PyBaMM -Python Battery Mathematical Modelling

User Workshop
1st April 2020

Valentin Sulzer[a], Scott G Marquis[b], Robert Timms[b,e], Tom Tranter[c,e], Ferran Brosa Planella[d,e], Jacqueline Edge[e], Martin Robinson[f], Thibault Lestang[f], S Jon Chapman[b,e]

[a]Department of Mechanical Engineering, University of Michigan, 1231 Beal Ave, Ann Arbor, MI 48109, United States

[b]Mathematical Institute, University of Oxford, Andrew Wiles Building, Woodstock Road, Oxford, UK, OX2 6GG, UK

[c]EIL, Department of Chemical Engineering, University College London, London, WC1E 7JE, UK

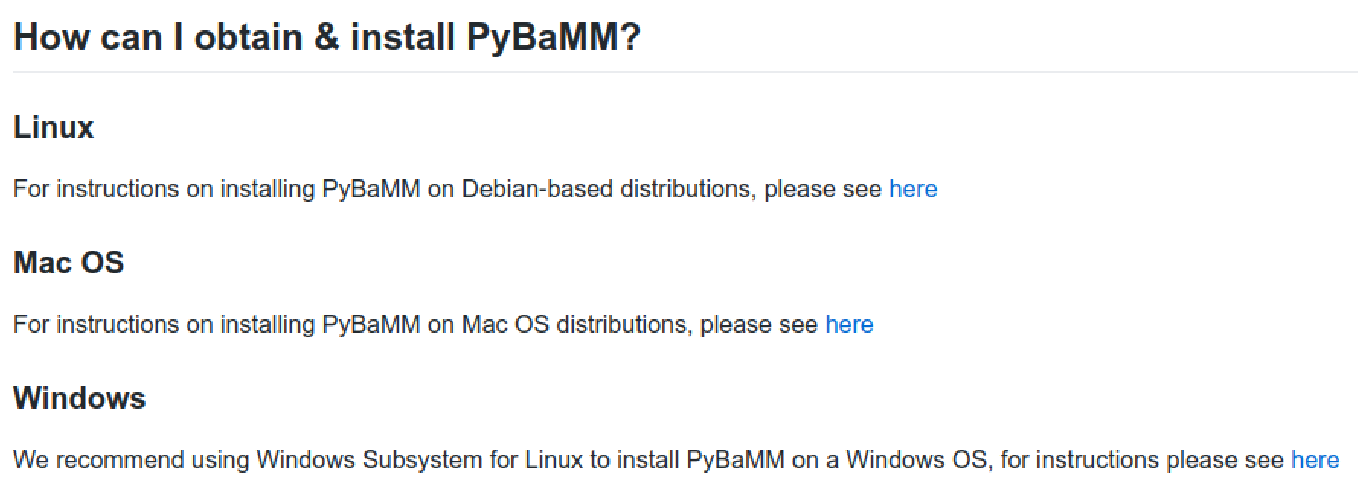
[d]WMG, University of Warwick, Gibbet Hill Road, Coventry, CV4 7AL, UK

[e]The Faraday Institution, Quad One, Becquerel Avenue, Harwell Campus, Didcot, OX11 0RA, UK

[f]Department of Computer Science, University of Oxford, 15 Parks Rd, Oxford OX1 3QD, UK

How to install

Instructions on the GitHub repo: https://github.com/pybamm-team/PyBaMM



How can I obtain & install PyBaMM?

Linux

For instructions on installing PyBaMM on Debian-based distributions, please see here

Mac OS

For instructions on installing PyBaMM on Mac OS distributions, please see here

Windows

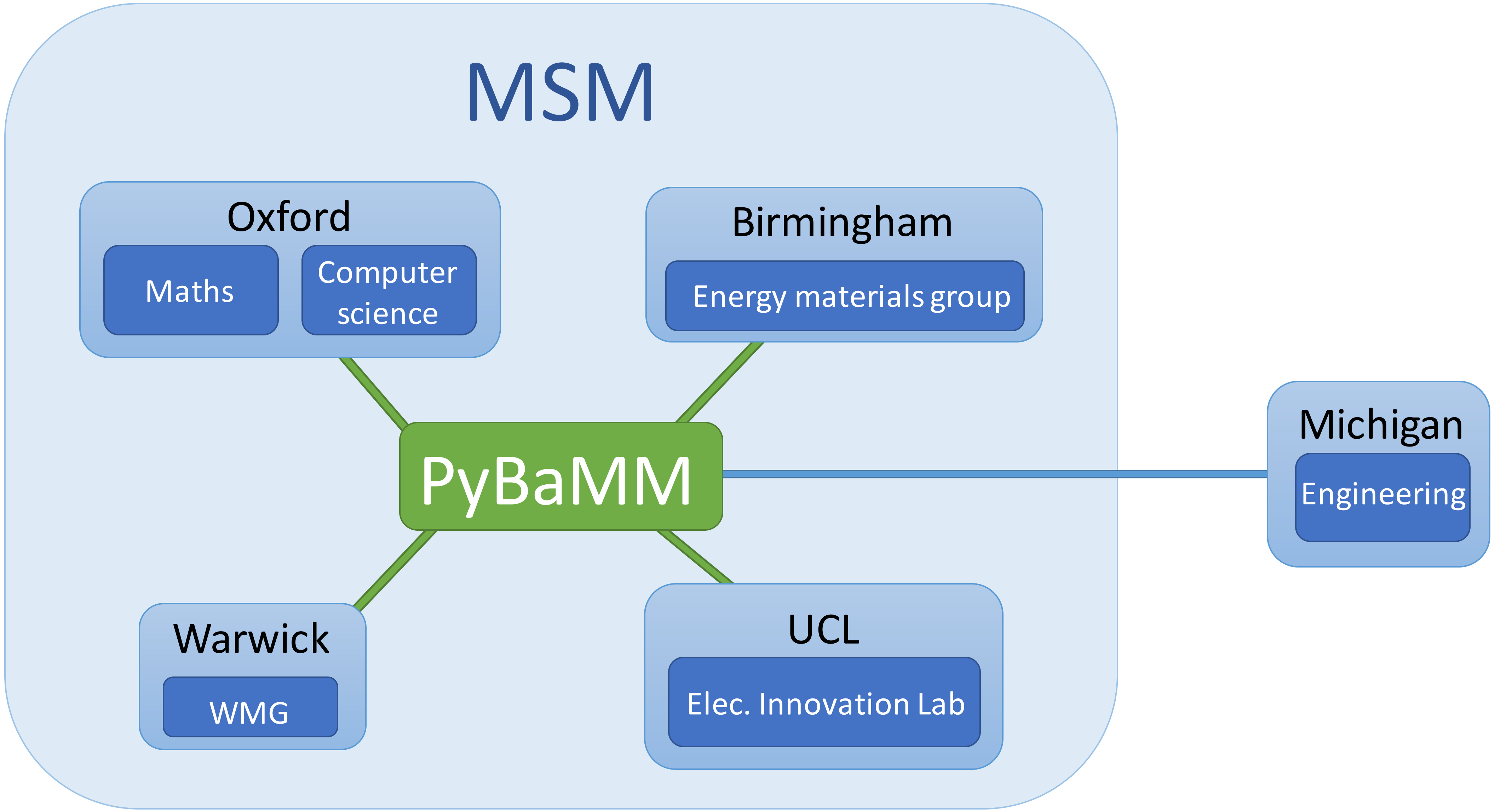
We recommend using Windows Subsystem for Linux to install PyBaMM on a Windows OS, for instructions please see here

We recommend that you install into a virtual environment. On Linux and Mac:

