PyBOP: Python Battery Model Optimisation and Parameterisation

Brady Planden, Nicola Courtier, David Howey

Engineering Science
University of Oxford

Faraday ECR, March 2024

Battery
Intelligence
Lab

Battery models are useful for:

Physical Insight

How well does the model predict future evolution?

How does parameter X affect variable Y?

Battery models are useful for:

Physical Insight

How well does the model predict future evolution?

How does parameter X affect variable Y?

Operational Improvements

What is the current state of the cell?

How fast can I charge the cell?

Battery models are useful for:

Physical Insight
How well does the model predict future evolution?
How does parameter X affect variable Y?

Operational Improvements
What is the current state of the cell?
How fast can I charge the cell?

Design Optimisation
How energy dense can the cell be?
What is the best configuration for this operation?

Well then, why isn't everyone modelling?

Parameter values

Geometric & non-geometric

Requires cell disassembly

Observability can be unclear

Well then, why isn't everyone modelling?

Parameter values

Geometric & non-geometric

Requires cell disassembly

Observability can be unclear

Domain expertise

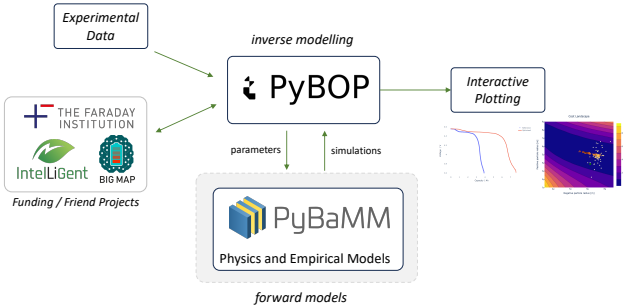
Is a more complex model needed?

Where should the complexity be added?

PyBOP is a platform to solve these challenges

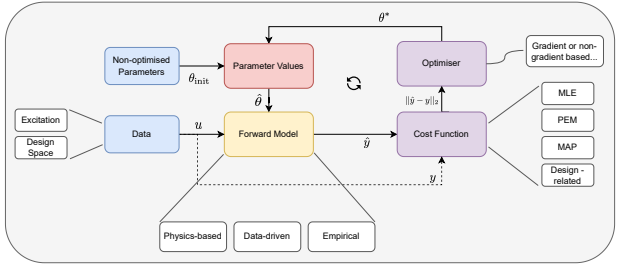
Provides workflows for parameter identification and exploitation:

- Physics-based parameter identification
- Frequentistic and Bayesian methods
- Fast empirical model fitting
- Optimised cell designs



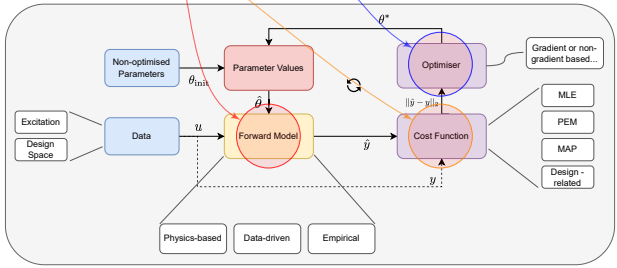
This is acheived through concise, inuitive design

```
# Generate problem, cost function, and optimisation class
problem = pybop.Problem(model, parameters, dataset)
cost = pybop.SumSquaredError(problem)
optim = pybop.Optimisation(cost, optimiser=pybop.GradientDescent)
x, final_cost = optim.run()
```



This is acheived through concise, inuitive design

```
# Generate problem, cost function, and optimisation class
problem = pybop.Problem(model, parameters, dataset)
cost = pybop.SumSquaredError(problem)
optim = pybop.Optimisation(cost, optimiser=pybop.GradientDescent)
x, final_cost = optim.run()
```



