An explained example on how to simulate a li-ion battery constant-current discharge using *Spectral li-ion SPM*

Adrien Bizeray, Jorn Reniers, Stephen Duncan and David Howey, Department of Engineering Science, University of Oxford, Parks Road, Oxford OX1 3PJ

This file explains how the single particle model is built and implemented in *Spectral li-ion SPM* and describes step-by-step how to run a 1C constant-current discharge simulation. This script is formatted using MATLAB® publish markups and an easy to read HTML version can be generated by hitting the publish command (this may take a few minutes the first time as pictures will be automatically generated).

Spectral li-ion SPM solves the so-called lithium-ion battery Single Particle Model (SPM). The SPM is an electrochemical model describing lithium transport, reaction kinetics and thermodynamics in lithium-ion batteries. This SPM implementation is also coupled to a bulk thermal model describing battery temperature. The SPM is an approximation to the electrochemical pseudo-two dimensional lithium-ion battery model where electrolyte transport limitations are neglected. The SPM is therefore only valid at low currents up to 1C or 2C depending on battery design. In *Spectral li-ion SPM* the diffusion partial-differential equations of the SPM are discretised in space using an efficient spectral numerical method called Chebyshev orthogonal collocation. This implementation of the SPM is similar to our implementation of the more complex pseudo-two dimensional battery model which is discussed in our paper:

- Bizeray, A.M. , Zhao, S., Duncan, S. R., and Howey, D. A., "Lithium-ion battery thermal-electrochemical model-based state estimation using orthogonal collocation and a modified extended Kalman filter", Journal of Power Sources, vol. 296, pp. 400-412, 2015. Publisher copy and Open access pre-print

If you use *Spectral li-ion SPM* in your work, please cite our paper. This code has been developed at the Department of Engineering Science of the University of Oxford. For information about our lithium-ion battery research, visit the Howey Research Group website. If you are interested in our energy research, check out our research group website Energy and Power Group.

For more information and comments, please contact 'david.howey@eng.ox.ac.uk'.

Contents

- Electrochemical Single Particle Model (SPM)
- Bulk thermal model
- Discretization using Chebyshev orthogonal collocation
- Example: constant-current discharge simulation
- Plotting some results

Electrochemical Single Particle Model (SPM)

The SPM is a simplification of the more complex pseudo-two dimensional lithium-ion battery model where the porous active material of each electrode - anode and cathode - is represented by a single spherical particle of active material. The radii of these particles are chosen to approximate the porosity of the electrodes' material. During battery discharge, lithium stored in the anode particle diffuses to the surface of the particle where it is de-intercalated and transfered in the electrolyte. Simultaneously, lithium ions from the electrolyte intercalate at the surface of the cathode particle and diffuse through the cathode particle. This process is reversed during battery charging. The SPM assumes that the transport of lithium in the electrolyte is fast compared to the solid-phase transport in the particle of active material. The electrolyte diffusion and migration are therefore unmodelled in the SPM, limiting its validity to low C-rates. We give a brief overview of the model in this document with the aim of explaining our numerical implementation using Chebyshev orthogonal collocation but we refer the reader to the literature for more details on the SPM and its derivation, e.g.:

- Guo M, Sikha G, White RE "Single-Particle Model for a Lithium-Ion Cell: Thermal Behavior", Journal of The Electrochemical Society, 158 (2) A122-A132 (2011)
- Chaturvedi N, Klein R, Christensen J, Ahmed J, Kojic A, "Algorithms for Advanced Battery-Management Systems" IEEE Control Systems Magazine, 30 (3) 49-68 (2010)

The transport of lithium in each electrode particle is governed by the following Fickian spherical diffusion equation:

$$\frac{\partial c\,(r,t)}{\partial t} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(D_{s,i}(T)r^2\frac{\partial c\,(r,t)}{\partial r}\right)$$

where t is the time, r is the radial coordinate, c the lithium concentration in $mol.m^{-3}$, $D_{s,i}(T)$ the temperature-dependent diffusion coefficient of lithium in the active material in $m^2.s^{-1}$ and $R_{s,i}$ the radius of the particle in electrode i. This diffusion equation is subject to the following Neumann boundary conditions at the centre and surface of each particle:

$$\left.\frac{\partial c\,(r,t)}{\partial r}\right|_{r=0} = 0 \text{ and } D_{s,i}(T)\left.\frac{\partial c\,(r,t)}{\partial r}\right|_{r=R_{s,i}} = -j_i\,(t)$$

The reaction rate j_i in $mol.m^{-2}.s^{-1}$ is the flux of lithium exchanged at the surface between the particle and the electrolyte. In the SPM this flux can directly be related to the battery input current I in A by conservation of charge. This is because the flux j_i is inherently assumed uniform along the electrode thickness since only one particle is considered for each electrode. The relationships between j_i and I in each electrode are given by:

$$j_- = \frac{+I}{a_{s,-}\mathcal{F}\mathcal{A}\delta_-} \text{ and } j_+ = \frac{-I}{a_{s,+}\mathcal{F}\mathcal{A}\delta_+}$$

where $a_{s,i} = 3\epsilon_{s,i}/R_{s,i}$ is the specific interfacial area in m^{-1}, $\mathcal{F}$ is the Faraday constant, $\mathcal{A}$ is the electrode surface area (assumed equal in anode and cathode) and δ_i is the thickness of electrode i in m.

