TOOFAB: User's Manual

Zuan Khalik
z.khalik@tue.nl
11-2020
Version 0.1

Abstract

This document describes the features and the use of the TOOlbox for FAst Battery simulation (TOOFAB). TOOFAB is a battery simulation toolbox that can accurately and efficiently simulate the Doyle-Fuller-Newman model, which is an electrochemical battery model that describes the internal electrochemical processes in a battery.

Contents

1 Before you begin

1.1 What is TOOFAB?

The main feature of TOOFAB (TOOlbox for FAst Battery simulation) is an accurate and computationally efficient simulation of the Doyle-Fuller-Newman (DFN) model [1] based on the implementation and results presented in [2]. The DFN model is an electrochemistry-based battery model that describes the internal electrochemical processes (i.e., the concentration and potential gradients). This model has been used in a variety of applications, such as battery cell design and battery-management-system functionalities such as state-of-charge estimation, state-of-health estimation and fast charging.

While there are already several battery simulation toolboxes (freely) available that solve the DFN model equations, e.g. [3, 4, 5], our developed toolbox provides several advantages compared to the currently available toolboxes. Firstly, with the implementation used that is presented in [2], the computation times are generally significantly smaller than the currently available toolboxes, as is shown in [2]. Secondly, because the model equations have been solved directly, without the use of any other software or MATLAB toolboxes, the developed battery simulation toolbox does not require installing any other toolboxes. Finally, for the same reason as given in the previous point, i.e., that the model equations are solved directly without any other toolboxes, the MATLAB code is fairly easily translatable into C-code, which can either be made into an executable for even faster simulation, or can be used to implement the model on an embedded system, which are generally programmed in C.

The complete feature list of TOOFAB is as follows:

- Accurate and computationally efficient simulation of the DFN model describing the internal electrochemical processes of a battery.
- Ability to easily use and change concentration-dependent parameters.
- Ability to easily configure the toolbox for a desired trade-off between accuracy and computational efficiency through selection of model simplifications and adjusting the coarseness of the discretization grid.
- Option to use a battery ageing model [6] describing the side reactions that lead to Li-ion loss due to the build-up SEI (solid-electrolyte interphase) layer.
- Option to use a lumped thermal model [7] describing the thermal dynamics.
- Ability to use efficiently use the model in a closed-loop setting.

1.2 Prerequisites

This toolbox only requires a working version of MATLAB. The toolbox has been tested with MATLAB R2019b, but should work with any MATLAB version equal to or newer than MATLAB R2016b. This compatibility requirement comes from the feature that allows local functions,

added to MATLAB since version R2016b. A legacy version compatible with older MATLAB versions can be created upon request.

1.3 Installing the Toolbox

To install the toolbox there are two options:

- Copy the `DFN.m` function file to the desired location where it is going to be used.
- Add the path of the `DFN.m` (wherever it is downloaded) to the path. This can be done by executing the following commands in the path where `DFN.m` is stored:

```
addpath(genpath(pwd))
savepath
```

2 Using the Toolbox

2.1 Running Simulations

Simulations can be run using the `DFN` function. The various arguments of the `DFN` are explained in Section 4.1. For example, if the desire is to simulate the battery with the default parameters with a constant current of 30 A, for a duration of 2000 seconds with 0% state-of-charge as the initial condition, then `DFN` can be called as follows:

```
out = DFN(30,2000,0)
```

In Section 3, more examples can be found of how `DFN` can be used.

2.2 Choosing Simplifications

The model simplifications as considered in [2] can be chosen using either `simp` or `set_simp`. The main conclusions on the simplifications made in [2] are as follows

- Simplification [S1] linearizes the Butler-Volmer equation describing the reaction rate. This simplification generally does not lead to a significant sacrifice in accuracy.
- Simplification [S2] relates to the simplification of concentration-dependent parameters. Simplification [S2-I] assumes that there is a 1 step delay in how the concentration affects the parameter. This simplification has a negligible effect on accuracy. Simplification [S2-II] assumes the given concentration-dependent parameters to be constant. This simplification does not lead to a significant impact on accuracy in most cases.
- Finally, Simplification [S3], approximates diffusion along the radial direction in the particles with a 2-parameter polynomial approximation. This simplification may sometimes lead to a significant loss in accuracy.