Example: Fast parameter estimation for circuit models

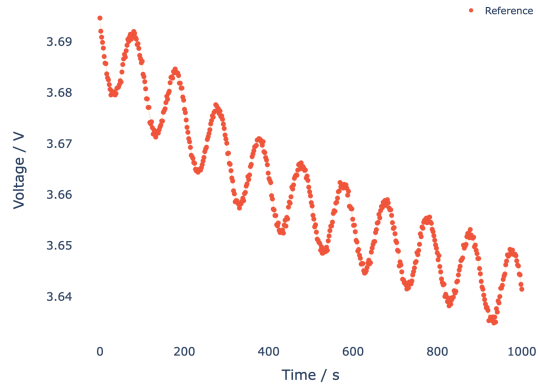
Benefiting from gradient and non-gradient optimisers

Model: 2-RC Thevenin model

Data: Synthetic data w/ unknown parameters

Conditions: discharge biased sinusoid function

Parameters: R_0, R_1, R_2, C_1, C_2



Example: Fast parameter estimation for circuit models

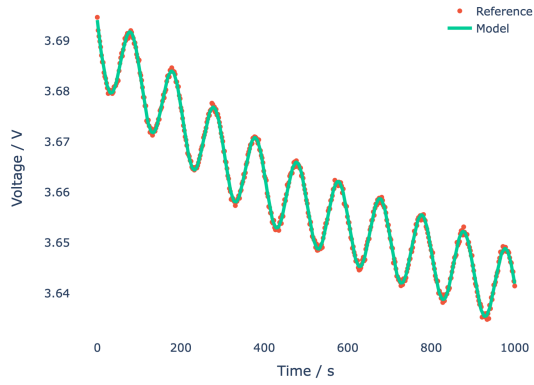
Benefiting from gradient and non-gradient optimisers

Model: 2-RC Thevenin model

Data: Synthetic data w/ unknown parameters

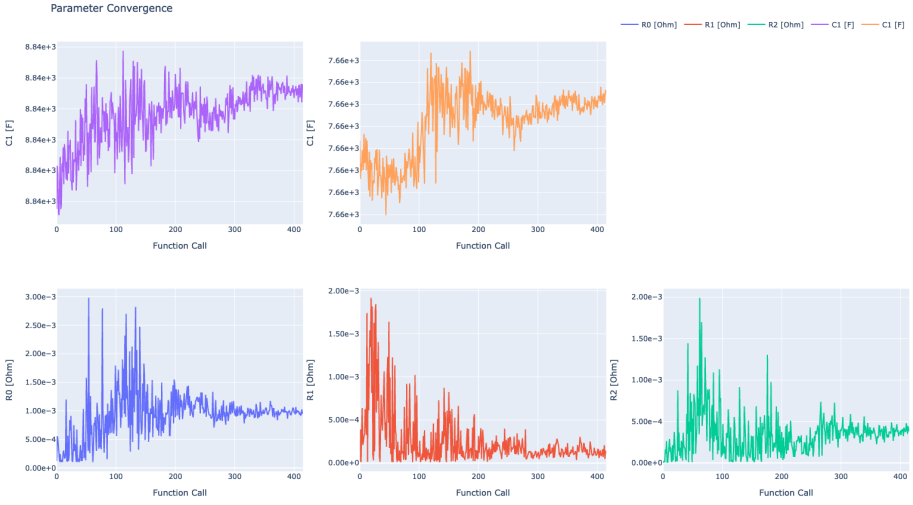
Conditions: discharge biased sinusoid function

Parameters: R_0, R_1, R_2, C_1, C_2



High dimensional parameter spaces still pose a challenge

Low identifiability on capacitance – excitation matters!



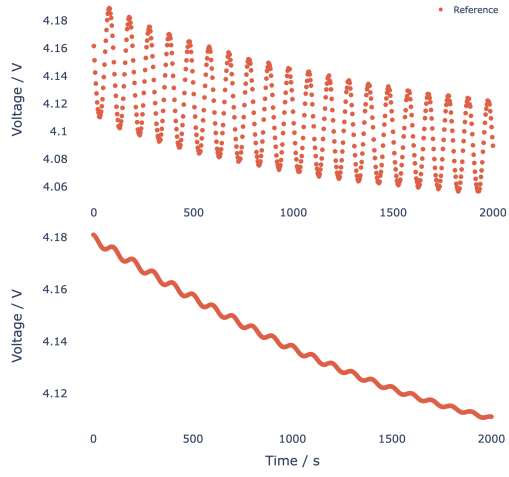
Example: SPM parameter identification

Electrode thickness values for a single particle model[a] are identified from synthetic data.

