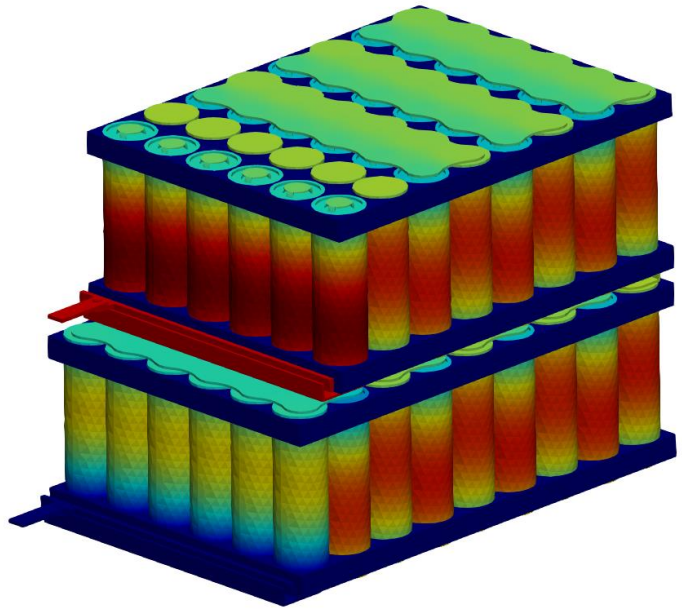
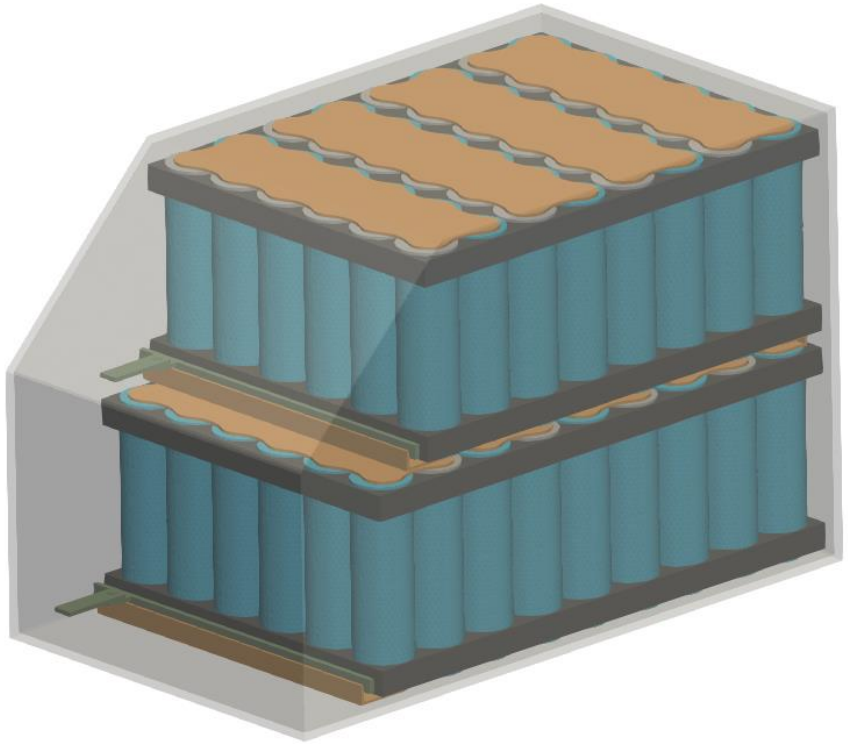
QUICKERSIM



Tutorial e-motorbike’s battery pack

Overview of model



This tutorial presents how to perform simulation of heat transfer in a e-motorbike's battery pack in Q-Bat software.

The model consists 102 groups of 21700 Li-ion cells (17s6p), terminals, connectors, holders and casing.

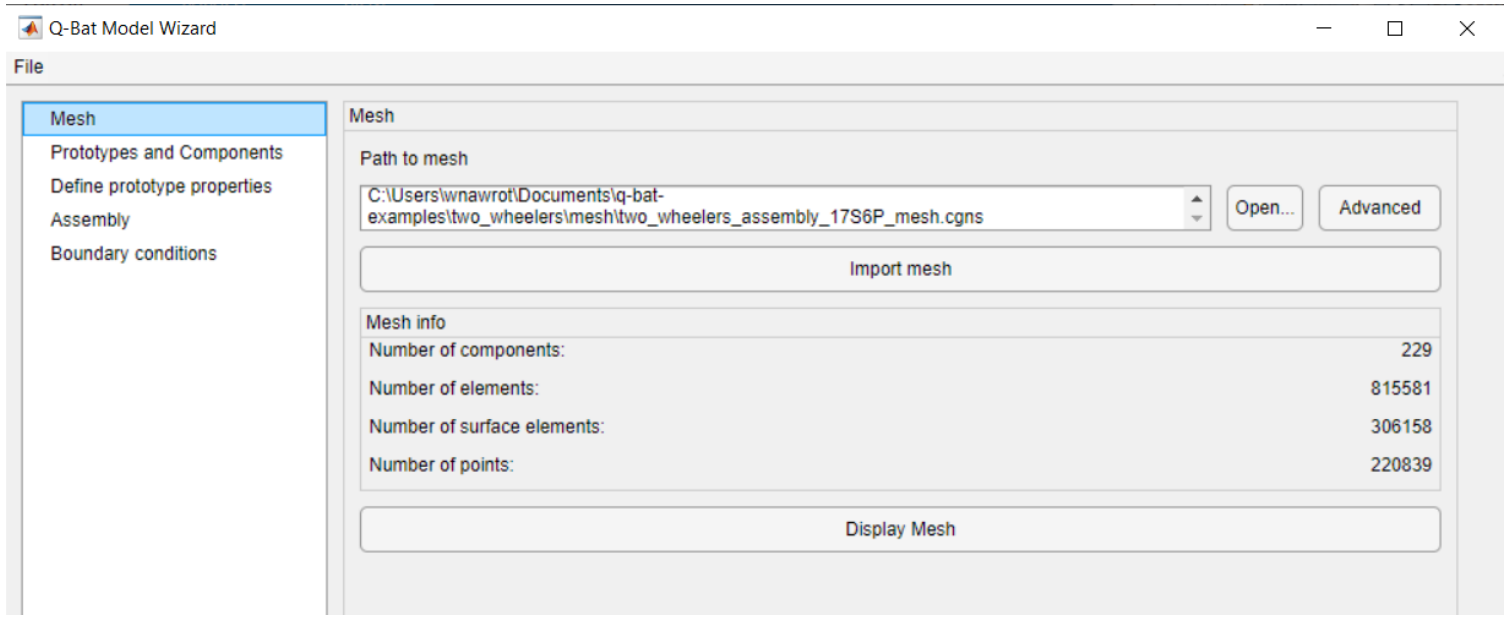
The geometry is presented below.

