Battery Modeling Software Documentation

Loan Kim Pham, Scott Moura
Energy, Controls, and Applications Lab (eCAL)
University of California, Berkeley

1 Introduction

This open-source battery modeling code simulates the electrochemical dynamics of lithium-ion batteries relevant for control [1–3], optimization [4,5], and analysis studies [6]. To run the model, all input and parameter files must be present and able to be opened within MATLAB. The main file `dfn.m` first calls upon `params.m`, which contains the parameters of the model including geometric, transport, and kinetic parameters. A representation of the parameters used in the model can be referenced in the code tree, under the file name `dfn_tree.jpg`.

1.1 Inputs and Model Equations

The model input data, namely current-time trajectories, can be input in two forms:

- Manual input
- Arrays of time and current in the form of `.mat` files

Within the code, current is designated in terms of A/m^2, since the equations model a cross section of the lithium-ion battery cell sandwich. Sample data can be found under the data folder of the software, which contains files such as `UDDSx2_batt_ObsData.mat` and `US06x3_batt_ObsData.mat`. These two input files provide current charge/discharge cycles for a battery in a plug-in hybrid electric vehicle undergoing an emissions certification driving schedules [1]. The user also has the option of inputting manual data for time and current. This is often used for conveniently testing simple protocols, such as constant or pulsed current. Within this model, a positive current corresponds to charge while a negative current corresponds to discharge. Information about manual data can be found in Section 3.

The model equations form a system of PDEs, ODEs in space, ODEs in time, and corresponding boundary/initial conditions [7]. More detail on the model equations and their implementation can be found in Section 2.

1.2 Outputs

A list of the available model outputs are detailed in Table 1. Outputs can be viewed graphically by running `plot_dfn.m` after `dfn.m` has been run. For example, when a constant positive current is used, then `plot_dfn.m` will plot SOC decreasing over time, meaning the battery is in a state of discharge. Examples with sample outputs can be found within Section 3 of this documentation.

Table 1: DFN Model Outputs (non-exclusive)

Description	Math Variable	Matlab Variable	Units
Voltage	$V(t)$	`Volt`	[V]
State of charge	$SOC(t)$	`SOC`	[-]
Solid surface concentration	$c_{ss}^{\pm}(x,t)$	`c_ss_n, c_ss_p`	[mol/cm^3]
Vol. averaged concentration	$\overline{c}_s^{\pm}(x,t)$	`c_avg_n, c_avg_p`	[mol/cm^3]
Electrolyte concentration	$c_e(x,t)$	`c_ex`	[mol/cm^3]
Solid potential	$\phi_s^{\pm}(x,t)$	`phi_s_n, phi_s_p`	[V]
Electrolyte potential	$\phi_e(x,t)$	`phi_e`	[V]
Ionic current	$i_e^{\pm}(x,t)$	`i_en, i_ep`	$[A/cm^3]$
Volumetric current density	$j_n^{\pm}(x,t)$	`jn, jp`	$[A/cm^3]$
Overpotential	$\eta^{\pm}(x,t)$	`eta_n, eta_p`	$[V]$
Temperature	$T(t)$	`T`	$[K]$

2 Modeling

2.1 Model Theory

We consider the Doyle-Fuller-Newman (DFN) model in to predict the evolution of lithium concentration in the solid $c_s^{\pm}(x,r,t)$, lithium concentration in the electrolyte $c_e(x,t)$, solid electric potential $\phi_s^{\pm}(x,t)$, electrolyte electric potential $\phi_e(x,t)$, ionic current $i_e^{\pm}(x,t)$, molar ion fluxes $j_n^{\pm}(x,t)$, and bulk cell temperature $T(t)$ [7]. The governing equations are given by

$$\frac{\partial c_s^{\pm}}{\partial t}(x,r,t) = \frac{1}{r^2}\frac{\partial}{\partial r}\left[D_s^{\pm} r^2 \frac{\partial c_s^{\pm}}{\partial r}(x,r,t)\right], \tag{1}$$