To simplify the model implementation we introduce the following change of variable $u = rc$. We also rescale the radial coordinate r by introducing the change of variable $x = r/R_{s,i}$. This maps the physical coordinates $r \in [0, R_{s,i}]$ onto the coordinates $x \in [0,1]$. The discretization using Chebyshev orthogonal collocation requires rescaling the domain on $[-1,1]$. However, mapping $r \in [0, R_{s,i}]$ onto $[-1,1]$ would result in discretization nodes clustered at both the surface and centre of the particles. A higher density of nodes is more sensible at the surface than the centre of particles because the concentration gradient is sharp at the surface, but relatively small at the centre. Instead, by introducing $x = r/R_{s,i}$ and solving the model on the domain $r \in [-R_{s,i}, R_{s,i}]$ mapped onto $x \in [-1,1]$, the Chebyshev nodes are only clustered at the surface, and by symmetry we only need to solve the equation on $x \in [0,1]$, i.e. $r \in [0, R_{s,i}]$. Introducing the changes of variable defined above results in the following partial-differential equation for the spherical diffusion:

$$\frac{\partial u\,(x,t)}{\partial t} = \frac{D_{s,i}(T)}{R_{s,i}^2}\frac{\partial^2 u\,(x,t)}{\partial x^2}$$

subject to:

$$\left.\frac{\partial u\,(x,t)}{\partial x}\right|_{x=1} - u\,(1,t) = \frac{-R_{s,i}^2}{D_{s,i}(T)}j_i(t)$$

By symmetry, we have that $c(r) = c(-r) \Leftrightarrow u(r) = -u(-r)$, or equivalently $u(x) = -u(-x)$. We must also ensure that $c = u/r$ remains finite as $r \to 0$, therefore it follows that $u(r=0) = 0$. More details on the model discretization are given in the 'Discretization using Chebyshev orthogonal collocation' section.

The transformed lithium concentration profiles $c(r,t) = u(r,t)/r$ in each particle and the temperature T are the only states of the SPM. The current I is the input of the model. The output voltage V can be computed from the states according to:

$$V = (U_+\,(\theta_+,T) + \eta_+) - (U_-\,(\theta_-,T) + \eta_-) - R_c i_{app}$$

The term $-R_c i_{app}$ is the voltage drop due to the contact resistance of the battery with R_c the contact resistance in $\Omega.m^2$ and $i_{app} = I/\mathcal{A}$ the current density in $A.m^{-2}$.

U_+ and U_- are the cathode and anode open-circuit potentials respectively. These open-circuit potentials are experimentally fitted functions of the surface stoichiometry of the active material $\theta_i = c_{s,i}^{surf}/c_{s,i}^{max}$, where $c_{s,i}^{max}$ is the theoretical maximum lithium concentration in the material. The open-circuit potentials are also temperature-dependent through a first-order Taylor series approximation with respect to temperature.

The overpotentials η_i are the voltage drops due to the departure from equilibrium potential associated with the intercalation/de-intercalation reaction in each electrode. The relationship between the reaction rate j_i and the overpotential η_i is given by the Butler-Volmer equation:

$$j_i = \frac{i_{0,i}}{\mathcal{F}} \left(\exp\left(\frac{\alpha_a \mathcal{F}}{RT} \eta_i \right) - \exp\left(\frac{-\alpha_c \mathcal{F}}{RT} \eta_i \right) \right)$$

where the exchange current density $i_{0,i}$ is a function of the reactants and products concentrations:

$$i_{0,i} = k_i(T) \mathcal{F} \sqrt{c_e^{avg}} \sqrt{c_{s,i}^{surf}} \sqrt{c_{s,i}^{max} - c_{s,i}^{surf}}$$

By assuming that the anodic and cathodic charge transfer coefficients α_a and α_c are equal, i.e. $\alpha_a = \alpha_c = 0.5$, we can express the overpotential η_i as a function of the reaction rate j_i (and therefore I):

$$\eta_i = \frac{2RT}{\mathcal{F}} \sinh^{-1}\left(\frac{j_i}{2 i_{0,i}} \right)$$

Bulk thermal model

The thermal model is a simple bulk thermal model, i.e. the cell temperature T is assumed uniform, governed by the following zero-dimensional heat equation:

$$\rho c_p \frac{dT}{dt} = \dot{q}_{gen} + \dot{q}_{conv}$$

where c_p is the lumped specific heat of the cell in $J.kg^{-1}.K^{-1}$ and ρ is the cell bulk density in $kg.m^{-3}$.

