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], **Agriya Khetarpal** [4], **Ferran Brosa Planella** [2,5], **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** Quansight PBC **5** 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Published: unpublished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 methods for estimating and optimising battery model parameters, offering both deterministic and stochastic approaches with example workflows. PyBOP enables parameter identification from data for various battery models, including electrochemical and equivalent circuit models from the open-source PyBaMM package (Sulzer et al., 2021). The same approaches enable design optimisation under user-defined operating conditions across various model structures and design goals. PyBOP facilitates optimisation and provides diagnostics to examine optimiser performance and convergence of the cost and parameters. Identified parameters can be used for prediction, online estimation and control, and design optimisation, accelerating battery research and development.

Statement of need

PyBOP provides a user-friendly, object-oriented interface for optimising battery model parameters. It leverages the open-source PyBaMM package (Sulzer et al., 2021) to formulate and solve battery models. Together, these tools serve a broad audience including students, engineers, and researchers in academia and industry, enabling advanced model use without specialised knowledge of battery modelling, parameter inference, or software development. PyBOP emphasises clear diagnostics and workflows to support users with varying domain expertise, and provides access to numerous optimisation and sampling algorithms. These capabilities are enabled through interfaces to PINTS (Clerx et al., 2019), SciPy (Virtanen et al., 2020), and PyBOP's implementations of algorithms including adaptive moment estimation with weight decay (AdamW) (Loshchilov & Hutter, 2017), gradient descent (Cauchy & others, 1847), and cuckoo search (Yang & Suash Deb, 2009).

PyBOP complements other lithium-ion battery modelling packages built around PyBaMM, such as `liionpack` for battery pack simulation (Tranter et al., 2022) and `pybamm-eis` for fast numerical computation of the electrochemical impedance of any battery model, as well as the battery parameter exchange (BPX) standard (Korotkin et al., 2023). Identified PyBOP parameters are easily exportable to other packages.

Architecture

PyBOP is structured around four core components: a `Simulator`, `Cost`, `Problem`, and `Optimiser`/`Sampler`, as shown in Figure 1. The purpose of the `Simulator` is to generate model predictions. For example, the `pybop.pybamm.Simulator` interfaces with PyBaMM to

efficiently construct, discretise and numerically solve a `PyBaMM` model for candidate parameter values. Custom or built-in `Cost` classes evaluate an error measure, likelihood or design metric for the candidate parameter values and simulation result. Multiple costs can be summed with optional weighting. The `Problem` class coordinates simulator and cost evaluation, and the `Optimiser/Sampler` classes perform parameter inference through optimisation algorithms or Monte Carlo sampling. This structure ensures extensibility for new optimisation problems with a consistent interface between models and optimisers.

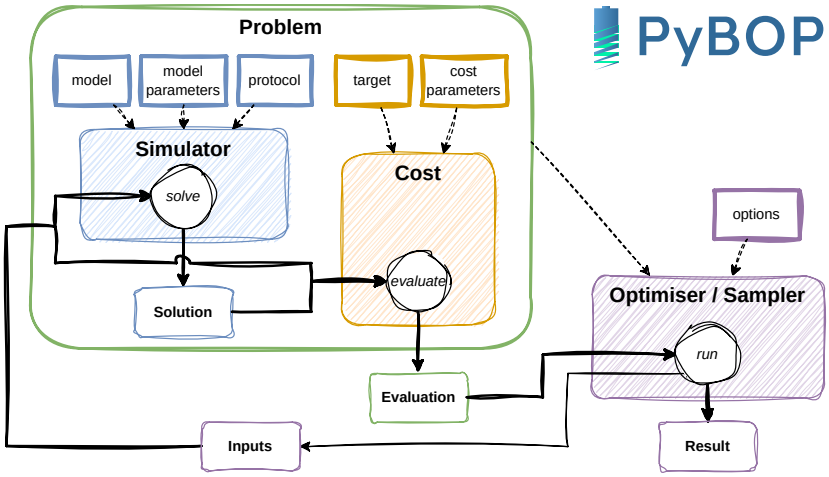

Figure 1: The core `PyBOP` architecture with base class interfaces. Each class provides a direct mapping to a step in the optimisation workflow.

