The Journal of Open Source Software

liionpack: A Python package for simulating packs of batteries with PyBaMM

T. G. Tranter[1, 2], **R. Timms**[2, 3], **V. Sulzer**[4], **F. Brosa Planella**[2, 5], **G. M. Wiggins**[6], **V. Karra**[6], **P. Agarwal**[7], **S. Chopra**[8], **S. Allu**[6], **P. Shearing**[1, 2], **and D. J. L. Brett**[1, 2]

1 Department of Chemical Engineering, University College London, London, WC1E 7JE, United Kingdom. **2** The Faraday Institution, Quad One, Becquerel Avenue, Harwell Campus, Didcot, OX11 0RA, United Kingdom. **3** Mathematical Institute, University of Oxford, OX2 6GG, United Kingdom. **4** Carnegie Mellon University, Scott Hall 5109, 5000 Forbes Ave, Pittsburgh, PA 15213, United States. **5** WMG, University of Warwick, Coventry, CV4 7AL, United Kingdom **6** Oak Ridge National Laboratory, 2360 Cherahala Boulevard, Knoxville, Tennessee 37932, United States. **7** Symbiosis Institute of Technology, Symbiosis International University, Lavale, Pune, Maharashtra 412115, India. **8** Cluster Innovation Centre, University of Delhi, GC Narang Road, Delhi, 110007, India.

DOI: 10.21105/joss.04051

Software

- Review
- Repository
- Archive

Editor: Tim Tröndle

Reviewers:

- @EricaEgg
- @mefuller
- @yangbai90

Submitted: 13 December 2021
Published: 18 February 2022

Summary

Electrification of transport and other energy intensive activities is of growing importance as it provides an underpinning method to reduce carbon emissions. With an increase in reliance on renewable sources of energy and a reduction in the use of more predictable fossil fuels in both stationary and mobile applications, energy storage will play a pivotal role and batteries are currently the most widely adopted and versatile form. Therefore, understanding how batteries work, how they degrade and how to optimize and manage their operation at large scales is critical to achieving emission reduction targets. The electric vehicle (EV) industry requires a considerable number of batteries even for a single vehicle, sometimes numbering in the thousands if smaller cells are used, and the dynamics and degradation of these systems, as well as large stationary power systems, is not that well understood. As increases in the efficiency of a single battery become diminishing for standard commercially available chemistries, gains made at the system level become more important and can potentially be realised more quickly compared with developing new chemistries. Mathematical models and simulations provide a way to address these challenging questions and can aid the engineer and designers of batteries and battery management systems to provide longer lasting and more efficient energy storage systems.

Statement of need

`liionpack` is a PyBaMM-affiliated Python package for simulating large systems of batteries connected in series and parallel. Python enables wrapping low-level languages (e.g., C) for speed without losing flexibility or ease-of-use in the user-interface. `liionpack` was designed to be used by physicists, engineers, students, academics and industrial researchers and system designers concerned with the dynamics of electric current and heat transport in large battery systems. Commercial battery pack simulation tools are available such as modules that can be included within Comsol®, Simulink® and STAR-CCM+™, but to our knowledge `liionpack` is the first to be released open-source. The commercial packages contain more advanced features such as GUI's for circuit design, and integration with CAD based thermal and fluid dynamics tools, but `liionpack` provides everything you need to model a pack of batteries with simple physics and can incorporate circuit definitions defined elsewhere and heat transfer coefficients that are calculated elsewhere. We hope that it will provide the battery community with a platform to build upon to add more features in the future and increase productivity, reproducibility and transparency in this research space.

The API for `liionpack` was designed to provide a simple and efficient extension to the `PyBaMM` (Sulzer et al., 2021) framework allowing users to scale up simulations from single cells to many thousands with a few extra lines of code. `PyBaMM` provides a number of classic physics-based single battery models with configurable options to investigate thermal effects and degradation, for example. The pack architecture introduced by `liionpack` can be defined as a number of batteries connected in series and parallel to one another using busbars and interconnections with defined resistances. A netlist may also be used to construct the pack which is more flexible and allows for configurable network topology and can be constructed graphically with packages such as `LTSpice` (Brocard, 2013) or simply created manually, specifying nodal connections as either current sources, voltage sources or resistors. Statistical distributions can be easily incorporated into the pack architecture elements through the use of input parameters that allow a single model to be solved with varying inputs.