$$\varepsilon_e \frac{\partial c_e}{\partial t}(x,t) = \frac{\partial}{\partial x}\left[\varepsilon_e D_e \frac{\partial c_e}{\partial x}(x,t) + \frac{1-t_c^0}{F} i_e^{\pm}(x,t)\right], \tag{2}$$

$$\frac{\partial \phi_s^{\pm}}{\partial x}(x,t) = \frac{i_e^{\pm}(x,t) - I(t)}{\sigma^{\pm}}, \tag{3}$$

$$\frac{\partial \phi_e}{\partial x}(x,t) = -\frac{i_e^{\pm}(x,t)}{\kappa} + \frac{2RT}{F}(1-t_c^0) \times \left(1 + \frac{d\ln f_{c/a}}{d\ln c_e}(x,t)\right)\frac{\partial \ln c_e}{\partial x}(x,t), \tag{4}$$

$$\frac{\partial i_e^{\pm}}{\partial x}(x,t) = a_s F j_n^{\pm}(x,t), \tag{5}$$

$$j_n^{\pm}(x,t) = \frac{1}{F} i_0^{\pm}(x,t) \left[e^{\frac{\alpha_a F}{RT}\eta^{\pm}(x,t)} - e^{-\frac{\alpha_c F}{RT}\eta^{\pm}(x,t)} \right], \tag{6}$$

$$\begin{aligned} \rho^{\text{avg}} c_P \frac{dT}{dt}(t) &= h_{\text{cell}} \left[T_{\text{amb}}(t) - T(t) \right] + I(t)V(t) \\ &\quad - \int_{0^-}^{0^+} a_s F j_n(x,t) \Delta T(x,t) dx, \end{aligned} \tag{7}$$

where $D_e, \kappa, f_{c/a}$ are functions of $c_e(x,t)$ and

$$i_0^{\pm}(x,t) = k^{\pm} \left[c_{ss}^{\pm}(x,t) \right]^{\alpha_c} \left[c_e(x,t) \left(c_{s,\max}^{\pm} - c_{ss}^{\pm}(x,t) \right) \right]^{\alpha_a}, \tag{8}$$

$$\begin{aligned} \eta^{\pm}(x,t) &= \phi_s^{\pm}(x,t) - \phi_e(x,t) \\ &\quad - U^{\pm}(c_{ss}^{\pm}(x,t)) - F R_f^{\pm} j_n^{\pm}(x,t), \end{aligned} \tag{9}$$

$$c_{ss}^{\pm}(x,t) = c_s^{\pm}(x, R_s^{\pm}, t)., \tag{10}$$

$$\Delta T(x,t) = U^{\pm}(\overline{c}_s^{\pm}(x,t)) - T(t) \frac{\partial U^{\pm}}{\partial T}(\overline{c}_s^{\pm}(x,t)), \tag{11}$$

$$\overline{c}_s^{\pm}(x,t) = \frac{3}{(R_s^{\pm})^3} \int_0^{R_s^{\pm}} r^2 c_s^{\pm}(x,r,t) dr \tag{12}$$

$$SOC(t) = \frac{mean_x[\overline{c}_s^{-}(x,t)]}{\overline{c}_{s,max}^{-}} \tag{13}$$

Along with these equations are corresponding boundary and initial conditions. For brevity, we only summarize the equations here. However, further details, including notation definitions, can be found in [7,8]. The parameters are taken from the publicly available DUALFOIL model, developed by Newman and his collaborators [9]. The simulations provided here correspond to a $LiCoO_2$ cell. However, the techniques are broadly applicable to any Li-ion chemistry.

2.2 Numerical Solution

Numerical solution of the coupled nonlinear PDAE (1)-(12) is, by itself, a nontrivial task. In fact, a body of literature exists on this singular topic (see, e.g. [10–12]). In our work the PDEs governing Fickian diffusion in the solid phase, (1), are discretized in the r-dimension via Padé approximates [12]. All the remaining PDEs are discretized in the x dimension via the central difference method [13], such that the mass of lithium is conserved. This ultimately produces a finite-dimensional continuous-time differential-algebraic equation (DAE) system

$$\dot{x}(t) = f(x(t), z(t), I(t)), \tag{14}$$

$$0 = g(x(t), z(t), I(t)), \tag{15}$$

where

$$x = \left[c_s^{\pm}, c_e, T\right]^T, \quad z = \left[\phi_s^{\pm}, i_e^{\pm}, \phi_e, j_n^{\pm}\right]^T. \tag{16}$$

This DAE model is then propagated forward in time via an implicit numerical scheme. In particular, the nonlinear discretized equations are solved via Newton's method, at each time step. A crucial step is to provide the scheme with analytic expressions for the Jacobian. This approach is rewarded with fast and accurate simulations.

