The Journal of Open Source Software

Cellpy – an open-source library for processing and analysis of battery testing data

Julia Wind [1], **Asbjørn Ulvestad** [1], **Muhammad Abdelhamid** [1], and **Jan Petter Mæhlen** [1]

1 Institute for Energy Technology, 2007 Kjeller, Norway

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Mojtaba Barzegari

Reviewers:

- @MaximevdHeijden
- @TomTranter

Submitted: 24 October 2023
Published: unpublished

Summary

Recent years have witnessed an exponential increase in battery research, driven by the need to develop efficient and sustainable energy storage systems. One of the main tools in battery research are battery cycling experiments, providing insights into performance, lifetime and quality of the battery. Due to the large variety of battery testing equipment and the resulting multitude of different and often proprietary data formats, combined with the large number of parameters involved, managing and processing battery testing data has often been a difficult and tedious task.

The Python library `cellpy` assists in solving these problems by

1. providing the tools to read different data formats,
2. converting those into one common data format that also includes relevant battery-specific meta-data, and
3. providing a data structure equipped with a set of methods that helps the user to easily perform simple and in-depth analyses of both single data sets and collections of data sets.

Statement of need

Typically, a battery-testing data set consists of simple time series data with voltage, current and capacity. Unfortunately, data from different equipment are measured and handled in different ways and stored in different, often proprietary, formats. Consequently, a direct and meaningful comparison of several cells tested under a variety of conditions can be challenging and requires more advanced data handling methodologies. Several open-source libraries focus on battery test-data extraction. However, most of them are dedicated to specific battery testing equipment: notably galvani (Echemdata, 2022) parses the proprietary BioLogic format, neware_reader (Beyonder et al., n.d.) parsing several versions of Neware data, and galv (formerly Galvanalyser) (Battery-Intelligence-Lab, 2023) supporting Maccor, Ivium and BioLogic formats. BEEP (Battery Evaluation and Early Prediction (Herring et al., 2020)) provides a structured interphase for collecting and processing battery test data and exports to text format.

`cellpy` provides powerful and versatile tools for the simple and efficient handling of battery testing data originating from different battery cell testers, all the way from data collection to data analysis and visualisation, ensuring consistency, accuracy and comparability. `cellpy` can directly parse the data from most common commercial battery testers (Arbin, Maccor, PEC, Neware, BioLogic), in addition to offering full flexibility by allowing the user to provide other file format specifications (in YAML format). The data is converted into and saved in a common format, accommodating not only data from diverse testers, but also thoughtfully embeding battery-specific metadata (*e.g.*, step-types, type of cell, type of chemistry, electrode properties, etc.). This makes subsequent data handling considerably easier and proves invaluable in interpreting and comparing results across tests and conditions. In addition to translating data to a common format, `cellpy` has a range of utilities for studying and analysing the data. These include methods for the extraction of key characteristics from tests, cell comparison, plotting and statistical analysis, as well as advanced tools such as incremental-capacity analysis (ICA, dQ/dV), OCV relaxation analysis and batch processing of results from many battery test (Andersen et al. (2019), Ulvestad et al. (2020), Huld et al. (2023), Spitthoff et al. (2023)).

The `cellpy` library provides a valuable toolset and has been in frequent use for both everyday and advanced tasks in battery research. The ability to effortlessly import and process the data through a simple but highly flexible API allows for quick and simple comparison of different cells. At the same time, `cellpy` serves as an excellent starting point for researchers leaning towards advanced analysis: `cellpy` can automatically convert data with different units, summarize and perform statistical evaluations all the way down to the individual cycle and step level, while giving the user fine-grained control of the behaviour through setting of parameters or directly by using a more advanced, deeper API. This eases further use of the data, *e.g.*, as features for machine learning algorithm, and promotes reproducibility and traceability throughout the entire process.