Import mesh

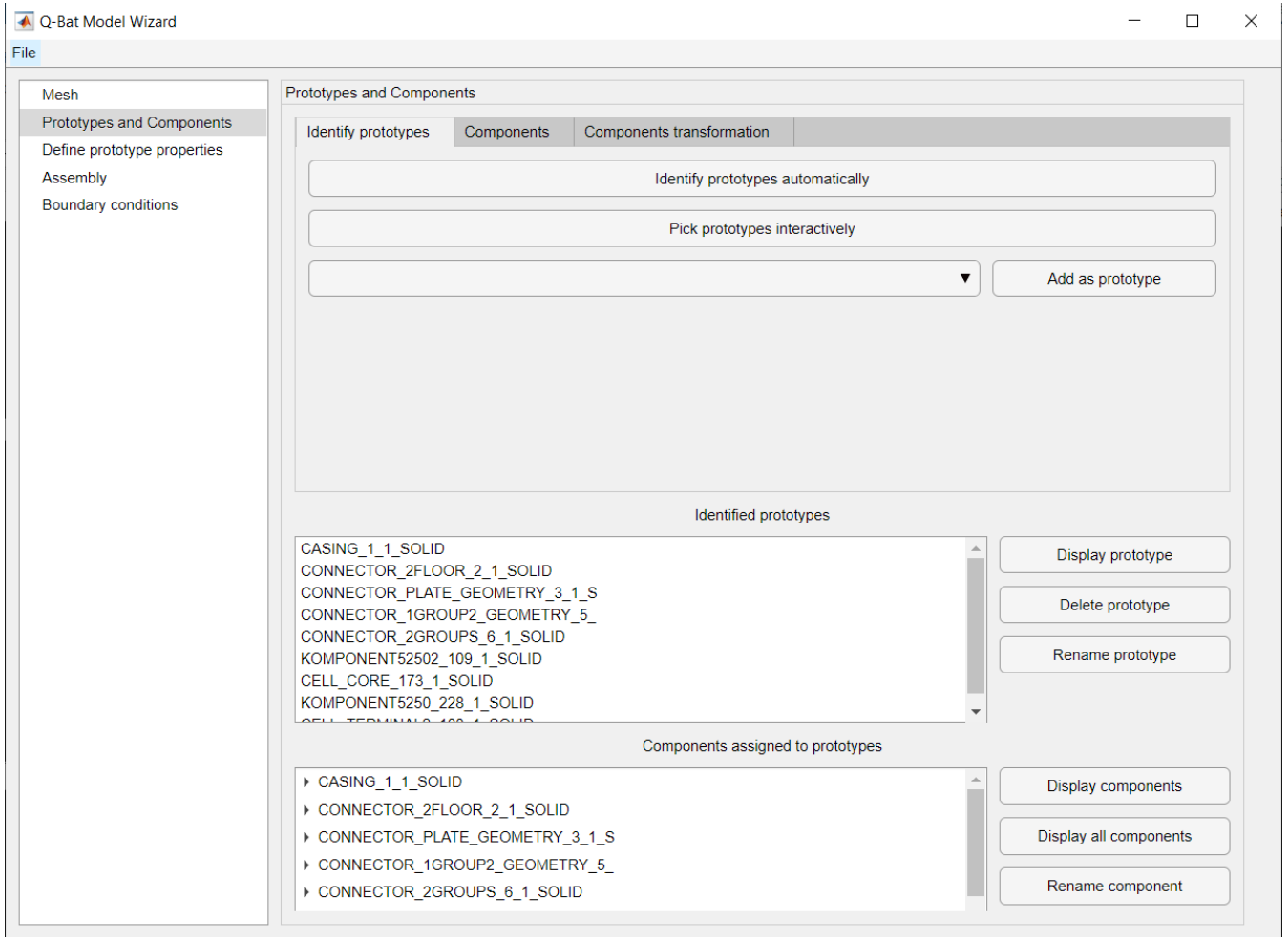
Let's start by enabling Q-Bat GUI. Open the MATLAB command window and use function *wizard.WizardGUI*.

Then, import the mesh file using section **Path to mesh**.

Now you can display the mesh and check mesh information.



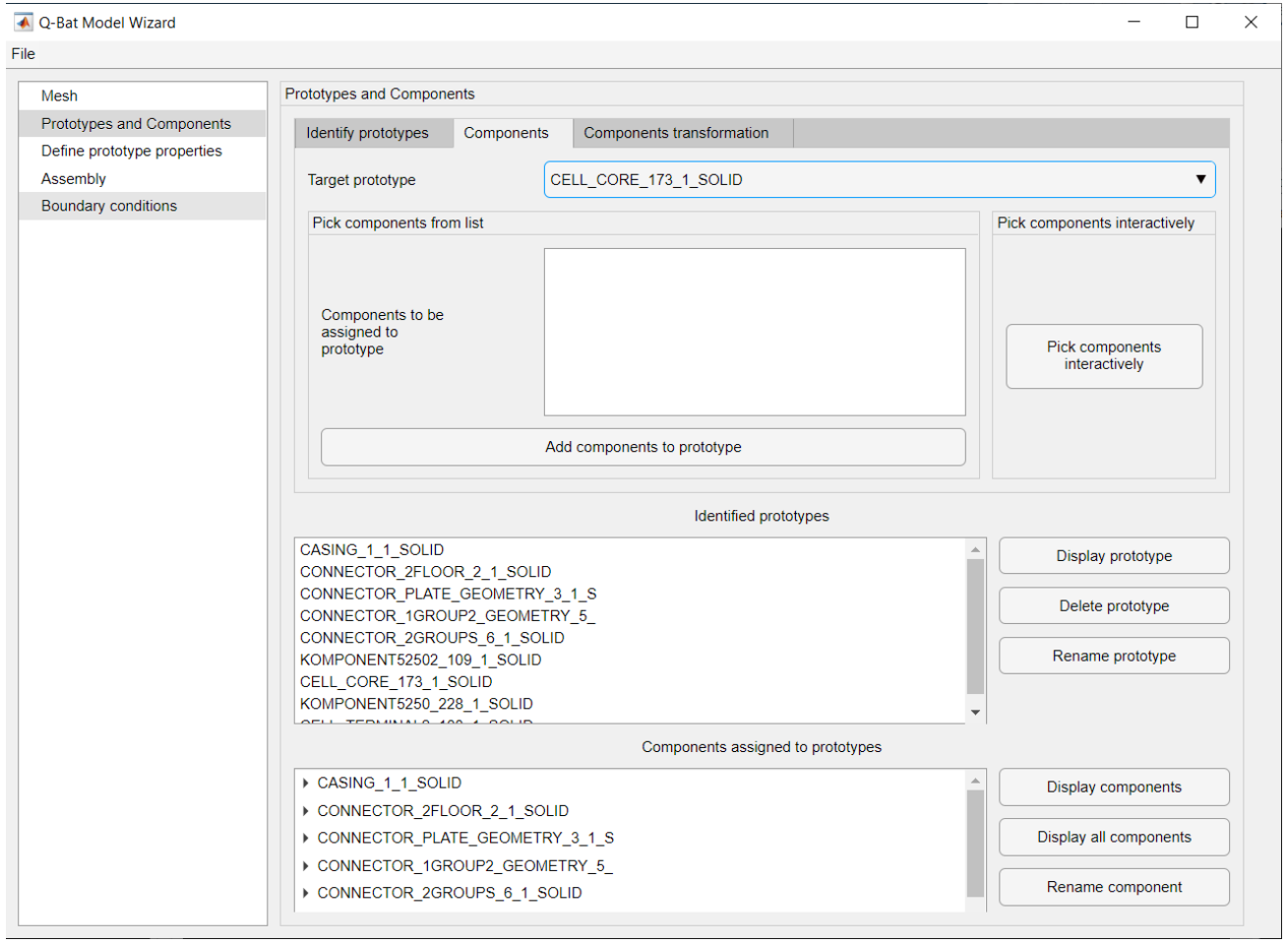
Identify prototypes



Go to **Prototypes and Components** and select section **Identify prototypes.**

You can add prototypes automatically, manually from the list or pick them interactively.

