The Journal of Open Source Software

COBRAPRO: A MATLAB toolbox for Physics-based Battery Modeling and Co-simulation Parameter Optimization

Sara Ha [1] **and Simona Onori** [2]

1 Mechanical Engineering, Stanford University, 440 Escondido Mall, Stanford 94305, CA, United States of America **2** Energy Science and Engineering, Stanford University, 367 Panama Mall, Stanford 94305, CA, United States of America

DOI: 10.21105/joss.06803

Software

- Review
- Repository
- Archive

Editor: Mojtaba Barzegari

Reviewers:

- @yuefan98
- @BradyPlanden
- @brosaplanella

Submitted: 02 March 2024
Published: 02 January 2025

Summary

COBRAPRO (**Co**-simulation **B**attery Modeling for **A**ccelerated **P**arameter **O**ptimization) is a physics-based battery modeling software with the capability to perform closed-loop parameter optimization using experimental data. COBRAPRO is based on the Doyle-Fuller-Newman (DFN) model (Doyle et al., 1993), which is the most widely accepted high-fidelity model that considers the lithium-ion transport and charge conservation in the liquid electrolyte and solid electrodes, and kinetics at the solid and liquid interface during lithium intercalation and deintercalation. Such physics-based models have found applications in battery design (Couto et al., 2023; Dai & Srinivasan, 2016) and advanced battery management systems to ensure reliable and safe operation of electric vehicles (Kolluri et al., 2020). The DFN model relies on physical parameters such as geometric, stoichiometric, concentration, transport, and kinetic properties. Accurate parameter values are essential for the model to predict real battery behavior under various usage conditions. However, parameters obtained from cell teardown experiments cannot be directly applied (Ha & Onori, 2024; Li et al., 2022), as the DFN model simplifies a real battery, assuming perfectly spherical particles, neglecting electrode heterogeneity, and considering internal dynamics only in one dimension. Parameter identification is therefore crucial for developing models that accurately capture battery dynamics. COBRAPRO enables users to identify these parameters for any battery using standard current-voltage data from a battery cycler, optimizing DFN parameters by minimizing the error between simulated and experimental results through an integrated optimization routine.

Statement of need

Although parameter calibration is required to accurately predict the dynamical behavior of real batteries, current DFN modeling tools lack the capability to perform parameter identification.

COMSOL Multiphysics© (COMSOL AB, Stockholm, Sweden, 2023) is a commercially available finite element modeling software commonly used to simulate the DFN model. Although COMSOL lacks a built-in parameter identification feature, it was demonstrated that COMSOL's *LiveLink™ for MATLAB©* can be used to establish communication between COMSOL and MATLAB for parameter optimization (Pozzato & Onori, 2023). This framework allows users to leverage the versatile suite of optimizers in MATLAB while running COMSOL to generate the model output. However, the expensive licensing fee and proprietary nature of COMSOL create barriers to public access, limiting collaboration and code reproducibility.

In contrast, several open-source DFN model simulation tools have emerged, such as PyBaMM (Sulzer et al., 2021), LIONSIMBA (Torchio et al., 2016), PETLION (Berliner et al., 2021),

DEARLIBS (Lee & Onori, 2021), fastDFN (Scott Moura, 2016), and MPET (Smith & Bazant, 2017). Among these packages, DEARLIBS is the only software equipped with a closed-loop parameter identification routine. Other open-source packages operate as open-loop software, requiring predefined parameters to simulate real battery behavior. Without parameter optimization capabilities, these open-loop tools typically depend on uncalibrated, experimentally measured parameters from the literature. Taking inspiration from DEARLIBS, COBRAPRO aims to address three primary challenges in the DFN model:

Challenge 1. Computational complexity

- **Issue:** The DFN model is also known as the pseudo-two-dimensional (P2D) model due to the coupling of the cell thickness (x-direction) and radial particle (r-direction) dimensions. This coupling of dimensions contributes to the model's computational complexity.
- **Solution:** COBRAPRO leverages the SUNDIALS IDA solver (Gardner et al., 2022; Hindmarsh et al., 2005) to enable fast computation speed compared to DEARLIBS, which uses MATLAB©'s Symbolic Math Toolbox and ode15 solver. For 1C discharge given 10 discretized points in the electrodes, separator, and particle domains ($N_p = N_s = N_n = N_{rp} = N_{rn} = 10$), COBRAPRO solves the DFN model in ~0.708 seconds, while DEARLIBS takes ~2.54 minutes, which is a two orders of magnitude improvement (~257 times). Under identical simulation conditions, LIONSIMBA and PyBaMM computed the model in ~1.13 seconds and ~0.237 seconds, respectively, demonstrating comparable performance to COBRAPRO. For larger discretization points, COBRAPRO exhibits up to three orders of magnitude improvement in computation speed compared to DEARLIBS. Refer to (Ha & Onori, 2024) for more details on the computation comparison study between COBRAPRO and other open-source DFN codes.

