The Journal of Open Source Software

PyBOP: A Python package for battery model optimisation and parameterisation

Brady Planden [1¶], **Nicola E. Courtier** [1,2], **Martin Robinson** [3], **Ferran Brosa Planella** [4], and **David A. Howey** [1,2]

1 Department of Engineering Science, University of Oxford, Oxford, UK **2** The Faraday Institution, Harwell Campus, Didcot, UK **3** Research Software Engineering Group, University of Oxford, Oxford, UK **4** Mathematics Institute, University of Warwick, Coventry, UK ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

The Python Battery Optimisation and Parameterisation (`PyBOP`) package provides a set of methods for the parameterisation and optimisation of battery models, offering both Bayesian and frequentist approaches with example workflows to assist the user. `PyBOP` can be used for parameter identification of various models, including the electrochemical and equivalent circuit models provided by the popular open-source package `PyBaMM` (Sulzer et al., 2021). Similarly, `PyBOP` can be used for design optimisation under user-defined operating conditions for a given parameter. `PyBOP` allows the user to parameterise battery models using a variety of methods and provides diagnostics on the performance and convergence of the optimisation. The identified parameters can be used for prediction, on-line control and design optimisation, all of which support improved battery utilisation and development.

Statement of need

`PyBOP` is designed to provide a user-friendly, object-oriented interface for the optimisation of battery models which have been implemented in existing battery modelling software, e.g. `PyBaMM` (Sulzer et al., 2021). `PyBOP` is intended to serve a broad audience of students, engineers, and researchers in both academia and the battery industry. `PyBOP` prioritises clear and informative diagnostics and workflows for both new and experienced users, while also leveraging advanced optimisation algorithms provided by `SciPy` (Virtanen et al., 2020), `PINTS` (Clerx et al., 2019), and internal implementations such as the adaptive moment estimation with weight decay (AdamW), as well as Cuckoo search.

`PyBOP` supports the Battery Parameter eXchange (BPX) standard (Korotkin et al., 2023) for sharing battery parameter sets. These parameter sets are costly to obtain due to a number of factors: the equipment and time spent on characterisation experiments, the requirement of battery domain knowledge, and the computational cost of parameter estimation. `PyBOP` reduces the barrier to entry and ongoing costs by providing an accessible workflows' that efficiently connects battery models with numerical optimisers, as well as explanatory examples of battery parameterisaton and design optimisation.

This package complements other tools in the field of lithium-ion battery modelling built around `PyBaMM`, such as `liionpack` for simulating battery packs (Tranter et al., 2022) as the identified parameters are easily exportable from `PyBOP` into packages aimed at predictive forward modelling.

Architecture

PyBOP is a Python package packaged through PyPI, which currently supports Python versions 3.9 — 3.12. The package composes the popular battery modelling package PyBaMM for forward modelling, while providing classes for parameterisation and optimisation. As shown in Figure 1, PyBOP composes the battery modelling package PyBaMM, enabling a consistent interface and robust objection construction process. With the forward model interface construction for parameter identification and optimisation, PyBOP provides statistical methods and optimisation algorithms to interface cleanly with the forward model predictions. Furthermore, identifiablity metrics are provided for the estimated parameters through Hessian approximation of the cost/likelihood functions in the frequentist workflows and posterior moments in the Bayesian workflows.

PyBOP's interface to supporting funding agencies, alongside a visualisation of the general workflow for parameterisation and optimisation

Figure 1: PyBOP's interface to supporting funding agencies, alongside a visualisation of the general workflow for parameterisation and optimisation

PyBOP formulates the optimisation workflow through four main classes, namely the model, problem, cost, and optimiser or sampler, as shown in Figure 2. Each of these objects represent a base class with children classes constructing differing functionality for specialised parameterisation or optimisation workflows. For example, the model class offers children classes for differing physics-based battery models, as well as emperical models. This allows for the underlying PyBaMM model to be constructed and validated for the different requirements between the physics-based models and the emperical. For a given set of model equations provided from PyBaMM, initial conditions, spatialy discretisation, and numerical solver initialisation is completed. By composing PyBaMM models directly into PyBOP, the underlying model structure can be modified, and optimally constructed for the optimisation tasks. One such example of this, is the spatially rediscretiation that is performed with geometric parameters are optimised. In this situation, PyBOP aims to minimally reconstruct the PyBaMM model while maintaining the problem, cost, and optimisation objects, providing improved performance benefits to end-users. In the typical optimisation workflow, the classes in Figure 2 are constructed in sequence.