Implementation and architecture

`cellpy` is implemented in python and can be used as either a library within python scripts, or as a stand-alone application for analysing battery cell test data. Internally, `cellpy` utilises the rich ecosystem of scientific tools available for python. In particular, `cellpy` uses pandas DataFrames as the "storage containers" for the collected data within the `cellpy` Data object. This offers full flexibility and makes it easy for the user to apply advanced methods, analyses of or transformations to the data in addition to the features implemented in `cellpy`.

The core of `cellpy` is the **CellpyCell** object (see Figure 1) that contains both the data (stored in the **Data** object) as well as central methods required to read, process and store battery testing data. The CellpyCell provides the appropriate interface and coordination of the resources needed, such as loading configurations (*e.g* default reader, default raw-data location), selecting readers for different data formats and exporters for saving the data.

The Journal of Open Source Software

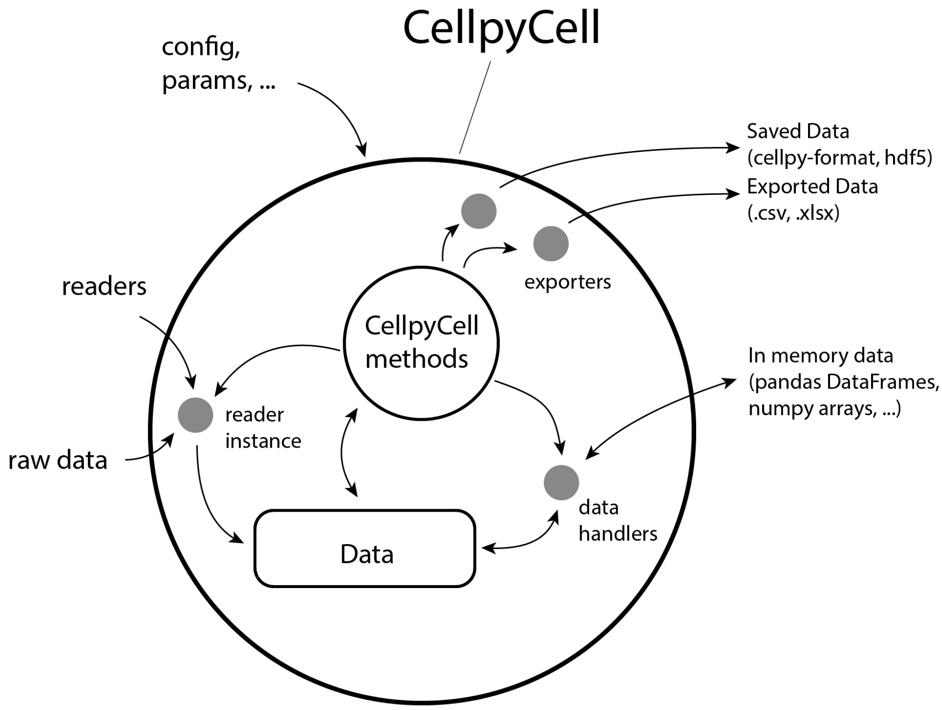

Figure 1: Illustration of the core object within `cellpy`, the **CellpyCell**.

The **CellpyCell Data** object stores both the battery test data as well as the corresponding meta data (see Figure 2). In addition to the central DataFrame containing the raw data (*raw*), the DataFrames *steps* and *summary* provide step- (*e.g.*, maximum current, mean voltage, type-of-step *vs.* step number) and cycle-based (*e.g.*, gravimetric charge capacity, coulombic efficiency, C-rates *vs.* cycle number) summaries and statistics respectively.

