Python Battery Mathematical Modelling Training Workshop

Oxford University 31st July - 1st August 2019

Exercise 1 – solving ODEs in PyBaMM

Using the examples available in the PyBaMM repository, write a script which solves the following system of ODEs:

$$\begin{aligned}\frac{\mathrm{d}x}{\mathrm{d}t} &= 2x, \quad x(0) = 1,\\ \frac{\mathrm{d}y}{\mathrm{d}t} &= -x, \quad y(0) = -0.5.\end{aligned}$$

Listing 1: Solving ODEs in PyBaMM.

```python
import pybamm
import numpy as np
import matplotlib.pyplot as plt

# 1. Initialise an empty model
model = pybamm.BaseModel()

# 2. Define variables
x = pybamm.Variable("x")
y = pybamm.Variable("y")

# 3. State governing equations
dxdt = 2 * x
dydt = -x

model.rhs = {x: dxdt, y: dydt}  # add equations to rhs dictionary

# 4. State initial conditions
model.initial_conditions = {x: pybamm.Scalar(1), y: pybamm.Scalar(-0.5)}

# 6. State output variables
model.variables = {"x": x, "y": y}

"Using the model"

# use default discretisation
disc = pybamm.Discretisation()
disc.process_model(model)
```

```python

# solve
solver = pybamm.ScipySolver()
t = np.linspace(0, 1, 20)
solution = solver.solve(model, t)

# post-process, so that the solutions can be called at any time t (using interpolation)
t_sol, y_sol = solution.t, solution.y
x = pybamm.ProcessedVariable(model.variables["x"], t_sol, y_sol)
y = pybamm.ProcessedVariable(model.variables["y"], t_sol, y_sol)

# plot
t_fine = np.linspace(0, t[-1], 1000)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(13, 4))
ax1.plot(t_fine, np.exp(2 * t_fine), t_sol, x(t_sol), "o")
ax1.set_xlabel("t")
ax1.legend(["exp(2*t)", "x"], loc="best")

ax2.plot(t_fine, -0.5 * np.exp(2 * t_fine), t_sol, y(t_sol), "o")
ax2.set_xlabel("t")
ax2.legend(["0.5*exp(2*t)", "y"], loc="best")

plt.tight_layout()
plt.show()
```