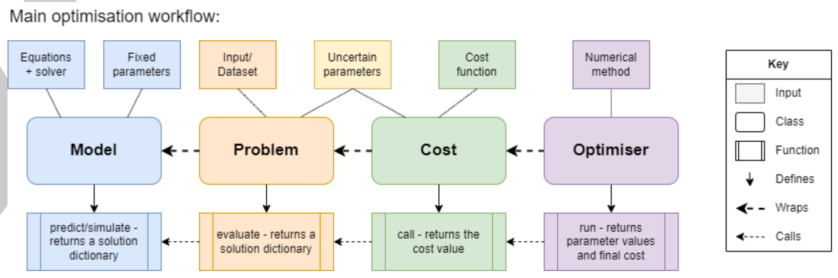
Main optimisation workflow:
Equations + solver
Fixed parameters
Input/ Dataset
Uncertain parameters
Cost function
Numerical method
Model
Problem
Cost
Optimiser
predict/simulate - returns a solution dictionary
evaluate - returns a solution dictionary
call - returns the cost value
run - returns parameter values and final cost
Key
Input
Class
Function
Defines
Wraps
Calls

Figure 2: The core PyBOP architecture, showcasing the base class interfaces. Each class provide direct mapping to a classical step in the optimisation workflow.

The currently implemented subclasses for the model, problem, and cost classes are listed in Table 1. From this point onwards, the parameterisation and optimisation tasks will simply be referred to as an optimisation task. This simplification can be justified by inspecting Equation 4 and Equation 5 and confirming parameterisation can be viewed as an minimising optimisation task for a distance-based cost function.

Table 1: List of available model, problem and cost classes.

Battery Models	Problem Types	Cost / Likelihood Functions
Single particle model (SPM)	Fitting problem	Sum of squared error
SPM with electrolyte (SPMe)	Design problem	Root mean squared error
Doyle-Fuller-Newman (DFN)	Observer	Gaussian log likelihood
Many particle model (MPM)		Maximum a posteriori
Multi-species multi-reaction (MSMR)		Unscented Kalman filter
Weppner Huggins		Gravimetric energy density
Equivalent circuit model (ECM)		Volumetric energy density

Likewise, the current optimisation algorithms available for usage in optimisation tasks in presented in Table 2. The cost functions in Table 1 are grouped by problem type, while the model and optimiser classes can be selected in combination with any problem-cost pair.

Table 2: The currently supported optimisation algorithms classifed by candidate solution type, inclusive of gradient information. (*) Scipy Minimize has gradient and non-gradient methods.

Gradient-based	Evolutionary Strategies	Swarm Intelligence	Direct Search
Adaptive moment estimation with weight decay (AdamW)	Covariance matrix adaptation (CMA-ES)	Particle swarm (PSO)	Nelder-Mead
Improved resilient backpropagation (iRProp-)	Exponential natural (xNES)	Cuckoo search	
Gradient descent	Separable natural (sNES)		
SciPy minimize (*)	SciPy differential evolution		

As discussed above, `PyBOP` offers Bayesian inference methods such as Maximum a Posteriori (MAP) presented alongside the frequentist methods in Table 2; however, for a full Bayesian framework, monte carlo sampling is implemented within `PyBOP`. These methods construct a posterior distribution of the unknown parameters which can used for uncertainty and practical identifiability diagnostics. The individual sampler classes are currently composed within `PyBOP` from the `Pints` library, with a base sampling class implemented for interopability and direct integration to the `PyBOP` model, problem, and likelihood classes. The currently support samplers are presented in Table 3.

Table 3: PyBOP's monte carlo sampling methods separated based on candidate suggestion method.