Challenge 2. Determining consistent initial conditions

- **Issue:** The partial differential equations (PDEs) governing the DFN model are discretized in x- and r-directions to form a system of ordinary differential equations (ODEs) and algebraic equations (AEs), referred to as differential-algebraic equations (DAEs). To solve the DAE system, it is essential to provide the correct initial conditions for the algebraic variables, which are typically not known *a priori* and must satisfy the AEs in the DFN model (Lee & Onori, 2021). Inconsistent initial conditions result in either a failure to start the simulation or the model diverging towards an incorrect solution (Methekar et al., 2011).
- **Solution:** The single-step approach (Lawder et al., 2015), a robust initialization method implemented in COBRAPRO and DEARLIBS, modifies the AEs into implicit ODEs to determine consistent initial conditions. This approach eliminates the need for nonlinear algebraic solvers, such as SUNDIALS IDACalcIC or CasADi's rootfinder, which are commonly used in other open-source DFN codes but have been shown to experience convergence issues (Methekar et al., 2011). In (Ha & Onori, 2024), a comparison study between the single-step approach and SUNDIALS IDACalcIC demonstrates that the single-step approach consistently determines initial conditions across different C-rates and node numbers, outperforming SUNDIALS IDACalcIC. For more details, refer to (Ha & Onori, 2024).

Challenge 3. Parameter identifiability and identification

- **Issue:** Extensive cell teardown experiments (Chen et al., 2020; Ecker et al., 2015; Schmalstieg et al., 2018) have measured properties such as cell geometry, open-circuit potentials (OCPs), electrolyte characteristics, and solid diffusion coefficients. However, directly using these parameters in the DFN model often leads to inaccurate simulation results. Systematic parameter optimization routines are necessary to develop a well-calibrated model that accurately predicts real battery dynamics. Furthermore, not all

parameters in the DFN model are identifiable due to overparametrization (Forman et al., 2012) and identifiability challenges (Goshtasbi et al., 2020), with identifiability depending on the type of data available.

- **Solution:** COBRAPRO features a co-simulation parameter optimization framework that utilizes particle swarm optimization (PSO) to minimize the multi-objective function (Allam & Onori, 2021; Xu et al., 2023), defined as the error between the experimental and simulated voltage and state of charge (SOC) in the electrodes (Ha & Onori, 2024). COBRAPRO leverages MATLAB's Parallel Computing Toolbox to accelerate PSO convergence through multicore processing. Additionally, COBRAPRO supports parameter identifiability analysis, allowing users to conduct local sensitivity analysis (LSA) and correlation analysis to identify a subset of parameters that can be reliably optimized using PSO.

Core Capabilities

- **Parameter identification routine:** PSO optimizes parameters using experimental current-voltage data
- **Parameter identifiability analysis:**
  - LSA: Perturbs parameters around their nominal values and evaluates their sensitivity with respect to voltage and SOC
  - Correlation analysis: Calculates linear correlation between two parameters
  - User-defined sensitivity and correlation index thresholds utilized to determine a set of identifiable parameters
- **DFN model implementation:**
  - PDEs along the x-direction spatially discretized with finite volume method (FVM)
  - PDE along the r-direction spatially discretized with FVM and finite difference method (FDM)
    - To estimate particle surface concentration using FVM, 3rd-order Hermite interpolation was implemented to account for sharp concentration gradients near the particle surface (Xu et al., 2023)
  - SUNDIALS IDA solver used to solve spatially discretized DFN equations
- **Determining consistent initial conditions:**
  - Single-step approach (Lawder et al., 2015)
  - SUNDIALS IDACalcIC (Gardner et al., 2022; Hindmarsh et al., 2005)
- **Simulating battery cycling:**
  - Constant current (CC) profiles
  - Hybrid pulse power characterization (HPPC) profiles
  - Dynamic current profiles

Example: Case Study on LG 21700-M50T Cells

To demonstrate the parameter identification process in COBRAPRO, a case study is conducted to parameterize a fresh LG 21700-M50T cell using the C/20 capacity test, HPPC, and driving cycle data (Pozzato et al., 2022). In this example, we break down the identification problem by systematically grouping parameters in each identification step, as shown in Figure 1. This multi-step approach is proposed to improve the identifiability of parameters instead of identifying all the unknown parameters simultaneously (Arunachalam & Onori, 2019). For detailed information on the multi-objective cost functions, identifiability analysis, and list of all the DFN parameters, refer to (Ha & Onori, 2024).