The Journal of Open Source Software

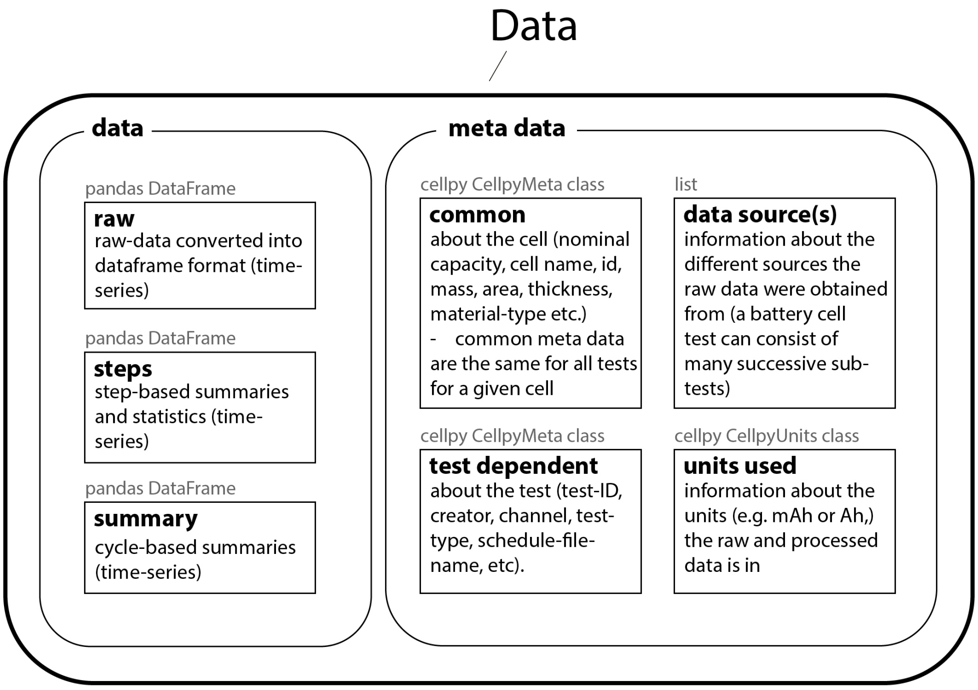

Figure 2: Summary of the types of contents in a **CellpyCell Data** object.

The most common data processing routines, such as extraction of charge/discharge voltage curves in different formats or selecting data for specified step-types, are implemented as methods on the CellpyCell object. In addition, the `cellpy` library also consists of a rich set of utilities (Figure 3) that can be used for further processing the data, both individually and within batch routines. Utility functions include *e.g.*, ICA tools, assisting in creating dQ/dV graphs (employing different data-smoothing algorithms), or tools for OCV relaxation analysis.