Hamiltonian-based	Adaptive	Slice Sampling	Evolutionary	Other
Monomial Gamma	Delayed Rejection Adaptive	Slice Doubling	Differential Evolution	Metropolis Random Walk
No-U-Turn	Haario Bardenet	Slice Rank Shrinking		Emcee Hammer
Hamiltonian	Rao Blackwell Haario	Slice Stepout		
Relativistic				Metropolis Adjusted Langevin

Background

Battery models

In general, battery models can be written in the form of a differential-algebraic system of equations:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = f(t, \mathbf{x}, \mathbf{y}, \mathbf{u}(t), \boldsymbol{\theta}), \tag{1}$$

$$\mathbf{y}(t) = g(t, \mathbf{x}, \mathbf{y}, \mathbf{u}(t), \boldsymbol{\theta}), \tag{2}$$

with initial conditions

$$\mathbf{x}(0) = \mathbf{x}_0(\boldsymbol{\theta}). \tag{3}$$

Here, t is time, $\mathbf{x}(t)$ are the spatially (discretised) states, $\mathbf{y}(t)$ are the outputs (for example the terminal voltage), $\mathbf{u}(t)$ are the inputs (e.g. the applied current) and $\boldsymbol{\theta}$ are the unknown parameters.

Common battery models include various types of equivalent circuit model (e.g. the Thévenin model), the Doyle–Fuller–Newman (DFN) model (Doyle et al., 1993; Fuller et al., 1994) based on porous electrode theory and its reduced-order variants including the single particle model (SPM) (Planella et al., 2022), as well as the multi-scale, multi-reaction (MSMR) model (Verbrugge et al., 2017).

Simplified models that retain good prediction capabilities at a lower computational cost are widely used, for example within battery management systems, while physics-based models are required to understand the impact of design parameters on battery performance.

Examples

Parameterisation

Battery model parameterisation is difficult due to the high ratio of the number of parameters to measurable outputs (Andersson et al., 2022; Miguel et al., 2021; Wang et al., 2022). A complete parameterisation often requires a step-by-step identification of smaller groups of parameters from a variety of different datasets (Chen et al., 2020; Chu et al., 2019; Kirk et al., 2023).

A generic data fitting optimisation problem may be formulated as:

$$\min_{\boldsymbol{\theta}} \ \mathcal{L}_{(\mathbf{y}_i)}(\boldsymbol{\theta}) \quad \text{subject to equations (1)-(3)} \tag{4}$$

in which $\mathcal{L} : \boldsymbol{\theta} \mapsto [0, \infty)$ is a cost (or likelihood) function that quantifies the agreement between the model and a sequence of data points $(\mathbf{y}_i)$ measured at times t_i. For gradient-based optimisers, the Jacobian of the cost function with respect to the unknown parameters, $(\frac{\partial L}{\partial \theta})$ is used as a directional metric for the algorithm when exploring the parameter space.

By way of example, we next demonstrate the fitting of some synthetic data for which we know the true parameter values.

Design optimisation

Design optimisation is supported within PyBOP to guide future development of the battery design by identifying parameter variations which may unlock improvements in battery performance. This optimisation task can be viewed similarly to the parameterisation workflows described above, however, with the aim of increasing the distance metric instead of minimising it. In the case of design optimisation for maximising gravimetric energy density, PyBOP minimises the negative of the cost function, where the cost metric is no longer a distance between two time-series vectors, instead it is the integrated energy from the vector normalised with the corresponding cell mass. This is typically quantified for operational conditions such as a 1C (the applied current required to discharge the cell in one hour) capacity.

Design optimisation can be written in the form of a constrained optimisation problem as:

$$\min_{\theta \in \Omega} \mathcal{L}(\boldsymbol{\theta}) \quad \text{subject to equations (1)-(3)} \tag{5}$$

in which $\mathcal{L} : \boldsymbol{\theta} \mapsto [0, \infty)$ is a cost function that quantifies the desirability of the design and Ω is the set of allowable parameter values.

As an example, let us consider the target of maximising gravimetric energy density subject to constraints on the geometric electrode parameters (Couto et al., 2023).

Acknowledgements

We gratefully acknowledge all contributors to this package. This work was supported by the Faraday Institution Multiscale Modelling (MSM) project (ref. FIRG059), UKRI's Horizon Europe Guarantee (ref. 10038031), and EU IntelLiGent project (ref. 101069765).