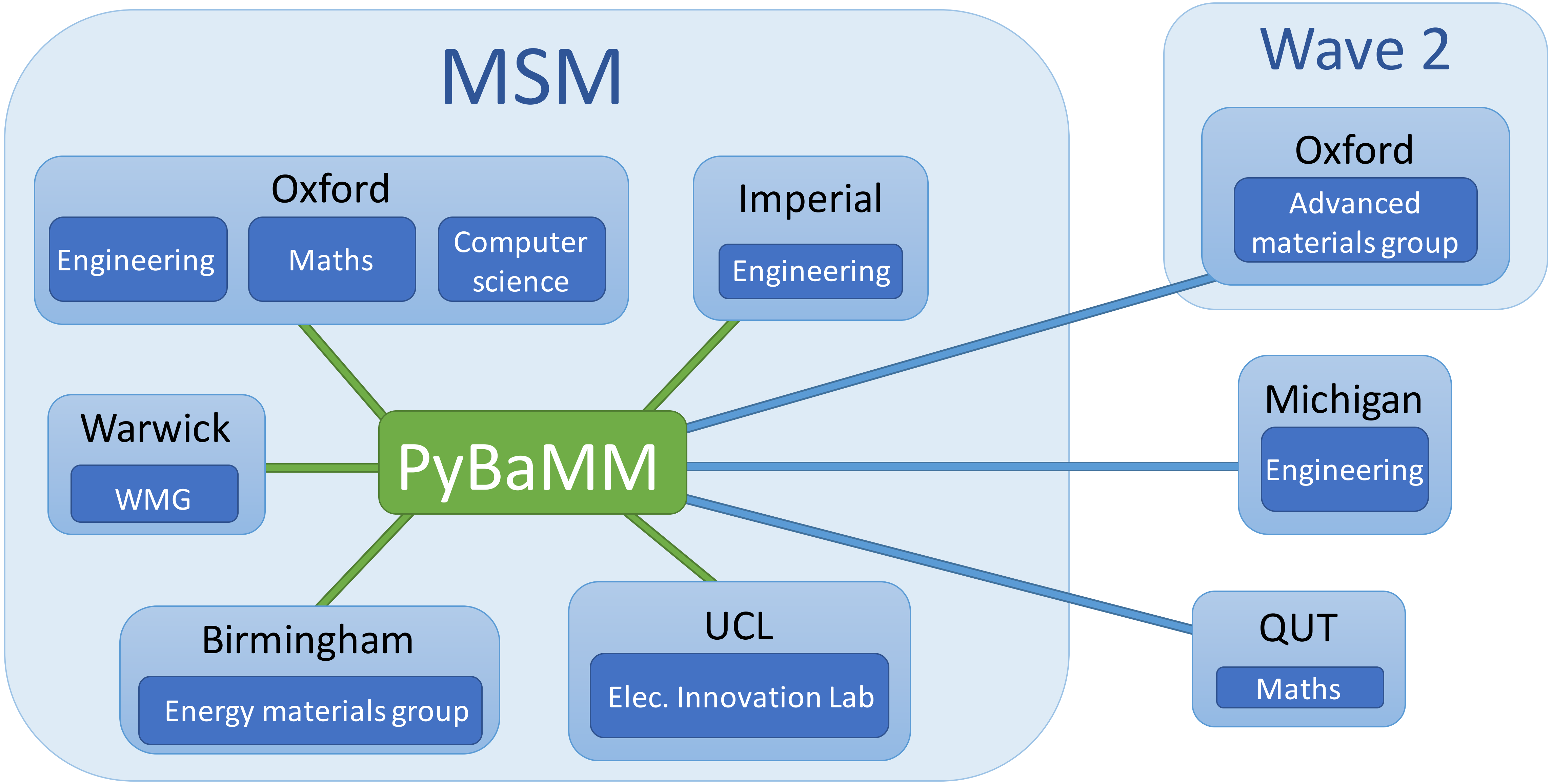
1. python3 -m venv env
2. source env/bin/activate
3. pip install pybamm

For Windows we recommend the use of Windows Subsystem for Linux (instructions online)

Who is PyBaMM? Current collaborators



Who is PyBaMM? Future collaborators



What is PyBaMM?

PyBaMM is a framework for building and solving battery models

FAST FLEXIBLE MODULAR

Many standard models already implemented and easy to reuse

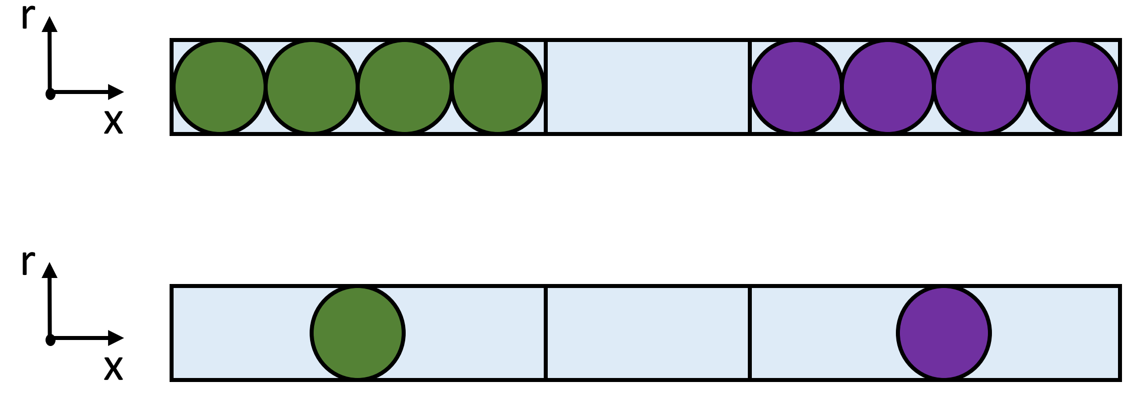
- Cell geometries: 1D, 2+1D single layer pouch cells
- Standard lithium-ion models: ODE, SPM, SPMe, DFN plus lead-acid models
- Chemistries: LCO, NCA, Graphite, LiPF6
- Fully coupled thermal models
- Experimental suite
  - Any voltage, current or power control (e.g. GITT, PITT, CCCV, drive cycle, etc.)
  - Easy interface to define new protocols
- Nonlinear parameters provided as functional form or as data
- Structured electrodes
- Interface with external software

Model structure

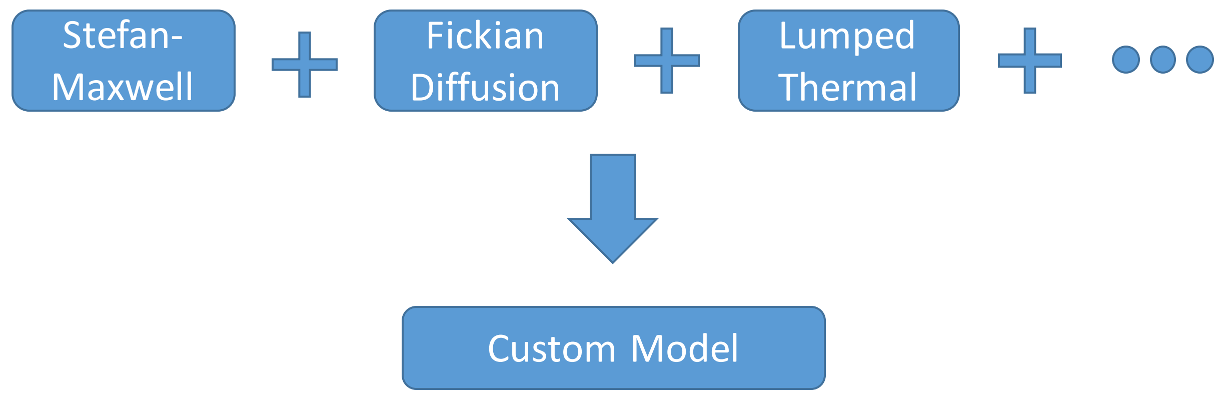
- Battery models are constructed from submodels

