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** [2,4], 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Published: unpublished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 has been developed to enable parameter identification of various battery models, including the electrochemical and equivalent circuit models provided by the popular open-source package PyBaMM (Sulzer et al., 2021). PyBOP

Similarly, PyBOP can be used for parameter design optimisation under user-defined operating conditions across a variety of model structures. PyBOP enables battery model parameterisation across a range of methods with diagnostics on the performance and convergence of the identified or optimised parameters. 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 a Python package designed to provide a user-friendly, object-oriented interface for the optimisation of battery models. PyBOP leverages the open-source PyBaMM (Sulzer et al., 2021) package for formulation and solving of these battery models. PyBOP is intended to serve a broad audience of students, engineers, and researchers in both academia and the battery industry, by enabling usage of predictive battery models where not previously possible. PyBOP prioritises clear and informative diagnostics and workflows for users of varying expertise, by providing advanced optimisation and sampling algorithms. These methods are provided through interfaces to PINTS (Clerx et al., 2019), SciPy (Virtanen et al., 2020), in addition to the PyBOP constructed algorithms such as Adaptive Moment Estimation with Weight Decay (AdamW), and Cuckoo search.

PyBOP supports the Battery Parameter eXchange (BPX) standard (Korotkin et al., 2023) for sharing battery parameter sets. As these parameter sets are costly to obtain due to: the equipment and time spent on characterisation experiments, the requirement of battery domain knowledge, and the computational cost of parameter estimation. PyBOP reduces these costs by enabling fast computational estimation with parameter set interoperability.

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) and pybamm-eis for numerical impedance spectroscopy as the identified parameters are easily exportable from PyBOP into these packages.

Architecture

PyBOP has a tiered data structure aimed at computing and processing the forward model predictions and packaging the required information to the optimisation and sampling algorithms. The forward model is solved using the popular battery modelling package, PyBaMM, with construction, parameterisation, and discretisation managed through PyBOP's model interface to PyBaMM. This approach enables a robust object construction process with consistent interfacing between the models and optimisers. The statistical methods and optimisation algorithms are then constructed to interface cleanly with the forward model predictions. Furthermore, identifiability metrics are provided alongside the estimated parameters through Hessian approximation of the cost functions in the frequentist workflows and posterior moments in the Bayesian workflows.

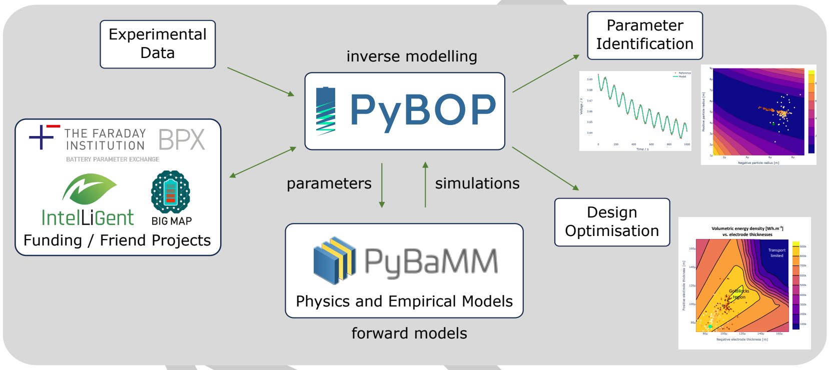
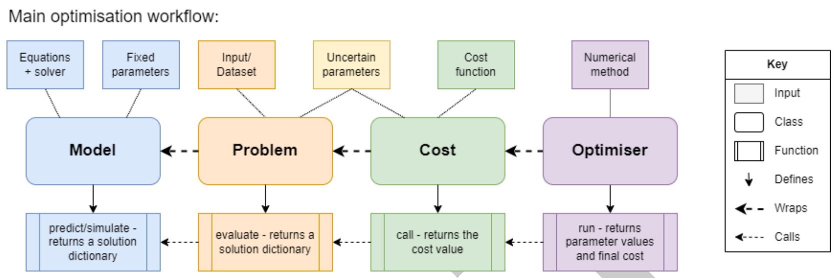


Figure 1: PyBOP's interface to supporting funding agencies, alongside a visualisation of the general workflow for parameterisation and optimisation