2.3 Implementation

Provided alongside this documentation is a code tree, which is an organizational representation of `dfn.m`. The code tree allows the user to view the structure of the entire model, including parameters and variables, in one diagram. The tree is read vertically, and is more specific as the user reads down the tiers. Every variable used in `dfn.m` can be referenced within the code tree. Variables are also grouped together by purpose under each section. If variables come from outputs of another function, the user can find which functions were called by looking for the .m files under the variables. The code tree is color coded for the convenience of the user to find .m files, parameters, and the initial conditions.

3 Examples

The user may alter the input data (current and time) to simulate various battery charge/discharge cycles. Within this section are samples that reflect states of charge, discharge, and a sample drive cycle. Manual data is used within Fig. 3 and Fig. 4 while data loaded from a .mat file is used in Fig. 6.

```
22      %% Input Signal
23
24      % Manual Data
25 -    t = -2:p.delta_t:(60*60/3);
26 -    Iamp = zeros(length(t),1);
27 -    Iamp(t >= 0) =-10;
28 -    I = Iamp;
29
30      % Experimental Data
31      % load('data/UDDSx2_batt_ObsData.mat');
32      % tdata = t;
33      % Tfinal = tdata(end);
34      % t = -2:p.delta_t:Tfinal;
35      % Iamp = interp1(tdata,I,t,'spline',0);
36      % Ah_amp = trapz(tdata,I)/3600;
37      % I = Iamp * (0.4*35)/Ah_amp;
38      % I = I/2;
39 -    NT = length(t);
```

Figure 1: Manual data using a current of $10A/m^2$

If the user wishes to use manual data, this can be done within `dfn.m` under the section titled Input Signal as seen in Fig. 1. The user can alter the variable I, which represents the input current, as well as the array t, which represents time starting at 0. Here, an input of 10 A/m^2 is assigned to current, and a total time of 3600 seconds is used. Outputs are located in the code within `dfn.m`, as seen in Fig. 2. To view the outputs plotted over time, the user can call `plot_dfn.m` after `dfn.m` has finished running. Within the command window, `dfn.m` will only show the outputs of voltage and SOC.

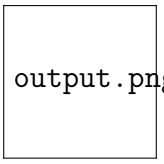

Figure 2: Output data within dfn.m

A state of discharge corresponds to a positive constant current, which can be seen because SOC is constantly decreasing. Within this state, $c_e(x,t)$ is constantly decreasing; this can be viewed in `plot_dfn.m`. A sample has been run using a constant current of 10 A/m^2 , as depicted in Fig. 3.

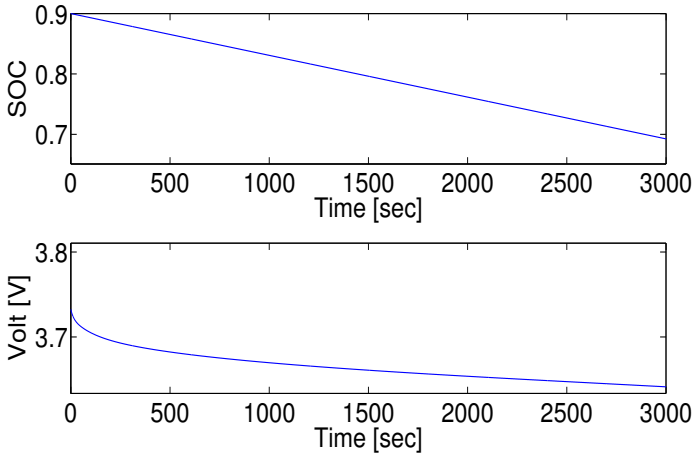

Figure 3: A decreasing SOC provided by constant positive current

Alternatively, a negative constant current will result in an increasing SOC. As SOC increases, $c_e(x,t)$ also increases. A sample has been run using a constant current of -10 A/m^2 , as depicted in Fig. 4. When using a negative current, time was reduced to 1200 seconds. If a time of 3600 seconds was used once again for this sample, the model would exceed its representative limits. Users can identify this situation when Matlab outputs imaginary values.