- Use or adapt pre-built models

- Create custom models



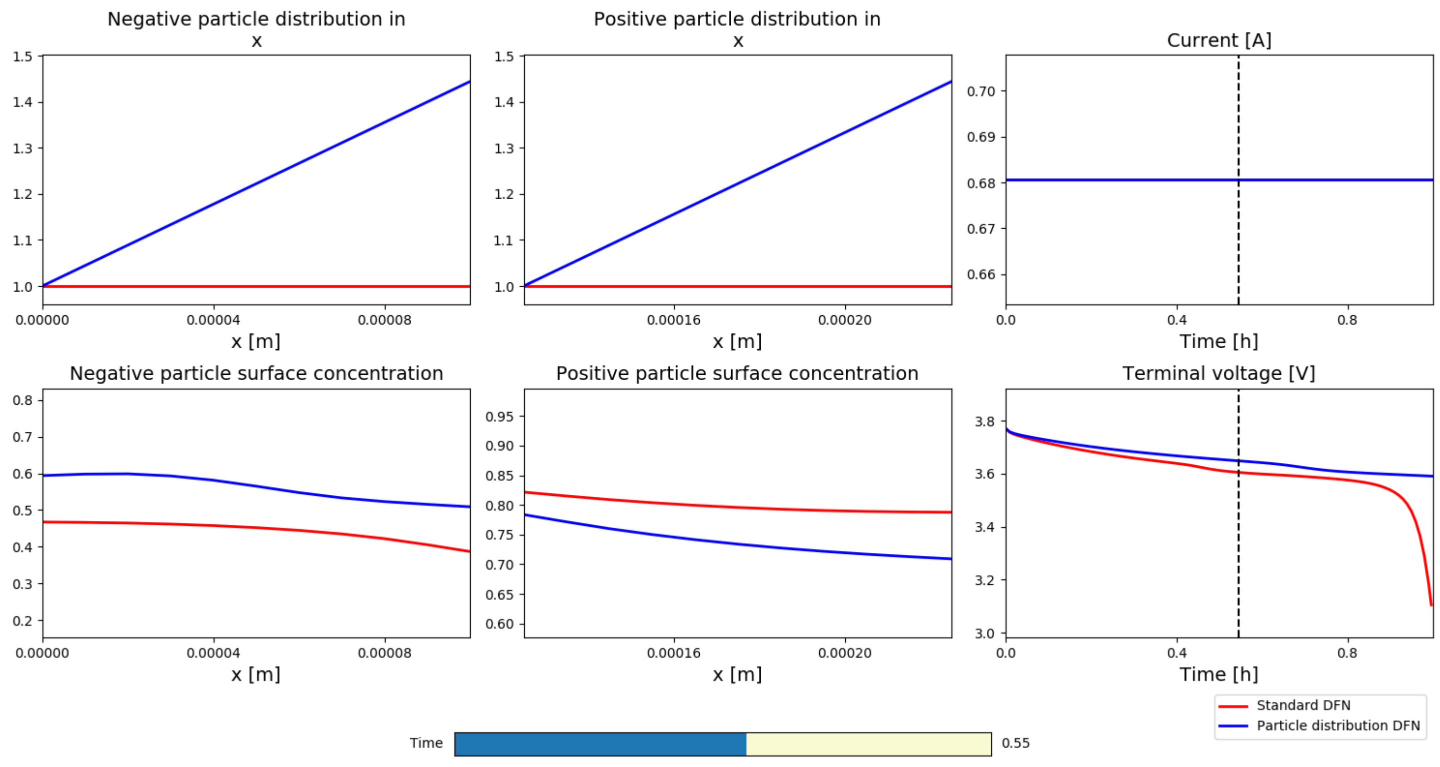
PyBaMM includes popular battery model such as the Doyle-Fuller-Newman model (P2D) and the Single Particle Model



Users can build custom models from submodels

Model structure

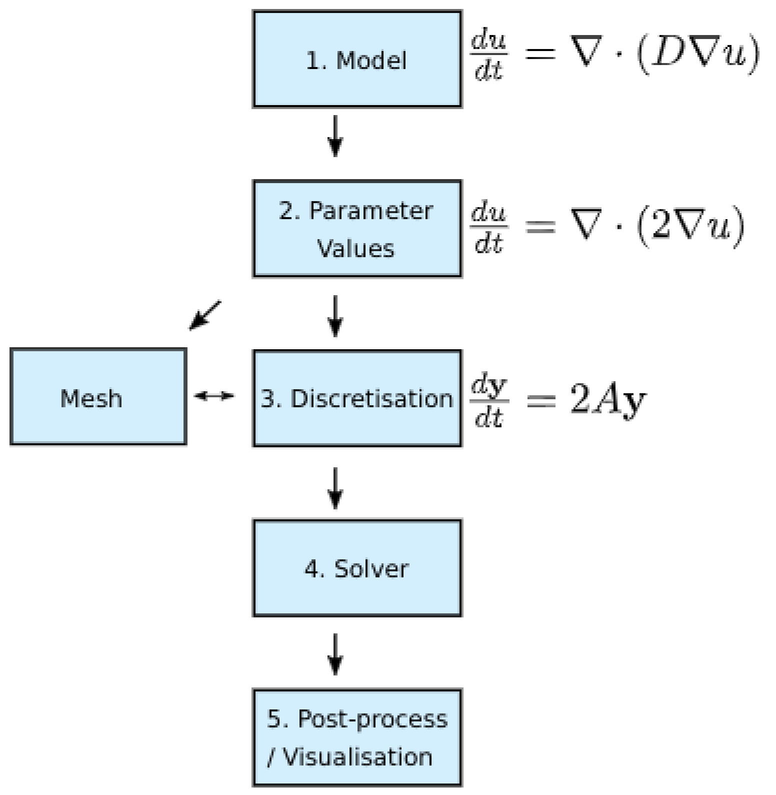
- Structure allows for "plug and play" physics
- New physics can be quickly implemented in an independent fashion
- Different physical mechanisms are readily included in both full and reduced-order models



Using PyBaMM it is easy to compare how different physics affects the model solution. Here we investigate the influence of graded electrodes (particle distribution in the through-cell direction x).

Numerical methods

- Choice of spatial discretisation
  - Finite Volume Method (macroscopic and particles)
  - Finite Element Method in 2D direction (current collectors in pouch cell models)
- Calculation of the analytic Jacobian from expression tree via automatic differentiation (using CasAdi)
- Ability to set up, solve and perform forward and adjoint sensitivity analysis
- Multiple ODE and DAE solvers (e.g. scipy, sundials CVODE and IDA)

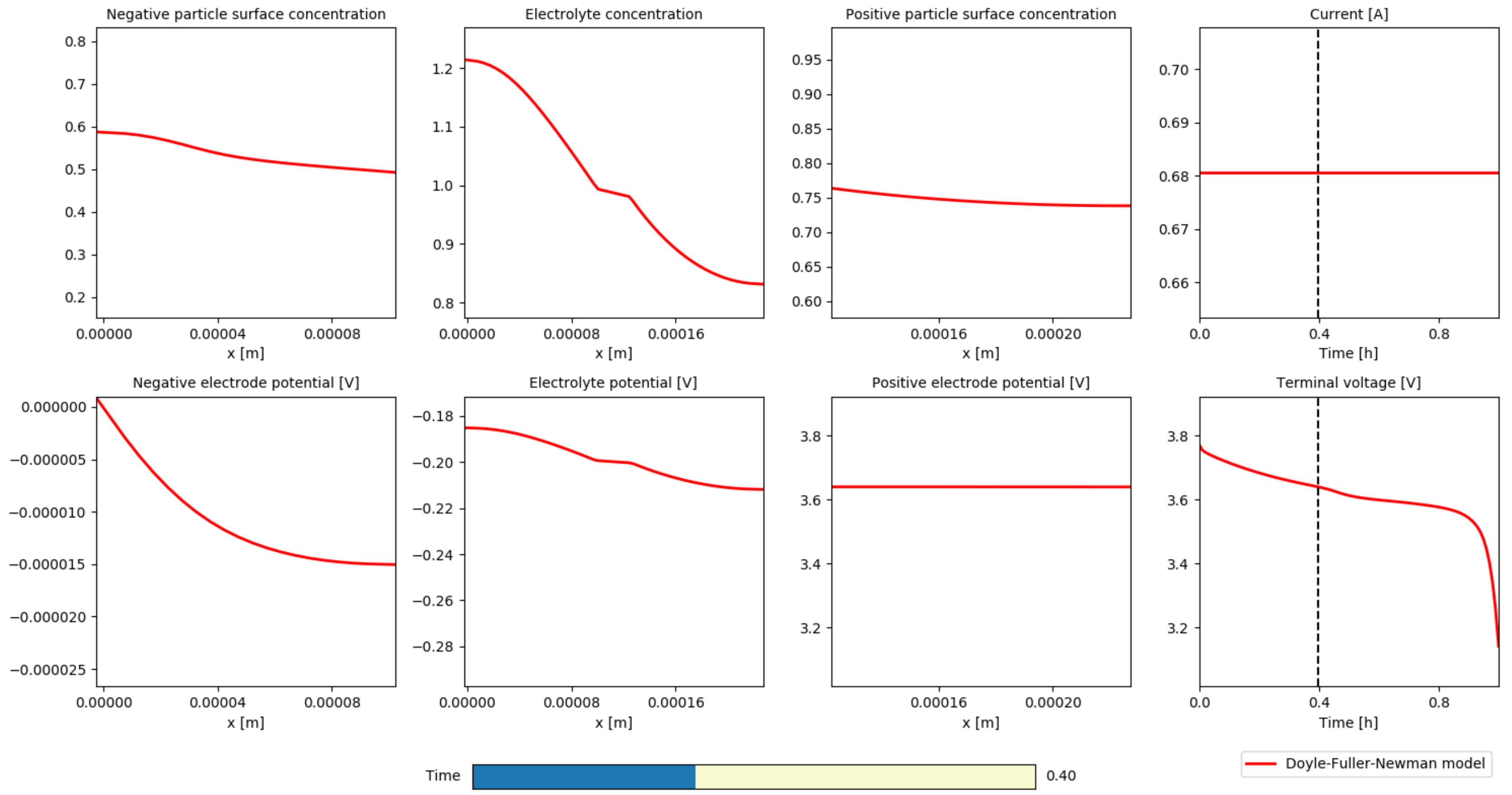

1. Model
$\frac{du}{dt} = \nabla \cdot (D\nabla u)$
2. Parameter Values
$\frac{du}{dt} = \nabla \cdot (2\nabla u)$
Mesh
3. Discretisation
$\frac{d\mathbf{y}}{dt} = 2A\mathbf{y}$
4. Solver
5. Post-process / Visualisation

By employing a pipeline process, PyBaMM separates the statement of model equations, from the inputted parameters, choice of spatial discretization, and choice of time solver.

