PyBaMM Implementation of Marinescu et al. (2018) Zero-Dimensional Model

Dr. Michael Cornish

May 6, 2020

Abstract

The zero-dimensional differential algebraic model of Marinescu et al. (2018) for Li-S batteries is implemented in PyBaMM. Various idiosyncrasies with the implementation are discussed.

1 Marinescu et al. (2018)

1.1 The Model

The DAE system is as follows,

$$\frac{dS_8^0}{dt} = -\frac{n_{s_8}M_{s_8}}{n_eF}i_H - k_sS_8^0, \tag{1}$$

$$\frac{dS_4^{2-}}{dt} = \frac{n_{s_8}M_{s_8}}{n_eF}i_H + \left(1 - \frac{f_s}{m_s}S_s\right)k_sS_8^0 - \frac{n_{s_4}M_{s_8}}{n_eF}i_L, \tag{2}$$

$$\frac{dS_2^{2-}}{dt} = \frac{n_{s_2}M_{s_8}}{n_eF}i_L, \tag{3}$$

$$\frac{dS^{2-}}{dt} = \frac{2n_sM_{s_8}}{n_eF}i_L - \frac{k_p}{v\rho_s}S_p(S^{2-} - S_\star^{2-}), \tag{4}$$

$$\frac{dS_p}{dt} = \frac{k_p}{v\rho_s}S_p(S^{2-} - S_\star^{2-}), \tag{5}$$

$$\frac{dS_s}{dt} = k_sS_8 \tag{6}$$

along with the algebraic system,

$$I = i_H + i_L. \tag{7}$$

The current functions are,

$$i_H = -2i_{H,0}a_r\sinh\left(\frac{n_eF\eta_H}{2RT}\right), \tag{8}$$

$$i_L = -2i_{L,0}a_r\sinh\left(\frac{n_eF\eta_L}{2RT}\right), \tag{9}$$

with surface overpotentials,

$$\eta_H = V - E_H, \tag{10}$$

$$\eta_L = V - E_L, \tag{11}$$

and potentials given via the Nernst equation,

$$E_H = E_H^0 + \frac{RT}{4F} \ln\left(f_H \frac{S_8^0}{(S_4^{2-})^2}\right), \tag{12}$$

$$E_L = E_L^0 + \frac{RT}{4F} \ln\left(f_L \frac{S_4^{2-}}{(S^{2-})^2 S_2^{2-}}\right). \tag{13}$$

It is important to note that $k_s = 0$ on discharge but $k_s = 0.0002$ on charge.

1.2 Conservation Equations

The above system does not adhere to mass conservation. We should expect that the mass conservation constraint,

$$m_s = \sum_i S_i, \tag{14}$$

should translate to the dynamical system by differentiating both sides by time to get,

$$0 = \sum_i \frac{dS_i}{dt}. \tag{15}$$

However, this is not the case. The right hand side equals $\left(1 - \frac{f_s}{m_s} S_s\right) k_s S_8^0$, which during discharge does equal zero since to $k_s = 0$. Indeed, there is an extra differential equation which we have hitherto not mentioned,

$$\frac{dS_l}{dt} = \frac{f_s}{m_s} S_s k_s S_8^0. \tag{16}$$

This equation was not mentioned because it does not dynamically effect the system. However, introducing this equation does not alleviate lack of mass conservation during charge, namely,

$$\sum_i \frac{dS_i}{dt} = k_s S_8^0. \tag{17}$$

What about energy conservation?