The total heat generation rate per unit volume $\dot{q}_{gen}$ is the sum of the reversible, reaction and ohmic heat contributions:

$$\dot{q}_{gen} = \dot{q}_{rev} + \dot{q}_{rx} + \dot{q}_c = -\frac{i_{app}}{L} T \left(\frac{dU_+}{dT} - \frac{dU_-}{dT} \right) + \frac{i_{app}}{L} \left(\eta_- - \eta_+ \right) + \frac{R_c A_s i_{app}^2}{V_c}$$

We assumed a convective boundary condition with the ambient and the convective heat removal rate is given by:

$$\dot{q}_{conv} = \frac{-h A_c \left(T - T^{amb} \right)}{V_c}$$

where h is the convective heat transfer coefficient in $W.m^{-2}.K^{-1}$, A_c is the cell surface area in m^2, V_c is the cell volume in m^3, and T^{amb} is the ambient temperature in K.

Discretization using Chebyshev orthogonal collocation

The partial-differential equation constituting the solid-phase particle model is discretized in space using the Chebyshev orthogonal collocation method. The underlying principle of Chebyshev orthogonal collocation consists in approximating the solution of the PDE by a sum of Chebyshev polynomials that interpolates the solution at the discretizing Chebyshev nodes defined on $[-1, 1]$. The derivative of this solution-interpolant can be conveniently calculated using Chebyshev differentiation matrices computed by the *chebdif.m* function of the MATLAB Differentiation Matrix Suite developed by Weideman and Reddy:

- JAC Weideman, SC Reddy, A MATLAB differentiation matrix suite, ACM Transactions of Mathematical Software, Vol 26, pp 465-519 (2000)

Denoting D_M^1 and D_M^2 the first- and second- derivative Chebyshev differentiation matrices with $M+1$ Chebyshev nodes, we have:

$$\frac{\partial u}{\partial x} \approx D_M^1 \mathbf{u}$$

and

$$\frac{\partial^2 u}{\partial x^2} \approx D_M^2 \mathbf{u}$$

where $\mathbf{u}$ is the vector containing the values of the solution u at the $M+1$ Chebyshev nodes.

The discrete approximation to the diffusion equation with the changes of variable $u = rc$ and $x = r/R_{s,i}$ can therefore be written:

$$\frac{\partial \mathbf{u}}{\partial t} = \frac{D_{s,i}(T)}{R_{s,i}^2} D_M^2 \mathbf{u}$$

Notations

- $\mathbf{u}_i$ denotes the i^{th} element of the vector $\mathbf{u}$.
- $[D_M^2]_{i,j}$ denotes the row i and column j of the matrix D_M^2.
- We also adopt the MATLAB convention $i:j$ for denoting the rows or columns ranging from i to j.

We define $M = 2N$ where N is an integer defined by the user ($N+1$ is the number of discretizing Chebyshev nodes in half of the sphere including the surface and centre). By recalling the symmetry $u(x) = -u(-x)$ and that $u(r=0) = 0 \Leftrightarrow \mathbf{u}_{N+1} = 0$, we can solve the diffusion equation for the first half of the states $\mathbf{u}_{1:N}$ only on $x \in [0,1]$ and write:

$$\frac{\partial \mathbf{u}_{1:N}}{\partial t} = \frac{D_{s,i}(T)}{R_{s,i}^2} \left([D_M^2]_{1:N,1:N} \mathbf{u}_{1:N} + [D_M^2]_{1:N,N+2:M+1} \mathbf{u}_{N+2:M+1} \right)$$

By symmetry, we have shown that:

$$u(r) = -u(-r) \Rightarrow \mathbf{u}_{N+2:M+1} = -P\mathbf{u}_{1:N}$$

where the permutation matrix $P = P^{-1}$ is the backward-identity matrix, or exchange matrix (matrix with 1 elements on the counter-diagonal and 0 elements everywhere else). Therefore, we can write the diffusion equation in terms of $\mathbf{u}_{1:N}$ only:

$$\frac{\partial \mathbf{u}_{1:N}}{\partial t} = \frac{D_{s,i}(T)}{R_{s,i}^2} \left([D_M^2]_{1:N,1:N} - [D_M^2]_{1:N,N+2:M+1} P \right) \mathbf{u}_{1:N}$$

For convenience, we define the matrices:

$$\tilde{D}_N^2 = [D_M^2]_{1:N,1:N} - [D_M^2]_{1:N,N+2:M+1} P$$

and

$$\tilde{D}_N^1 = [D_M^1]_{1:N,1:N} - [D_M^1]_{1:N,N+2:M+1} P$$

And the diffusion equation further simplifies to:

$$\frac{\partial \mathbf{u}_{1:N}}{\partial t} = \frac{D_{s,i}(T)}{R_{s,i}^2} \tilde{D}_N^2 \mathbf{u}_{1:N}$$