The `pybamm.Simulator` returns a solution with corresponding sensitivities, where possible, to enable gradient-based optimisation. Bayesian inference is provided by sampler classes, with Monte Carlo algorithms provided by `PINTS`. In the typical workflow, the classes in Figure 1 are constructed in sequence, from left to right. The optimisation result includes a log of the candidate parameters and corresponding cost values. Beyond convergence information, identifiability metrics are provided through Hessian approximation and Sobol sampling from the `salib` package.

Beyond the core architecture, `PyBOP` provides specialised inference and optimisation features. Parameter inference from electrochemical impedance spectroscopy (EIS) simulations is handled through the `pybop.pybamm.EISSimulator`, which discretises and linearises the EIS forward model into sparse mass matrix form with an auto-differentiated Jacobian. The result is returned in the frequency domain and is compatible with the same cost classes as time-domain simulations.

The currently available optimisation algorithms are presented in Table 1. Note that SciPy minimize includes several gradient-based and gradient-free methods. Hereafter, point-based parameterisation and design-optimisation tasks are referred to as optimisation tasks. This simplification can be justified by comparing Equation 5 and Equation 7; deterministic parameterisation is an optimisation task to minimise distance-based cost between model output and measured values.

Table 1: Currently supported optimisers classified by candidate solution type, including gradient information.

Gradient-based	Evolutionary	(Meta)heuristic
Weight decayed adaptive moment estimation (AdamW)	Covariance matrix adaptation (CMA-ES)	Particle swarm (PSO)
Gradient descent	Exponential natural (xNES)	Nelder-Mead
SciPy minimize	Separable natural (sNES)	Cuckoo search
Improved resilient backpropagation (iRProp-/+)	SciPy differential evolution	Simulated annealing

Beyond deterministic optimisers (Table 1), PyBOP provides Monte Carlo sampling routines to estimate parameter distributions within a Bayesian framework. These methods construct posterior parameter distributions that can be used to assess uncertainty and practical identifiability. Individual sampler classes are composed from the PINTS library, with a base sampler class implemented for interoperability and direct integration with the `Problem` class. Currently supported samplers are listed in Table 2.

Table 2: Sampling methods supported by PyBOP, classified according to the candidate proposal method.

Gradient-based	Adaptive	Slicing	Other
Monomial gamma	Delayed rejection adaptive	Rank shrinking	Metropolis random walk
No-U-turn	Haario Bardenet	Doubling	Emcee hammer
Hamiltonian	Haario	Stepout	Metropolis adjusted Langevin
Relativistic	Rao Blackwell		Differential evolution

Background

Battery models

In general, battery models (after spatial discretisation) can be written in the form of a differential-algebraic system of equations,

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

$$0 = g(t, \mathbf{x}, \boldsymbol{\theta}), \tag{2}$$

$$\mathbf{y}(t) = h(t, \mathbf{x}, \boldsymbol{\theta}), \tag{3}$$

with initial conditions

$$\mathbf{x}(0) = \mathbf{x}_0(\boldsymbol{\theta}). \tag{4}$$

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

Common battery models include equivalent circuit models (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 reduced-order variants including the single particle model (SPM) (Brosa Planella et al., 2022) and the multi-species multi-reaction (MSMR) model (Verbrugge et al., 2017). Simplified models that retain acceptable predictive accuracy at lower computational cost are widely used, for example in battery management systems, while physics-based models

are required to understand the impact of physical parameters on performance. This separation of complexity traditionally results in multiple parameterisations for a single battery type, depending on model structure.

Examples

Parameterisation

Battery model parameterisation is challenging due to the large number of parameters compared to the number of measurable outputs (Andersson et al., 2022; Miguel et al., 2021; Wang et al., 2022). A complete parameterisation often requires stepwise identification of parameter subsets from a variety of excitations and datasets (Chen et al., 2020; Chu et al., 2019; Kirk et al., 2023; Lu et al., 2021). Parameter identifiability can be poor for some excitations and datasets, requiring improved experimental design and uncertainty-capable identification methods (Aitio et al., 2020).