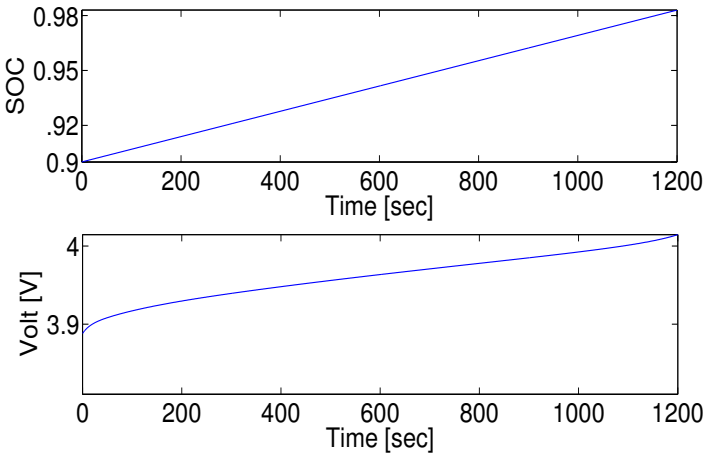

Figure 4: An increasing SOC provided by constant negative current

To use experimental data, data is loaded from a .mat file. The data used in this example comes from the file `UDDSx2_batt_ObsData.mat` which can be found in the data folder of the model. As seen in Fig. 5, the user can simply change the file name in line 31 to use a separate set of data. A drive cycle that represents a gradual decline in SOC is shown in Fig. 6. In this cycle, the current input fluctuates over time, which causes SOC and voltage in relation to fluctuate. Within this specific drive cycle the user can see that the battery is discharging.

```
22      %% Input Signal
23
24      % Manual Data
25      % t = -2:p.delta_t:(60*60/3);
26      % Iamp = zeros(length(t),1);
27      % Iamp(t >= 0) =-10;
28      % I = Iamp;
29
30      % Experimental Data
31 -    load('data/UDDSx2_batt_ObsData.mat');
32 -    tdata = t;
33 -    Tfinal = tdata(end);
34 -    t = -2:p.delta_t:Tfinal;
35 -    Iamp = interp1(tdata,I,t,'spline',0);
36 -    Ah_amp = trapz(tdata,I)/3600;
37 -    I = Iamp * (0.4*35)/Ah_amp;
38 -    I = I/2;
39 -    NT = length(t);
```