The Journal of Open Source Software

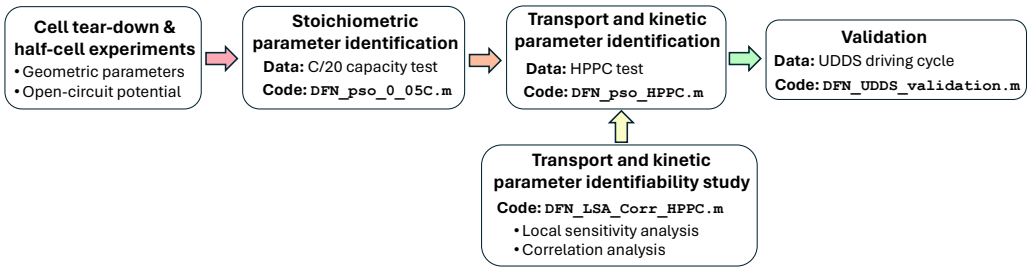

Figure 1: Case study: Parameter identification procedure on LG 21700-M50T cells.

Based on the analysis from (Ha & Onori, 2024), the geometric parameters, OCPs, transference number, and equilibrium electrolyte coefficients are determined to be redundant parameters from structural identifiability analysis. Their values are fixed to values measured from cell tear-down and half-cell experiments on LG 21700-M50 cells, as reported by (Chen et al., 2020). Next, the C/20 capacity test data is used to identify the stoichiometric parameters in the example code `DFN_pso_0_05C.m`. A parameter identifiability study is then performed in `DFN_LSA_Corr_HPPC.m`, which includes LSA and correlation analysis to select parameters that exhibit high sensitivity to HPPC voltage and SOC, while maintaining low correlation with other parameters. The identified subset of parameters is then optimized using HPPC data in the `DFN_pso_HPPC.m` code. Finally, the identified parameters are validated with urban dynamometer driving schedule (UDDS) data in the `DFN_UDDS_validation.m` code. All case study codes are located in the Examples folder.

View COBRAPRO's README on Github to view a list of all available example codes.

C/20 Capacity Test Identification

In `Examples/Parameter_Identification_Routines/DFN_pso_0_05C.m`, the `User Input` section is used to define the parameter names, PSO settings, experimental data, etc. A preview of the `User Input` section is provided here.

First, load the `Parameters_LG_INR21700_M50.m` function, which outputs a `param` structure containing the nominal (experimentally measured) parameters of an LG INR21700-M50 cell (Chen et al., 2020), as well as the DFN simulation settings, e.g., discretization method, method to determine consistent initial condition, constant or variable current type, etc.:

```
%% User Input
% Load nominal parameters
param = Parameters_LG_INR21700_M50;
```

Enter a mat file name to save the PSO results. The mat file will contain an updated `param` structure that replaces the nominal stoichiometric values with the identified parameters:

```
% Enter mat file name where your PSO results will be stored
file_name = 'identified_parameters_0_05C';
```

Define the names of the parameters you want to identify. Here, we identify the stoichiometric parameters θ_p^{100} (`theta100_p`), θ_n^{100} (`theta100_n`), θ_p^0 (`theta0_p`), and θ_n^0 (`theta0_n`):

```
% Enter names of parameters to identify (make sure names match the
% parameter names in "param" structure containing the nominal parameters)
param_CC = {'theta100_p', 'theta100_n', 'theta0_p', 'theta0_n'};
```

Define the lower and upper bounds of the parameters defined in `param_CC`:

```
% Enter lower and upper bounds of parameters to identify
% theta100_p
lower_bounds.theta100_p = 0.22;
```

```
upper_bounds.theta100_p = 0.34;
% theta100_n
lower_bounds.theta100_n = 0.7;
upper_bounds.theta100_n = 1;
% theta0_p
lower_bounds.theta0_p = 0.7;
upper_bounds.theta0_p = 1;
% theta0_n
lower_bounds.theta0_n = 0.015;
upper_bounds.theta0_n = 0.04;
```

Load your time, current, and voltage experimental data. In this example, load the C/20 capacity test data:

```
% Load Experimental Data
%-------------------------------------------------------------------------
%   t: Should be a vector consisting of your time experiment data      [s] (Mx1)
%   I: Should be a vector consisting of your current experiment data   [A] (Mx1)
%   V: Should be a vector consisting of your voltage experimental data [V] (Mx1)
%   -> where M is the total number of data points in your experiment
%-------------------------------------------------------------------------
% C/20 capacity test conducted on LG INR21700 M50T cells
load('data_INR21700_M50T/capacity_test_data_W8_Diag1.mat')
% Assign your data variables to t, I, and V
t = t_data;
I = I_data;
V = V_data;
```

Once all user inputs have been defined, run the `DFN_pso_0_05C.m` code to start the PSO. Once the PSO is finished, the code prints the identified parameter values, and the HPPC voltage and SOC objective function values to the Command Window:

```
Displaying identified values...
------------------------
theta100_p:
Identified value: 0.26475
0.22(lower) | 0.27(initial) | 0.34(upper)
------------------------
theta100_n:
Identified value: 0.77842
0.7(lower) | 0.9014(initial) | 1(upper)
------------------------
theta0_p:
Identified value: 0.89385
0.7(lower) | 0.9084(initial) | 1(upper)
------------------------
theta0_n:
Identified value: 0.029818
0.015(lower) | 0.0279(initial) | 0.04(upper)

Displaying objective function values...
------------------------
J_V =0.0033403 [-]
J_V =11.8445 [mV]
J_SOCp =0.030231 [%]
J_SOCn =0.019037 [%]
J_tot =0.003833 [-]
```

The code also plots the simulation results generated from the identified parameters and the experimental data, as shown in Figure 2 and Figure 3.