Simple interface: CC discharge

```
import pybamm
model = pybamm.lithium_ion.DFN()
sim = pybamm.Simulation(model)
sim.solve()
sim.plot()
```

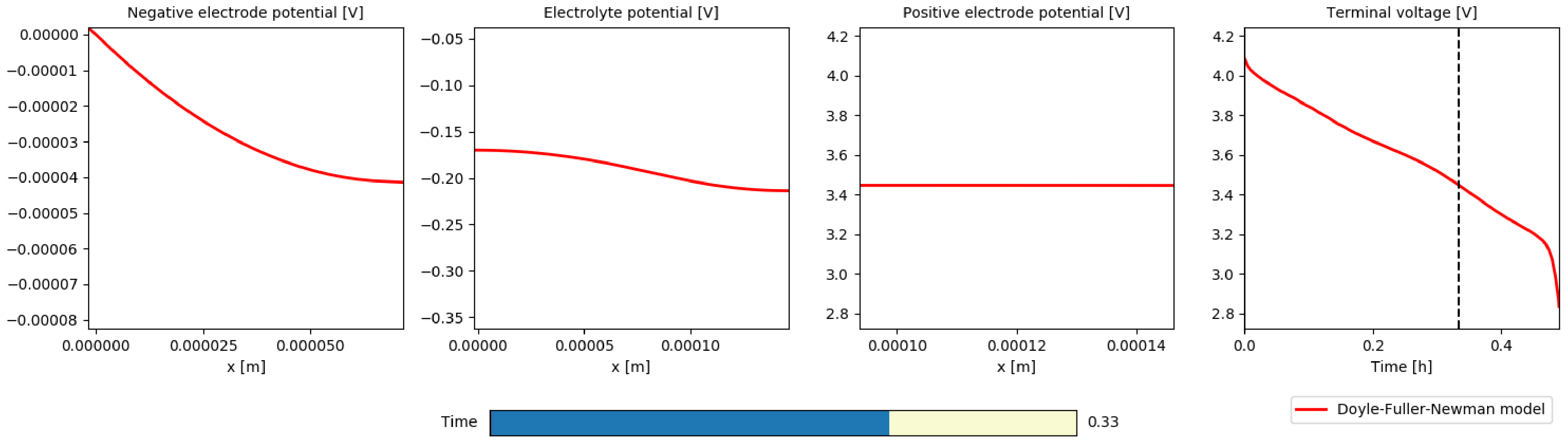
- Easy interface for running simulations
- Model, simulation and plots can be customised



Simple interface: Change parameters

```python
import pybamm
model = pybamm.lithium_ion.DFN()
chemistry = pybamm.parameter_sets.NCA_Kim2011
parameter_values = pybamm.ParameterValues(chemistry=chemistry)
sim = pybamm.Simulation(model, parameter_values=parameter_values, C_rate=2)
sim.solve()
sim.plot()
```

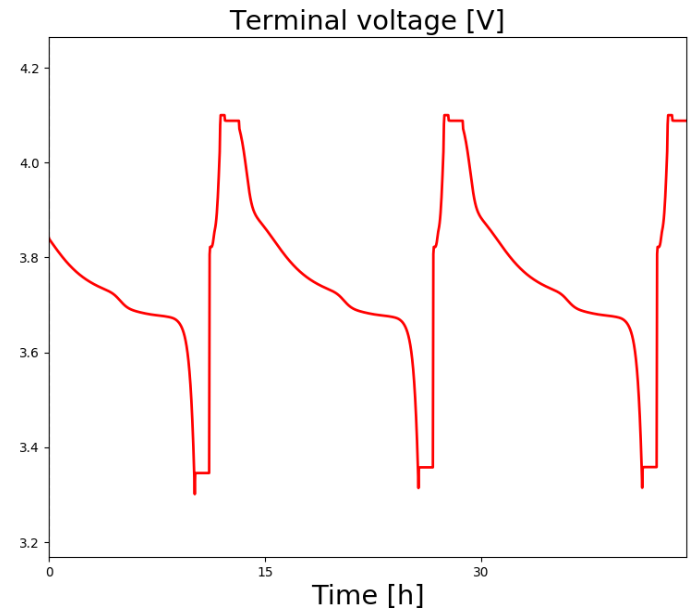
Easy to change parameters and model options



Simple interface: Simulation of experimental protocols

PyBaMM allows users to specify experimental protocols using keyword strings

```
experiment = pybamm.Experiment(
    [
        "Discharge at C/10 for 13 hours or until 3.3 V",
        "Rest for 1 hour",
        "Charge at 1 A until 4.1 V",
        "Hold at 4.1 V until 50 mA",
        "Rest for 1 hour",
    ]
    * 3,
)
model = pybamm.lithium_ion.DFN()
sim = pybamm.Simulation(model, experiment=experiment)
sim.solve()
```



- Can easily define custom protocols
- Specify current, power or voltage
- Standard protocols developed by Emma Kendrick’s group are also implemented but not public yet

Solution times

Doyle-Fuller-Newman Model – 1C CC Discharge

Grid points per domain	States	CasAdi ("fast" mode)	CasADi ("safe" mode)	IDA KLU	Scikits DAE
10	281	0.155s	0.793s	4.860s	2.102s
20	961	0.226s	1.571s	4.776s	10.120s
40	3521	0.566s	5.101s	6.992s	1m 47s
80	13441	1.977s	18.951 s	11.642s	-
160	52481	8.865s	1m 41s	34.289s	-

All simulation performed on a laptop computer (i5, 2.1 GHz, 16 Gb of RAM) using both absolute and relative tolerances of 10^{-6} . The parameters were taken from the papers:

1) M. Ecker, T.K.D. Tran, P. Dechent, S. Kabitz, A. Warnecke, and D.U. Sauer. *Parameterization of a physico-chemical model of a lithium-ion battery. I. Determination of parameters.* Journal of The Electrochemical Society, 162 (2015), pp. A1836-A1848.
2) M. Ecker, S. Kabitz, I. Laresgoiti, and D.U. Sauer. *Parameterization of a physico-chemical model of a lithium-ion battery. II. Model validation.* Journal of The Electrochemical Society, 162 (2015), pp. A1849-A1857.

Case study: Single Particle model with electrolyte (SPMe)

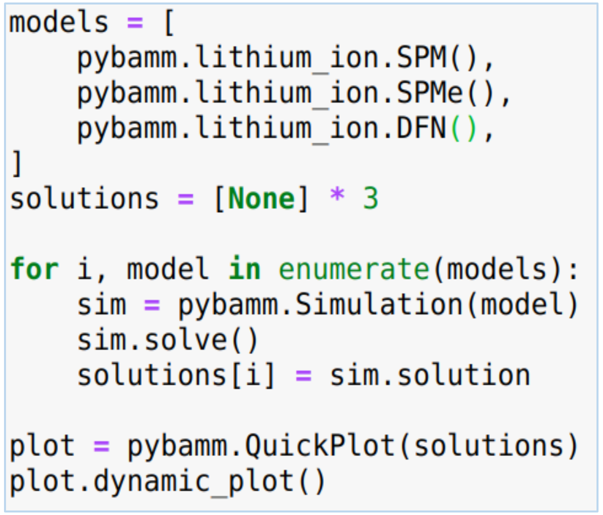
Marquis, Timms (Oxford, Maths); Sulzer (Michigan)

