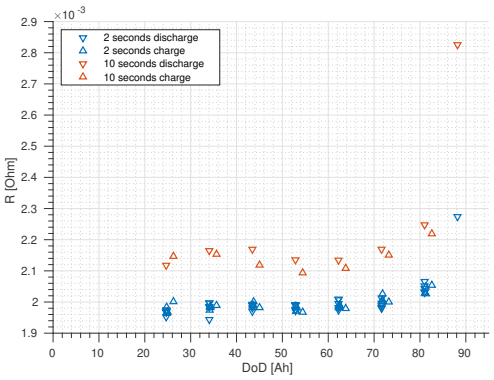

Figure 2: Equivalent resistance at 2 and 10 seconds versus depth of discharge (DoD).

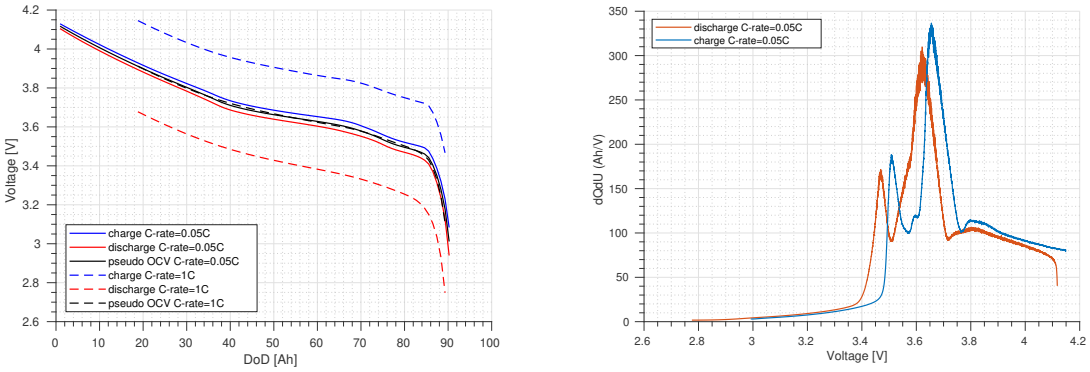

Figure 3: Pseudo OCV (left) and ICA (right) plots. In pseudo OCV plot line colors are: red for discharge half cycle, blue for charge half cycle, black for pseudo OCV; continuous lines for C/20 cycle, dashed lines for 1C cycle. In ICA plot line colors are: red for discharge half cycle, blue for charge half cycle.

resistance can be measured (at beginning of each charge/discharge half cycle of capacity measurement), although this was not necessarily the original intention of the experimenter. To solve this problem, the *dattes_configure* function allows the user to limit the search range of the current pulses used for the resistance calculation. Other restrictions can also be set, such as minimum/maximum pulse duration or minimum rest period before a pulse.

Phases 183 to 185 have been identified as relevant for characterising the low-current behavior of the battery. DATTES allows the analysis of the open circuit voltage (OCV), as shown in Figure 3 (left plot). In this figure, two cycles with a current regime of 1C and C/20 have been taken into account. Thanks to the *dattes_configure* function, the user can limit the analysis of low current phases by defining a maximum current threshold above which the phases are not taken into account. Figure 3 (right plot) shows the results of an incremental capacitance analysis (ICA). For this type of analysis, the filter type is an example of a user-definable parameter [10].

4 Impact

While DATTES stands out as the first MATLAB software to support the entire data processing chain for battery experiments. Some of its features are also unique.

The number and variety of input data formats that DATTES can handle is very large. Currently, data produced by the Arbin, Bitrode, Biologic, Digatron, and Neware cyclers are supported by DATTES, and other data formats can be easily adapted. DATTES also provides conversion of raw experimental data into a standard format, XML, following the structure defined in the VEHLIB library. DATTES is thus natively compatible with the energy management software VEHLIB [6].

To date and at our best knowledge, DATTES is the open source experimental data processing software that offers the largest number of analysis tools. One of the reasons for this ability to handle different types of experiments is the method of experiment segmentation according to battery usage phases. This facilitates the analysis of patterns suitable for each type of analysis (e.g. impedance, pseudo-OCV or ICA, etc.).

