

1st Online Developer Meeting



(Pending new logo)

The PyBOP Team 

21st March 2024

Agenda




- ▶ Presentation on current version of PyBOP (Nicola, 15 min)
- ▶ Quick introduction from everyone (5 min)
- ▶ Discussion of open issues (30 min)
- ▶ Planning and assignment of tasks (10 min)

Introduction



With thanks to:

- ▶ the PyBaMM developers  PyBaMM
- ▶ the PINTS developers 



1. Parameter estimation from cell data

- How will my battery respond to different excitations?
- How is degradation affecting my battery performance?

2. Model comparison

- What can I learn from a more detailed model?
- Is it worth the extra computational effort?



3. Design optimisation within practical constraints

- How energy dense can I make my battery (at some rate)?

Build a model

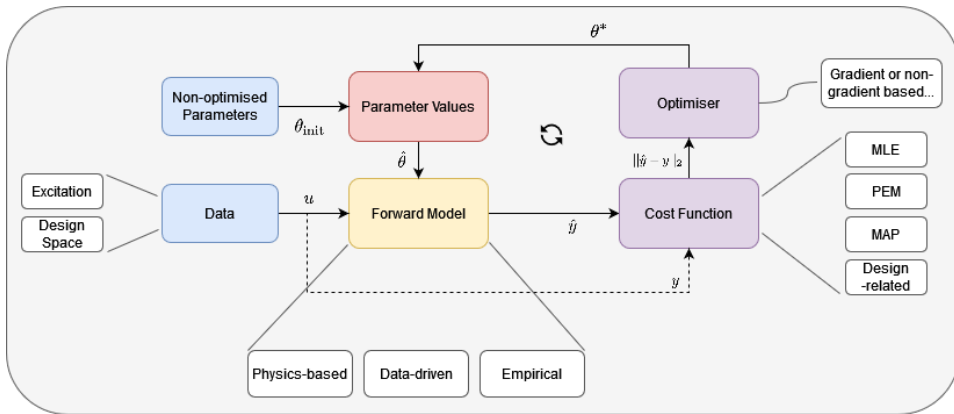


Perform an optimisation

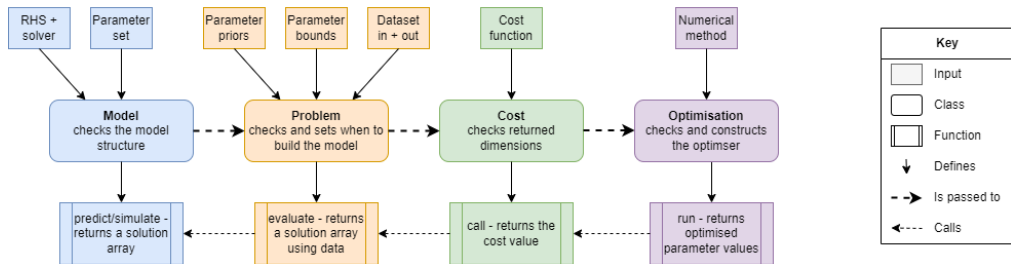


Learn from the result

Icons made by Uniconlabs, RaftelDesign and Freepik from www.flaticon.com



Main optimisation workflow:



Demo/examples



- ▶ Parameterisation
- ▶ Design optimisation

1. Define a model (PyBaMM's SPM) and fitting parameters

```
import pybop
import numpy as np

# Define model
parameter_set = pybop.ParameterSet.pybamm("Chen2020")
model = pybop.lithium_ion.SPM(parameter_set=parameter_set)

# Fitting parameters
parameters = [
    pybop.Parameter(
        "Negative particle radius [m]",
        prior=pybop.Gaussian(6e-06, 0.1e-6),
        bounds=[1e-6, 9e-6],
    ),
    pybop.Parameter(
        "Positive particle radius [m]",
        prior=pybop.Gaussian(4.5e-06, 0.1e-6),
        bounds=[1e-6, 9e-6],
    ),
]
```


2. Generate synthetic data and define problem, cost and optimisation

```
# Generate data
sigma = 0.001
t_eval = np.arange(0, 900, 2)
values = model.predict(t_eval=t_eval)
corrupt_values = values["Voltage [V]"].data + np.random.normal(0, sigma, len(t_eval))

# Form dataset
dataset = pybop.Dataset(
    {
        "Time [s]": t_eval,
        "Current function [A]": values["Current [A]"].data,
        "Voltage [V]": corrupt_values,
    }
)

# Generate problem, cost function, and optimisation class
problem = pybop.FittingProblem(model, parameters, dataset)
cost = pybop.SumSquaredError(problem)
optim = pybop.Optimisation(cost, optimiser=pybop.CMAES)
optim.set_max_iterations(100)
```