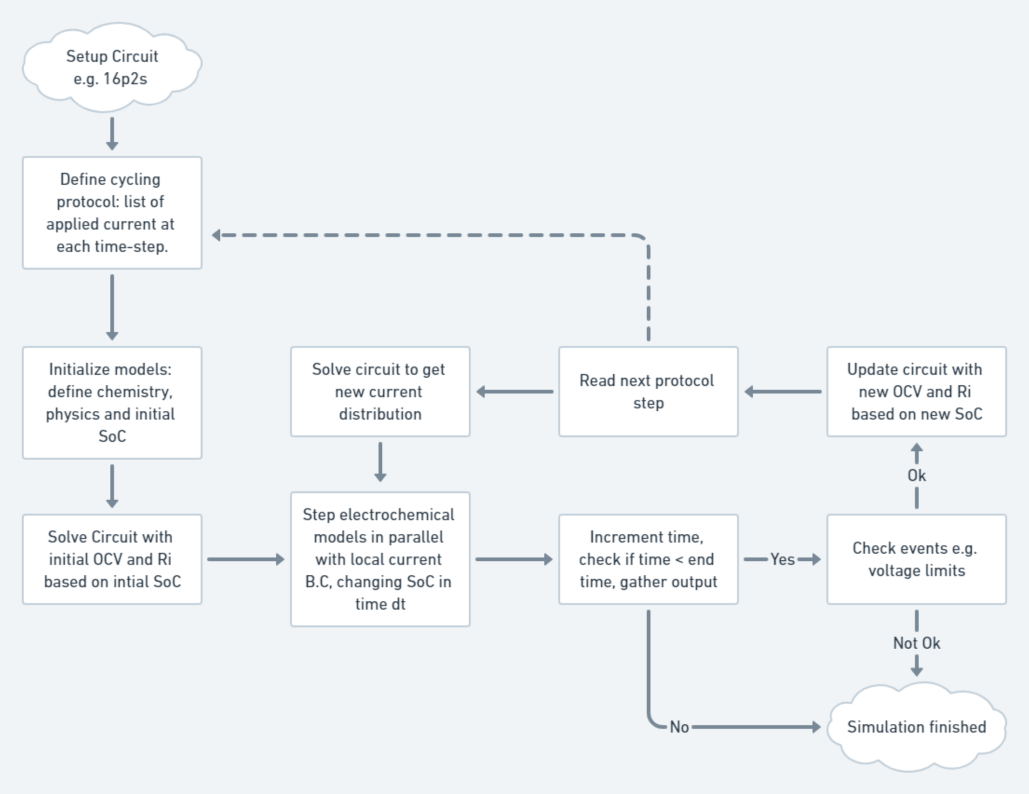
Setup Circuit
e.g. 16p2s
Define cycling protocol: list of applied current at each time-step.
Initialize models: define chemistry, physics and initial SoC
Solve Circuit with initial OCV and Ri based on intial SoC
Solve circuit to get new current distribution
Step electrochemical models in parallel with local current B.C, changing SoC in time dt
Read next protocol step
Increment time, check if time < end time, gather output
Yes
Update circuit with new OCV and Ri based on new SoC
Ok
Check events e.g. voltage limits
Not Ok
No
Simulation finished

Figure 1: Coupled system solution algorithm.

Algorithm

The algorithm to solve the coupled system of batteries is shown in Figure 1. The nature of the solving process facilitates parallel processing of the electrochemical problem for each battery during each time-step formulated as an integrable 1D DAE. The system is coupled electrically at the global level via the busbars and interconnections in the circuit and solving this linear algebraic system between electrochemical time-steps determines the current balance and boundary conditions for each battery at the next time-step. The combination of a global circuit solve and local electrochemical solve repeatedly iterated over in time in a see-saw fashion provides the most simple and efficient way of coupling the system without repeating time-steps. Results for solving a single battery forming a circuit with negligible busbar resistance deviates by less than 0.01% from a pure PyBaMM simulation.

At present, the circuits that are solved may only contain three different types of element: namely current sources, voltage sources and resistors. Resistors are used to represent the busbars and interconnections in the pack as well as the internal resistance of the batteries. The open circuit voltage is used for the voltage sources in the circuit and modified nodal analysis (MNA) (Ho et al., 1975) is used to solve the circuit problem determining the distribution of current in the pack. A typical 4p1s pack architecture is shown below in Figure 2 which was produced using Lcapy (The Lcapy developers, 2014).