PyBOP formulates the inference process into four key classes, namely the model, problem, cost, and optimiser/sampler, as shown in Figure 2. Each of these objects represent a base class with child classes constructing specialised functionality for inference or optimisation workflows. The model class constructs a PyBaMM forward model for a given set of model equations provided from PyBaMM, initial conditions, spatial discretisation, and numerical solver. By composing PyBaMM directly into PyBOP, specialised models can be constructed alongside the default models which can be modified, and optimally constructed for the inference tasks. One such example of this, is the spatial rediscretisation that is performed when geometric parameters are optimised. In this situation, PyBOP minimally rediscretises the PyBaMM model while maintaining the problem, cost, and optimiser objects, providing improved performance benefits to users. Alongside construction of the forward model, PyBOP's model class provides methods for acquiring sensitivities from the prediction, enabling gradient based optimisation algorithms. This prediction alongside it's corresponding sensitivities are provided to the problem class for processing and exception control. A standardised data structure is then provided to the cost classes, which provides a distance, design, or likelihood based metric for optimisation. For deterministic optimisation, the optimisers minimise the corresponding cost function or the negative log-likelihood if a likelihood class is provided. Bayesian inference is provided through Monte Carlo sampling classes, which accept the child cost class, LogPosterior and samples from it using Pints' based Monte Carlo algorithms at the time of submission. In the typical workflow, the classes in Figure 2 are constructed in sequence.

In addition to the core architecture, PyBOP offers multiple specialised inference and optimisation processes. One such instance is numerical electrochemical impedance spectroscopy predictions by discretising the forward model into sparese mass matrix form with accompanying auto-differentiation generated jacobian. These objects are then translated into the frequency domain with a linear solve used to compute the battery model impedance. In this situation, the forward models are constructed within the spatial rediscretisation workflow, allowing for geometric parameter inference from EIS forward model predictions. Furthermore, PyBOP builds upon the JAX (Bradbury et al., 2018) numerical solvers provided by PyBaMM by offering JAX-based cost function for auto-differentiation of the forward model with respect to the parameters. This functionality provides a performance improvement alongside an interface to JAX-based inference packages, such as Numpyro (Phan et al., 2019), BlackJAX (Cabezas et al., 2024), and Optax (DeepMind et al., 2020).



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. 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 1: List of available model, problem and cost (or likelihood) classes.

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

Likewise, the current algorithms available for optimisation tasks are presented in Table 2. From this stage onwards, the point-based parameterisation and design optimisation tasks will simply be referred to as optimisation tasks. This simplification can be justified by inspecting Equation 4 and Equation 6 and confirming that deteriminstic parameterisation can be viewed as an optimisation task to minimise a distance-based cost function.

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

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

Gradient-based	Evolutionary Strategies	(Meta)heuristic
	Cuckoo search	

As discussed above, `PyBOP` offers Bayesian inference methods such as Maximum a Posteriori (MAP) presented alongside the point-based methods in Table 1; however, for a full Bayesian framework, Monte Carlo sampling is implemented within `PyBOP`. These methods construct a posterior distribution on the inference parameters which can used for uncertainty and practical identifiability. The individual sampler classes are currently composed within `PyBOP` from the `PINTS` library, with a base sampling class implemented for interoperability and direct integration to the `PyBOP` model, problem, and likelihood classes. The currently supported samplers are presented in Table 3.

Table 3: PyBOP's supported 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	Slice Stepout		Metropolis Adjusted Langevin
Relativistic	Haario			

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 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 its reduced-order variants including the single particle model (SPM) (Planella et al., 2022), as well as the multi-species, multi-reaction (MSMR) model (Verbrugge et al., 2017). Simplified models that retain acceptable 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 physical parameters on battery performance. This separation of complexity conventionally results in multiple parameterisations for a single battery type, dependent on the model structure.

Examples

Parameterisation

Battery model parameterisation is challenging due to the high number of parameters required to identify compared 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) and excitations.

A generic data fitting optimisation problem may be formulated as:

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

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 observations $(\mathbf{y}_i)$ measured at times t_i. For gradient-based optimisers, the Jacobian of the cost function with respect to the unknown parameters, $(\frac{\partial \mathcal{L}}{\partial \theta})$ is computed for step size and directional information.