2 General Numerical Method

PyBaMM requires initiating the model in some particular ways. We need to clearly define the variables and ensure any functions of the variables are consistent with the PyBaMM library of functions. As such, for later testing we are required to implement all of the functions with the Numpy counterparts to handle numerical data. The specific implementation of all variables, parameters, functions, and solver can be found in the Jupyter notebook. However, it should be noted that in the current iteration of the PyBaMM model, the current is taken as a constant. Therefore, any change in the current requires re-initializing the model. This is discuss in the next section in more detail. PyBaMM DAE solvers are, for all intents and purposes, black-box solvers. However, we do know that the solver using an implicit method. Therefore, an education guess would suggest that the numerical method produces a nonlinear system of equations at each time instant which some root-finding algorithm must solve. Such methods do not guarantee unique solutions and as such it may be a good idea to attempt to find alternative solutions within some small region of the previous time step. For example, search for solutions in a ball of radius defined by the current solution of the PyBaMM solver. If a solution appears, then this brings to question the method. No solution helps validate the model's implementation.

3 Initial Conditions and the Ramp-up

Due to the nonlinear solver, the computer requires a good initial guess which is consistent with the algebraic condition. The algebraic condition and mass conservation[1] gives two constraints for the seven variables. Zero current further constrains the voltage to both E_H an E_L, which subsequently equal one another, but this is really a further specification of the algebraic condition. Hence, we have a total of three constraints for seven variables. Therefore, S_8, S_4, S, and S_s are pre-specified while the remaining three variables are solved given the constraints.

The initial condition for all implementations are therefore found under the zero-current condition. To allow for alternative currents, a ramp-up period is implemented whereby the current is quickly changed from zero to the desired initial current. The PyBaMM solver finds the consistent species concentrations at each time step. This process is completed in as little time as possible to keep the initial conditions close to the zero-current conditions.

4 Validation of Implementation

4.1 Validation Metrics

We validate all simulations with three metrics. All three metrics require the L_2 norm[2], or some minor deviation thereof. First, we check that the algebraic condition is satisfied. This is performed simply by evaluating the algebraic condition at each time step with the data produced by the method. The L_2 norm is applied to this data. Second, we evaluate the backward Euler estimate for the derivative of each species,

$$\left.\frac{dS_i}{dt}\right|_{t=j\Delta t} \approx \frac{S_i(j\Delta t) - S_i((j-1)\Delta t)}{\Delta t}, \tag{18}$$

and compare with the value of the functions on the right hand side of the dynamical system, evaluated at $t = j\Delta t$. The L_2 norm of the difference between these values for each time step is then implemented. The norm is not expected to be equal to zero, and in particular greater than the other norms, for two reasons. First, the numerical derivative is inexact and should not equal the derivative function. Moreover, the derivative functions will had some small error due to numerical evaluations and floating point arithmetic. Finally, we take the maximum L_2 norm across all species.

Finally, we consider the pseudo mass conservation condition. The sum of the right hand side functions of the ODE system will add to a value which should be zero if mass was conserved. This system does not have mass conservation, but we can sum the functions nonetheless and compare the values with the expected values. In particular,

$$\sum_i f_i(\boldsymbol{S}; \boldsymbol{p}) - \left(1 - \frac{f_s}{m_s} S_s\right) k_s S_8^0 = 0. \tag{19}$$

We evaluate the left hand side of the above equation at each time step. The L_2 norm is then taken and compared with the expected value of zero.

[1] Mass conservation is not held by this system and requires further discussion in subsection **??**

[2] This is simply the sum of squares for some data, divided by the quantity of data.

4.2 Zero Current

Zero current is an equilibrium condition for the system. Therefore we should expect all variables are constant and equal to their initial conditions for this simulation.

4.3 Comparison with Matlab Implementation

We may compare the Matlab output by the L_2 norm metric, which is simply sum of the squares of the difference between each data point at each time instant,

$$M = \sum_i \sum_j (S_{ij}^p - S_{ij}^m)^2, \tag{20}$$

where S_{ij}^p is the PyBaMM data of species i (including voltage) at time step j, and similarly for the Matlab data S_{ij}^m.

4.3.1 Ramp-up

4.3.2 Discharge

4.3.3 Charge

5 Initial state Solver

Given a voltage time series, we should be able to retrieve the initial state. This methodology is further discussed in the document "initial State Determination" .