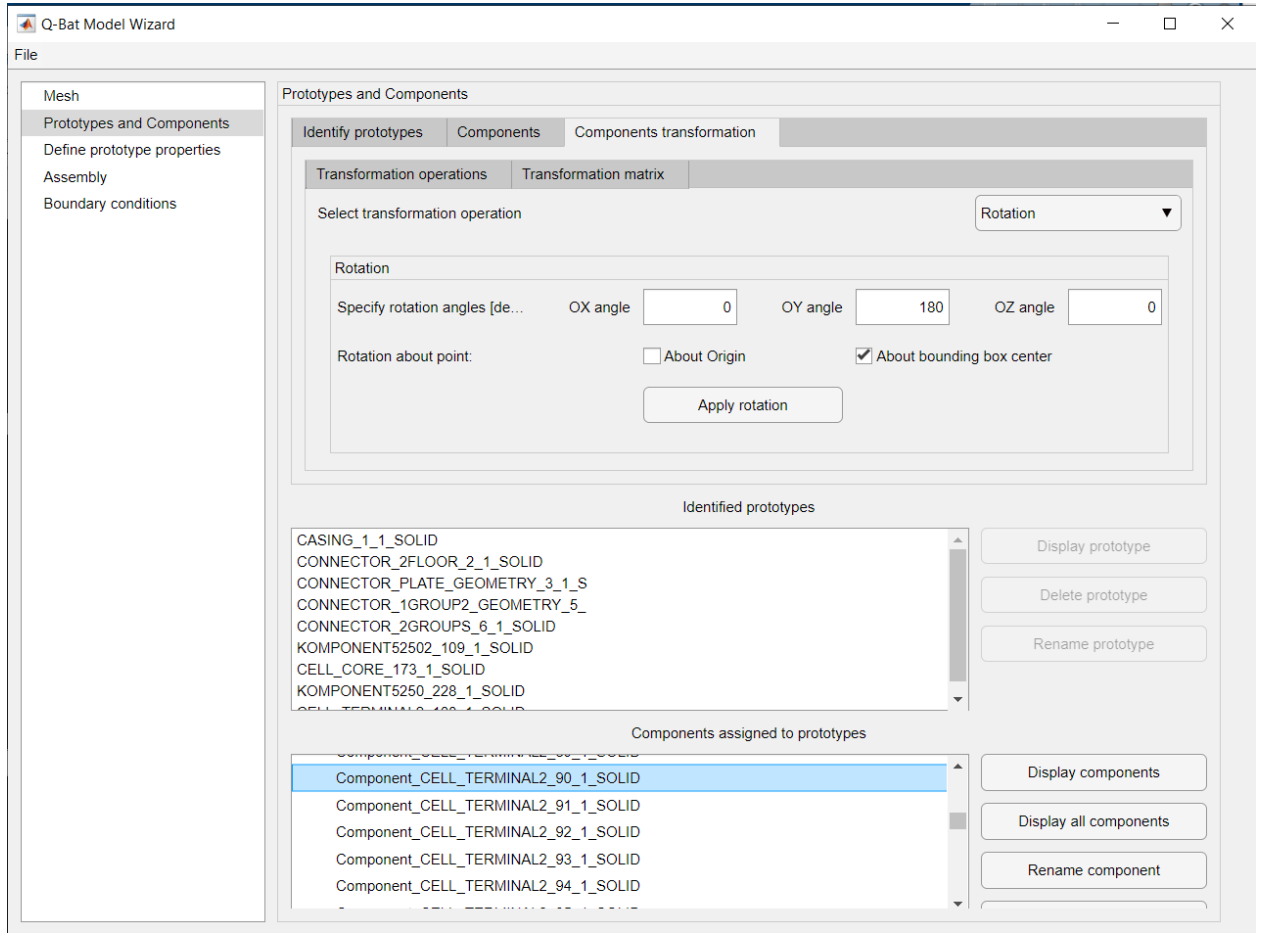
Define components



In section **Components** you can assign components to the proper prototypes.

The selected components can be displayed and renamed.