A generic data-fitting optimisation problem may be formulated as:

$$\min_{\boldsymbol{\theta}} \ \mathcal{L}_{(\hat{\boldsymbol{y}}_i)}(\boldsymbol{\theta}) \quad \text{subject to equations (1)-(4)} \tag{5}$$

where $\mathcal{L} : \boldsymbol{\theta} \mapsto [0, \infty)$ is a cost function that quantifies the agreement between the model output $\mathbf{y}(t)$ and a sequence of observations $(\hat{\boldsymbol{y}}_i)$ measured at times t_i. For gradient-based optimisers, the Jacobian of the cost function with respect to unknown parameters, $\partial\mathcal{L}/\partial\theta$, is computed for step-size and directional information.

We demonstrate the fitting of synthetic data where the model parameters are known, using PyBaMM's SPM with contact resistance. We target two parameters: the lithium diffusivity in the negative electrode active material particles ("negative particle diffusivity") and the contact resistance, with true values [3.3e-14 m^2/s, 10 mOhm]. We generate time-domain data for a one-hour discharge from 100% to 0% state of charge (1C rate) followed by 30 minutes relaxation. The output voltage is corrupted with zero-mean Gaussian noise of amplitude 2 mV (blue dots in Figure 2 (left)). Initial states are assumed known, although this is not generally necessary. The PyBOP repository contains example notebooks illustrating similar inference processes. The underlying cost landscape to be explored by the optimiser is shown in Figure 2 (right), with the initial position and true values marked. In general, the true values are unknown.

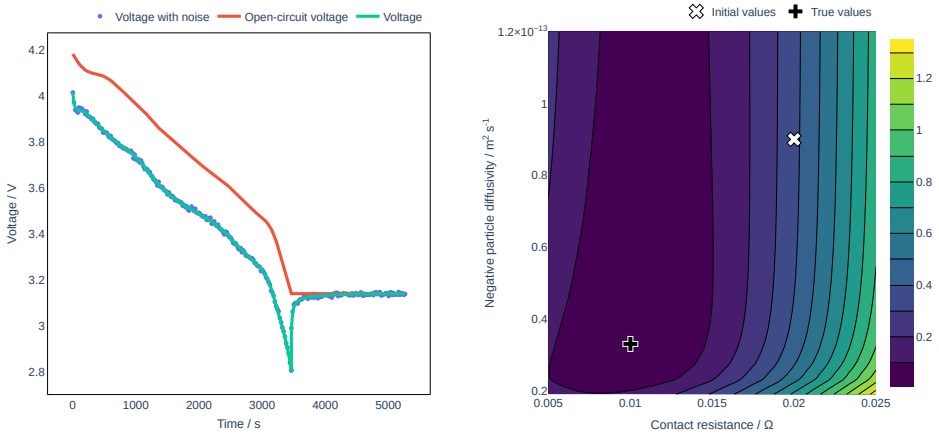

Figure 2: A synthetic dataset (left) and cost landscape (right) depicting a time-series parameterisation problem using the root-mean-squared error cost function.

PyBOP can generate and fit EIS data using methods from `pybamm-eis` (Dhoot et al., 2024). Using PyBaMM's SPM with double-layer capacitance and contact resistance, Figure 3 shows numerical EIS predictions alongside the cost landscape for the corresponding inference task. At the time of publication, gradient-based optimisation and sampling methods are not available for EIS simulators.