Run `Examples/Parameter_Identification_Routines/DFN_pso_0_05C_identification.m` to replicate the results shown here (code will load the identified parameters shown in this case study).

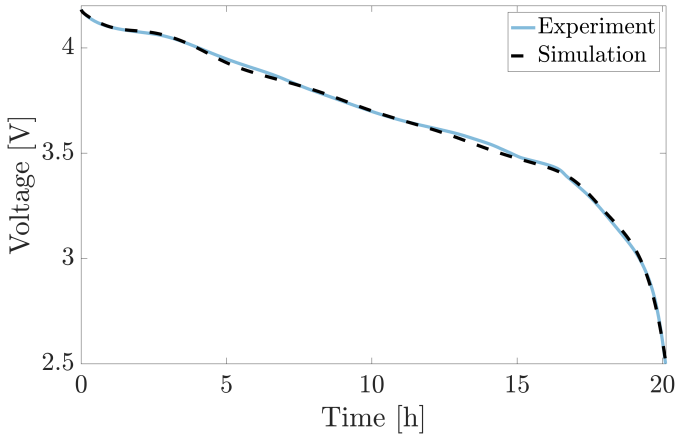

Figure 2: C/20 capacity test voltage identification results.

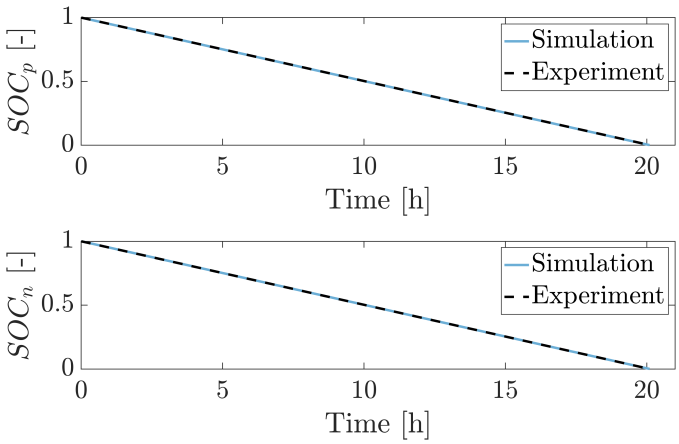

Figure 3: C/20 capacity test positive and negative electrode SOC identification results.

Parameter Identifiability Analysis

In `Examples/Parameter_Identifiability_Analysis/DFN_LSA_Corr_HPPC.m`, parameter identifiability analysis is conducted to determine a subset of identifiable parameters given the HPPC profile. In the `User Input` section, load the `param` structure that contains the identified stoichiometric parameter values from `DFN_pso_0_05C.m`:

```
%% User Input
% Load nominal parameters and identified stoichiometric parameters
% from C/20 capacity test data
load('identified_parameters_0_05C.mat','param')
```

Next, enter the names of the parameters for the identifiability analysis. These include the remaining parameters, excluding the structurally redundant parameters and stoichiometric coefficients, as outlined in the analysis by (Ha & Onori, 2024):

```
%--------------------------------------------------------------------------
% Enter names of parameters to conduct LSA (make sure names match the
% parameter names in "param" structure containing nominal parameters)
%--------------------------------------------------------------------------
param_LSA_HPPC = {'Rc' 'Dsp' 'Dsn' 'kp' 'kn' 'De' 'Kappa' 'sigmap' 'sigman'};
```

Enter the name of the mat file to store the list of identifiable parameters:

```
%--------------------------------------------------------------------------
% Enter mat file name where the identifiable parameters will be stored
%--------------------------------------------------------------------------
file_name = 'HPPC_identifiable_params';
```

Set the LSA perturbation coefficient and the correlation coefficient threshold values (refer to (Ha & Onori, 2024) for more details). In this example, five different correlation coefficient thresholds ($\beta = 0.8, 0.9, 0.95, 0.98, 0.99$) are investigated, resulting in five identifiable parameter sets, one for each threshold value:

```
%--------------------------------------------------------------------------
% Perturbation coefficient for LSA [-]
%--------------------------------------------------------------------------
pct=0.05;

%--------------------------------------------------------------------------
% Define correlation coefficient threshold
%--------------------------------------------------------------------------
% Correlation coefficient threshold used to determine the uncorrelated parameter
% vector prioritized by sensitivity, which will be considered the "identifiable"
% parameters for the HPPC profile (input vector to yield multiple identifiable
% parameter sets for each correlation threshold)
%--------------------------------------------------------------------------
beta_corr = [0.8 0.9 0.95 0.98 0.99];
```