Similarly, we can write the discrete approximation to the boundary condition at the particle surface $r = R \Leftrightarrow x = 1$ as:

$$\left[\tilde{D}_N^1\right]_{1,:} \mathbf{u}_{1:N} - \mathbf{u}_1 = \frac{-R_{s,i}^2}{D_{s,i}(T)} j_i(t)$$

or equivalently:

$$\left[\tilde{D}_N^1\right]_{1,2:N} \mathbf{u}_{2:N} + \left(\left[\tilde{D}_N^1\right]_{1,1} - 1\right) \mathbf{u}_1 = \frac{R_{s,i}^2}{D_{s,i}(T)} j_i(t)$$

This allows us to express the value of u at the surface $\mathbf{u}_1$ in terms of the value of u at the other nodes $\mathbf{u}_{2:N}$ as:

$$\mathbf{u}_1 = \frac{\left[\tilde{D}_N^1\right]_{1,2:N}}{1 - \left[\tilde{D}_N^1\right]_{1,1}} \mathbf{u}_{2:N} + \frac{1}{1 - \left[\tilde{D}_N^1\right]_{1,1}} \frac{R_{s,i}^2}{D_{s,i}(T)} j_i(t)$$

We now substitute the expression of $\mathbf{u}_1$ into the diffusion equation for $\mathbf{u}_{1:N}$ to obtain the following reduced diffusion equation for the inner nodes only $\mathbf{u}_{2:N}$, which automatically satisfies the surface boundary condition:

$$\frac{\partial \mathbf{u}_{2:N}}{\partial t} = \frac{D_{s,i}(T)}{R_{s,i}^2} \left(\left[\tilde{D}_N^2\right]_{2:N,2:N} + \frac{\left[\tilde{D}_N^2\right]_{2:N,1} \left[\tilde{D}_N^1\right]_{1,2:N}}{1 - \left[\tilde{D}_N^1\right]_{1,1}} \right) \mathbf{u}_{2:N} + \frac{\left[\tilde{D}_N^2\right]_{2:N,1}}{1 - \left[\tilde{D}_N^1\right]_{1,1}} j_i(t)$$

And we can cast this diffusion equation into the following state-space form:

$$\frac{\partial \mathbf{u}_{2:N}}{\partial t} = D_{s,i}(T) A \mathbf{u}_{2:N} + B j_i(t)$$

and

$$\mathbf{c}_{2:N+1} = C \mathbf{u}_{2:N} + \frac{D}{D_{s,i}(T)} j_i(t)$$

where $\mathbf{c}$ is the vector containing the lithium concentration values at the Chebyshev nodes, and the matrices A, B, C and D are defined as follows:

$$A = \frac{1}{R_{s,i}^2} \left(\left[\tilde{D}_N^2\right]_{2:N,2:N} + \frac{\left[\tilde{D}_N^2\right]_{2:N,1} \left[\tilde{D}_N^1\right]_{1,2:N}}{1 - \left[\tilde{D}_N^1\right]_{1,1}} \right)$$

$$B = \frac{\left[\tilde{D}_N^2\right]_{2:N,1}}{1 - \left[\tilde{D}_N^1\right]_{1,1}}$$

$$C = \begin{bmatrix} \frac{1}{R_{s,i}} \frac{\left[\tilde{D}_N^1\right]_{1,2:N}}{1 - \left[\tilde{D}_N^1\right]_{1,1}} & & \\ \frac{1}{\mathbf{r}_2} & & \\ & \ddots & \\ & & \frac{1}{\mathbf{r}_N} \end{bmatrix}$$

$$D = \begin{bmatrix} \frac{R_{s,i}}{1 - \left[\tilde{D}_N^1\right]_{1,1}} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

The value of concentration at the centre of the particle $\mathbf{c}_{N+1}$ can be computed by recalling the original boundary condition:

$$\left.\frac{\partial c(r,t)}{\partial x}\right|_{x=1} = \frac{-R_{s,i}}{D_{s,i}(T)} j_i(t)$$

Recalling that $c(x) = c(-x)$, this can be written in discrete form:

$$\left([D_M^1]_{1,1:N} + [D_M^1]_{1,N+2:M+1} P\right) \mathbf{c}_{1:N} + [D_M^1]_{1,N+1} \mathbf{c}_{N+1} = \frac{-R_{s,i}}{D_{s,i}(T)} j_i(t)$$

which gives the expression for the concentration at the centre of the particle:

$$\mathbf{c}_{N+1} = -\frac{[D_M^1]_{1,1:N} + [D_M^1]_{1,N+2:M+1} P}{[D_M^1]_{1,N+1}} \mathbf{c}_{1:N} - \frac{1}{[D_M^1]_{1,N+1}} \frac{R_{s,i}}{D_{s,i}(T)} j_i(t)$$

Example: constant-current discharge simulation

In this example script, we will simulate the constant-current discharge of a lithium-ion battery using *Spectral li-ion SPM*. The parameters provided are for a LCO cell and were found in the literature. Please see our paper for more details:

- Bizeray A.M. , Zhao, S., Duncan, S. R., and Howey, D. A., "Lithium-ion battery thermal-electrochemical model-based state estimation using orthogonal collocation and a modified extended Kalman filter", Journal of Power Sources, vol. 296, pp. 400-412, 2015. Publisher copy and Open access pre-print

We first need to add the source files to the MATLAB path. Also, make sure the *chebdif.m* file from the *MATLAB Differentiation Matrix Suite* is available on the MATLAB path. The *MATLAB Differentiation Matrix Suite* can be downloaded at http://uk.mathworks.com/matlabcentral/fileexchange/29-dmsuite. We suggest to drop the unzip *dmsuite* folder into the source folder of *Spectral li-ion SPM* and then run the following commands at the beginning of your script.

```
% Adding required folder / subfolder on the MATLAB path
clear; close all;
addpath(genpath('source'));
addpath(genpath('model_parameters'));

%{
    Note: Make sure the 'MATLAB Differentiation Matrix Suite' developed by
    Weideman and Reddy is on your MATLAB path.
    Ref: JAC Weideman, SC Reddy, A MATLAB differentiation matrix suite,
    ACM Transactions of Mathematical Software, Vol 26, pp 465-519 (2000).)

    The 'MATLAB Differentiation Matrix Suite' is available for download at:
    http://uk.mathworks.com/matlabcentral/fileexchange/29-dmsuite
%}
```

All the model parameters are defined in the get_modelData.m file. The open-circuit potential and entropic coefficient for both electrodes are defined in the get_openCircuitPotential.m file. You should modify these files to define your own model parameters. A *data* structure containing the parameters is created by calling the get_modelData.m function:

```
data = get_modelData;     % Load the model parameters for a cell
%{
    Model parameters can be defined by the user by editing the file
    'model_parameters/get_modelData.m'
    The open-circuit potential and entropy coefficients of each electrode
    are defined in the file 'model_parameters/openCircuitPotential.m'
%}
```

The model is built by calling the get_model.m function which creates a *matrices_spm* structure containing the differentiation matrices and the matrices A, B, C, and D of the diffusion state-space model. The Chebyshev nodes are computed by the get_nodes.m function with $N+1$ the number of Chebyshev nodes.

```
% Building the model
N = 6;       % N+1 is the number of Chebyshev nodes used to discretize
             % the diffusion equation in each particle
nodes            = get_nodes(data,N);          % Create the Chebyshev nodes
matrices_spm     = get_model(data,nodes,N);   % Create the SPM model
```

The user must supply initial conditions to the model, i.e. the initial active material stoichiometry in anode *x1_init* and cathode *y3_init* and the initial cell temperature *T*. The initial stoichiometry is assumed uniform within each particle (cell at equilibrium). The initial state vector is then created in the *initSPM* structure as *initSPM.y0* by calling the get_init.m function.

```
% Initial conditions
x1_init = data.x1_soc1;
y3_init = data.y3_soc1;
T_init  = data.T_amb;
%{
    The user must define 3 initial conditions:
        x1_init : initial stoichiometry in the anode particle (assumed uniform)
        y3_init : initial stoichiometry in the cathode particle (assumed uniform)
        T_init  : initial cell temperature
%}
initSPM = get_init(x1_init,y3_init,T_init,data,nodes,matrices_spm);
```

The input of the model is the current I in Amperes. Any current input can be set by the user by defining an anonymous function *I* of time, which returns a current vector associated with a time vector.

```
% Input current
C_rate = 1;     % C-rate
I = @(t) C_rate*data.C_nom*ones(size(t));    % Applied current [A]
```

The time span for the time integration can be set by defining the *tspan* vector. If the *tspan* vector contains the initial and final times only, the time steps chosen by the solver will be returned. If the *tspan* vector contains more than two values, the user-defined time steps will be returned by the solver (see ode45 help). The voltage limits to stop the simulation are defined in a *V_limit* vector containing the lower and upper cut-off voltages. This vector is then provided to the cutOffVoltage.m function.

```
% Time integration & Terminal conditions
% There are 2 terminal conditions:
%   the simulation time span (i.e. model 1 hour)
%   the voltage limits (i.e. the minimum and maximum voltage is reached)
% Whichever occurs first will stop time integration
tspan = 0:10:3600;      % Simulation time span
V_limit = [3.0 4.2];    % Minimum and maximum voltage
```

The model is then integrated in time by running the following code using the ode45 ODE solver. The function cutOffVoltage.m stops the simulation if the voltage reaches the limits defined in the *V_limit* vector. The derivs_spm.m function is the actual single particle model and returns the time derivative of the state vector $\partial \mathbf{y}/\partial t$ at a given time t and state $\mathbf{y}$.

```
% Solution
event = @(t,y) cutOffVoltage(t,y,I,data,matrices_spm,V_limit);
fun = @(t,y) derivs_spm(t,y,I,data,matrices_spm);
opt = odeset('Events',event);
```

```
[result.time,result.state] = ode45(fun,tspan,initSPM.y0,opt);
```