- Fitting signals: Terminal voltage and predicted OCV
- Initial conditions $\sim \mathcal{N}(\mu, \sigma^2)$
- Maximium iters: 150

Ground truth parameter values:

$$[\mathrm{L_n} = 8.52\text{e-}05,\ \mathrm{L_p} = 7.56\text{e-}05]$$



[a] All models are tested daily on the cloud!

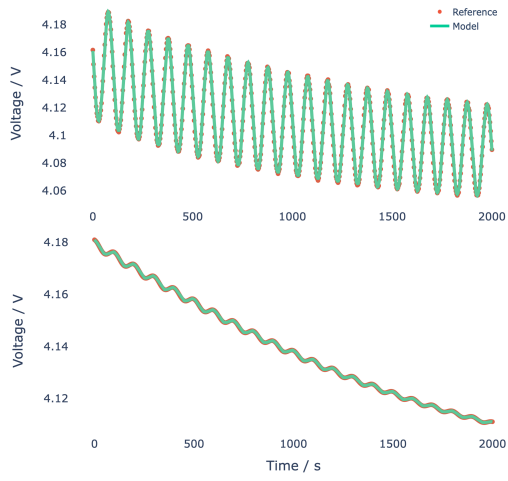
Example: SPM parameter identification

Electrode thickness values for a single particle model[a] are identified from synthetic data.

- Fitting signals: Terminal voltage and predicted OCV
- Initial conditions $\sim \mathcal{N}(\mu, \sigma^2)$
- Maximium iters: 150

Ground truth parameter values:

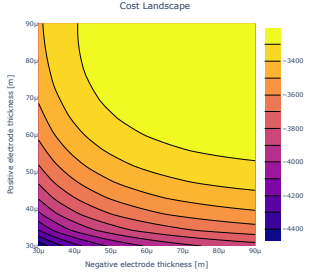
$$[\mathrm{L_n} = 8.52\text{e-}05,\ \mathrm{L_p} = 7.56\text{e-}05]$$



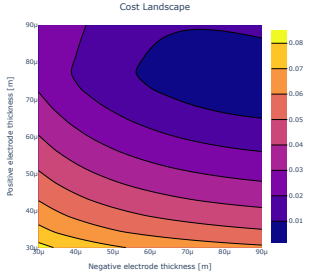
[a] All models are tested daily on the cloud!