Load the HPPC simulation conditions from `HPPC_W8_Diag1.mat`, which include the current magnitude (`curr_vec`) and time duration (`time_vec`) for each step in the HPPC profile:

```
%--------------------------------------------------------------------------
% Load HPPC profile input current vector [A] and simulation time [s]
%--------------------------------------------------------------------------
% Load HPPC_W8_Diag1.mat file that consists of curr_vec and time_vec
load('HPPC_W8_Diag1.mat')
```

Once all the user inputs have been defined, run the code to start the identifiability analysis. The code will output figures showing plots of parameter sensitivities as a function of time, the sensitivity index of each parameter, and the correlation matrix. The identifiable parameter subsets determined for each correlation coefficient threshold are printed to the Command Window:

```
-----------------------------------------------
Identifiable parameters from LSA and correlation analysis:
-----------------------------------------------
Rc Dsp Dsn (threshold = 0.8)
Rc Dsp Dsn (threshold = 0.9)
Rc Dsp Dsn De (threshold = 0.95)
Rc Dsp Dsn De (threshold = 0.98)
Rc Dsp Dsn kp De (threshold = 0.99)
```

HPPC Identification

In `Examples/Parameter_Identification_Routines/DFN_pso_HPPC.m`, the identifiable parameters determined from `DFN_LSA_Corr_HPPC.m` are identified using PSO. First, load your `param` structure, which contains the identified stoichiometric parameter values from `DFN_pso_0_05C.m`. In this example, we load the `identified_parameters_0_05C.mat` file, which contains a `param` structure with the identified stoichiometric parameter values from `DFN_pso_0_05C.m`:

```
%% User Input
% Load nominal parameters and identified stoichiometric parameters
% from C/20 capacity test data
load('identified_parameters_0_05C.mat','param')
```

When defining the names of the HPPC parameters to identify, users can manually type the parameters (Option 1) or load the parameter identifiability results generated from `DFN_LSA_Corr_HPPC.m` (Option 2).

Here, Option 2 is demonstrated by loading `HPPC_identifiable_params.mat`, which is generated from `DFN_LSA_Corr_HPPC.m` and contains five identifiable parameter sets for $\beta = 0.8, 0.9, 0.95, 0.98, 0.99$. In this example, the parameter set corresponding to $\beta = 0.95$ is selected for PSO, by defining `beta_value = 0.95`:

```
%--------------------------------------------------------------------------
% Option 2: Load identifiable parameters from identifiability analysis
% conducted in "Examples/Local_Sensitivity_Analysis/DFN_LSA_Corr_HPPC.m"
%--------------------------------------------------------------------------
load('HPPC_identifiable_params.mat')
% Enter desired beta value
beta_value = 0.95;
```

Enter the MAT file name to save an updated param structure containing the PSO results:

```
% Enter mat file name where your PSO results will be stored
file_name = 'identified_parameters_HPPC_0_95Corr';
```

Define the upper and lower bounds for each parameter in `param_HPPC`:

```
% Enter lower and upper bounds of parameters to identify
% Dsp
pct = 0.2; % perturbation coeff
lower_bounds.Dsp = 10^(log10(param.Dsp)*(1+pct));
upper_bounds.Dsp = 10^(log10(param.Dsp)*(1-pct));
...
```

Load the time, current, and voltage vectors generated from the HPPC data:

```
% Load Experimental Data
% HPPC test conducted on LG INR21700 M50T cells
load('HPPC_data_W8_Diag1.mat')
```

PSO settings can also be modified in the `User Input` section:

```
particle_num = 100;
exit_type = 'MaxStallIterations';
max_iterations = 5;
max_stall_tolerance = 0.5e-6;
social_adjustment = 3.6;
self_adjustment = 0.3;
```

Once all user inputs have been defined, run the code to start the PSO. Once the PSO is completed, the identified parameter and objective function values are printed to the Command Window:

```
Displaying identified values...
------------------------
Rc:
Identified value: 0.02339
0.001(lower) | 0.02339(initial) | 0.02339(upper)
------------------------
De:
Identified value: 7.1555e-11
6.6922e-13(lower) | 7.1555e-11(initial) | 7.6509e-09(upper)
------------------------
Dsp:
Identified value: 1.3732e-12
3.7936e-16(lower) | 1.3732e-12(initial) | 4.9706e-09(upper)
------------------------
Dsn:
Identified value: 1.6399e-11
9.5335e-15(lower) | 1.6399e-11(initial) | 2.8208e-08(upper)

Displaying objective function values...
------------------------
J_V =0.0050556 [-]
J_V =16.1016 [mV]
J_SOCp =0.13285 [%]
J_SOCn =0.17272 [%]
J_tot =0.0081113 [-]
```

Similar to `DFN_pso_0_05C.m`, the simulation results generated from the identified parameters are plotted against the experimental data, as shown in Figure 4 and Figure 5.