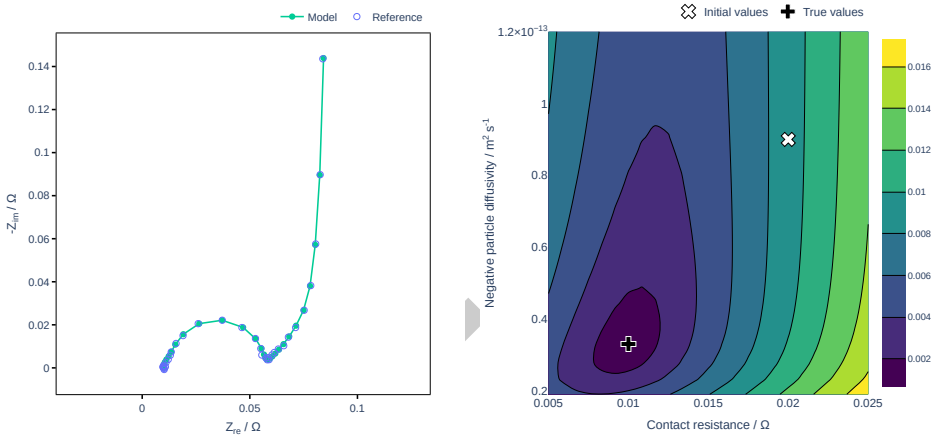

Figure 3: Data and model fit (left) and cost landscape (right) for a frequency-domain EIS parameterisation, at 5% SOC, using the root-mean-squared error cost function.

We continue with time-domain identification (Figure 2), however time- and frequency-domain problems may be combined for improved parameterisation. As gradient information is available for our time-domain example, the choice of distance-based cost function and optimiser is unconstrained. Due to the difference in magnitude between the two parameters, we apply a logarithmic transformation which transforms the search space of the optimiser to allow for a common step size, improving convergence. As a demonstration of PyBOP's parameterisation capabilities, Figure 4 (left) shows convergence rates for distance-minimising cost functions, while Figure 4 (right) shows analogous results for likelihood maximisation. Optimisation is performed using SciPy minimize with the gradient-based BFGS method.

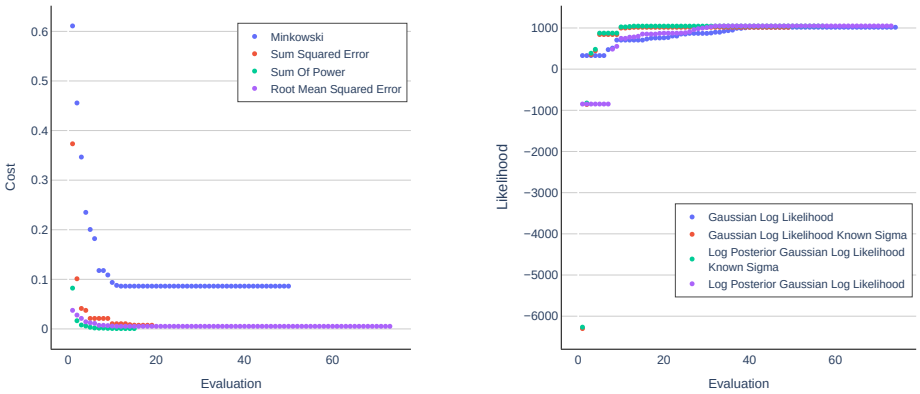

Figure 4: Convergence of the BFGS method for various cost (left) and likelihood (right) functions.

Using the same model and parameters, we compare example convergence rates of various algorithms across several categories: gradient-based methods in Figure 5 (left), evolutionary

strategies in Figure 5 (middle) and (meta)heuristics in Figure 5 (right) using a mean-squared-error cost. Figure 6 shows the optimiser's exploration of the cost landscape, with the three rows showing the gradient-based optimisers (top), evolution strategies (middle), and (meta)heuristics (bottom). Optimiser performance depends on the cost landscape, initial guess or prior for each parameter, and the hyperparameters for each problem.