- Systematic derivation of SPMe using asymptotic methods

  Electrolyte diffusion time << discharge time

  Large electrode and electrolyte conductivity

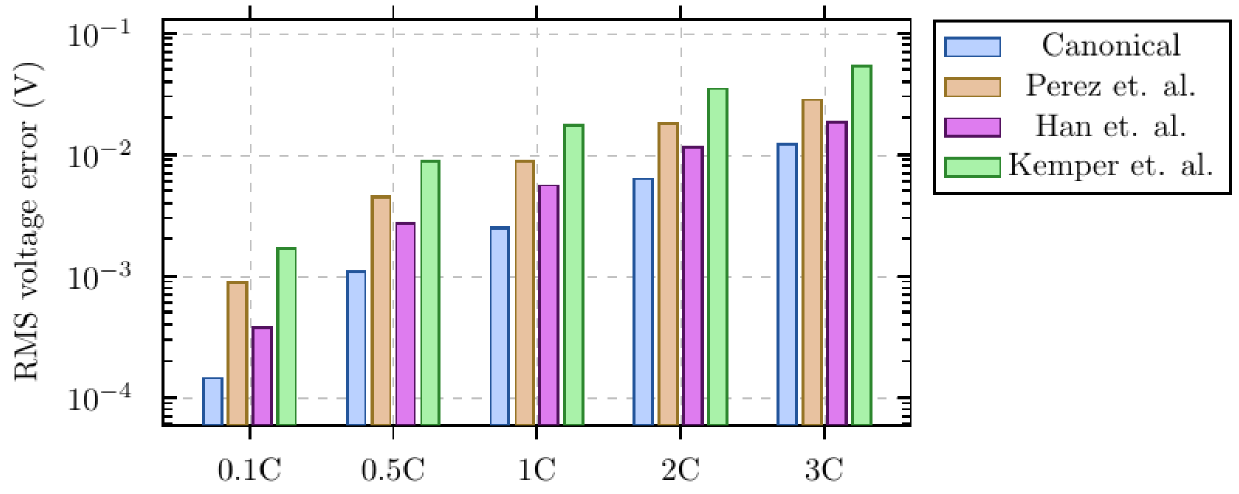
- Outperforms other (ad hoc) models in literature of similar complexity
- Achieves order of magnitude reduction in computation cost at small loss of accuracy
- Extends the validity of SPM to higher C-rates.

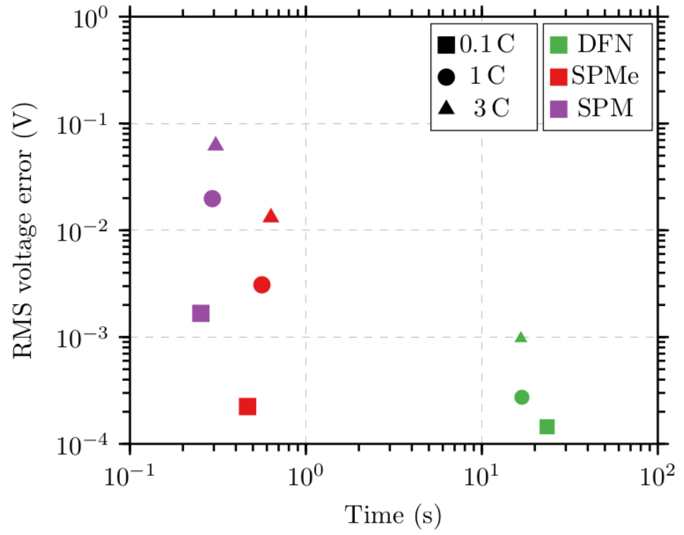


```
models = [
    pybamm.lithium_ion.SPM(),
    pybamm.lithium_ion.SPMe(),
    pybamm.lithium_ion.DFN(),
]
solutions = [None] * 3

for i, model in enumerate(models):
    sim = pybamm.Simulation(model)
    sim.solve()
    solutions[i] = sim.solution

plot = pybamm.QuickPlot(solutions)
plot.dynamic_plot()
```

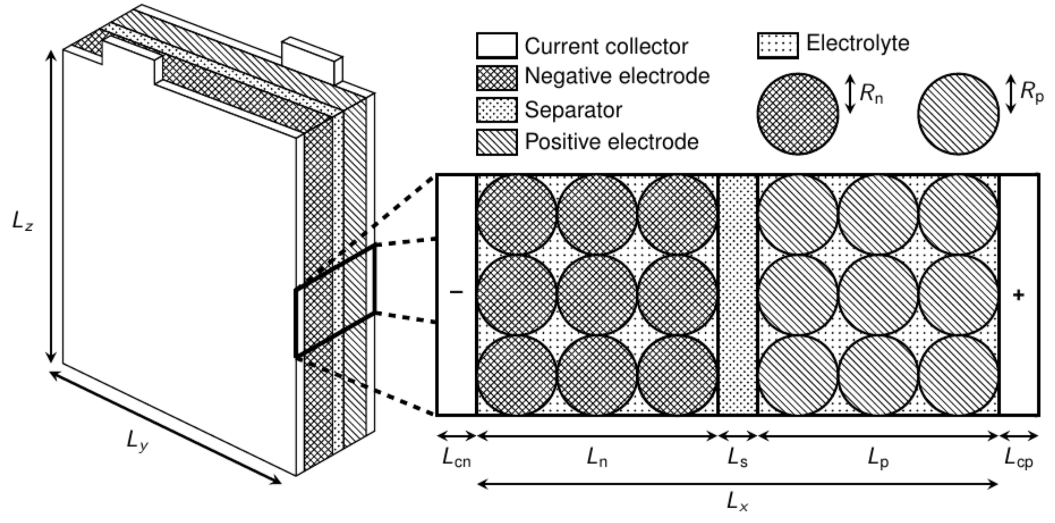




Case study: 2+1D Pouch Cell Model

Timms, Marquis (Oxford, Maths); Sulzer (Michigan)

- Larger batteries exhibit non-uniform behaviour which may adversely affect battery performance and lifetime
- Exploit the geometry to systematically derive 2+1D model
- Combine with other limits to derive a hierarchy of reduced-order models
- Influence of cell geometry can be investigated efficiently

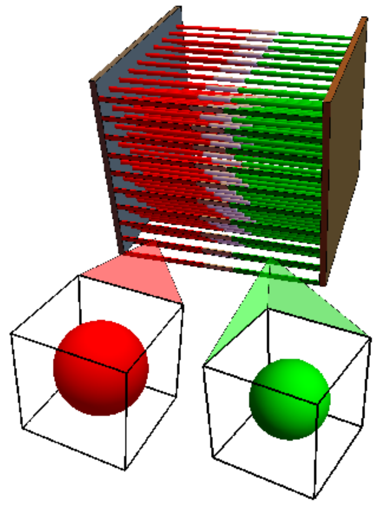


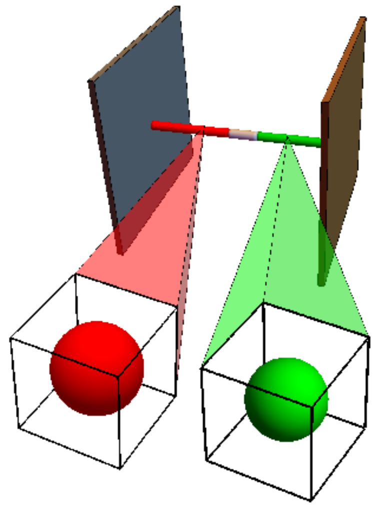
High conductivity

$$\frac{\sigma L_x}{\sigma_c L_c} \sim \frac{L_x^2}{L_z^2} \quad \Rightarrow \quad \text{2+1D model}$$

Very high conductivity

$$\frac{\sigma L_x}{\sigma_c L_c} \sim \frac{L_x^3}{L_z^3} \quad \Rightarrow \quad 2+\bar{1}\text{D model}$$