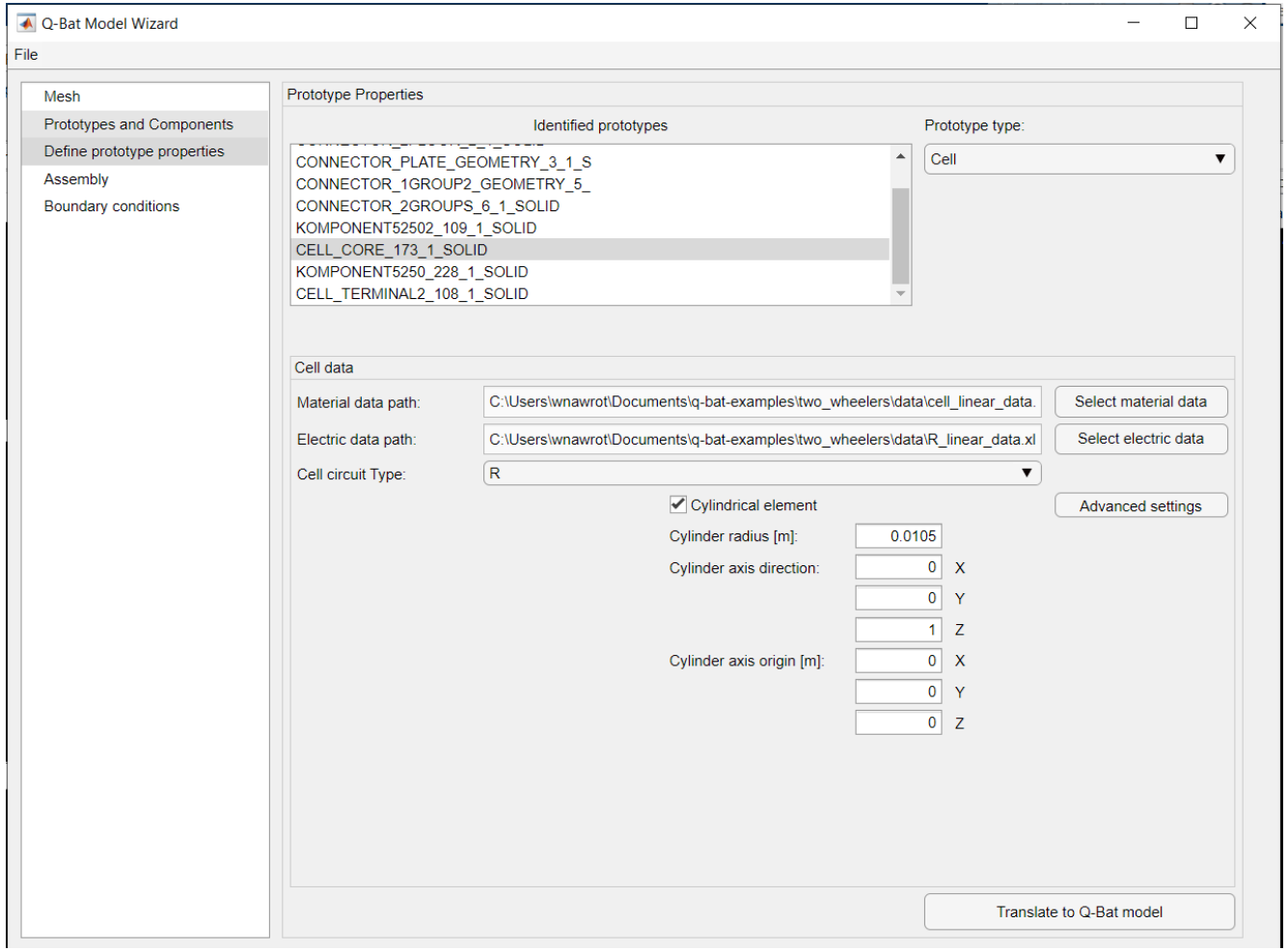
Components transformation



You can do some transformation operation in section **Components transformation**.

Choose Rotation and specify rotation angle at 180° to set terminals in the proper place in the model.

Prototype properties



Go to **Define prototypes properties** to assign prototype type from the list and define material data.

You can select type of cell circuit and import electrical data.

If there are cylindrical element you should mark the checkbox and determine the required data.

When you finish this part,

translate your work

to Q-Bat model

and save the file.

Prototype properties

You need to define the following data (SI units):

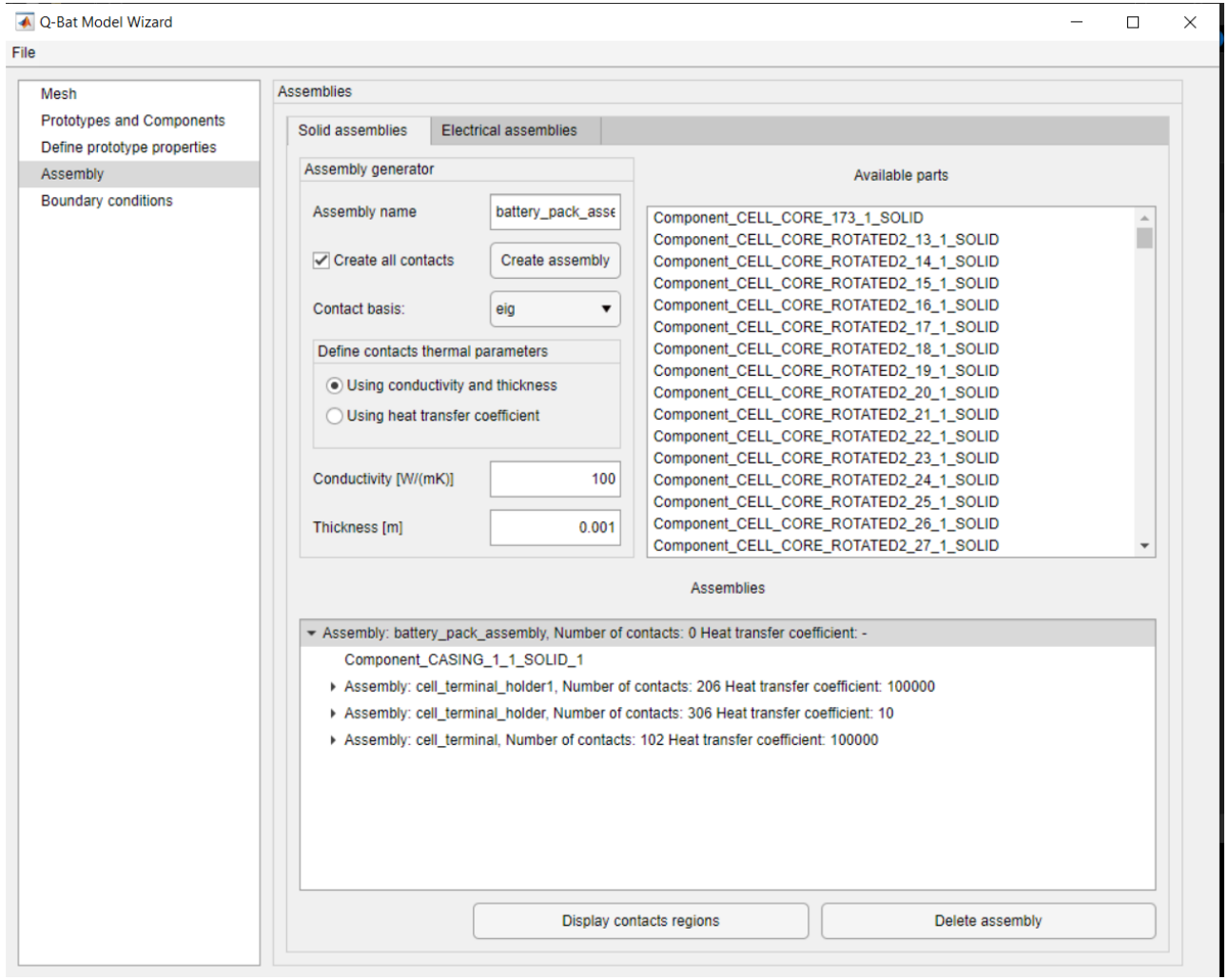
material data for cells and heat components:

- density,
- specific heat capacity,
- thermal conductivity in axis OX, OY and OZ.

electric data for cells:

- for R cricuit type:
  - capacity,
  - voltage,
  - resistance $R_{0.}$

Create assembly

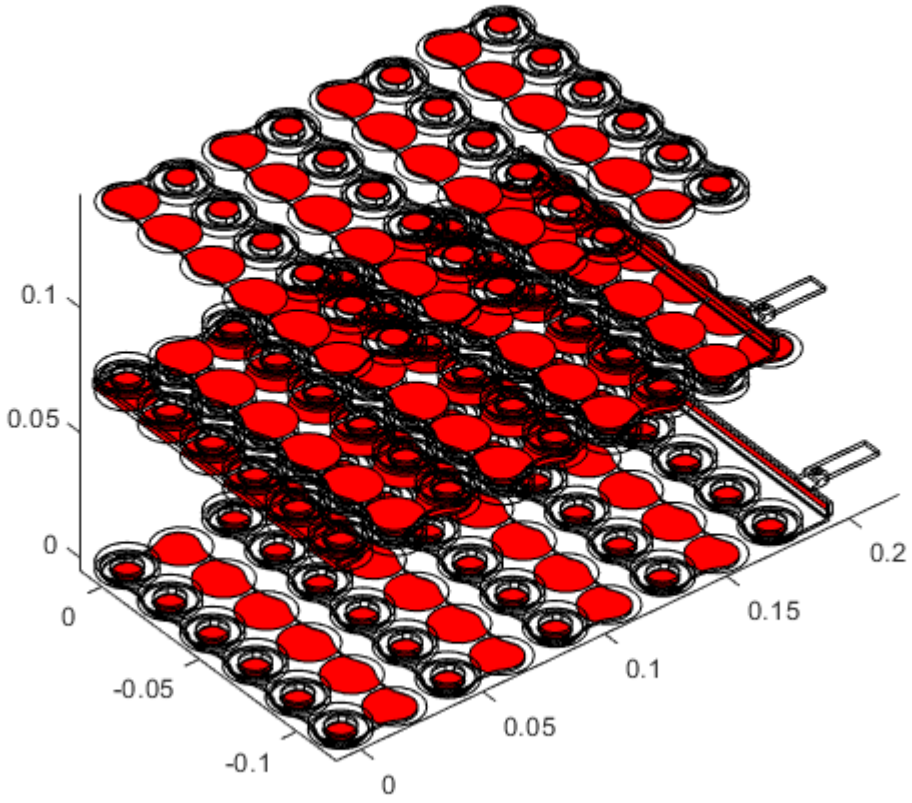


Go to **Assembly** section **Solid assemblies** to aggregate components in the assemblies and create contacts between them.