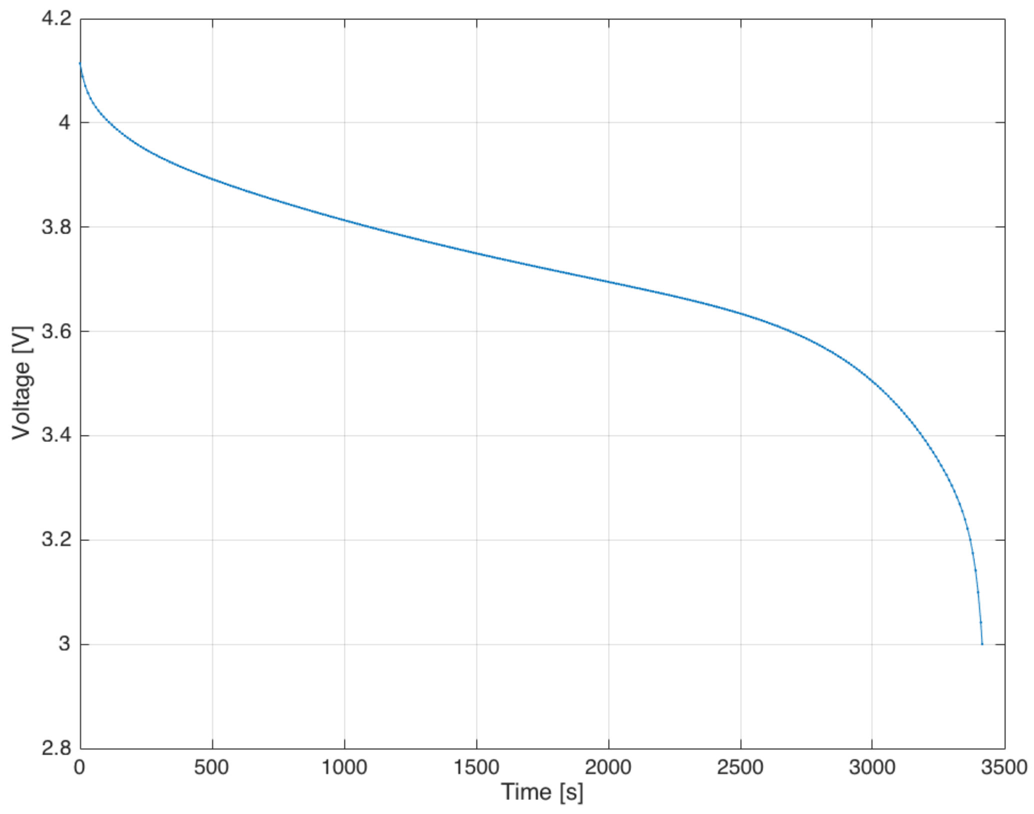
The ODE solver ode45 only returns the time steps and associated states of the model, other quantities of interest such as voltage, temperature, SOC, concentration profiles etc. are computed by the get_postproc.m function. The *result* structure containing the *time* and *state* vectors computed by the ODE solver is provided to the get_postproc.m function, which returns a structure with additional quantities of interest.

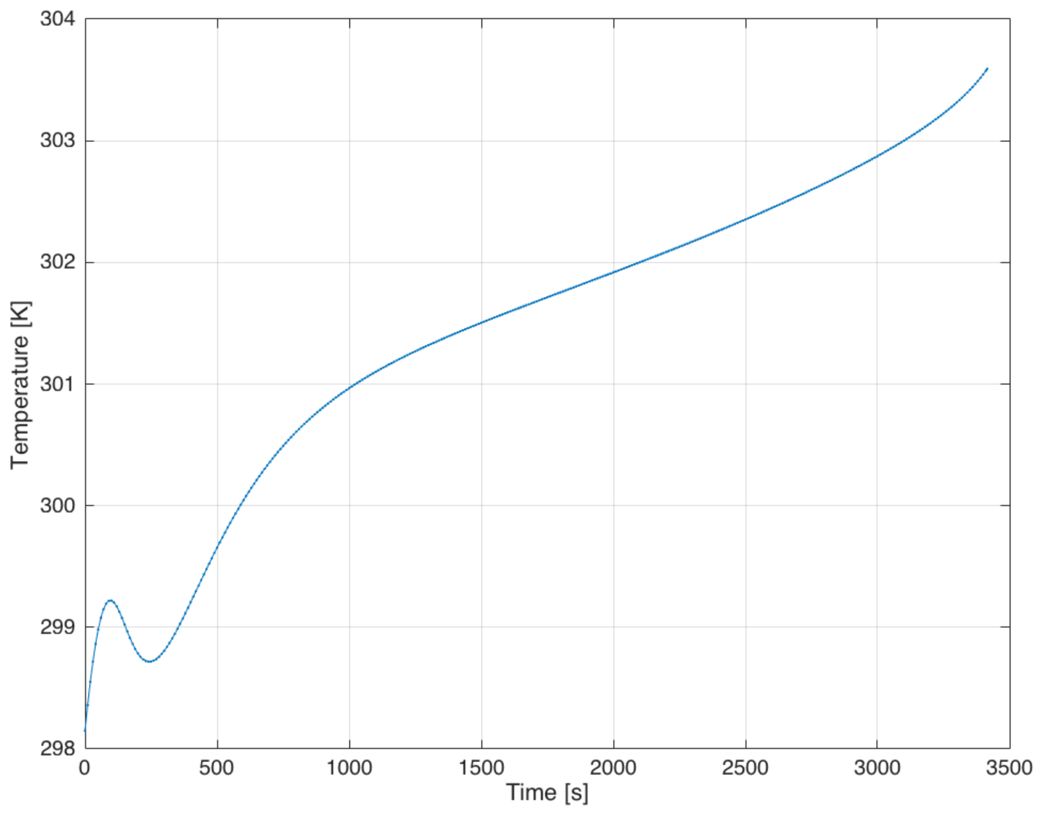
```
% Postprocessing result (compute voltage, temperature, etc. from states)
result = get_postproc( result,data,nodes,matrices_spm,I);
```