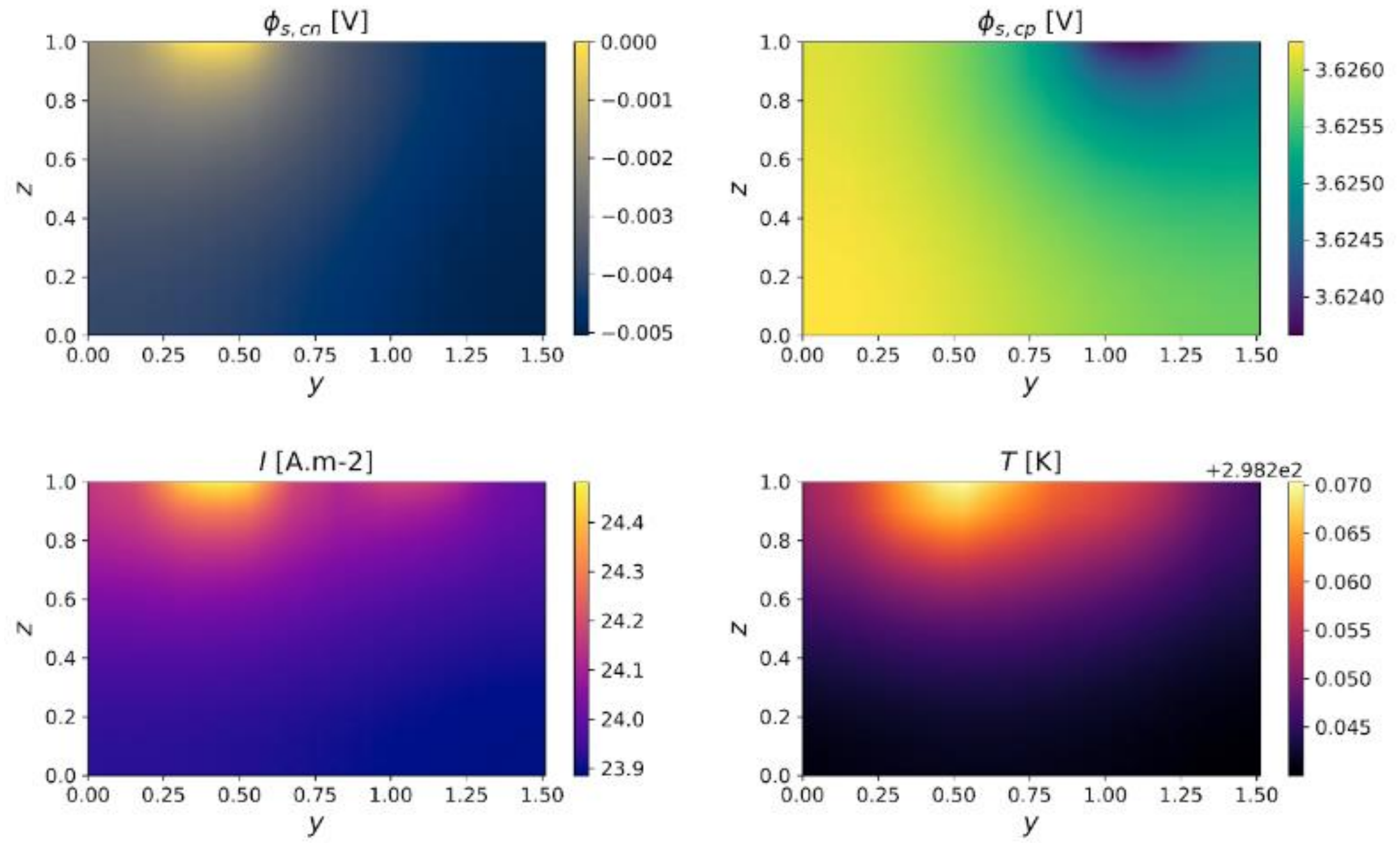




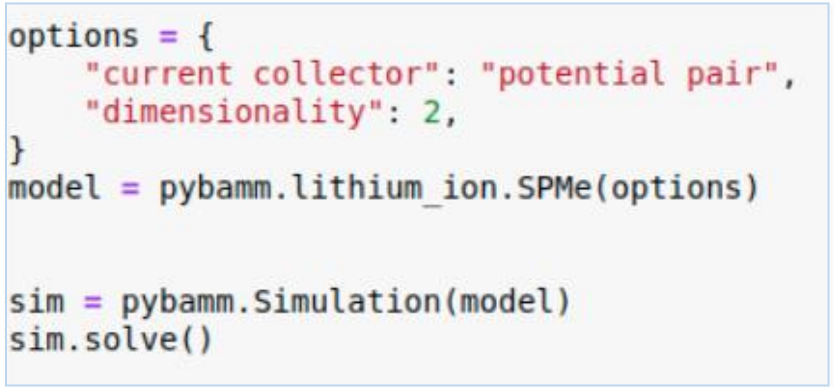
The full 3D model is reduced to a collection of 1D models, coupled via 2D current collectors

Case study: 2+1D Pouch Cell Model

Timms, Marquis (Oxford, Maths); Sulzer (Michigan)



Example result: 100 local Single Particle Models with temperature, solution time approx. 22s for 1C discharge



```
options = {
    "current collector": "potential pair",
    "dimensionality": 2,
}
model = pybamm.lithium_ion.SPMe(options)


sim = pybamm.Simulation(model)
sim.solve()
```

Case study:

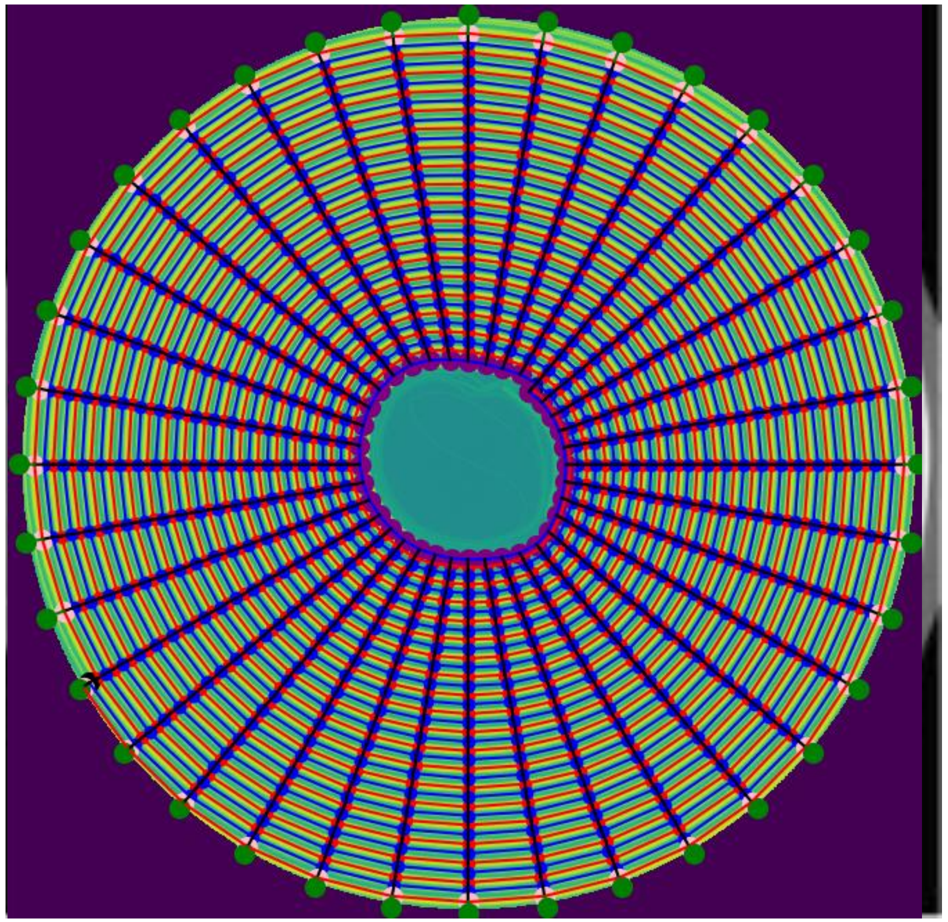
Tranter (UCL); Timms (Oxford, Maths)

A coupled multiscale model of electrochemistry and thermal transport for Li-ion cells

- Internal temperature differences lead to non-uniformities in current flow, state of charge, particle stress and levels of degradation

- Li-ion battery heat generation and transport problem coupling:
    - particle scale electrochemical modelling (*PyBaMM)*
    - cell level electrical and thermal modelling (*OpenPNM)*