3. Run the optimisation and plot the results

```
# Run the optimisation
x, final_cost = optim.run()
print("True parameters:", true_x)
print("Estimated parameters:", x)

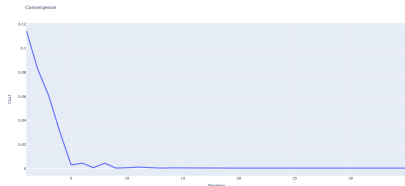
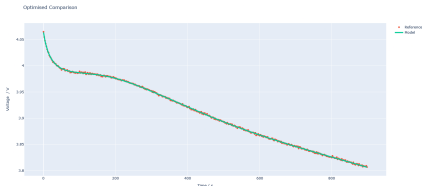
# Plot the timeseries output
pybop.quick_plot(problem, parameter_values=x, title="Optimised Comparison")

# Plot convergence and parameter traces
pybop.plot_convergence(optim)

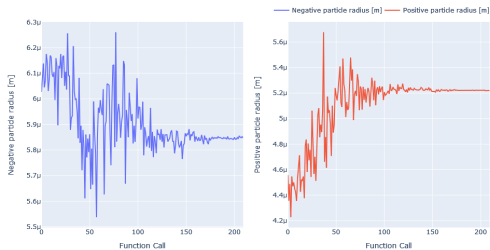
# Plot the parameter traces
pybop.plot_parameters(optim)

# Plot the cost landscape with optimisation path
pybop.plot2d(optim, steps=15)
```

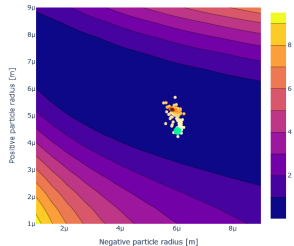
Parameterisation results



Parameter Convergence



Cost Landscape



1. Define a model (PyBaMM's SPM) and design parameters

```
# Define parameter set and model
parameter_set = pybop.ParameterSet.pybamm("Chen2020")
model = pybop.lithium_ion.SPM(parameter_set=parameter_set)

# Fitting parameters
parameters = [
    pybop.Parameter(
        "Positive electrode thickness [m]",
        prior=pybop.Gaussian(7.56e-05, 0.05e-05),
        bounds=[65e-06, 10e-05],
    ),
    pybop.Parameter(
        "Positive particle radius [m]",
        prior=pybop.Gaussian(5.22e-06, 0.05e-06),
        bounds=[2e-06, 9e-06],
    ),
]
```

2. Set the target experiment and define problem, cost and optimisation

```
# Define test protocol
experiment = pybop.Experiment(
    ["Discharge at 1C until 2.5 V (5 seconds period)",
]
init_soc = 1 # start from full charge
signal = ["Voltage [V]", "Current [A]"]

# Generate problem
problem = pybop.DesignProblem(
    model, parameters, experiment, signal=signal, init_soc=init_soc
)

# Generate cost function and optimisation class:
cost = pybop.GravimetricEnergyDensity(problem)
optim = pybop.Optimisation(
    cost, optimiser=pybop.PSO, verbose=True, allow_infeasible_solutions=False
)
optim.set_max_iterations(15)
```

3. Run the optimisation and plot the results

```
# Run optimisation
x, final_cost = optim.run()
print("Estimated parameters:", x)
print(f"Initial gravimetric energy density: {-cost(cost.x0):.2f} Wh.kg-1")
print(f"Optimised gravimetric energy density: {-final_cost:.2f} Wh.kg-1")

# Plot the timeseries output
if cost.update_capacity:
    cost.problem._model.approximate_capacity(x)
pybop.quick_plot(problem, parameter_values=x, title="Optimised Comparison")

# Plot the cost landscape with optimisation path
pybop.plot2d(optim, steps=3)
```

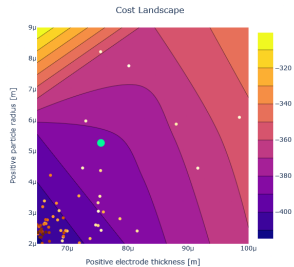
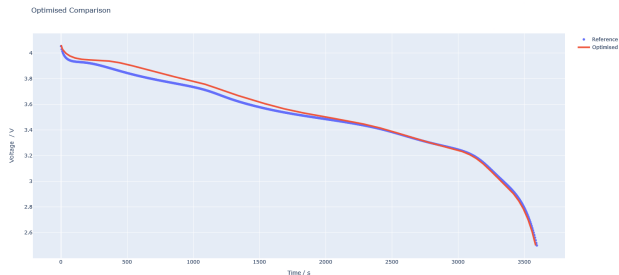
Output