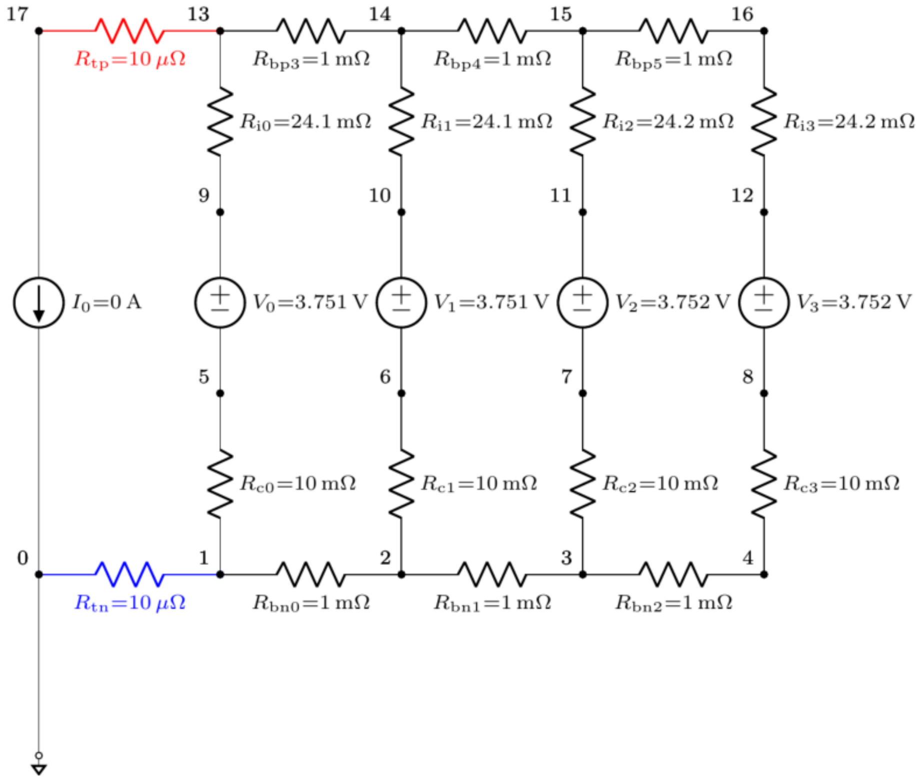

Figure 2: Typical pack architecture.

Presently, the thermal problem is solved in a non-coupled way with each battery acting as an independent heat source and interacting with it's environment in a "lumped" sense with a volume-averaged heat transfer coefficient. Heat generation and conduction through the busbars and from cell to neighbouring cells is likely to occur in some scenarios and can be accounted for by solving a transient thermal problem on the network architecture (Tranter et al., 2020), which will be implemented in future releases. Heat transfer coefficients may also be easily adjusted on a cell-by-cell basis and also throughout the simulation solving process to reflect heterogenous and time-dependent cooling conditions.

Several distributed solvers are provided and can be selected through a common function with a simple function argument. These are Casadi (Andersson et al., 2019) which uses multi-threading and works well for single workstations and ray (Moritz et al., 2018) and dask

(Dask Development Team, 2016) which are designed for running on clusters and use multiprocessing. Many of the functions and models that can be found in `PyBaMM` should work in exactly the same way in `liionpack` and examples are provided showing how to set up and configure different battery models for running in the pack system. Several visualization tools are also provided for analysis of the results.