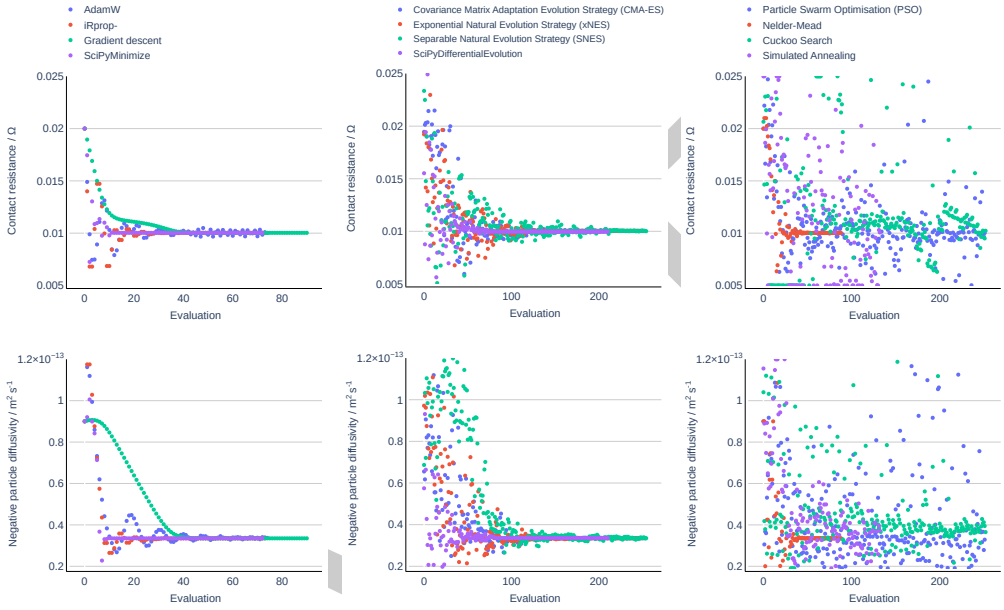

Figure 5: Convergence in the parameter values for optimisation algorithms available in PyBOP.

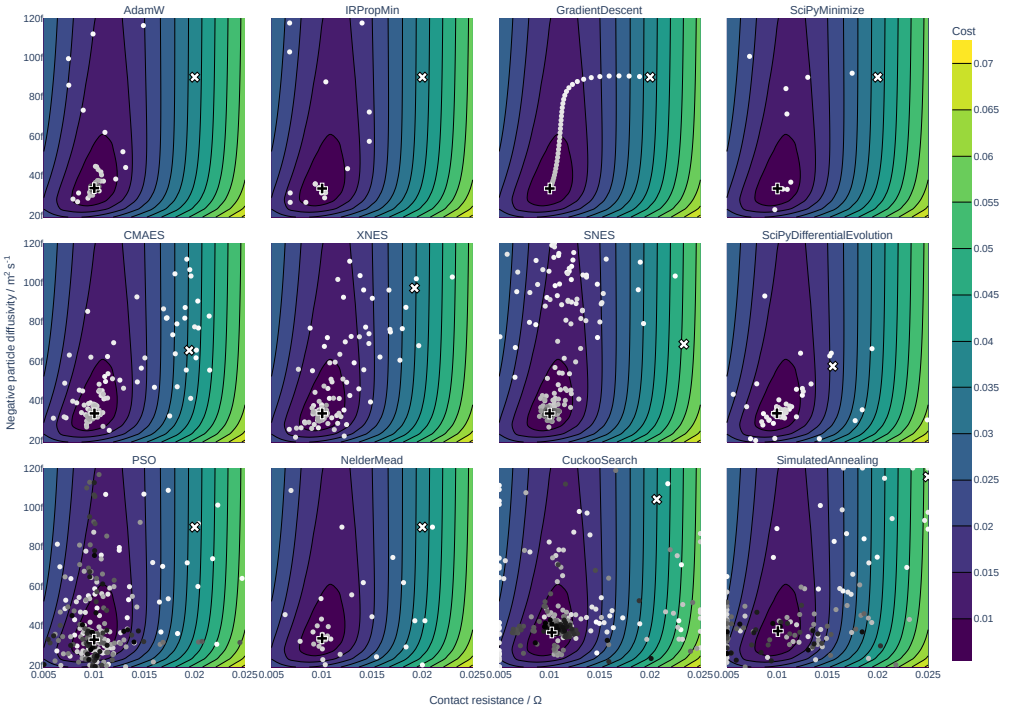

Figure 6: Cost landscape plots showing the optimisation traces of 12 different optimisers.