2.3 Choosing Grid Parameters

The grid parameters can be chosen either through `grid` or `set_grid`. The main conclusions on the selection of the grid parameters made in [2] are as follows.

- For a high-energy cell, generally n_n, n_s, n_p have to be chosen relatively large, as diffusion throughout the cell is generally more restricted in a high-energy cell. On the other hand, $n_{r,n}$ and $n_{r,p}$ can be chosen relatively small, as diffusion throughout the particles is less restrictive.
- For the considered high-power cell, the opposite was observed. Here, n_n, n_s, n_p were chosen relatively small, while $n_{r,n}$, $n_{r,p}$ was chosen relatively large.

3 Examples

This section will be written soon.

4 Functions

4.1 DFN

`DFN`: The `DFN` function is the main function used to interface with the toolbox.

4.1.1 Syntax

```
out = DFN(input_current,tf,init_cond,param)
```

4.1.2 Input Arguments

`input_current`: This field contains information about the current profile. This field can be provided either as a scalar value, an array which contains the current levels at each specified sample time, or as a function handle, which points to a function that takes the output voltage, current, concentration and potentials, and the parameters as input and mainly provides the current as output. The latter form is especially useful when the battery is desired to be controlled in closed-loop.

When provided as a scalar value, the value is used as the applied current for time `0` until time `tf`.

When provided as an array, the first column of the array must specify the time samples and the second column of the array must specify the current levels at each specified time sample. How the current is interpolated and extrapolated in between the time samples can be specified with the parameters `current_interp` and `current_extrap`, respectively.

When provided as a function handle, the function that it points to must have the following input arguments, in the following specific order

1. `iter` [-] is the current iteration (or time step) of the main time integration loop.
2. `t` [s] is the current simulation time.
3. `i_app` [A] is the complete vector of the applied current up to the current time step.
4. `V` [V] is the complete voltage vector up to the current time step.
5. `soc` [-] is the complete state-of-charge vector up to the current time step.
6. `cs` [mol/m^3] is the complete solid-phase concentration matrix up to the current time step.

7. `ce` [mol/m^3] is the complete electrolyte-phase concentration matrix up to the current time step.
8. `phis` [V] is the complete solid-phase potential matrix up to the current time step.
9. `phie` [V] is the complete electrolyte-phase potential matrix up to the current time step.
10. `mem` is a MATLAB `struct` that can be used to store memory variables to be used throughout the simulation.
11. `param` is a MATLAB `struct` containing all the defined parameter fields.

Furthermore, the function that the function handle points to must have the following output arguments, in the following specific order

1. `i_app_next` [A] is the applied current at the next time step
2. `mem` is a necessary output argument to ensure that the memory functionality works. Note that the contents of `mem` can be accessed after the simulation in the
3. `end_simulation` is a flag that can be set to 1 to stop the simulation after the current time step. Set this flag to 0 if its use is not desired.
4. `dt` [s] is an optional output argument that specifies the step size for the next time step (if desired).

Note that when `current_interp` is specified as a function handle, the step size `dt` in `param` can be changed at each time step, which may be useful when a variable time discretization grid is desired. However, changing the step size also required the re-computation of the system matrices, and therefore changing the step size often throughout the simulation is not recommended.

`tf`: Final integration time.

`init_cond`: This field specifies the initial conditions of the cell. This field can be specified either as a scalar value or as a `struct` containing information on the desired initial conditions. When specified as a scalar value, the toolbox assumes steady-state conditions. Then, the value can be either chosen to represent an initial voltage or as an initial state-of-charge. To specify an initial state-of-charge, a value between 0 and 1 must be chosen, representing the state-of-charge. When a value larger than 2 is given for `init_cond`, the toolbox assumes that an initial voltage is specified.