In `Examples/Parameter_Identification_Results/DFN_pso_HPPC_identification.m`, the results shown here can be replicated by defining the following in the `User Input` section:

```
%-------------------------------------------------------------------------
% Enter your identified parameters
%-------------------------------------------------------------------------
load('identified_parameters_HPPC_0_95Corr.mat')
```

This loads the mat file containing the identified HPPC values using $\beta = 0.95$. Note that the identified HPPC values for $\beta = 0.90$ or $\beta = 0.99$ can also be viewed by loading `identified_parameters_HPPC_0_90Corr.mat` or `identified_parameters_HPPC_0_99Corr.mat` instead.

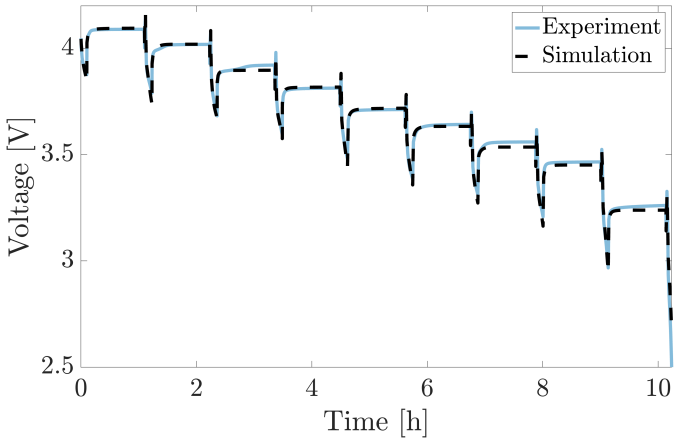

Figure 4: HPPC voltage identification results using $\beta = 0.95$.

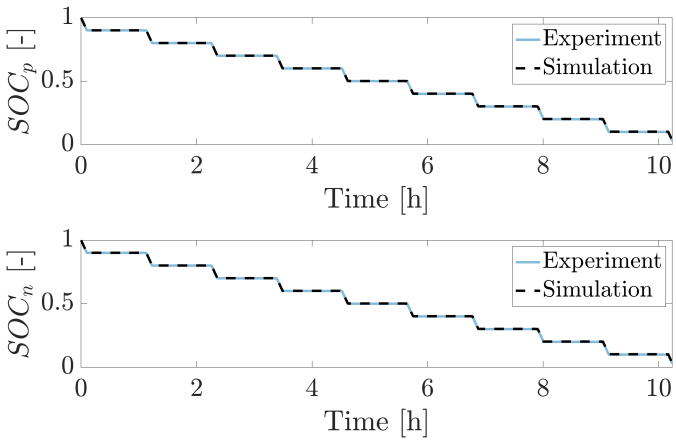

Figure 5: HPPC positive and negative electrode SOC identification results using $\beta = 0.95$.

UDDS Driving Cycle Validation

In `Examples/Parameter_Identification_Results/DFN_pso_UDDS_validation.m`, the identified parameters from the C/20 capacity test and HPPC data are validated using the UDDS driving cycle. The model is simulated under the UDDS profile and compared against the experimental UDDS data.

In the `User Input` section, load the parameter values identified from C/20 and HPPC data:

```
%% User Input
% Load identification results
load('identified_parameters_HPPC_0_95Corr.mat','param')
```

Note that to run the UDDS validation for the HPPC identification results using $\beta = 0.90$ and $\beta = 0.99$, load `identified_parameters_HPPC_0_90Corr.mat` or `identified_parameters_HPPC_0_99Corr.mat` instead.

Load the experimental UDDS data:

```
% Load Experimental Data
% HPPC test conducted on LG INR21700 M50T cells
load('data_INR21700_M50T/UDDS_W8_cyc1.mat')
```

The objective function is printed to the Command Window:

```
Displaying objective function values...
------------------------
J_V =0.0037372 [-]
J_V =13.6512 [mV]
J_SOCp =0.032023 [%]
J_SOCn =0.01561 [%]
J_tot =0.0042135 [-]
```

The simulation results and experimental data are plotted as shown in Figure 6 and Figure 7.

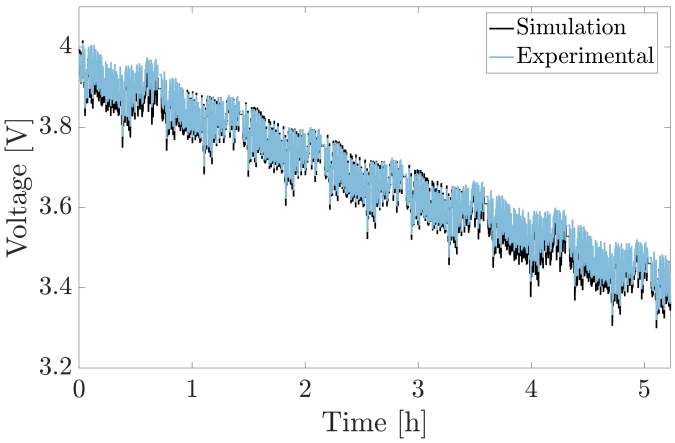

Figure 6: UDDS voltage identification results using $\beta = 0.95$.