Discussion Points

- Performance (multiprocessing)
- ~~Construction of PyBaMM models (geometric and non-geometric identification)~~
- Feasability checks on identified parameters
- Spatial identification methods?
- Documentation supported at https://pybop-docs.readthedocs.io/en/latest/
- Benchmarks provided at https://pybop-team.github.io/pybop-bench/
- Plotting classes via Plotly (cost landscapes, gradient landscapes)
- Test suite provided by pytest (~98% coverage)
- Standalone implementations (Bring your own model)

References

Andersson, M., Streb, M., Ko, J. Y., Löfqvist Klass, V., Klett, M., Ekström, H., Johansson, M., & Lindbergh, G. (2022). Parametrization of physics-based battery models from input-output data: A review of methodology and current research. *Journal of Power Sources*, *521*(November 2021), 230859. https://doi.org/10.1016/j.jpowsour.2021.230859

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

Chu, Z., Plett, G. L., Trimboli, M. S., & Ouyang, M. (2019). A control-oriented electrochemical model for lithium-ion battery, Part I: Lumped-parameter reduced-order model with constant phase element. *Journal of Energy Storage*, *25*(August), 100828. https://doi.org/10.1016/j.est.2019.100828

Clerx, M., Robinson, M., Lambert, B., Lei, C. L., Ghosh, S., Mirams, G. R., & Gavaghan, D. J. (2019). Probabilistic inference on noisy time series (PINTS). *Journal of Open Research Software*, *7*(1), 23. https://doi.org/10.5334/jors.252

Couto, L. D., Charkhgard, M., Karaman, B., Job, N., & Kinnaert, M. (2023). Lithium-ion battery design optimization based on a dimensionless reduced-order electrochemical model. *Energy*, *263*(PE), 125966. https://doi.org/10.1016/j.energy.2022.125966

Doyle, M., Fuller, T. F., & Newman, J. (1993). Modeling of Galvanostatic Charge and Discharge of the Lithium/Polymer/Insertion Cell. *Journal of The Electrochemical Society*, *140*(6), 1526–1533. https://doi.org/10.1149/1.2221597

Fuller, T. F., Doyle, M., & Newman, J. (1994). Simulation and optimization of the dual lithium ion insertion cell. *Journal of The Electrochemical Society*, *141*(1), 1. https://doi.org/10.1149/1.2054684

Kirk, T. L., Lewis-Douglas, A., Howey, D., Please, C. P., & Jon Chapman, S. (2023). Nonlinear electrochemical impedance spectroscopy for lithium-ion battery model parameterization. *Journal of The Electrochemical Society*, *170*(1), 010514. https://doi.org/10.1149/1945-7111/acada7

Korotkin, I., Timms, R., Foster, J. F., Dickinson, E., & Robinson, M. (2023). Battery parameter eXchange. In *GitHub repository*. The Faraday Institution. https://github.com/FaradayInstitution/BPX

Miguel, E., Plett, G. L., Trimboli, M. S., Oca, L., Iraola, U., & Bekaert, E. (2021). Review of computational parameter estimation methods for electrochemical models. *Journal of Energy Storage*, *44*(PB), 103388. https://doi.org/10.1016/j.est.2021.103388

Planella, F. B., Ai, W., Boyce, A. M., Ghosh, A., Korotkin, I., Sahu, S., Sulzer, V., Timms, R., Tranter, T. G., Zyskin, M., Cooper, S. J., Edge, J. S., Foster, J. M., Marinescu, M., Wu, B., & Richardson, G. (2022). A Continuum of Physics-Based Lithium-Ion Battery Models Reviewed. *Progress in Energy*, *4*(4), 042003. https://doi.org/10.1088/2516-1083/ac7d31

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), 14. https://doi.org/10.5334/jors.309

Tranter, T. G., Timms, R., Sulzer, V., Planella, F. B., Wiggins, G. M., Karra, S. V., Agarwal, P., Chopra, S., Allu, S., Shearing, P. R., & Brett, D. J. I. (2022). Liionpack: A python package for simulating packs of batteries with PyBaMM. *Journal of Open Source Software*, *7*(70), 4051. https://doi.org/10.21105/joss.04051

Verbrugge, M., Baker, D., Koch, B., Xiao, X., & Gu, W. (2017). Thermodynamic model for substitutional materials: Application to lithiated graphite, spinel manganese oxide, iron phosphate, and layered nickel-manganese-cobalt oxide. *Journal of The Electrochemical Society*, *164*(11), E3243. https://doi.org/10.1149/2.0341708jes

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Wang, A. A., O'Kane, S. E. J., Brosa Planella, F., Houx, J. L., O'Regan, K., Zyskin, M., Edge, J., Monroe, C. W., Cooper, S. J., Howey, D. A., Kendrick, E., & Foster, J. M. (2022). Review of parameterisation and a novel database (LiionDB) for continuum Li-ion battery models. *Progress in Energy*, *4*(3), 032004. https://doi.org/10.1088/2516-1083/ac692c