When `init_cond` is given by a `struct`, the toolbox assumes non-steady-state conditions. The struct must contain the following required fields

- `cs` [mol/m^3] is the solid-phase concentration vector at the initial time step
- `ce` [mol/m^3] is the electrolyte-phase concentration vector at the initial time step
- `soc` [-] is the state-of-charge at the initial time step

Additionally, if thermal dynamics are enable, i.e. the parameter `thermal_dynamics` is set to 1, the following field must also be specified

- `T` [K] is the temperature at the initial time step

Additionally, if ageing is enabled, i.e., the parameter `ageing` is set to 1, the following fields must also be specified

- `Closs` is the amount of charge lost due to Li-ion loss at the initial time step.

`param`: This is an optional input argument, given as a `struct`. Here, any desired non-default parameter fields of the configurable parameters given in Section 5 can be specified. Configurable parameters that have not been specified will be assumed a certain default value.

4.1.3 Output Arguments

`out`: is a `struct` that contains all the output variables. The variables that are contained in `out` are

- `x` [-] is a vector indicating the position along the width of the cell at every discretization node, normalized to the total width of the cell. Useful for plotting. The values at the row indexes of the DFN model states that are output correspond to the position across the cell according to `x` (except for `cs`).
- `t` [s] is the time vector in seconds. The values at the column indexes of the DFN model states that are output correspond to the time according to `t`.
- `cs` [mol/m^3] is the solid-phase concentration. The values of `cs` at every time step are given by the rows of `cs`. The values of the concentration in the negative electrode are given from row index 1 through `n_n*n_rn`. The concentration values along the radial direction in the negative electrode are given by every `n_rn` rows for every of the `n_n` particles. The values of the concentration in the positive electrode are given from row index `n_n*n_rn+1` to the final row index. The concentration values along the radial direction in the positive electrode are given by every `n_rp` rows for every of the `n_p` particles.
- `ce` [mol/m^3] is the electrolyte-phase concentration.
- `phis` is the solid-phase potential
- `phie` is the electrolyte-phase potential
- `stoich` is the solid-phase concentration in the solid-phase at the surface of the particles
- `stoich` is the stoichiometry of the particles at the surface
- `jn` [mol/m^2/s] is the net molar flux exiting the particles.
- `U` [V] is the equilibrium potential
- `eta` [V] is the main-reaction over-potential
- `V` [V] is the terminal voltage
- `i_app` [V] is the applied current
- `T` [K] is the cell temperature
- `soc` [-] is the cell state-of-charge
- `param` is the `struct` containing the configuration of the configurable parameters that have been used for the simulation. The fields in `param` are explained in Section 5.

- `mem` is the memory variable used when providing a function handle to `input_current`
- `out_states` is a `struct` containing the final state of the necessary states that have to be specified when `init_cond` is given by a `struct`.
- `ageing` is a `struct` that is only output when parameter `ageing` is set to 1. It contains the following fields:
  - `Cbat` [C] is the aged capacity of the battery at the end of the simulation.
  - `s0_neg_aged` [-] is the aged the stoichiometric value at 0% SoC in the negative electrode.
  - `s100_neg_aged` [-] is the aged the stoichiometric value at 100% SoC in the negative electrode.
  - `s0_pos_aged` [-] is the aged the stoichiometric value at 0% SoC in the positive electrode.
  - `s100_pos_aged` [-] is the aged the stoichiometric value at 100% SoC in the positive electrode.
  - `Closs` [C] is the amount of charge lost due to Li-ion loss. Note that the fresh capacity of the cell minus `Closs` does not equal the aged capacity of the cell. Please see [ACC ref] for more information.
  - `Rf` [Ω/m^2] is the SEI film resistance at the end of the simulation.
  - `j2` [$mol/m^2/s$] is the side-reaction flux
  - `eta2` [V] is the side-reaction over-potential

5 User-Configurable Parameters

The user-configurable parameters are given as an (optional) input to the `DFN` function in a `struct`. The fields that can be specified in this `struct` are given as follows.