You can set the parameters like contact thickness and contact conductivity.

QUICKERSIM

Create assembly



You can display created contacts regions.

Switch from GUI to script

To speed up model creation process, we prepared script in MATLAB Editor, which performs next steps.

Open MATLAB Editor and create a new document.

Read the file you create using GUI (before creating assembly).

battery_pack_17S6P.m

```
1 -  clear;
2 -  close all;
3
4    %% Read components
5 -  model = loadModel('battery_pack_17S6P_components.mat');
6
```

QUICKERSIM

Create assembly

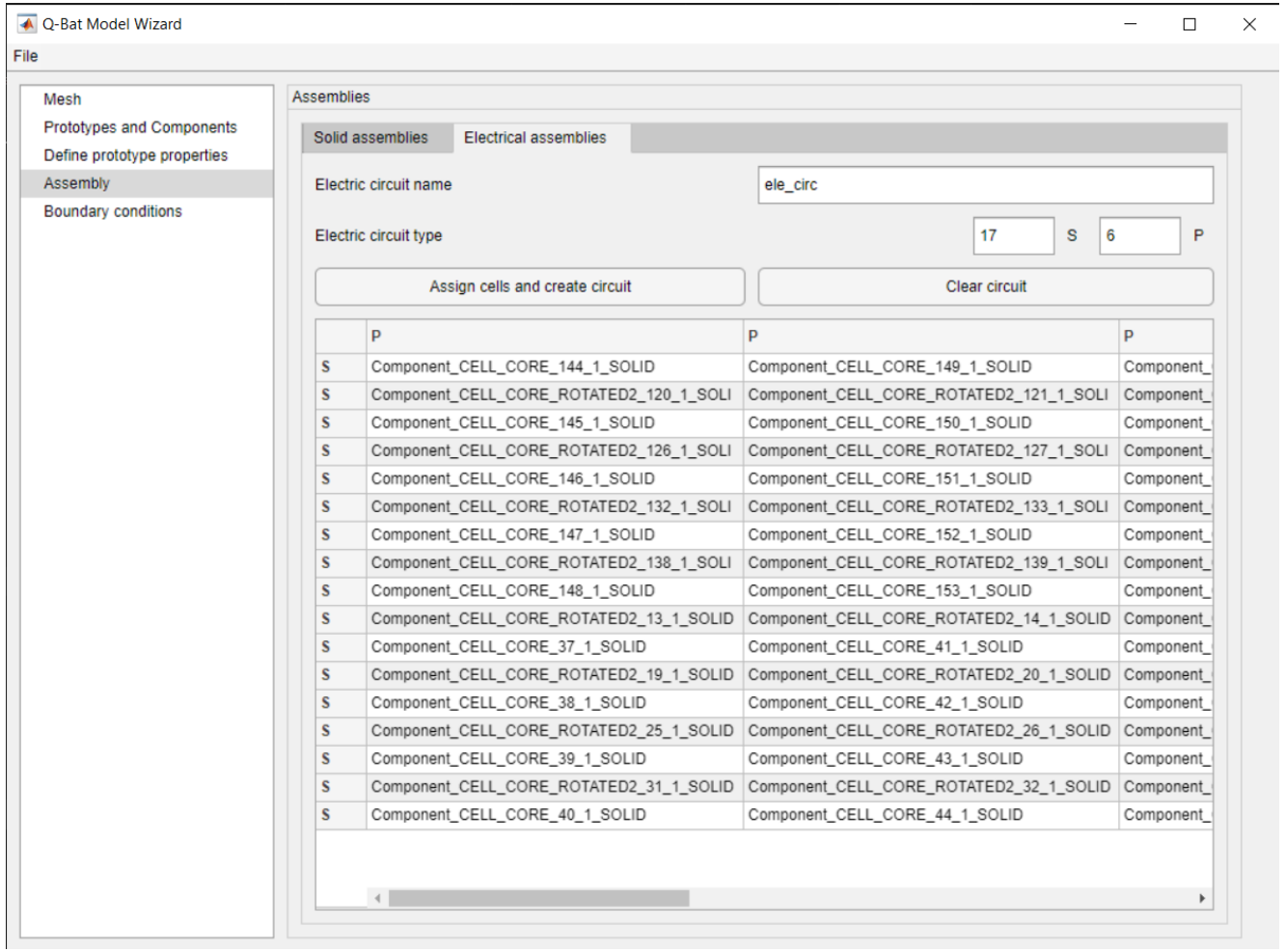
battery_pack_17S6P.m

```
%% Create assembly
tic;

model.config.contact_radius_treshold = 0.01;

model.createAssembly('cell_terminal_assembly', ["Component_CELL.*"], ...
     'create_all_contacts', true,...
     'contact_thickness', 1e-4, 'contact_conductivity', 100, ...
     'number_of_max_contact_modes', 1, 'basis_type', 'eigs',...
     'search_for_cylindrical_contacts', false);

model.createAssembly('cell_terminal_holder_assembly', ...
     ["cell_terminal_assembly","Component_KOMPONENT.*"], ...
     'create_all_contacts', true,...
     'contact_thickness', 1e-3, 'contact_conductivity', 0.01, ...
     'number_of_max_contact_modes', 1, 'basis_type', 'eigs',...
     'search_for_cylindrical_contacts', true);

model.createAssembly('cell_terminal_holder_connector_assembly',...
     ["cell_terminal_holder_assembly","Component_CONNECTOR.*"], ...
     'create_all_contacts', true,...
     'contact_thickness', 1e-3, 'contact_conductivity', 100, ...
     'number_of_max_contact_modes', 1, 'basis_type', 'eigs',...
     'search_for_cylindrical_contacts', false);

model.createAssembly('battery_pack_assembly',...
    ["cell_terminal_holder_connector_assembly","Component_CASING.*"], ...
     'create_all_contacts', false);

assembly_time = toc;
```

Aggregate components in the assemblies and create contacs between tchem.

You can also set contact thickness and contact conductivity.