We next demonstrate the fitting of synthetic data of which the system parameters are known. In this example problem, we employ `PyBaMM`'s implementation of the single particle model (SPM) with an added contact resistance submodel. We assume that the battery model is already parameterised except for two dynamic parameters, namely the lithium diffusivity of the negative electrode active material particle (denoted "negative particle diffusivity") and the contact resistance. We generate synthetic data from a one-hour discharge from 100% state of charge, to 0% (denoted as 1C rate), followed by 30 minutes of relaxation. This data is then corrupted with zero mean gaussian noise of amplitude 2mV, shown by the dots in Figure 3 (left). The initial states are assumed known, although such an assumption is not necessary in general. The underlying cost landscape explored by the optimiser is shown in Figure 3 (right).

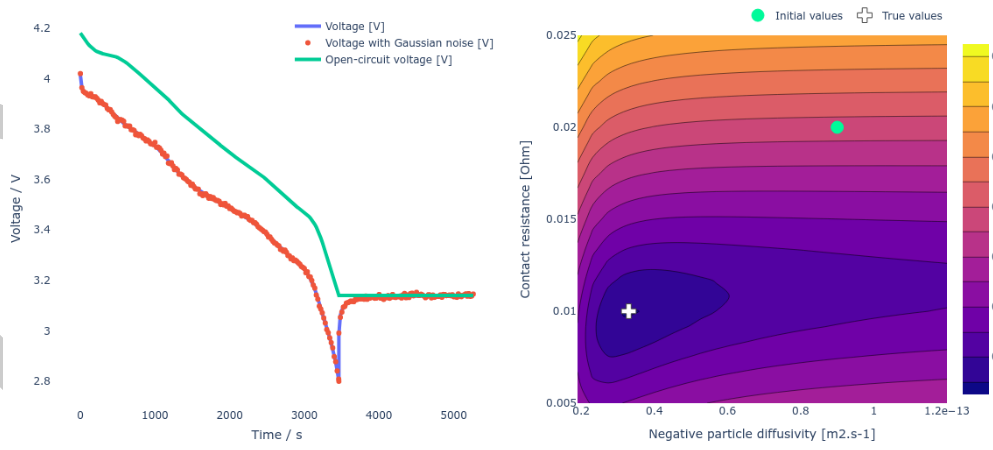


Figure 3: The cost landscape for the parameterisation problem using the root mean square error.

As gradient information is available for this problem, the choice of distance-based cost function and optimiser is not constrained. Due to the vastly different magnitudes of the two parameters, we apply two of the parameter transformations offered by `PyBOP`, namely the log transformation for the negative particle diffusivity and the scaled transformation (with a coefficient of 100) for the contact resistance. This application transforms the optimisers search space, enabling a shared step-size between the parameters; however, in general is not required. As a demonstration of PyBOP's parameterisation capabilities, Figure 4 (left) shows the rate of convergence for each of the distance-minimising cost functions, while Figure 4 (right) displays analogous results for maximising a likelihood. Here, the optimisation is performed with SciPy Minimize using the gradient-based L-BFGS-B method.

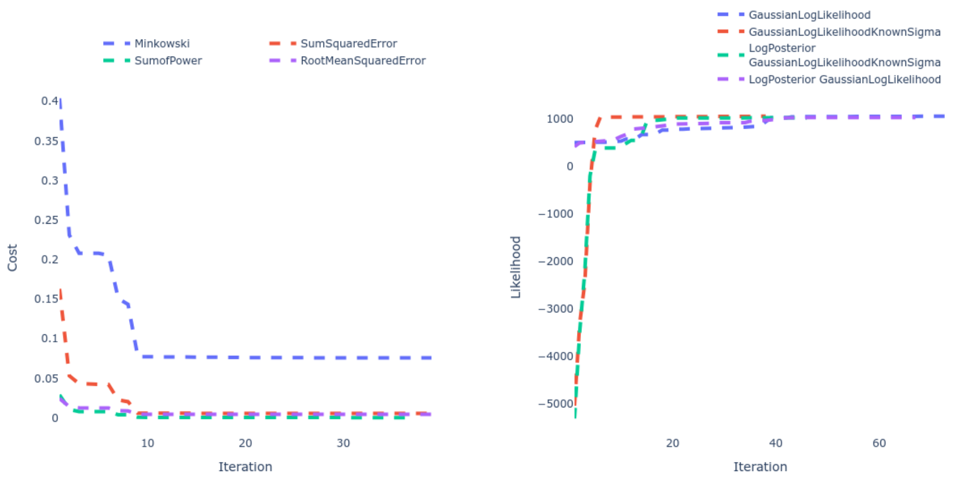


Figure 4: Convergence in the likelihood functions obtained using various likelihood functions and SciPy minimize.