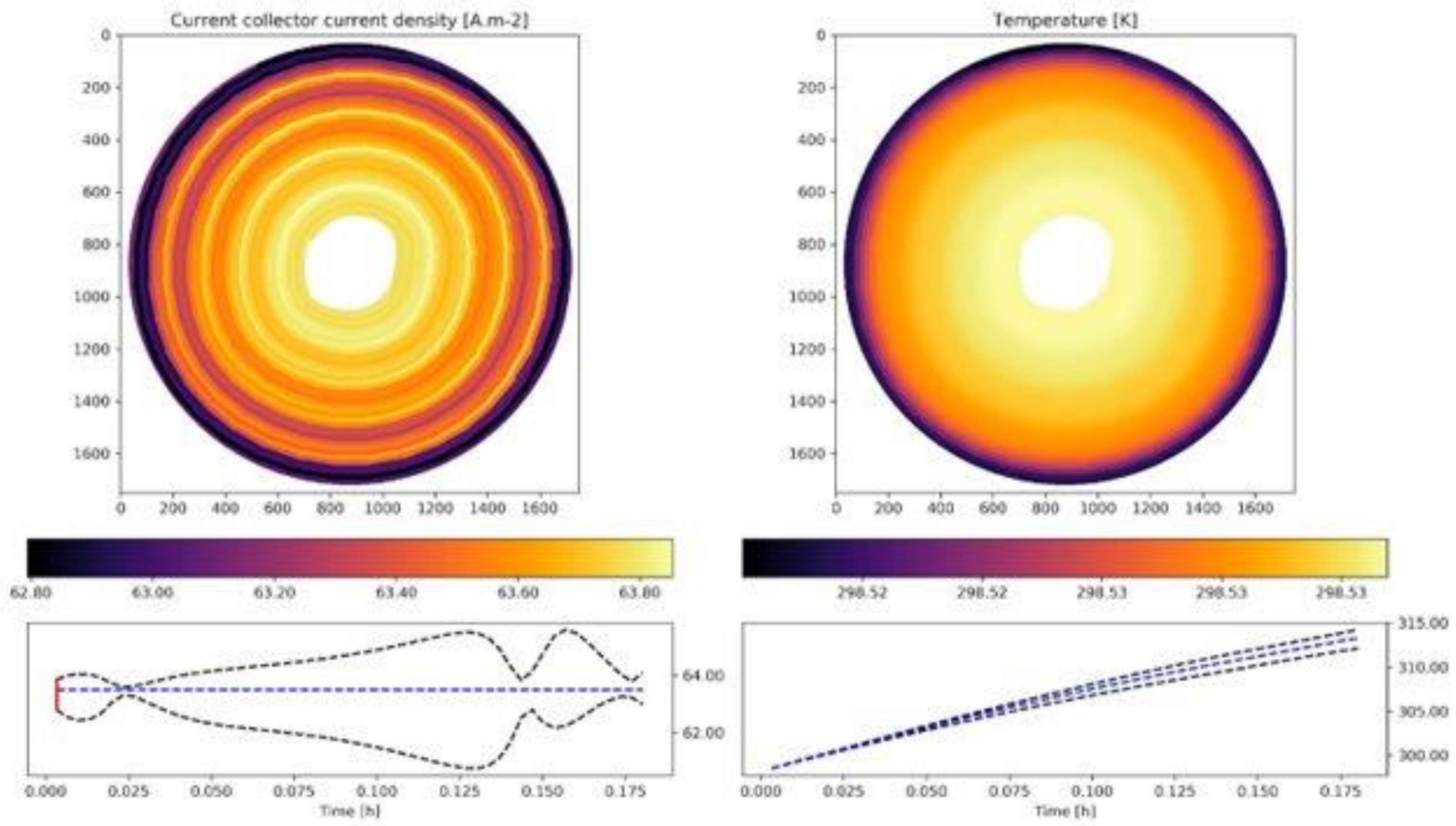
- Example: a 2D heat-transfer problem for spirally rolled cylindrical batteries using tomography data



Case study:

Tranter (UCL); Timms (Oxford, Maths)

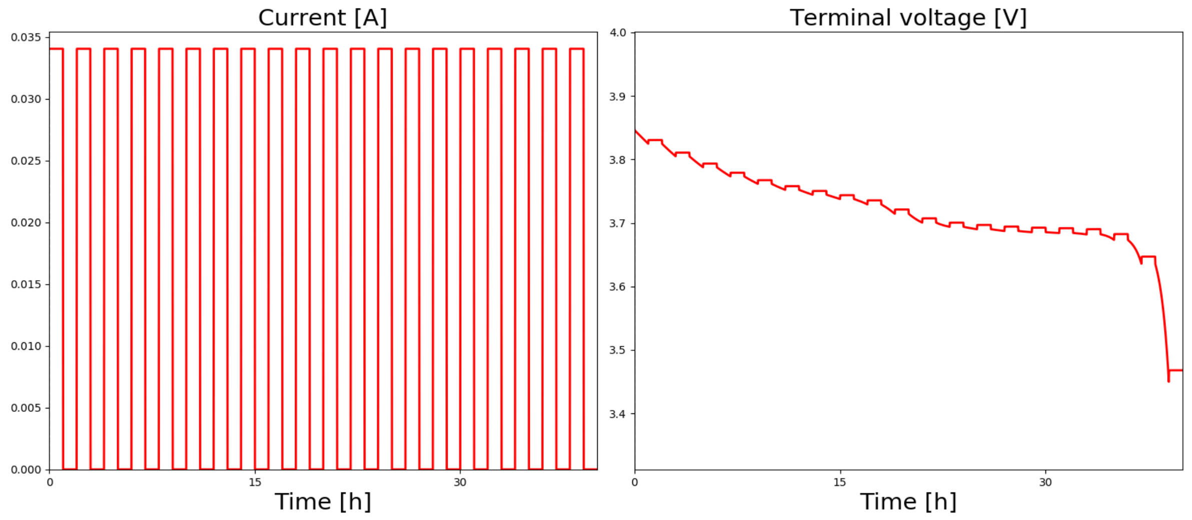
A coupled multiscale model of electrochemistry and thermal transport for Li-ion cells

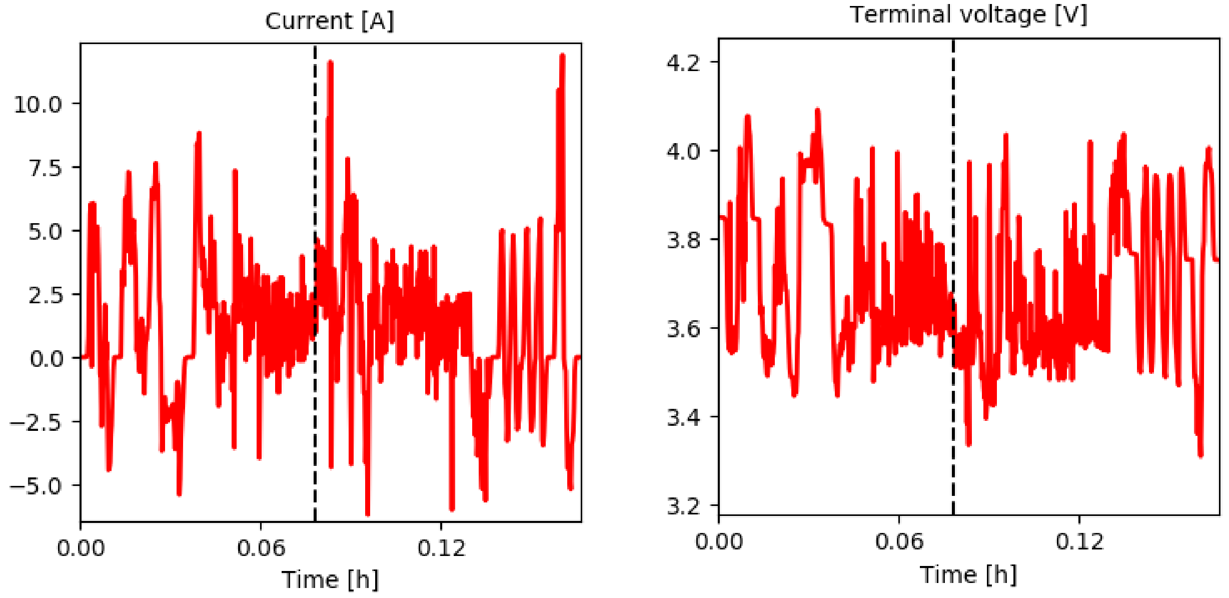


HTC = 5 [W.m-2], σ = 3e7 [S.m-1], #tabs = 5, Current = 5 [A]

Case study: Simulation of experimental protocols

```
experiment = pybamm.Experiment(
    [
        "Discharge at C/20 for 1 hour",
        "Rest for 1 hour"
    ] * 20,
)
model = pybamm.lithium_ion.DFN()
sim = pybamm.Simulation(model, experiment=experiment)
sim.solve()
sim.plot()
```