Electric circuit

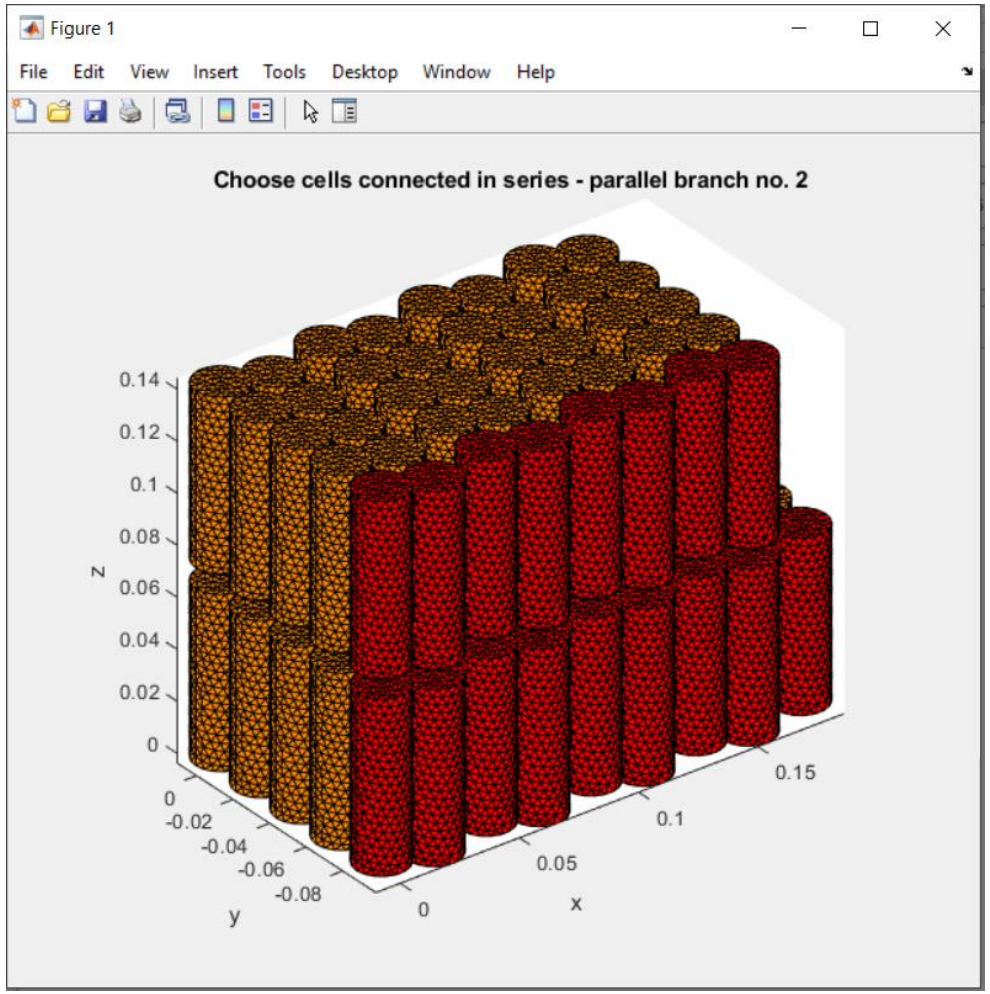


	P	P	P
S	Component_CELL_CORE_144_1_SOLID	Component_CELL_CORE_149_1_SOLID	Component_
S	Component_CELL_CORE_ROTATED2_120_1_SOLI	Component_CELL_CORE_ROTATED2_121_1_SOLI	Component_
S	Component_CELL_CORE_145_1_SOLID	Component_CELL_CORE_150_1_SOLID	Component_
S	Component_CELL_CORE_ROTATED2_126_1_SOLI	Component_CELL_CORE_ROTATED2_127_1_SOLI	Component_
S	Component_CELL_CORE_146_1_SOLID	Component_CELL_CORE_151_1_SOLID	Component_
S	Component_CELL_CORE_ROTATED2_132_1_SOLI	Component_CELL_CORE_ROTATED2_133_1_SOLI	Component_
S	Component_CELL_CORE_147_1_SOLID	Component_CELL_CORE_152_1_SOLID	Component_
S	Component_CELL_CORE_ROTATED2_138_1_SOLI	Component_CELL_CORE_ROTATED2_139_1_SOLI	Component_
S	Component_CELL_CORE_148_1_SOLID	Component_CELL_CORE_153_1_SOLID	Component_
S	Component_CELL_CORE_ROTATED2_13_1_SOLID	Component_CELL_CORE_ROTATED2_14_1_SOLID	Component_
S	Component_CELL_CORE_37_1_SOLID	Component_CELL_CORE_41_1_SOLID	Component_
S	Component_CELL_CORE_ROTATED2_19_1_SOLID	Component_CELL_CORE_ROTATED2_20_1_SOLID	Component_
S	Component_CELL_CORE_38_1_SOLID	Component_CELL_CORE_42_1_SOLID	Component_
S	Component_CELL_CORE_ROTATED2_25_1_SOLID	Component_CELL_CORE_ROTATED2_26_1_SOLID	Component_
S	Component_CELL_CORE_39_1_SOLID	Component_CELL_CORE_43_1_SOLID	Component_
S	Component_CELL_CORE_ROTATED2_31_1_SOLID	Component_CELL_CORE_ROTATED2_32_1_SOLID	Component_
S	Component_CELL_CORE_40_1_SOLID	Component_CELL_CORE_44_1_SOLID	Component_

In section **Electrical assemblies** you can create electric circuit from previously defined cell components.

Set number of cells connected in series and in parallel.

Electric circuit



In the next step, you can assign cells interactively.