5.1 Simulation Parameters

`dt`: [s] the step size. Default value is 1.

`grid`: is a struct where each of its fields specify the coarseness of the grid parameters. Note that the grid parameters can also be changed (more effectively) through `set_grid`, however setting the simplifications in `grid` is more intuitive. The fields in `grid` are:

- `nn` [-] number of FVM nodes for the negative electrode. Default value is 10.
- `ns` [-] number of FVM nodes for the separator. Default value is 10.
- `np` [-] number of FVM nodes for the positive electrode. Default value is 10.
- `nrn` [-] number of FDM nodes for the negative electrode along the radial direction. Default value is 10.

- `nrp` [-] number of FDM nodes for the positive electrode along the radial direction. Default value is 10.

`set_grid`: vector containing the grid parameters, of which its elements can be specified in the following order: `set_grid = [nn ns np nrn nrp]`. This field is especially useful when it is desired to change the set of grid parameters effectively. Note that the grid parameters can be changed either through this parameter, or through the above parameters, seperately. If both `set_grid` and either of the other parameters are given, then the values specified in `set_grid` always take precedence. For an overview of how to effectively make a trade-off between model accuracy and computation time using the grid parameters, please see [2] and Section 2.3. Default value is `[10 10 10 10 10]`.

`tol`: [-] is the tolerance used to terminate the Newton's algorithm for solving the set of algebraic equations. Default value is 1e-2.

`iter_max`: [-] specifies the maximum number of iterations that the Newton's method at each time step may take. Default value is 100.

`Vmin`: [V] specifies the minimum allowable terminal voltage. Exceeding this minimum voltage terminates the simulation. Default value is 2.

`Vmax`: [V] specifies the maximum allowable terminal voltage. Exceeding this maximum voltage terminates the simulation. Default value is 5.

`verbose`: specifies the level of verbosity for the simulation. Can be specified either as 0,1,2. Default value is 2.

`current_interp`: specifies how the applied current should be interpolated when `input_current` is given by an array. Can be chosen as any of the methods specified in the documentation of the MATLAB `griddedInterpolant` function. Default value is `'linear'`.

`current_extrap`: specifies how the applied current should be extrapolated when `input_current` is given by an array. Can be chosen as any of the methods specified in the documentation of the MATLAB `griddedInterpolant` function. Default value is `'linear'`.

`fvm_method`: specifies how the face values of the FVM discretization are computed. Can be chosen as either 1,2,3, where 1 specifies the harmonic mean, 2 specifies assuming a linear variation, and 3 specifies the weighted mean. Default value is 1.

`thermal_dynamics`: toggles the thermal dynamics. When specified as 1 thermal dynamics are calculated, when specified as 0 thermal dynamics are ignored. Default value is 1.

`T_amb`: [K] specifies the ambient temperature. Default value is 298.15.

`ageing`: toggles ageing. When specified as 1 ageing is enabled, when specified as 0 ageing is disabled. Default value is 0.

`simp`: is a struct where each of its fields specify the type of simplifications that the user wants to make. Note that the simplifications can also be changed (more effectively) through `set_simp`, however setting the simplifications in `simp` is more intuitive. The fields in `simp` are:

- `kappa` specifies the type of simplification to make on the parameter `kappa`, where `0` indicates no simplification, `1` indicates that [S2-I] is applied and `2` indicates that [S2-II] is applied. Default value is 1.

- `De` specifies the type of simplification to make on the parameter `De`, where `0` indicates no simplification, `1` indicates that [S2-I] is applied and `2` indicates that [S2-II] is applied. Default value is 1.
- `dlnfdce` specifies the type of simplification to make on the parameter `dlnfdce`, where `0` indicates no simplification, `1` indicates that [S2-I] is applied and `2` indicates that [S2-II] is applied. Default value is 1.
- `Ds_pos` specifies the type of simplification to make on the parameter `Ds_pos`, where `0` indicates no simplification, `1` indicates that [S2-I] is applied and `2` indicates that [S2-II] is applied. Default value is 1.
- `butler_volmer` specifies the type of simplification to make on the Butler-Volmer equation, where `0` indicates no simplification, and `1` indicates that [S1] is applied. Default value is 0.
- `solid_phase_diffusion` specifies the type of simplification to make on the solid-phase diffusion equation equation, where `0` indicates no simplification, and `1` indicates that [S3] is applied. Default value is 0.