Plotting some results

We plot here some results for the 1C constant-current discharge simulation, such as the voltage and temperature.

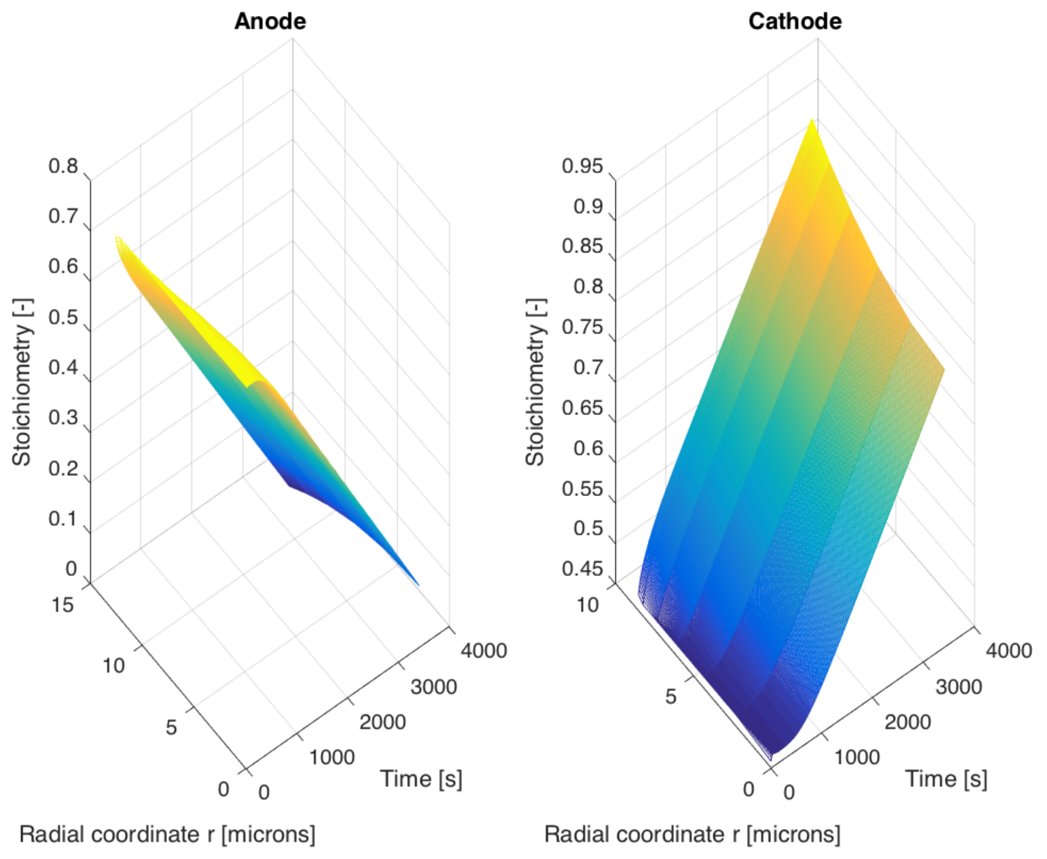
```
% Voltage vs time
figure;
plot(result.time,result.voltage,'.-');
xlabel('Time [s]');
ylabel('Voltage [V]');
grid on;

% Temperature vs time
figure;
plot(result.time,result.temperature,'.-');
xlabel('Time [s]');
ylabel('Temperature [K]');
grid on;
```