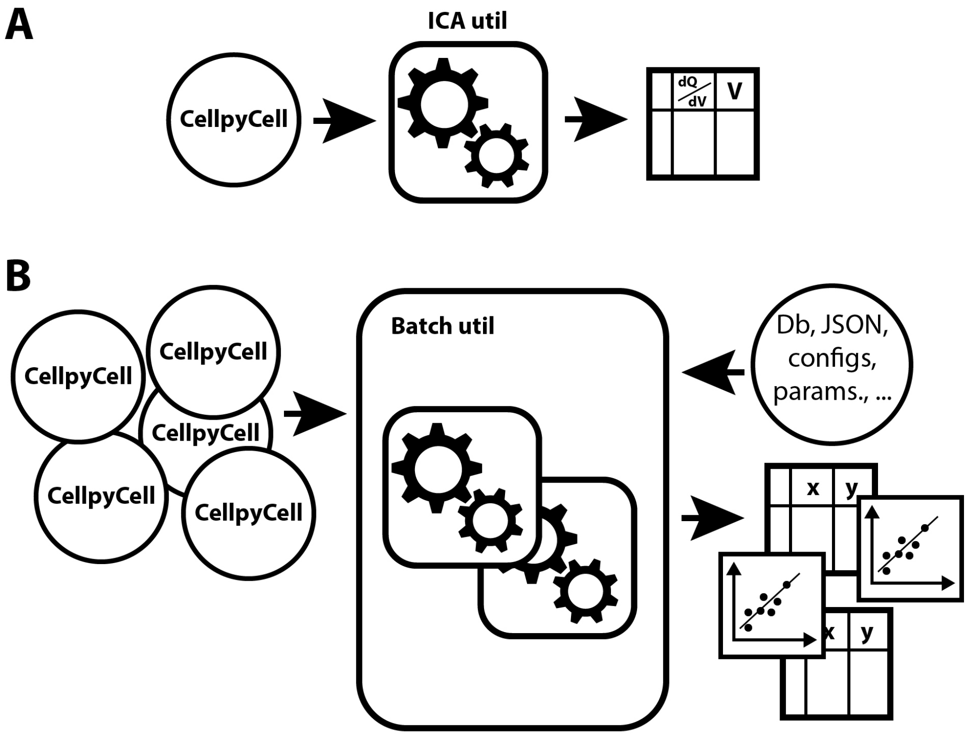

Figure 3: The `cellpy` library contains multiple utilities that assists in data analysis. A utility can work on (A) a single **CellpyCell** object, or (B) a set of CellpyCell objects such as the Batch utility that helps the user in automating and comparing results from many data sets.

The `cellpy`-file format (usually stored in hdf5 format) contains all the data contained in the Data object together with additional relevant meta data, including information about the file version.

Acknowledgements

The development of `cellpy` was supported by the Research Council of Norway through the ENERGIX Projects No.280985 ("KPN Silicon on the Road"), No.324077 ("KSP MoreIsLess"), No.320760 ("KSP SecondLife"), No.326866 ("KSP LongLife"), No.344317 ("KSP CellMap"), and FME-MoZEES (project No. 257653), co-sponsored by the Research Council of Norway and 40 partners from research, industry, and the public sector. The development was also supported through the EU-funded SIMBA project (HORIZON 2020, GA No. 963542).

The authors are thankful to the numerous inputs and comments from our colleagues and collaborators, and in particular Dr. Preben J.S. Vie and Dr. Martin Kirkengen.

References

Andersen, H. F., Foss, C. E. L., Voje, J., Tronstad, R., Mokkelbost, T., Vullum, P. E., Ulvestad, A., Kirkengen, M., & Mæhlen, J. P. (2019). Silicon-carbon composite anodes from industrial battery grade silicon. *Scientific Reports*, *9*(1), 14814. https://doi.org/10.1038/s41598-019-51324-4

Battery-Intelligence-Lab. (2023). Galv. In *GitHub repository*. GitHub. https://github.com/Battery-Intelligence-Lab/galv

Beyonder, Corvus Energy, & IFE. (n.d.). Neware reader. In *GitHub repository*. GitHub. https://github.com/FTHuld/neware_reader

Echemdata. (2022). Galvani. In *GitHub repository*. GitHub. https://github.com/echemdata/galvani

Herring, P., Gopal, C. B., Aykol, M., Montoya, J. H., Anapolsky, A., Attia, P. M., Gent, W., Hummelshøj, J. S., Hung, L., Kwon, H.-K., Moore, P., Schweigert, D., Severson, K. A., Suram, S., Yang, Z., Braatz, R. D., & Storey, B. D. (2020). BEEP: A python library for battery evaluation and early prediction. *SoftwareX*, *11*, 100506. https://doi.org/10.1016/j.softx.2020.100506

Huld, F. T., Mæhlen, J. P., Keller, C., Lai, S. Y., Eleri, O. E., Koposov, A. Y., Yu, Z., & Lou, F. (2023). Revealing silicon's delithiation behaviour through empirical analysis of galvanostatic charge and discharge curves. *Batteries*, *9*(5). https://doi.org/10.3390/batteries9050251

Spitthoff, L., Vie, P. J. S., Wahl, M. S., Wind, J., & Burheim, O. S. (2023). Incremental capacity analysis (dQ/dV) as a tool for analysing the effect of ambient temperature and mechanical clamping on degradation. *Journal of Electroanalytical Chemistry*, *944*, 117627. https://doi.org/10.1016/j.jelechem.2023.117627

Ulvestad, A., Reksten, A. H., Andersen, H. F., Carvalho, P. A., Jensen, I. J. T., Nagell, M. U., Mæhlen, J. P., Kirkengen, M., & Koposov, A. Y. (2020). Crystallinity of silicon nanoparticles: Direct influence on the electrochemical performance of lithium ion battery anodes. *ChemElectroChem*, *7*(21), 4349–4353. https://doi.org/10.1002/celc.202001108