For an overview of how to effectively make a trade-off between model accuracy and computation time using these simplifications, please see [2] and Section 2.2

`set_simp`: This variable is a vector of 6 integers of the form `set_simp = [v1 v2 v3 v4 v5 v6]`, where `v1`, `v2`, `v3`, `v4`, `v5`, and `v6` relate to the simplifications made to `kappa`, `De`, `dlnfdce`, `Ds_pos`, the Butler-Volmer equation, and the solid-phase diffusion dynamics, respectively. The values of the elements are chosen in the same way as described in `simp`. Note that if both `set_simp` and any of the fields in `simp` are given, then the values specified in `set_simp` always take precedence. Default value is `[1 1 1 1 0 0]`.

5.2 Material-Specific Parameters

5.2.1 DFN Model Parameters

`F`: [C/mol] Faraday's constant

`R`: [J/mol/K] Ideal gas constant

`delta_neg`: [m] Negative electrode thickness

`delta_sep`: [m] Separator thickness

`delta_pos`: [m] Positive electrode thickness

`Rneg`: [m] Particle radius in the negative electrode

`Rpos`: [m] Particle radius in the positive electrode

`epse_neg`: [-] Electrolyte volume fraction at the negative electrode

`epse_sep`: [-] Electrolyte volume fraction at the separator

`epse_pos`: [-] Electrolyte volume fraction at the positive electrode

`epss_neg`: [-] Active material volume fraction at the negative electrode

`epss_pos`: [-] Active material volume fraction at the positive electrode

`p_neg`: [-] Bruggeman porosity exponent at the negative electrode

`p_sep`: [-] Bruggeman porosity exponent at the separator

`p_pos`: [-] Bruggeman porosity exponent at the positive electrode

`Ds_neg`: [m^2/s] Solid-phase Li diffusion coefficient at the negative electrode.

`Ds_pos`: [m^2/s] Solid-phase Li diffusion coefficient at the positive electrode. Can be specified either as a constant value or an anonymous function with concentration and temperature as inputs.

`De`: [m^2/s] Electrolyte-phase Li diffusion coefficient. Can be specified either as a constant value or an anonymous function with concentration and temperature as inputs.

`dlnfdce`: [-] Mean molar activity coefficient. Can be specified either as a constant value or an anonymous function with concentration and temperature as inputs.

`kappa`: [S/m] Ionic conductivity. Can be specified either as a scalar value or an anonymous function with concentration and temperature as inputs.

`sigma_neg`: [S/m] Electronic conductivity of the negative electrode.

`sigma_pos`: [S/m] Electronic conductivity of the positive electrode.

`Rf_neg`: [Ω/m^2] SEI film resistance in the negative electrode. Can be specified either as a scalar value or a vector value indicating the resistance value at each spatial node in the negative electrode.

`Rf_pos`: [Ω/m^2] SEI film resistance in the positive electrode.

`t_plus`: [-] Transference number of Li-ions.

`A_surf`: [-] Electrode plate surface area.

`alpha_a`: [-] Anodic charge-transfer coefficient. Note that the cathodic charge-transfer coefficient `alpha_c` is computed from `alpha_a` as `alpha_c = 1-alpha_a`.

`k0_neg`: [$mol^{-3/2}m^{-1/2}s^{-1}$] Kinetic constant in the negative electrode. Can be specified either as a constant value or an anonymous function with temperature as input.

`k0_pos`: [$mol^{-3/2}m^{-1/2}s^{-1}$] Kinetic constant in the positive electrode. Can be specified either as a constant value or an anonymous function with temperature as input.