Furthermore, we can also compare the performance of the various optimisation algorithms divided by category: gradient-based in Figure 6 (left), evolution strategies in Figure 6 (middle) and (meta)heuristics in Figure 6 (right) for a sum squared error cost function. Note that optimiser performance depends on the cost landscape, prior information, and corresponding hyperparameters for each specific problem.

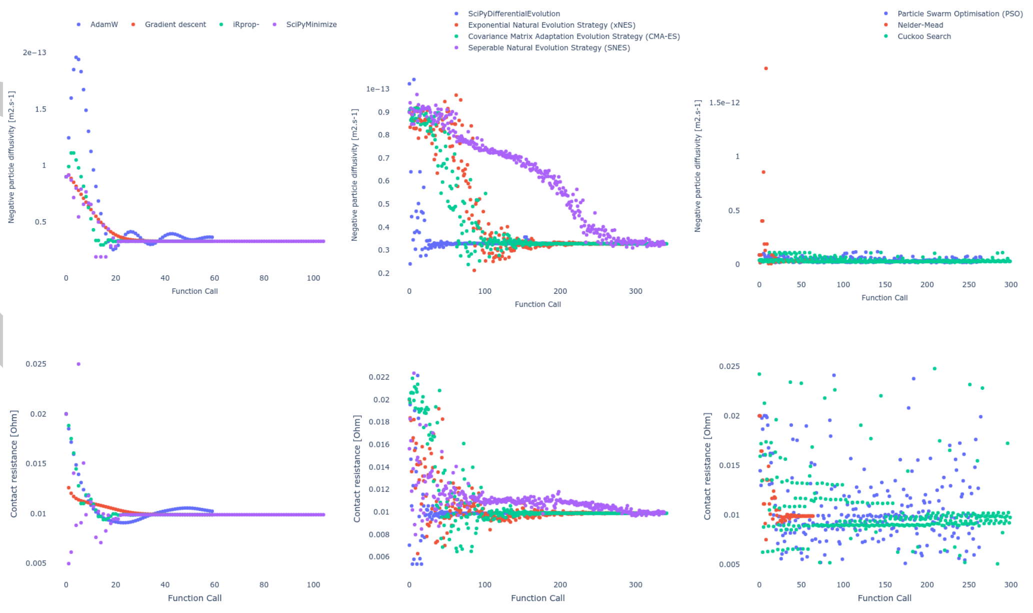


Figure 5: Convergence in the parameter values obtained for various (meta)heuristics.

