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:

$$\frac{dx}{dt} = 4x - 2y, \quad x(0) = 1,$$
$$\frac{dy}{dt} = 3x - y, \quad y(0) = 2.$$

You can try to write the script from scratch, or copy the code below and fill in the blanks.

Listing 1: Solving ODEs script

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

"Setting up the model"

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

# 2. Define variables
## DEFINE YOUR VARIABLES HERE ##

# 3. State governing equations
## WRITE THE EQUATIONS HERE ##

model.rhs = {}

# 4. State initial conditions
## ADD INITIAL CONDITIONS HERE ##

model.initial_conditions = {}

# 6. State output variables
## STATE OUTPUT VARIABLES HERE ##

"Using the model"

# use default discretisation
```

```
disc = pybamm.Discretisation()
## PROCESS MODEL USING THE GIVEN DISCRETISTAION ##

# solve
solver = pybamm.ScipySolver()
## SOLVE MODEL USING THE GIVEN SOLVER ##

# post-process, so that the solutions can be called at any time t (using ↩
    interpolation)
## PROCESS THE SOLUTION FOR PLOTTING ##

# plot
## PLOT SOLUTION ##
```

Exercise 2 – solving PDEs in PyBaMM

Write a script to solve the problem of linear diffusion on a unit sphere,

$$\frac{\partial c}{\partial t} = \nabla \cdot (\nabla c),$$

with the following boundary and initial conditions:

$$\left.\frac{\partial c}{\partial r}\right|_{r=0} = 0, \quad \left.\frac{\partial c}{\partial r}\right|_{r=1} = 2, \quad c|_{t=0} = 1.$$

Try solving the model again with different boundary or initial conditions.

Exercise 3 – extending the PDE model

In PyBaMM, paramater objects can be used to define paramaters whose value is set during processing of the model. In practice, paramater values can be read in from and external source, such as a .csv file, but they can also be set in a dictionary before model processing. Try to extend your model to include a diffusion coefficient D, i.e. solve

$$\frac{\partial c}{\partial t} = \nabla \cdot (D \nabla c),$$

Listing 2: Adding a parameter and setting its value

```
D = pybamm.Parameter("Diffusion coefficient")
param = pybamm.ParameterValues({"Diffusion coefficient": 0.5})
```

Try adding more paramters to your model, or changing the parameter values.

You can also add additional output variables to your model which can be accessed after the

solve. For instance, you may interested in the flux as well as the concentration. Extra output variables are easily added to the model.variables dictionary in PyBaMM.

Listing 3: Adding extra output variables

```
model.variables = {"Concentration": c, "Flux": N}
```

Exercise 4 – the negative partice problem

Now it is time to solve a real-life battery problem! Adapt your linear diffusion model to solve the problem of diffusion in the negetive electrode particle within the single paticle model. That is,

$$\frac{\partial c}{\partial t} = \nabla \cdot (D \nabla c),$$

with the following boundary and initial conditions:

$$\left.\frac{\partial c}{\partial r}\right|_{r=0} = 0, \quad \left.\frac{\partial c}{\partial r}\right|_{r=R} = -\frac{j}{FD}, \quad c|_{t=0} = c_0,$$

where c is the concentration, r the radial coordinate, t time, R the particle radius, D the diffusion coefficient, j the interfacial current density, F Faraday's constant, and c_0 the initial concentration. Use the paramaters from Table 1.

Symbol	Units	Value
R	m	10×10^{-6}
D	$\mathrm{m^2\,s^{-1}}$	3.9×10^{-14}
j	$\mathrm{A\,m^{-2}}$	1.4
F	$\mathrm{C\,mol^{-1}}$	96485
c_0	$\mathrm{mol\,m^{-3}}$	2.5×10^4

Table 1: Parameter values for use in Exercise 4.

Exercise 5 – making a model class