Example

An example of a small pack is included below. A 4p1s configuration is defined with busbar resistance of 1 $m\Omega$ and interconnection resistance of 10 $m\Omega$. The `Chen2020` (Chen et al., 2020) parameter set is used to define the battery cell chemistry which was gathered using an LG M50 cylindrical cell of 21700 format. By default the single particle model `SPM` is used to define the electrochemical battery model system but a suite of others are available (Marquis et al., 2020) and can be configured using a custom simulation.

```
import liionpack as lp
import pybamm

# Generate the netlist
netlist = lp.setup_circuit(Np=4, Ns=1, Rb=1e-3, Rc=1e-2)

# Define some additional variables to output
output_variables = [
    'X-averaged negative particle surface concentration [mol.m-3]',
    'X-averaged positive particle surface concentration [mol.m-3]',
]

# Cycling experiment, using PyBaMM
experiment = pybamm.Experiment([
    "Charge at 5 A for 30 minutes",
    "Rest for 15 minutes",
    "Discharge at 5 A for 30 minutes",
    "Rest for 30 minutes"],
    period="10 seconds")

# PyBaMM battery parameters
chemistry = pybamm.parameter_sets.Chen2020
parameter_values = pybamm.ParameterValues(chemistry=chemistry)

# Solve the pack problem
output = lp.solve(netlist=netlist,
                  parameter_values=parameter_values,
                  experiment=experiment,
                  output_variables=output_variables,
                  initial_soc=0.5)

# Display the results
lp.plot_output(output)

# Draw the circuit at final state
lp.draw_circuit(netlist, cpt_size=1.0, dpi=150, node_spacing=2.5)
```

The output for the examples is shown below as a pack summary in Figure 3 and an example of a cell variable plot showing each battery current in Figure 4.

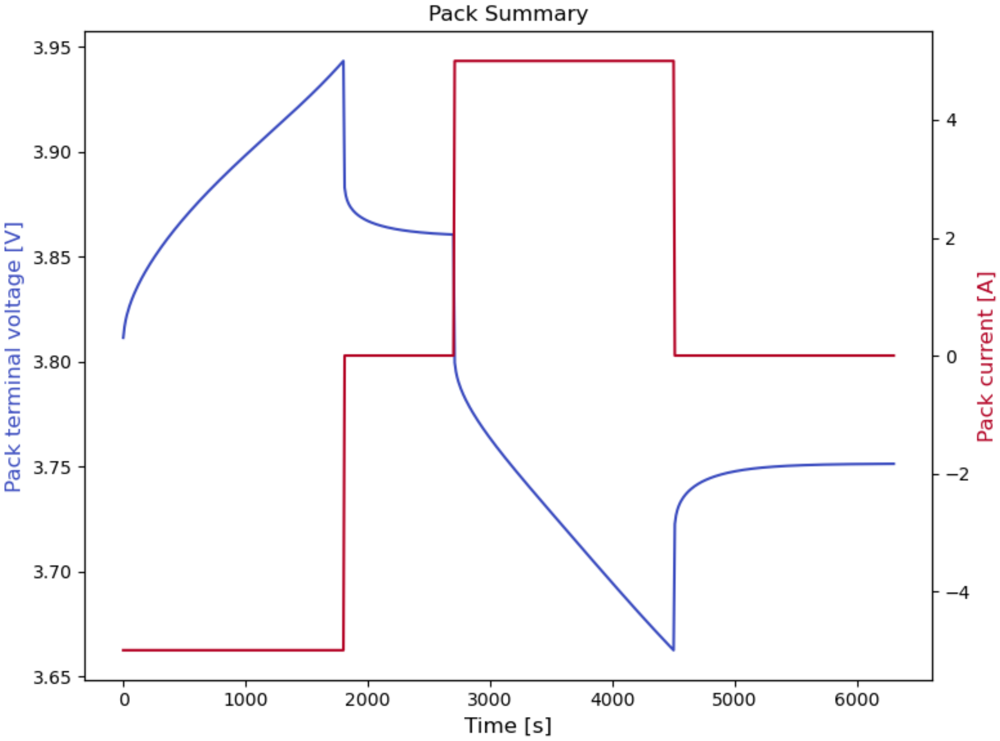

Figure 3: Pack summary showing the pack terminal voltage and total current.