PyBOP provides multiple cost functions for parameter inference

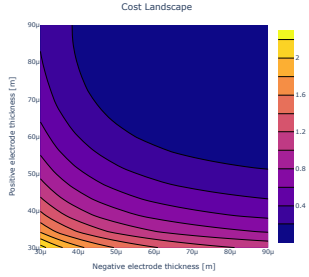
Maximum Likelihood



Root Mean Squared Error

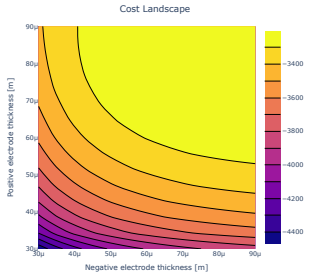


Sum of Squared Error

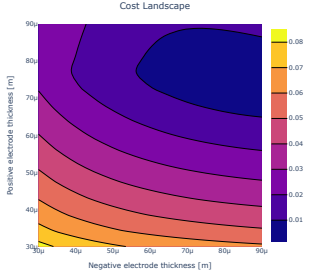


PyBOP provides multiple cost functions for parameter inference

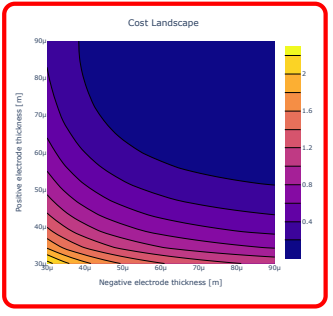
Maximum Likelihood



Root Mean Squared Error

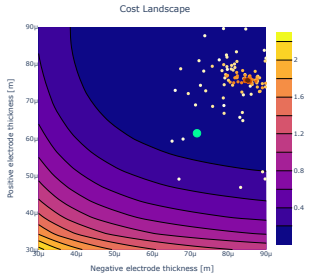


Sum of Squared Error

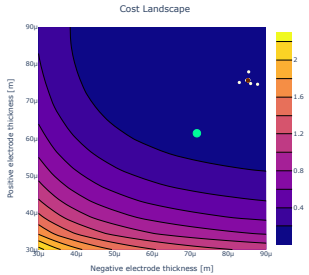


Verfiying optimality with convergence plots

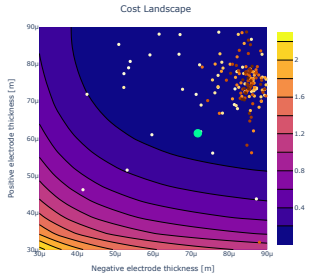
CMA-ES



Differential Evolution

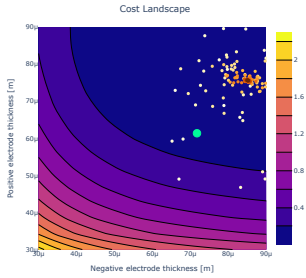


Particle Swam

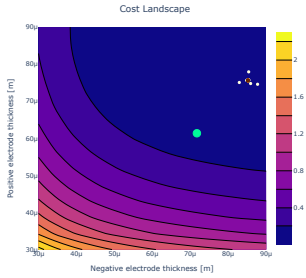


Verfiying optimality with convergence plots

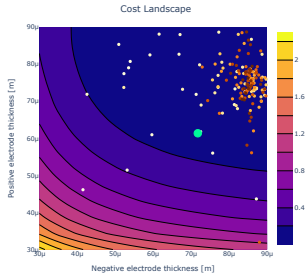
CMA-ES



Differential Evolution



Particle Swam



Example: Physics-based design optimisation

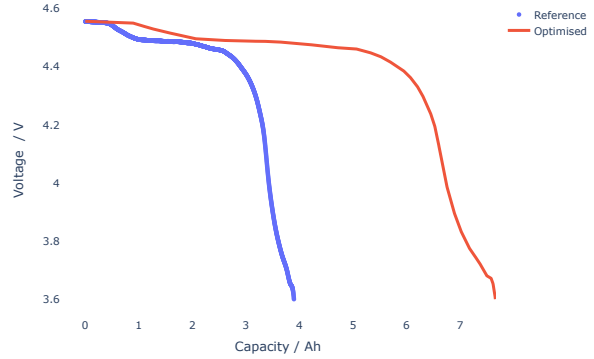
Chemistry: LNMO-Graphite/SiOx

Target: Maximize cell-level specific energy density

Parameters: Electrode coating thickness

Conditions: 1C cycling

Optimal:
$[L_n = 97\text{e-}06,\ L_p = 126\text{e-}06]$



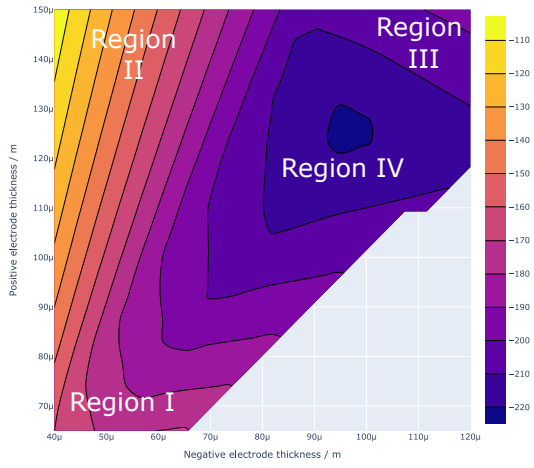
PyBOP enables physical insights into design optimisation

Region I: Electrodes too thin compared to inactive material

Region II: Poor N:P balancing

Region III: Electrodes too thick and cause transport limitations

Region IV: Ideal N:P balance and optimised wrt active material and transport



Get involved and become a developer and/or user!

GitHub Repository:
pybop-team/pybop

Installation:
`pip install pybop`

Examples

Docs

Benchmarks

Future Projects:

- Expansion of Bayesian methods
- New optimisation algorithms
- Improved workflows for battery relevant model optimisation
- Just released v24.3!