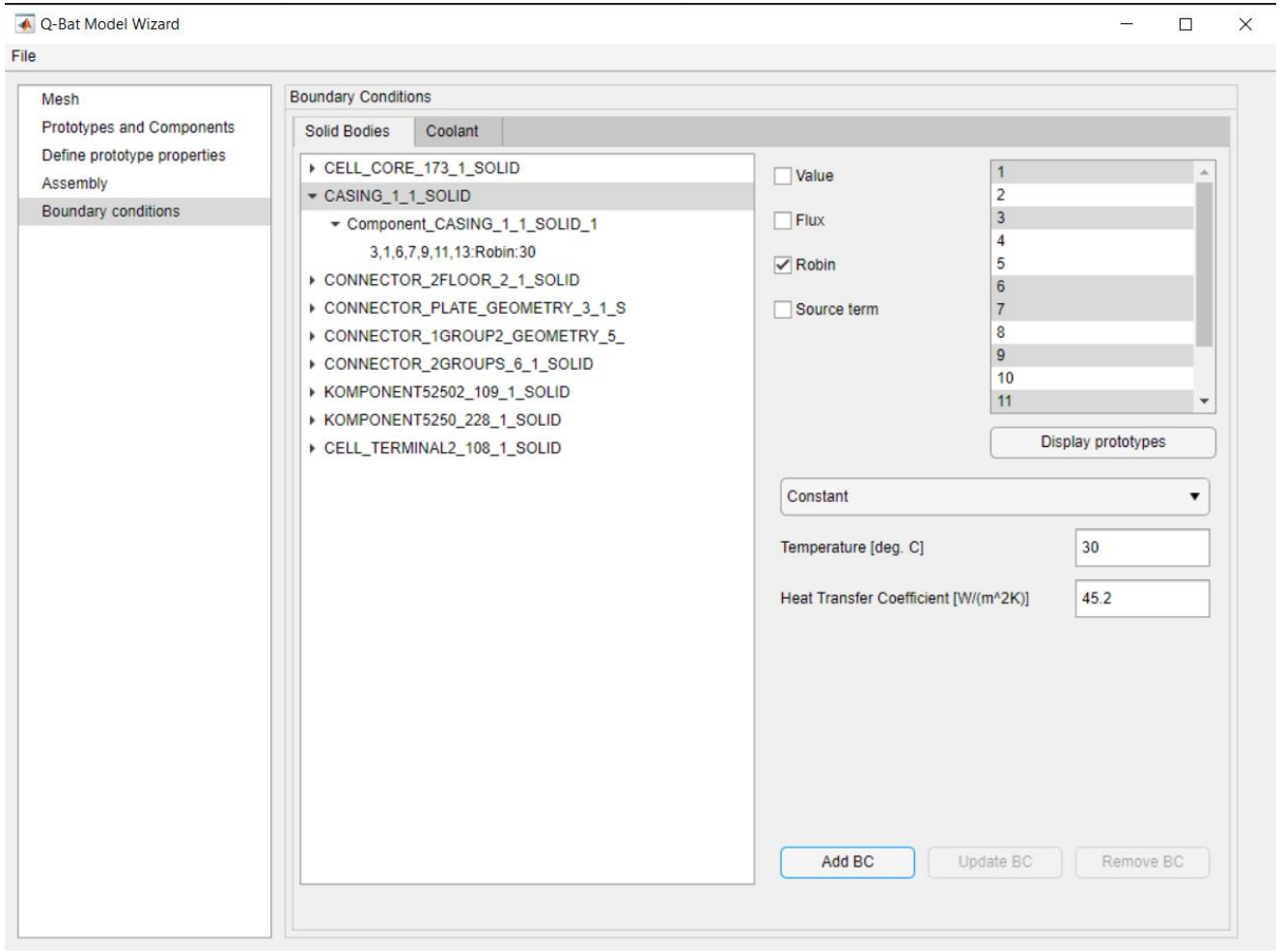
Electric circuit

You can also create electric circuit from previously defined cell components in the script.

battery_pack_17S6P.m

```
157
158     %% Electro circuit
159 -   model.createElectricCircuit("ele_circ", [[cell_1floor_parallel_part2_row1],...
160         [cell_1floor_parallel_part1(1:6)], [cell_1floor_parallel_part2_row2],...
161         [cell_1floor_parallel_part1(7:12)], [cell_1floor_parallel_part2_row3],...
162         [cell_1floor_parallel_part1(13:18)], [cell_1floor_parallel_part2_row4],...
163         [cell_1floor_parallel_part1(19:24)], [cell_1floor_parallel_part2_row5],...
164         ...
165         [cell_2floor_parallel_part2_row1], [cell_2floor_parallel_part1(1:6)],...
166         [cell_2floor_parallel_part2_row2], [cell_2floor_parallel_part1(7:12)], ...
167         [cell_2floor_parallel_part2_row3], [cell_2floor_parallel_part1(13:18)],...
168         [cell_2floor_parallel_part2_row4], [cell_2floor_parallel_part1(19:24)]],...
169         "17s6p");
```

Initial and Boundary Conditions



In **Boundary condition** section **Solid Bodies** you can add the boundary conditions to the compoponents (assign Robin boundary condition to the casing outer boundaries).

You can display prototypes to check the bouandaries' ID.

Save the project
and go to the script.

Initial and Boundary Conditions

battery_pack_17S6P.m

```
170
171     %% IC and BC
172 -   model.addAir("air", 'data', 'data/air_linear_data.xlsx',...
173         'dimensions', [242, 141, 158],...
174         'starting_point', [-0.016, -0.126, -0.0075], ...
175         'air_contact_bodies',...
176         [cell_1floor_parallel_part1', cell_1floor_parallel_part2_row1',...
177         cell_1floor_parallel_part2_row2', cell_1floor_parallel_part2_row3', ...
178         cell_1floor_parallel_part2_row4', cell_1floor_parallel_part2_row5',...
179         cell_2floor_parallel_part1', cell_2floor_parallel_part2_row1',...
180         cell_2floor_parallel_part2_row2', cell_2floor_parallel_part2_row3', ...
181         cell_2floor_parallel_part2_row4', "Component_CASING.*"], ...
182         'air_contact_surf_ids',...
183         [repmat({[1]}, 1, 102), {[2, 4, 5, 8, 10, 11, 13]}], ...
184         'air_contact_alpha', 5, 'n_partitions', [10, 10, 10]);
185
186 -   model.setIC('.*', 'T', 30);
187
188 -   model.addBC("Component_CASING.*", [1, 3, 6, 7, 9, 11, 13],...
189         'type', 'Robin', 'alpha', 45.2, 'T_inf', 30);
190
```

In the script, add to the model the air, which will be modeled as a finite volume body.

Set the initial conditions (temperature of the battery pack) and the boundary conditions (assign Robin boundary condition to the casing outer boundaries).

Model preparation and run

battery_pack_17S6P.m

```
%% Model Preparation and run
model.saveModel("battery_pack_17S6P_before_prep");

tic;
model.flag_is_parallel = false;
model_struct = model.prepare();
prep_time = toc;

model.saveModel("battery_pack_17S6P_after_prep");

current = readtable("data/current.xlsx");
current.Properties.VariableNames = {'t','current'};
model_struct.ele_circ_electro = current;

tic;
model.run(10, 100, 'electroSubsteps',100, 'signals_data', model_struct,...
    'solver', 'Direct')
run_time = toc;
```

If you have parallel toolbox installed start it.

Prepare model.

Define the current profile from the spreadsheet.

Specify the time step, next the number of time steps, the number of time steps of the electrical calculation and run the simulation.

Post-processing and export

Using different functions, you can plot:

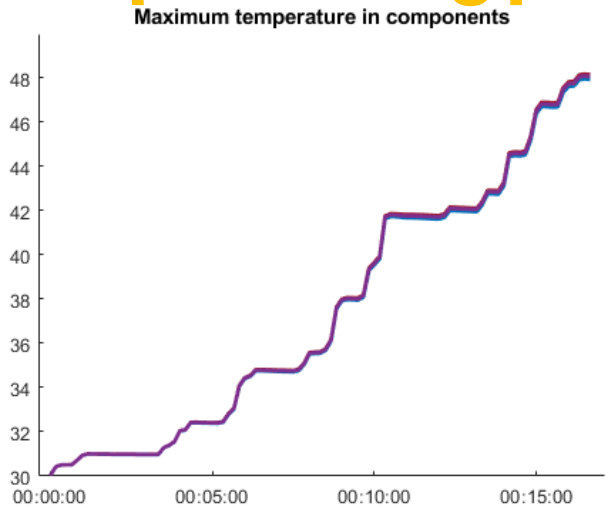
- max, mean and min temperatures over time in chosen bodies,
- state of charge over time,
- circuit voltage and current over time.