This example parameterisation task can also be approached from a Bayesian perspective, using PyBOP's sampler methods. First, we introduce Bayes' rule,

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}, \tag{6}$$

where $P(\theta|D)$ is the posterior parameter distribution, $P(D|\theta)$ is the likelihood function, $P(\theta)$ is the prior parameter distribution, and $P(D)$ is the model evidence, or marginal likelihood, which acts as a normalising constant. For maximum likelihood estimation or maximum a posteriori estimation, one wishes to maximise $P(D|\theta)$ or $P(\theta|D)$, respectively, formulated as an optimisation problem as per Equation 5.

To estimate the full posterior parameter distribution, however, one must use sampling or other inference methods to reconstruct $P(\theta|D)$. The posterior distribution provides information about the uncertainty of the identified parameters, e.g., by calculating the variance or other moments. Monte Carlo methods available from the probabilistic inference on noisy time-series (PINTS) package include gradient-based methods such as no-u-turn (Hoffman & Gelman, 2011) and Hamiltonian (Brooks et al., 2011), heuristic methods such as differential evolution (Braak, 2006), and conventional methods based on random sampling with rejection criteria (Metropolis et al., 1953). Figure 7 shows sampled posteriors for the synthetic model using an adaptive covariance-based sampler called Haario Bardenet (Haario et al., 2001).

The Journal of Open Source Software

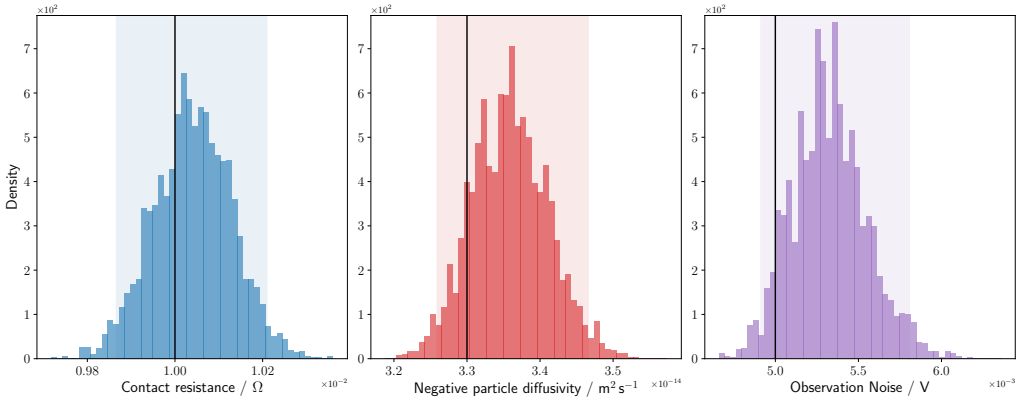

Figure 7: Posterior distributions of model parameters and observation noise; shaded areas show the 95th percentile credible interval.

Design optimisation

PyBOP supports design optimisation to guide device design development by identifying parameter sensitivities that can unlock improvements in performance. Design workflows are similar to parameterisation workflows, but the aim is to maximise a design metric rather than minimise a distance-based cost function. PyBOP performs maximisation by minimising the negative cost. An example design metric is the gravimetric energy (or power) density given by the integral of the discharge energy (or power) normalised by the cell mass. Such metrics are typically quantified for operating conditions such as a 1C discharge, at a given temperature.

In general, design optimisation can be written as a constrained optimisation problem,

$$\min_{\boldsymbol{\theta}\in\Omega} \mathcal{L}(\boldsymbol{\theta}) \quad \text{subject to equations (1)-(4)}, \tag{7}$$

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

We consider maximising gravimetric energy density subject to constraints on two of the geometric electrode parameters (Couto et al., 2023). We use the PyBaMM single particle model with electrolyte (SPMe) to investigate the impact of positive electrode thickness and active material volume fraction on energy density. Since the total volume fraction must sum to unity, the positive electrode porosity is defined relative to the active material volume fraction. The 1C rate can also be optimised (via the nominal capacity parameter) or defined as a function of the parameters for each design.