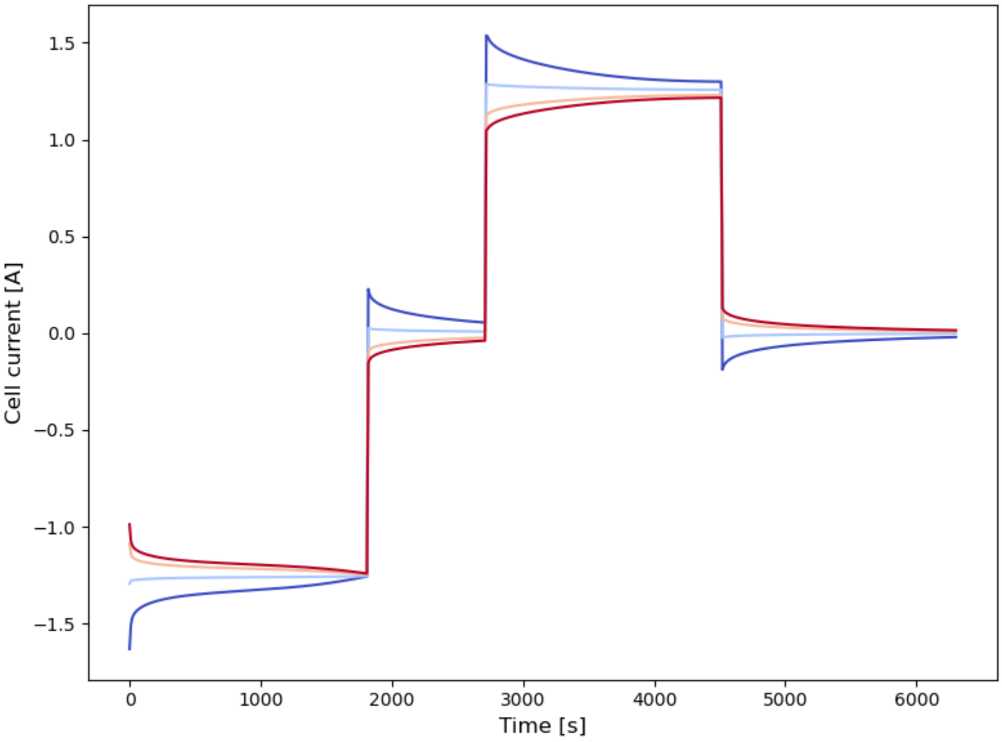

Figure 4: An example of individual cell variable data, any variable defined by the `PyBaMM` model should be accessible.

Acknowledgements

PyBaMM-team acknowledges the funding and support of the Faraday Institution's multi-scale modelling project under grant number EP/S003053/1, FIRG025.

The development work carried out by members at Oak Ridge National Laboratory was partially sponsored by the Office of Electricity under the United States Department of Energy (DOE).

References

Andersson, J. A., Gillis, J., Horn, G., Rawlings, J. B., & Diehl, M. (2019). CasADi: A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, *11*(1), 1–36. https://doi.org/10.1007/s12532-018-0139-4

Brocard, G. (2013). *The LTspice IV simulator: Manual, methods and applications*. Würth Elektronik.

Chen, C.-H., Brosa Planella, F., O'Regan, K., Gastol, D., Widanage, W. D., & Kendrick, E. (2020). Development of experimental techniques for parameterization of multi-scale lithium-ion battery models. *Journal of The Electrochemical Society*, *167*(8), 080534. https://doi.org/10.1149/1945-7111/ab9050

Dask Development Team. (2016). *Dask: Library for dynamic task scheduling*. https://dask.org

Ho, C.-W., Ruehli, A., & Brennan, P. (1975). The modified nodal approach to network analysis. *IEEE Transactions on Circuits and Systems*, *22*(6), 504–509. https://doi.org/10.1109/TCS.1975.1084079

Marquis, S. G., Timms, R., Sulzer, V., Please, C. P., & Chapman, S. J. (2020). A suite of reduced-order models of a single-layer lithium-ion pouch cell. *Journal of The Electrochemical Society*, *167*(14), 140513. https://doi.org/10.1149/1945-7111/abbce4

Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Elibol, M., Yang, Z., Paul, W., Jordan, M. I., & Stoica, I. (2018). *Ray: A distributed framework for emerging AI applications* (pp. 561–577). USENIX Association. ISBN: 978-1-939133-08-3

Sulzer, V., Marquis, S. G., Timms, R., Robinson, M., & Chapman, S. J. (2021). Python battery mathematical modelling (PyBaMM). *Journal of Open Research Software*, *9*(1). https://doi.org/10.5334/jors.309

The Lcapy developers. (2014). *Lcapy, a python package for linear circuit analysis*. https://lcapy.readthedocs.io

Tranter, T., Timms, R., Heenan, T., Marquis, S., Sulzer, V., Jnawali, A., Kok, M., Please, C., Chapman, S., Shearing, P., & others. (2020). Probing heterogeneity in li-ion batteries with coupled multiscale models of electrochemistry and thermal transport using tomographic domains. *Journal of The Electrochemical Society*, *167*(11), 110538. https://doi.org/10.1149/1945-7111/aba44b