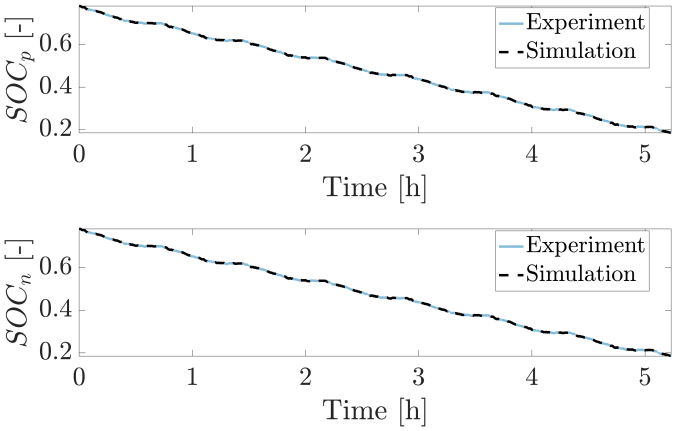

Figure 7: UDDS positive and negative electrode SOC identification results using $\beta = 0.95$.

Acknowledgements

The authors thank the Bits and Watts Initiative within the Precourt Institute for Energy at Stanford University for its partial financial support. We thank Dr. Le Xu for all the insightful discussions that greatly contributed to the enhancement of COBRAPRO. We extend our thanks to Alexis Geslin, Joseph Lucero, and Maitri Uppaluri for testing COBRAPRO and providing valuable feedback.

References

Allam, A., & Onori, S. (2021). Online Capacity Estimation for Lithium-Ion Battery Cells via an Electrochemical Model-Based Adaptive Interconnected Observer. *IEEE Transactions on Control Systems Technology*, *29*(4), 1636–1651. https://doi.org/10.1109/TCST.2020.3017566

Arunachalam, H., & Onori, S. (2019). Full Homogenized Macroscale Model and Pseudo-2-Dimensional Model for Lithium-Ion Battery Dynamics: Comparative Analysis, Experimental Verification and Sensitivity Analysis. *Journal of The Electrochemical Society*, *166*(8), A1380–A1392. https://doi.org/10.1149/2.0051908jes

Berliner, M. D., Cogswell, D. A., Bazant, M. Z., & Braatz, R. D. (2021). Methods—PETLION: Open-Source Software for Millisecond-Scale Porous Electrode Theory-Based Lithium-Ion Battery Simulations. *Journal of The Electrochemical Society*, *168*(9), 090504. https://doi.org/10.1149/1945-7111/ac201c

Chen, C.-H., Brosa Planella, F., O'Regan, K., Gastol, D., Widanage, W. D., & Kendrick, E. (2020). Development of Experimental Techniques for Parameterization of Multi-scale Lithium-ion Battery Models. *Journal of The Electrochemical Society*, *167*(8), 080534. https://doi.org/10.1149/1945-7111/ab9050

COMSOL AB, Stockholm, Sweden. (2023). *COMSOL Multiphysics®; v. 6.2*. www.comsol.com

Couto, Luis. D., Charkhgard, M., Karaman, B., Job, N., & Kinnaert, M. (2023). Lithium-Ion Battery Design Optimization Based on a Dimensionless Reduced-Order Electrochemical Model. *Energy*, *263*, 125966. https://doi.org/10.1016/j.energy.2022.125966

Dai, Y., & Srinivasan, V. (2016). On Graded Electrode Porosity as a Design Tool for Improving the Energy Density of Batteries. *Journal of The Electrochemical Society*, *163*(3), A406–A416. https://doi.org/10.1149/2.0301603jes

Doyle, M., Fuller, T. F., & Newman, J. (1993). Modeling of Galvanostatic Charge and Discharge of the Lithium/Polymer/Insertion Cell. *Journal of The Electrochemical Society*, *140*(6), 1526–1533. https://doi.org/10.1149/1.2221597

Ecker, M., Tran, T. K. D., Dechent, P., Käbitz, S., Warnecke, A., & Sauer, D. U. (2015). Parameterization of a Physico-Chemical Model of a Lithium-Ion Battery: I. Determination of Parameters. *Journal of The Electrochemical Society*, *162*(9), A1836–A1848. https://doi.org/10.1149/2.0551509jes

Forman, J. C., Moura, S. J., Stein, J. L., & Fathy, H. K. (2012). Genetic Identification and Fisher Identifiability Analysis of the Doyle–Fuller–Newman Model from Experimental Cycling of a LiFePO4 Cell. *Journal of Power Sources*, *210*, 263–275. https://doi.org/10.1016/j.jpowsour.2012.03.009

Gardner, D. J., Reynolds, D. R., Woodward, C. S., & Balos, C. J. (2022). Enabling New Flexibility in the SUNDIALS Suite of Nonlinear and Differential/Algebraic Equation Solvers. *ACM Transactions on Mathematical Software (TOMS)*, *48*(3), 1–24. https://doi.org/10.1145/3539801