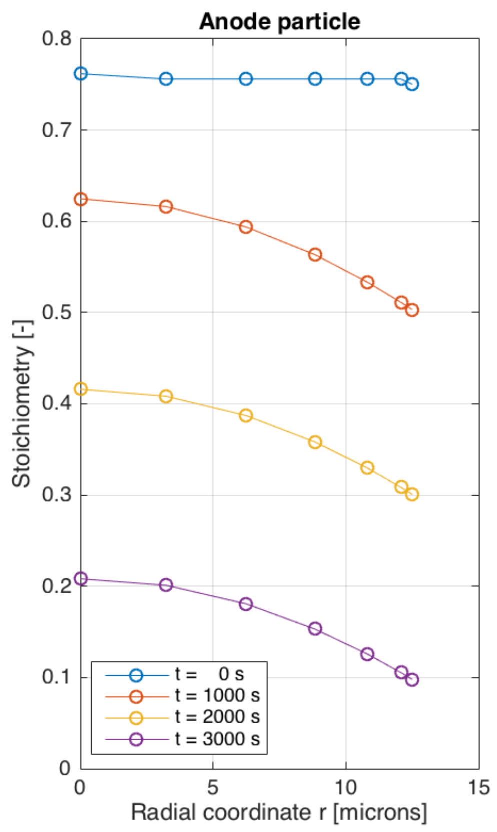
We can also plot the states of the model. For instance the evolution of the lithium concentration profile in each electrode particle is plotted here as a surface plot.

```
% Concentration profiles vs time
[R1,T] = meshgrid(nodes.xc2xp(nodes.xr,'r1')*1e6,result.time);
[R3,~] = meshgrid(nodes.xc2xp(nodes.xr,'r3')*1e6,result.time);

figure;
subplot(121)
mesh(T,R1,result.cs1/data.cs1_max); grid on;
title('Anode')
xlabel('Time [s]');
ylabel('Radial coordinate r [microns]');
zlabel('Stoichiometry [-]');

subplot(122)
mesh(T,R3,result.cs3/data.cs3_max); grid on;
title('Cathode');
xlabel('Time [s]');
ylabel('Radial coordinate r [microns]');
zlabel('Stoichiometry [-]');
```

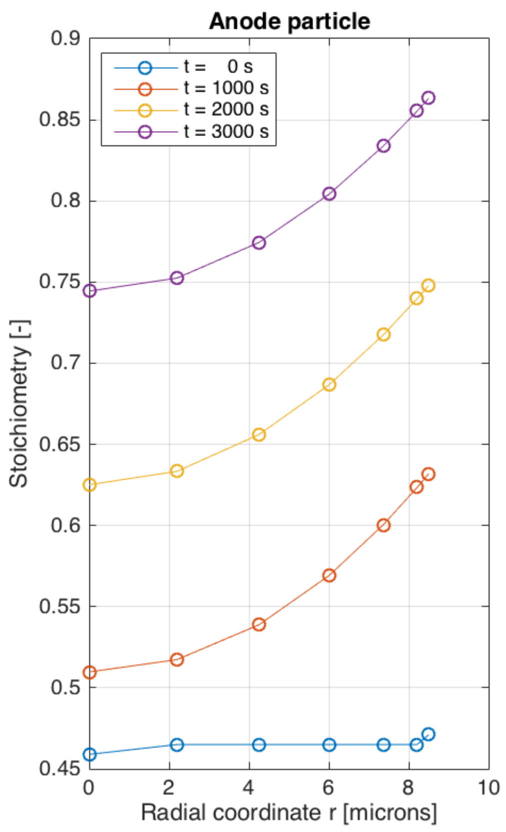


Or, if you prefer 2D graphs:

```
figure;
plot_idx = [1,101,201,301];      % Chosen times to plot

for i = 1:length(plot_idx)
    subplot(121)
    h1(i) = plot(nodes.xc2xp(nodes.xr,'r1')*1e6 , ...
        result.cs1(plot_idx(i),:)/data.cs1_max,'o-');
    xlabel('Radial coordinate r [microns]');
    ylabel('Stoichiometry [-]');
    title('Anode particle');
    hold on; grid on;

    subplot(122)
    h2(i) = plot(nodes.xc2xp(nodes.xr,'r3')*1e6 , ...
        result.cs3(plot_idx(i),:)/data.cs3_max,'o-');
    xlabel('Radial coordinate r [microns]');
    ylabel('Stoichiometry [-]');
    title('Anode particle');
    hold on; grid on;
end
legend_label = [ ...
    repmat('t = ',length(plot_idx),1),num2str(result.time(plot_idx)) , ...
    repmat(' s',length(plot_idx),1) ];

legend(h1,legend_label,'Location','Best');
legend(h2,legend_label,'Location','Best');
```



Anode particle
Stoichiometry [-]
Radial coordinate r [microns]
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
5
10
15
t = 0 s
t = 1000 s
t = 2000 s
t = 3000 s



Anode particle
Stoichiometry [-]
Radial coordinate r [microns]
0.45
0.5
0.55
0.6
0.65
0.7
0.75
0.8
0.85
0.9
0
2
4
6
8
10
t = 0 s
t = 1000 s
t = 2000 s
t = 3000 s