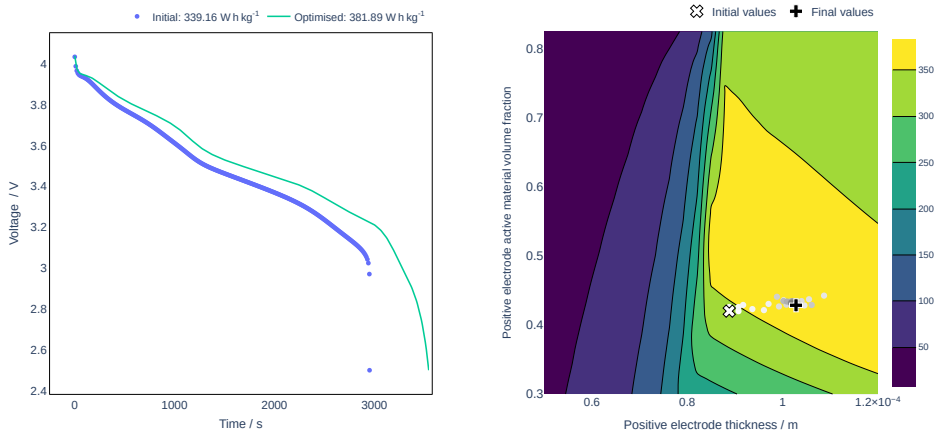

Figure 8: Initial and optimised voltage profiles alongside the gravimetric energy density cost landscape.

Figure 8 (left) shows the predicted improvement in the discharge profile between the initial and optimised parameter values for a fixed-rate 1C discharge selected from the initial design and (right) the Nelder-Mead search over the parameter space.

Acknowledgements

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

References

Aitio, A., Marquis, S. G., Ascencio, P., & Howey, D. (2020). Bayesian parameter estimation applied to the li-ion battery single particle model with electrolyte dynamics. *IFAC-PapersOnLine*, *53*(2), 12497–12504. https://doi.org/10.1016/j.ifacol.2020.12.1770

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

Braak, C. J. F. T. (2006). A Markov Chain Monte Carlo version of the genetic algorithm Differential Evolution: Easy Bayesian computing for real parameter spaces. *Statistics and Computing*, *16*(3), 239–249. https://doi.org/10.1007/s11222-006-8769-1

Brooks, S., Gelman, A., Jones, G., & Meng, X.-L. (2011). *Handbook of markov chain monte carlo*. Chapman; Hall/CRC. https://doi.org/10.1201/b10905

Brosa Planella, F., 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

Cauchy, A., & others. (1847). Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, *25*(1847), 536–538.

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

Dhoot, R., Timms, R., & Please, C. (2024). *PyBaMM EIS: Efficient Linear Algebra Methods to Determine Li-ion Battery Behaviour* (Version 0.1.4). https://www.github.com/pybamm-team/pybamm-eis

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

Haario, H., Saksman, E., & Tamminen, J. (2001). An Adaptive Metropolis Algorithm. *Bernoulli*, *7*(2), 223. https://doi.org/10.2307/3318737

Hoffman, M. D., & Gelman, A. (2011). *The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo*. https://arxiv.org/abs/1111.4246

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

Loshchilov, I., & Hutter, F. (2017). *Decoupled Weight Decay Regularization*. arXiv. https://doi.org/10.48550/ARXIV.1711.05101

Lu, D., Scott Trimboli, M., Fan, G., Zhang, R., & Plett, G. L. (2021). Implementation of a physics-based model for half-cell open-circuit potential and full-cell open-circuit voltage estimates: Part II. Processing full-cell data. *Journal of The Electrochemical Society*, *168*(7), 070533. https://doi.org/10.1149/1945-7111/ac11a5

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, *21*(6), 1087–1092. https://doi.org/10.1063/1.1699114

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

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. l. (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

Yang, X.-S., & Suash Deb. (2009). Cuckoo Search via levy flights. *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, 210–214. https://doi.org/10.1109/NABIC.2009.5393690