Goshtasbi, A., Chen, J., Waldecker, J. R., Hirano, S., & Ersal, T. (2020). Effective Parameterization of PEM Fuel Cell Models—Part I: Sensitivity Analysis and Parameter Identifiability. *Journal of The Electrochemical Society*, *167*(4), 044504. https://doi.org/10.1149/1945-7111/ab7091

Ha, S., & Onori, S. (2024). COBRAPRO: An Open-source Software for the Dolye-Fuller-Newman Model with Co-simulation Parameter Optimization Framework. *Journal of The Electrochemical Society*. https://doi.org/10.1149/1945-7111/ad7292

Hindmarsh, A. C., Brown, P. N., Grant, K. E., Lee, S. L., Serban, R., Shumaker, D. E.,

& Woodward, C. S. (2005). SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers. *ACM Transactions on Mathematical Software (TOMS)*, *31*(3), 363–396. https://doi.org/10.1145/1089014.1089020

Kolluri, S., Aduru, S. V., Pathak, M., Braatz, R. D., & Subramanian, V. R. (2020). Real-Time Nonlinear Model Predictive Control (NMPC) Strategies using Physics-Based Models for Advanced Lithium-ion Battery Management System (BMS). *Journal of The Electrochemical Society*, *167*(6), 063505. https://doi.org/10.1149/1945-7111/ab7bd7

Lawder, M. T., Ramadesigan, V., Suthar, B., & Subramanian, V. R. (2015). Extending explicit and linearly implicit ODE solvers for index-1 DAEs. *Computers & Chemical Engineering*, *82*, 283–292. https://doi.org/10.1016/j.compchemeng.2015.07.002

Lee, S. B., & Onori, S. (2021). A Robust and Sleek Electrochemical Battery Model Implementation: A MATLAB® Framework. *Journal of The Electrochemical Society*, *168*(9), 090527. https://doi.org/10.1149/1945-7111/ac22c8

Li, W., Demir, I., Cao, D., Jöst, D., Ringbeck, F., Junker, M., & Sauer, D. U. (2022). Data-Driven Systematic Parameter Identification of an Electrochemical Model for Lithium-Ion Batteries with Artificial Intelligence. *Energy Storage Materials*, *44*, 557–570. https://doi.org/10.1016/j.ensm.2021.10.023

Methekar, R. N., Ramadesigan, V., Pirkle, J. C., & Subramanian, V. R. (2011). A Perturbation Approach for Consistent Initialization of Index-1 Explicit Differential–Algebraic Equations Arising from Battery Model Simulations. *Computers & Chemical Engineering*, *35*(11), 2227–2234. https://doi.org/10.1016/j.compchemeng.2011.01.003

Pozzato, G., Allam, A., & Onori, S. (2022). Lithium-Ion Battery Aging Dataset Based on Electric Vehicle Real-Driving Profiles. *Data in Brief*, *41*, 107995. https://doi.org/10.1016/j.dib.2022.107995

Pozzato, G., & Onori, S. (2023). A General Matlab and COMSOL Co-simulation Framework for Model Parameter Optimization: Lithium-Ion Battery and Gasoline Particulate Filter Case Studies. *Automotive Technical Papers*, 2023-01-5047. https://doi.org/10.4271/2023-01-5047

Schmalstieg, J., Rahe, C., Ecker, M., & Sauer, D. U. (2018). Full Cell Parameterization of a High-Power Lithium-Ion Battery for a Physico-Chemical Model: Part I. Physical and Electrochemical Parameters. *Journal of The Electrochemical Society*, *165*(16), A3799–A3810. https://doi.org/10.1149/2.0321816jes

Scott Moura. (2016). *fastDFN*. https://github.com/scott-moura/fastDFN

Smith, R. B., & Bazant, M. Z. (2017). Multiphase Porous Electrode Theory. *Journal of The Electrochemical Society*, *164*(11), E3291–E3310. https://doi.org/10.1149/2.0171711jes

Sulzer, V., Marquis, S. G., Timms, R., Robinson, M., & Chapman, S. J. (2021). Python Battery Mathematical Modelling (PyBaMM). *Journal of Open Research Software*, *9*(1), 14. https://doi.org/10.5334/jors.309

Torchio, M., Magni, L., Gopaluni, R. B., Braatz, R. D., & Raimondo, D. M. (2016). LIONSIMBA: A Matlab Framework Based on a Finite Volume Model Suitable for Li-Ion Battery Design, Simulation, and Control. *Journal of The Electrochemical Society*, *163*(7), A1192–A1205. https://doi.org/10.1149/2.0291607jes

Xu, L., Cooper, J., Allam, A., & Onori, S. (2023). Comparative Analysis of Numerical Methods for Lithium-Ion Battery Electrochemical Modeling. *Journal of The Electrochemical Society*, *170*(12), 120525. https://doi.org/10.1149/1945-7111/ad1293