Figure 5: Experimental data from `UDDSx2_batt_ObsData.mat`

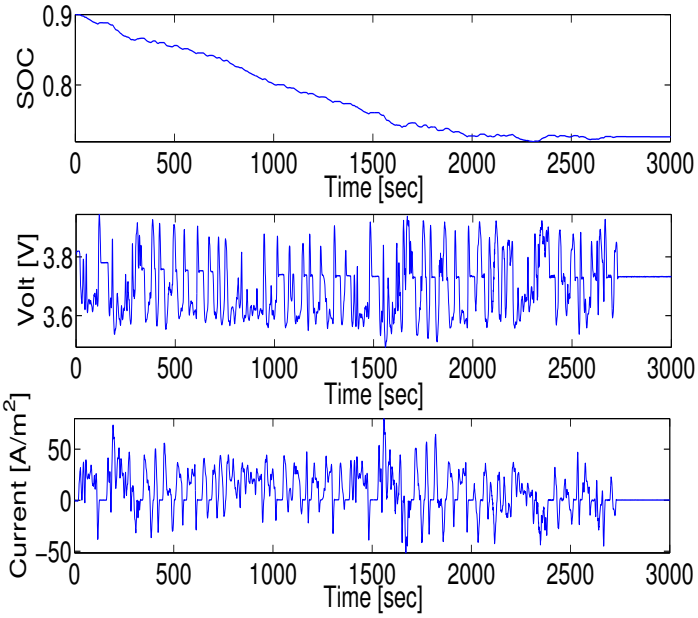

Figure 6: Sample drive cycle using data provided by `UDDSx2_batt_ObsData.mat`

4 Open Source Statement

This code made available by the Energy, Controls, and Applications Lab (eCAL) is for informational purposes only, and distributed "as is" with no support and/or warranty of any kind intended, implied, or provided.

References

[1] S. Moura, J. Stein, and H. Fathy, "Battery-Health Conscious Power Management in Plug-In Hybrid Electric Vehicles via Electrochemical Modeling and Stochastic Control," *Control Systems Technology, IEEE Transactions on*, vol. 21, no. 3, pp. 679–694, 2013.

[2] S. J. Moura, N. Chaturvedi, and M. Krstic, "Adaptive PDE Observer for Battery SOC/SOH Estimation via an Electrochemical Model," *ASME Journal of Dynamic Systems, Measurement, and Control, to appear*, 2013.

[3] S. Moura, N. Chaturvedi, and M. Krstic, "Constraint management in Li-ion batteries: A modified reference governor approach," in *American Control Conference (ACC), 2013*. IEEE, 2013, pp. 5332–5337.

[4] S. Bashash, S. J. Moura, J. C. Forman, and H. K. Fathy, "Plug-in hybrid electric vehicle charge pattern optimization for energy cost and battery longevity," *Journal of Power Sources*, vol. 196, no. 1, pp. 541 – 549, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378775310011390

[5] J. C. Forman, S. J. Moura, J. L. Stein, and H. K. Fathy, "Genetic identification and Fisher identifiability analysis of the Doyle-Fuller-Newman model from experimental cycling of a LiFePO4 cells," *Journal of Power*, vol. 210, pp. 263–275, 2012.

[6] S. Bashash, S. J. Moura, and H. K. Fathy, "On the aggregate grid load imposed by battery health-conscious charging of plug-in hybrid electric vehicles," *Journal of Power Sources*, vol. 196, no. 20, pp. 8747 – 8754, 2011.

[7] K. Thomas, J. Newman, and R. Darling, *Advances in Lithium-Ion Batteries*. New York, NY USA: Kluwer Academic/Plenum Publishers, 2002, ch. 12: Mathematical modeling of lithium batteries, pp. 345–392.

[8] N. A. Chaturvedi, R. Klein, J. Christensen, J. Ahmed, and A. Kojic, "Algorithms for advanced battery-management systems," *IEEE Control Systems Magazine*, vol. 30, no. 3, pp. 49 – 68, 2010.

[9] J. Newman. (2008) Fortran programs for the simulation of electrochemical systems. [Online]. Available: http://www.cchem.berkeley.edu/jsngrp/fortran.html

[10] K. A. Smith, C. D. Rahn, and C.-Y. Wang, "Control oriented 1D electrochemical model of lithium ion battery," *Energy Conversion and Management*, vol. 48, no. 9, pp. 2565 – 2578, 2007. [Online]. Available: http://dx.doi.org/10.1016/j.enconman.2007.03.015

[11] V. R. Subramanian, V. Boovaragavan, V. Ramadesigan, and M. Arabandi, "Mathematical model reformulation for lithium-ion battery simulations: Galvanostatic boundary conditions," *Journal of the Electrochemical Society*, vol. 156, no. 4, pp. A260 – A271, 2009.

[12] J. C. Forman, S. Bashash, J. L. Stein, and H. K. Fathy, "Reduction of an electrochemistry-based li-ion battery model via quasi-linearization and pade approximation," *Journal of the Electrochemical Society*, vol. 158, no. 2, pp. A93 – A101, 2011.

[13] R. L. Burden and D. Faires, *Numerical Analysis, Ed. 7*. Brooks/Cole, 2001.