Special attention has also been paid to data traceability. Raw data are first structured and enriched with metadata using **.meta* files (*dattes_structure*), e.g. nominal cell properties, experiment date/time/equipment, etc.). After that, each step of the DATTES workflow (*dattes_configure*, *dattes_analyse*) is traceable. The result delivered by DATTES contains information about each analysis method (current levels, phase selection conditions, signal filtering parameters, etc.). The DATTES result structure contains raw experiment profiles (date, time, current, voltage, etc.), metadata, analysis results, and analysis configuration. This data structure is fully documented [10].

Finally, DATTES facilitates the reuse of the analysis data produced. The function *dattes_export*

allows to convert the analysis results into different formats such as **.csv*, **.json* or **.m*.

This software has been developed according to Free and Open Source Software (FOSS) and FAIR [11] principles, and considerable effort has been made to write clear and complete documentation . The developers hope that the energy storage community will take advantage of the software, as new users are seen as potential reviewers who can catch bugs, ask for new features, and accelerate the rise of computing standards.

Finally, DATTES is designed to be modular and easily coupled with other existing software. It can enable the energy storage community to design better batteries, build robust models, and make more accurate predictions with standardized, peer-reviewed methodologies.

5 Conclusions

Processing experimental data is a central task in the daily life of energy storage researchers. Until now, most research teams have focused on proprietary tools, which severely limits the reproducibility of research. The lack of open tools available for MATLAB , the most popular language in the field, has been a major obstacle to the widespread use of open tools. This paper presents DATTES, the first experimental data processing software for batteries written in MATLAB code and distributed under an open license. The software provides a comprehensive and customisable toolkit for extracting, analysing, and visualising experimental data. It also provides gateways to other open software and tools. DATTES thus enables users to make the most of their experimental data and to engage in open and reproducible science.

6 Conflict of Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

References

[1] S. Hettrick, 2014:Software in research survey (2018). doi:10.5281/zenodo.1183562.

[2] M. Hassini, E. Redondo-Iglesias, P. Venet, Lithium-ion battery data: From production to prediction, under review (2023).

[3] P. Herring, C. B. Gopal, M. Aykol, J. H. Montoya, A. Anapolsky, P. M. Attia, W. Gent, J. S. Hummelshøj, L. Hung, H.-K. Kwon, et al., Beep: A python library for battery evaluation and early prediction, SoftwareX 11 (2020) 100506. doi:10.1016/j.softx.2020.100506.

[4] V. Sulzer, S. G. Marquis, R. Timms, M. Robinson, S. J. Chapman, Python battery mathematical modelling (pybamm), Journal of Open Research Software 9 (1) (2021). doi:10.5334/JORS.309.

[5] M. D. Murbach, B. Gerwe, N. Dawson-Elli, L.-k. Tsui, impedance. py: A python package for electrochemical impedance analysis, Journal of Open Source Software 5 (52) (2020) 2349. doi:10.21105/joss.02349.

[6] VEHLIB.
URL https://gitlab.univ-eiffel.fr/eco7/vehlib

[7] M. Hassini, E. Redondo-Iglesias, P. Venet, S. Gillet, Y. Zitouni, Second Life Batteries in a Mobile Charging Station: Model Based Performance Assessment, in: EVS35, 35nd International Electric Vehicle Symposium & Exhibition, Oslo, Norway, 2022.
URL https://hal.science/hal-03708744

[8] M. Hassini, E. Redondo-Iglesias, P. Venet, S. Gillet, Y. Zitouni, Second Life Batteries in a Mobile Charging Station: Experimental Performance Assessment, working paper or preprint (2022).
URL https://hal.science/hal-03713844

[9] M. Hassini, E. Redondo-Iglesias, P. Venet, Second-life batteries modeling for performance tracking in a mobile charging station, World Electric Vehicle Journal 14 (4) (2023) 94. doi:10.3390/wevj14040094.

[10] DATTES : Data Analysis Tools for Tests on Energy Storage.
URL https://gitlab.com/dattes/dattes/

[11] R. C. Jiménez, M. Kuzak, M. Alhamdoosh, M. Barker, B. Batut, M. Borg, S. Capella-Gutierrez, N. C. Hong, M. Cook, M. Corpas, et al., Four simple recommendations to encourage best practices in research software, F1000Research 6 (2017). doi:10.12688/f1000research.11407.1.