Estimated parameters: $[6.51159429 \times 10^{-5} \ 2.32340723 \times 10^{-6}]$

Initial gravimetric energy density: 386.31 Wh.kg⁻¹

Optimised gravimetric energy density: 410.78 Wh.kg⁻¹

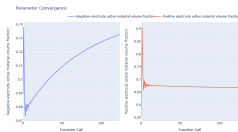
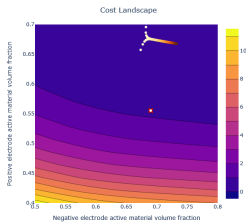
Results



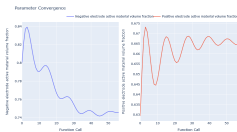
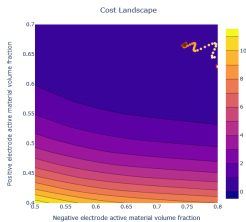
Choice of optimisers



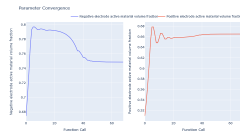
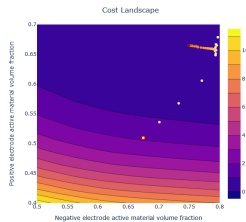
Gradient descent



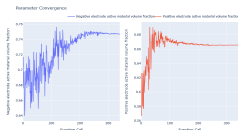
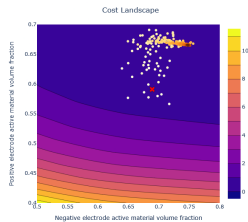
Adaptive moment (ADAM)



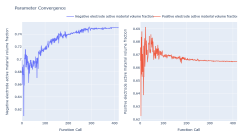
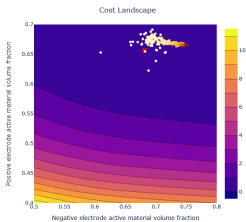
Resilient backpropagation (IRPropMin)



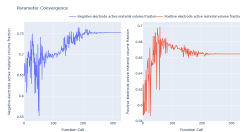
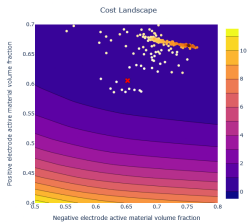
Stochastic natural evolution strategy (SNES)



Exponential natural evolution strategy (SNES)



Covariance matrix adaptation (CMA-ES)



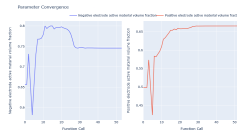
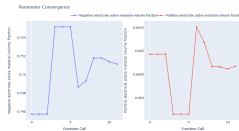
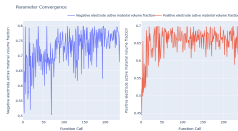
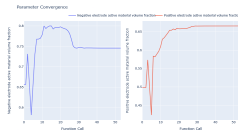
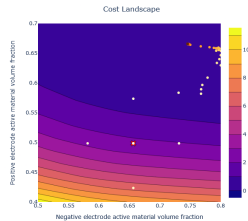
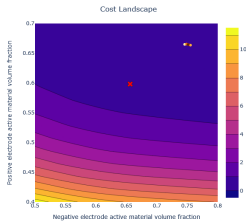
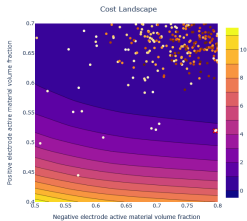
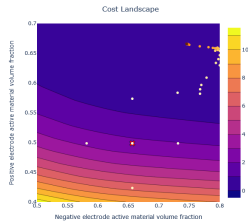
Other optimisers

Bound quadratic
approx. (BOBYQA)

Particle swarm
optimisation

SciPy differential
evolution

SciPy minimise
(COBYLA)



Questions?

Discussion



Let's introduce everyone:

- ▶ Where are you joining from?
- ▶ What is your research area of interest?
- ▶ How would you like to use/contribute to PyBOP?

Code of Conduct

PyBOP aims to foster a broad consortium of developers and users, both across and outside the Faraday Institution community.

Our values are:



Open-source (code and ideas should be shared)



Inclusivity and fairness (those who want to contribute may and input is appropriately recognised)



Inter-operability (aiming for modularity to enable maximum impact and inclusivity)



User-friendliness (putting user requirements first, thinking about user-assistance & workflows)

Icons made by Good Ware, Dewi Sari, Nhor Phai and Vectors Tank from www.flaticon.com

PyBOP

[https://github.com/
pybop-team/PyBOP](https://github.com/pybop-team/PyBOP)



SCAN ME

pybop-docs.readthedocs.io

Open issues



<https://github.com/pybop-team/PyBOP/issues>

Next steps



- ▶ How can you contribute?
- ▶ Project board: <https://github.com/orgs/pybop-team/projects/6/views/1>
- ▶ Date for next meeting