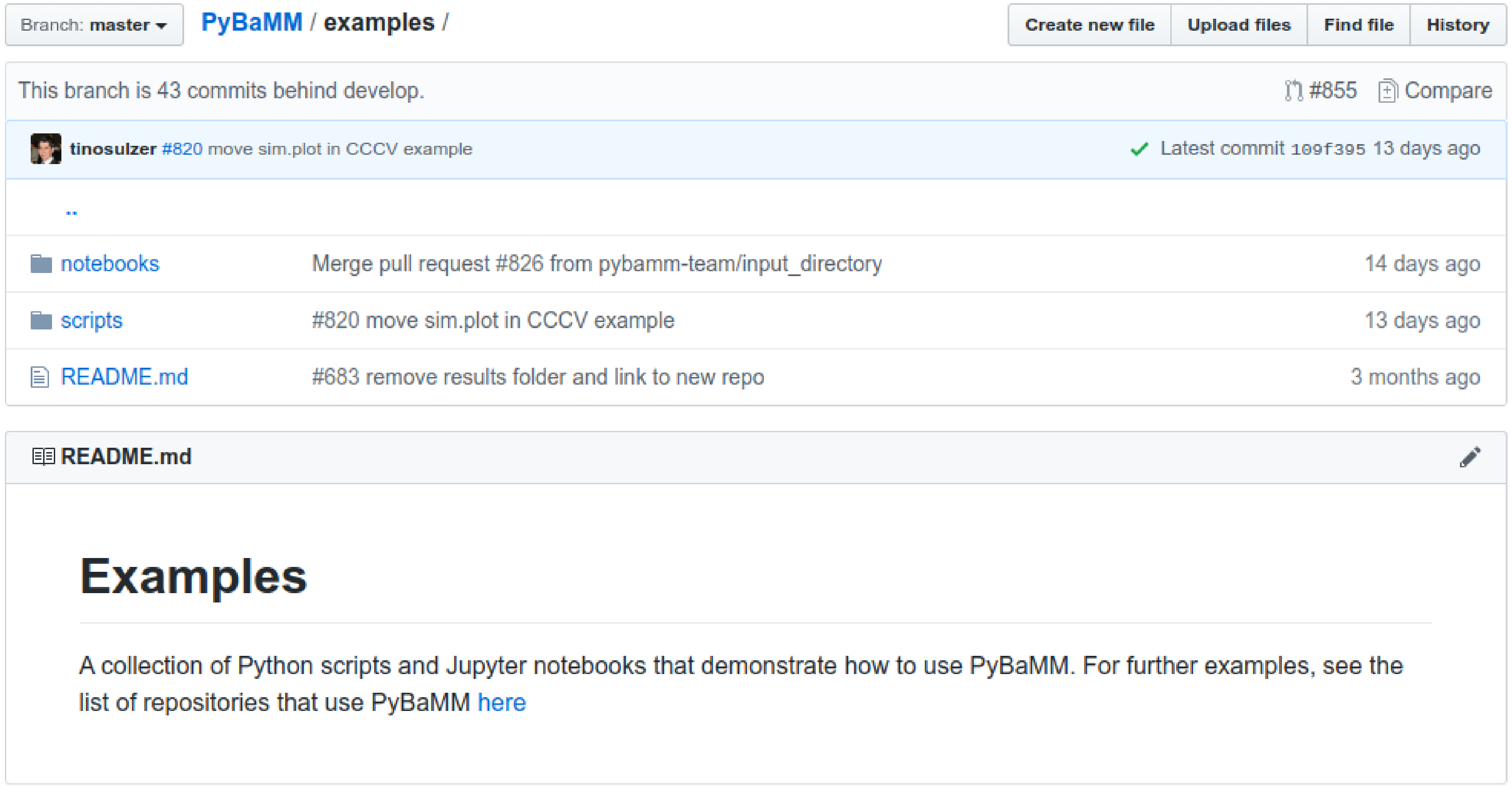




```
model = pybamm.lithium_ion.DFN()
param = model.default_parameter_values
param["Current function [A]"] = "[current data]US06"
sim = pybamm.Simulation(model, parameter_values=param)
sim.solve()
sim.plot()
```

More examples online

https://github.com/pybamm-team/PyBaMM/tree/master/examples



PyBaMM / examples /

This branch is 43 commits behind develop.

tinosulzer #820 move sim.plot in CCCV example — Latest commit 109f395 13 days ago

..

notebooks	Merge pull request #826 from pybamm-team/input_directory	14 days ago
scripts	#820 move sim.plot in CCCV example	13 days ago
README.md	#683 remove results folder and link to new repo	3 months ago

README.md

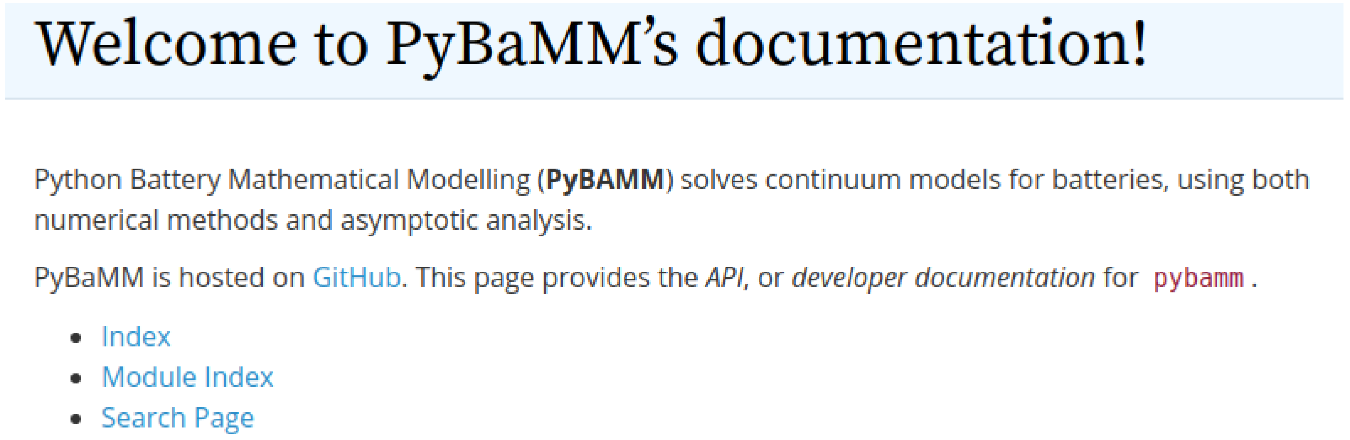
Examples

A collection of Python scripts and Jupyter notebooks that demonstrate how to use PyBaMM. For further examples, see the list of repositories that use PyBaMM here

And documentation

https://pybamm.readthedocs.io/en/latest/

API docs:



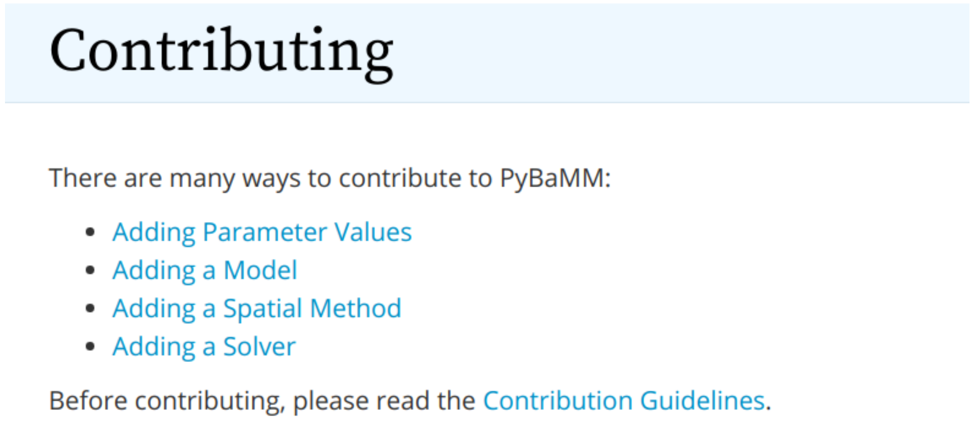
Welcome to PyBaMM's documentation!

Python Battery Mathematical Modelling (**PyBAMM**) solves continuum models for batteries, using both numerical methods and asymptotic analysis.

PyBaMM is hosted on GitHub. This page provides the *API*, or *developer documentation* for `pybamm` .

- Index
- Module Index
- Search Page

Tutorials:



Contributing

There are many ways to contribute to PyBaMM:

- Adding Parameter Values
- Adding a Model
- Adding a Spatial Method
- Adding a Solver

Before contributing, please read the Contribution Guidelines.