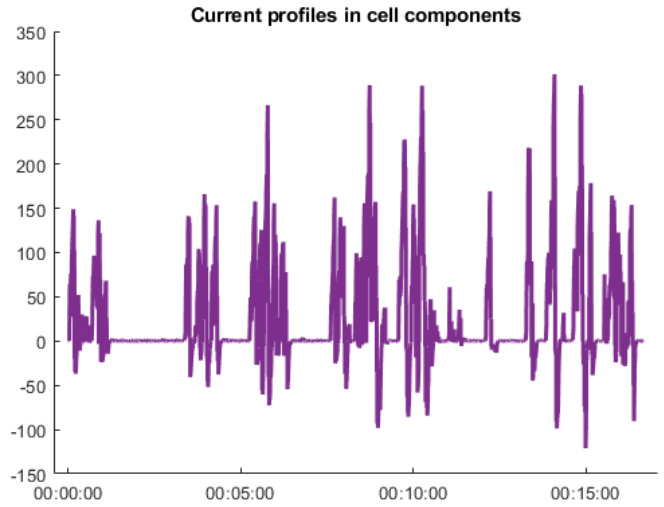
You can also plot the solution for the cells and the whole battery module for a selected time step.

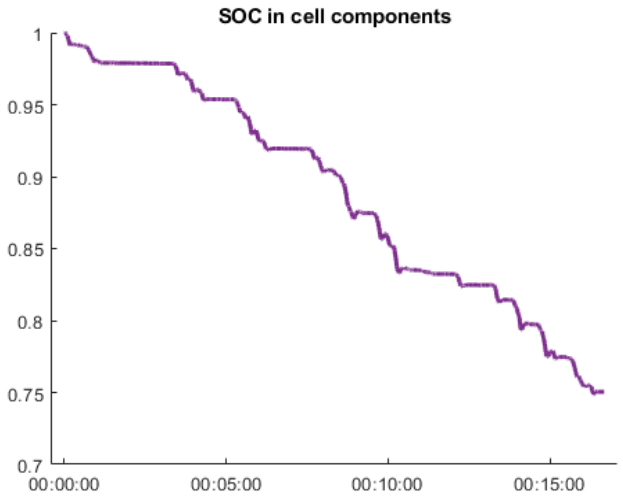
Lastly, you can export solution to extensions which are compatible with other software such as Paraview.

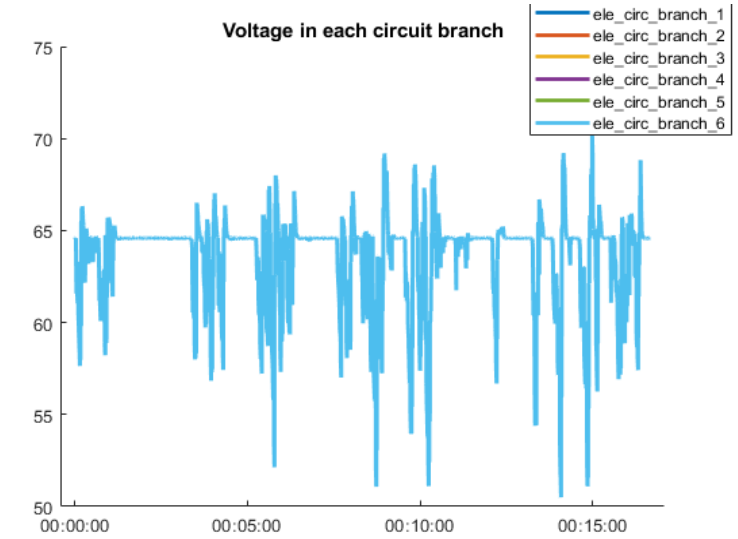
```
%% Post
% Elctro
model.plotCircuitVoltage("ele_circ")
model.plotSOCOverTime()
model.plotCurrentOverTime()

%% Thermal
model.plotMaxTempOverTime("Component_CELL_CORE.*")
model.plotMinTempOverTime("Component_CELL_CORE.*")
model.plotMeanTempOverTime("Component_CELL_CORE.*")

model.plotSolution("cell_terminal_holder_connector_assembly", 3)
model.plotSolution("cell_terminal_holder_connector_assembly", 50)
model.plotSolution("cell_terminal_holder_connector_assembly", 100)

model.exportSolutionToCGNS("battery_pack_assembly", "battery_pack_17S6P_result.cgns");
model.saveModel("battery_pack_17S6P_results");
```

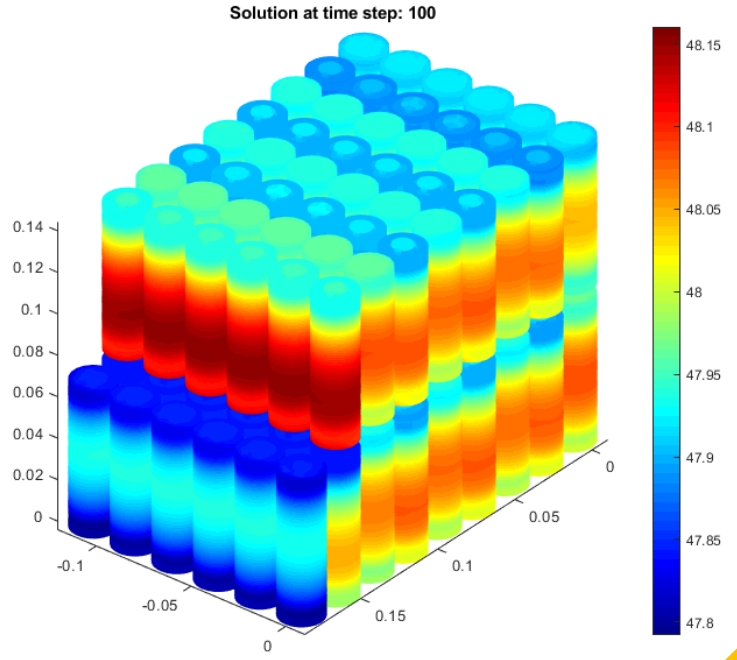
Post-processing plot











QUICKERSIM

Summary

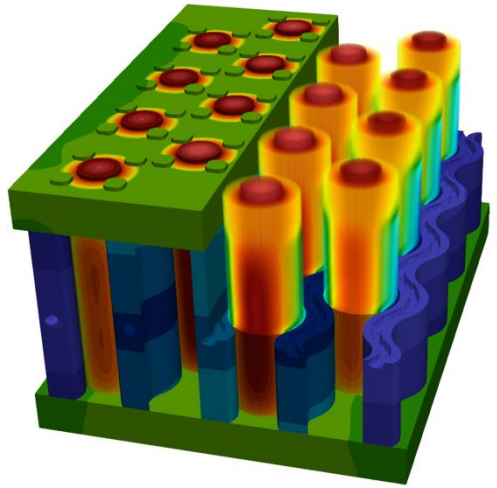
The model consists:

- 9 prototypes,
- 229 components,
- 815 581 mesh elements,
- 614 contacts regions.

The simulation time is only 15 minutes.

Learn more

- Q-Bat is a MATLAB-based product for real-time battery thermal simulation in 3D with CFD-like accuracy. Its main features are:
  - Near real-time execution
  - Accurate 3D data of battery temperature distribution
  - The capability of exporting the model to the Simulink
  - Fast model definition via dedicated GUI and TUI.



- To learn more:
  - QuickerSim https://emobility.quickersim.com/
  - Q-Bat product page https://www.mathworks.com/products/connections/product_detail/quickersim-q-bat.html

- For a free Q-Bat lite license, visit QuickerSim licensing website https://licensing.quickersim.com/

- To get **full version trial** write to q-bat@quickersim.com