`ce_0`: [mol/m^3] Average electrolyte concentration.

`Cbat`: [C] Reversible capacity of the battery.

`R_cc`: [Ω/m^2] Current collector resistance.

`s0_neg`: [-] Stoichiometry at 0% state-of-charge in the negative electrode of a fresh cell.

`s100_neg`: [-] Stoichiometry at 100% state-of-charge in the negative electrode of a fresh cell.

`s0_pos`: [-] Stoichiometry at 0% state-of-charge in the positive electrode of a fresh cell.

`s100_pos`: [-] Stoichiometry at 100% state-of-charge in the positive electrode of a fresh cell.

`U_neg`: [V] Negative electrode equilibrium potential, given by an anonymous function, with the stoichiometry as the input and equilibrium potential the output.

`U_pos`: [V] Positive electrode equilibrium potential, given by an anonymous function, with the stoichiometry as the input and equilibrium potential the output.

`EMF`: [V] Electro-motive force of the cell, given by an anonymous function, with cell state-of-charge as the input. Note that this field is optional and when this field is provided in `param`, the toolbox will automatically compute a corresponding `U_pos` and `U_neg`, based on either the default `U_neg` or a specified `U_neg`. Further note that if all three fields `U_neg`, `U_pos`, and `EMF` are provided, the toolbox will compute the equilibrium potentials with the provided `U_neg` and `U_pos`.

5.2.2 Thermal Model Parameters

`dU_dT_neg`: [V/K] Entropy change in the negative electrode, given by either by a scalar or an anonymous function, with the stoichiometry as the input.

`dU_dT_pos`: [V/K] Entropy change in the positive electrode, given by either by a scalar or an anonymous function, with the stoichiometry as the input.

`m`: [kg] Reversible capacity of the battery.

`hc`: [W/K] Convective heat transfer coefficient

`Cp`: [kg] Reversible capacity of the battery.

5.2.3 Ageing Model Parameters

`U2`: [V] Equilibrium potential of the side reactions. Chosen as a constant value.

`i02`: [A/m^2] Exchange current density of the side reactions, given by either by a scalar or an anonymous function, with the cell temperature as the input.

`V_SEI`: [m^3/mol] Molar volume of the SEI

`sigma_SEI`: [$1/(\Omega \cdot \mathrm{m})$] Conductivity of the SEI

Index

References

[1] M. Doyle, T. F. Fuller, J. Newman, Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell, J. Electrochem. Soc. 140 (6) (1993) 1526–1533. doi:10.1149/1.2221597.

[2] Z. Khalik, M. C. F. Donkers, H. J. Bergveld, Model simplifications and their impact on computational complexity for an electrochemistry-based battery modeling toolbox, J. Power Sources (2021).

[3] Fortran programs for the simulation of electrochemical systems, http://www.cchem.berkeley.edu/jsngrp/fortran.html (accessed November 10, 2020).

[4] M. Torchio, L. Magni, R. B. Gopaluni, R. D. Braatz, D. M. Raimondo, LIONSIMBA: A matlab framework based on a finite volume model suitable for Li-ion battery design, simulation, and control, J. Electrochem. Soc. 163 (7) (2016) A1192–A1205. doi:10.1149/2.0291607jes.

[5] L. Cai, R. E. White, Mathematical modeling of a lithium ion battery with thermal effects in COMSOL Inc. Multiphysics (MP) software, J. Power Sources 196 (14) (2011) 5985–5989. doi:10.1016/j.jpowsour.2011.03.017.

[6] P. Ramadass, B. Haran, P. M. Gomadam, R. White, B. N. Popov, Development of First Principles Capacity Fade Model for Li-Ion Cells, Journal of The Electrochemical Society 151 (2) (2004) A196. doi:10.1149/1.1634273.

[7] L. Rao, J. Newman, Heat-Generation Rate and General Energy Balance for Insertion Battery Systems, J. Electrochem. Soc. 144 (8) (1997) 2697–2704. doi:10.1149/1.1837884.