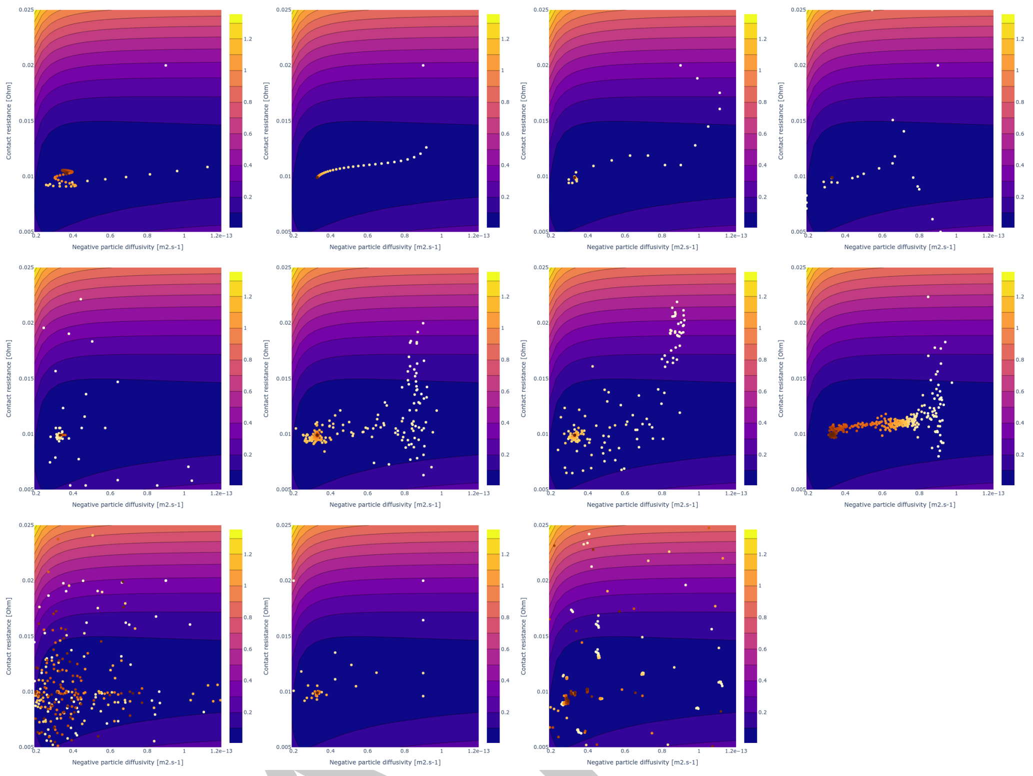


Figure 6: Counter plot for all optimisers.

This parameterisation task can also be approached from the Bayesian perspective, which we will present below using PyBOP's sampler methods. The optimisation equation presented in equation Equation 4 does not represent the Bayesian parameter identification task, and as such we introduce Bayes Theorem as,

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)} \tag{5}$$

where, $P(\theta|D)$ is the posterior and represents the probability density function of the parameter. $P(D|\theta)$ is the likelihood function and assesses the parameter values alongside a noise model. $P(\theta)$ encapsulates the prior knowledge on the parameters, and finally $P(D)$ is the model evidence and acts as a normalising constant such that the final posterior is a correctly scaled density function. Our aim in parameter inference is to identify the parameter values with the highest probability, which can be presented from point-based metric or as the posterior distribution, which provides additional information on the uncertainty of the identified parameters. To acquire this posterior distribution, we provide Monte-Carlo sampling methods. These methods sample from the posterior through a variety of methods, including gradient-based such as No-U-Turn (Hoffman & Gelman, 2011) and Hamiltonian (Brooks et al., 2011) as well as heuristic methods such as Differential Evolution (Braak, 2006), and finally conventional methods based on random sampling with rejection criteria (Metropolis et al., 1953). PyBOP offers a sampling class which provides an interface for these samplers, which are supported from the Probabilistic Inference of Noise Time-Series (PINTS) package. Figure 7 below presents the sampled posterior for the synthetic workflow described above, using an adaptive covariance based sampler, Haario Bardenet (Haario et al., 2001).

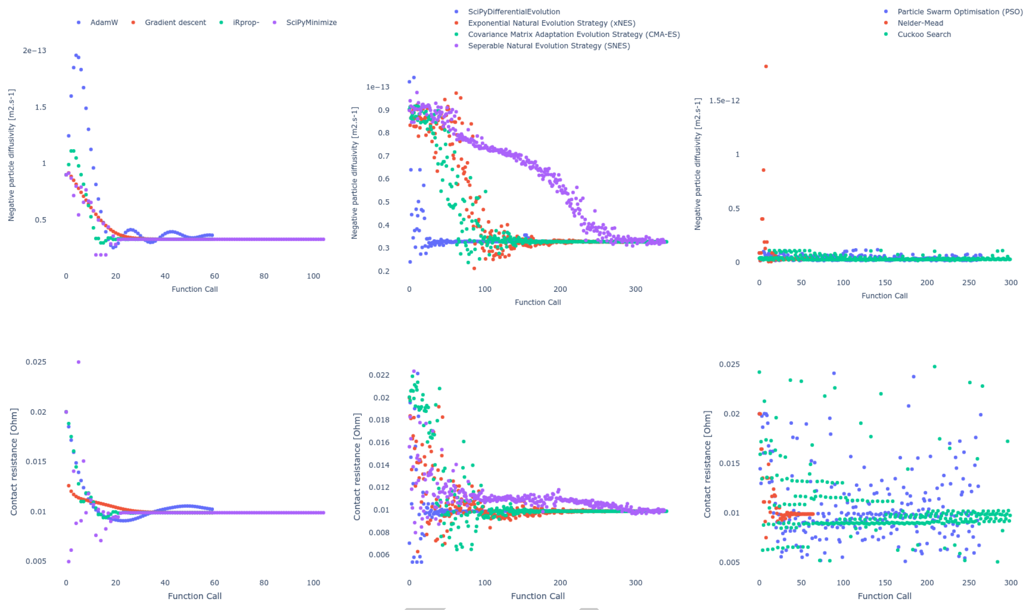


Figure 7: [Placeholder]: Posterior distributions for inferred parameters.

Design optimisation

Design optimisation is supported within PyBOP to guide future development of battery design by identifying parameter sensitivities which may unlock improvements in battery performance. This problem can be viewed similarly to the parameterisation workflows described above; however, with the aim of maximising a 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, but instead is the integrated energy from the vector normalised with the corresponding cell mass. This is typically quantified for operational conditions such as a 1C discharge, at a given temperature.

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

$$\min_{\theta \in \Omega} \mathcal{L}(\theta) \quad \text{subject to equations (1)-(3)} \tag{6}$$

in which $\mathcal{L} : \theta \mapsto [0, \infty)$ 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 problem of maximising gravimetric energy density subject to constraints on two of the geometric electrode parameters (Couto et al., 2023). For this example, we use PyBaMM's implementation of the single particle model with electrolyte (SPMe) to investigate the impact of the positive electrode thickness and active material volume fraction on the target cost. In this problem, we estimate the 1C rate from the theoretical capacity for each iteration of the design.

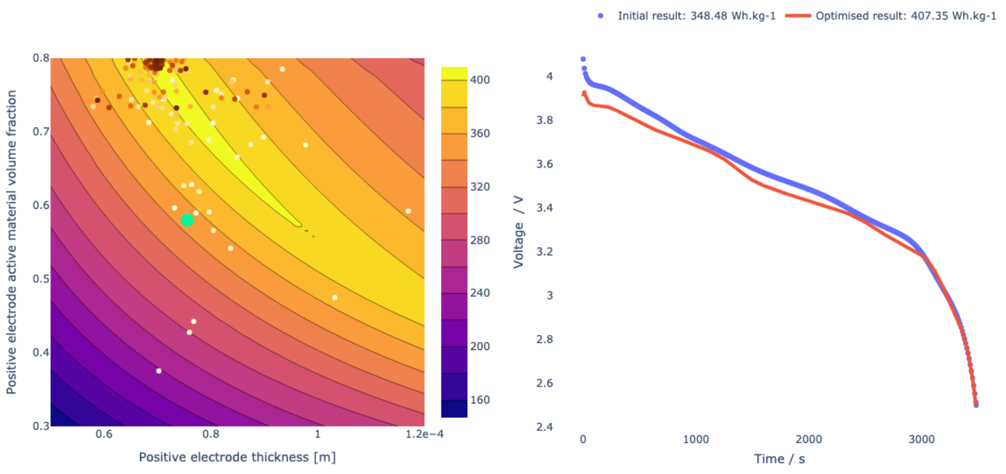


Figure 8: The voltage profiles for a 1C discharge predicted using the initial and optimised parameter values.

Figure 8 (left) shows the optimiser's search for the maximum gravimetric energy density within the parameter space. For this example, we employ the particle swarm optimisation (PSO) algorithm. The predicted improvement in the discharge profile between the initial and optimised parameter values is shown in Figure 8 (right).

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).

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

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

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs* (Version 0.3.13). http://github.com/jax-ml/jax

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

Cabezas, A., Corenflos, A., Lao, J., & Louf, R. (2024). *BlackJAX: Composable Bayesian inference in JAX*. https://arxiv.org/abs/2402.10797

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

DeepMind, Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., Buchlovsky, P., Budden, D., Cai, T., Clark, A., Danihelka, I., Dedieu, A., Fantacci, C., Godwin, J., Jones, C., Hemsley, R., Hennigan, T., Hessel, M., Hou, S., … Viola, F. (2020). *The DeepMind JAX Ecosystem*. http://github.com/google-deepmind

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

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

Phan, D., Pradhan, N., & Jankowiak, M. (2019). Composable effects for flexible and accelerated probabilistic programming in NumPyro. *arXiv Preprint arXiv:1912.11554*.

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