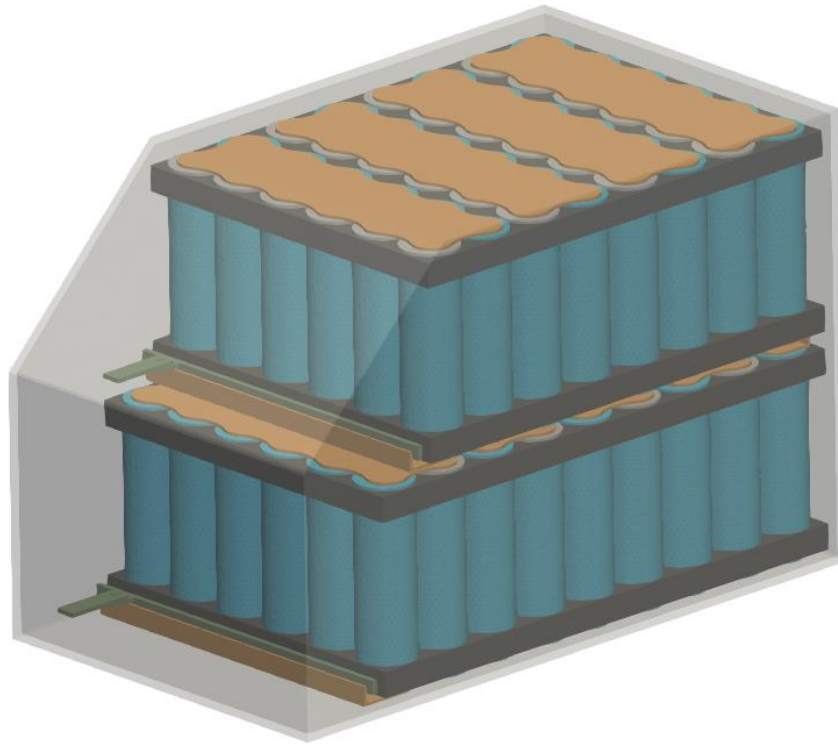


# Tutorial e-motorbike's battery pack

# 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.

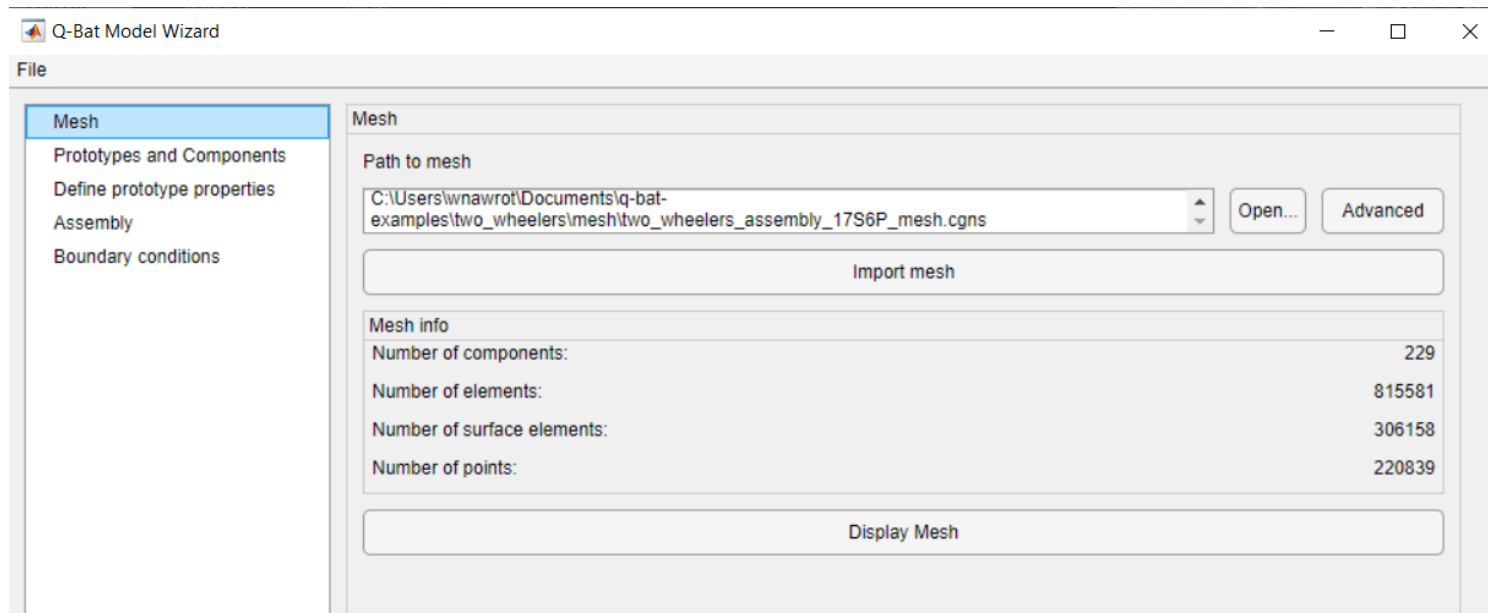
The geometry is presented below.

# Import mesh

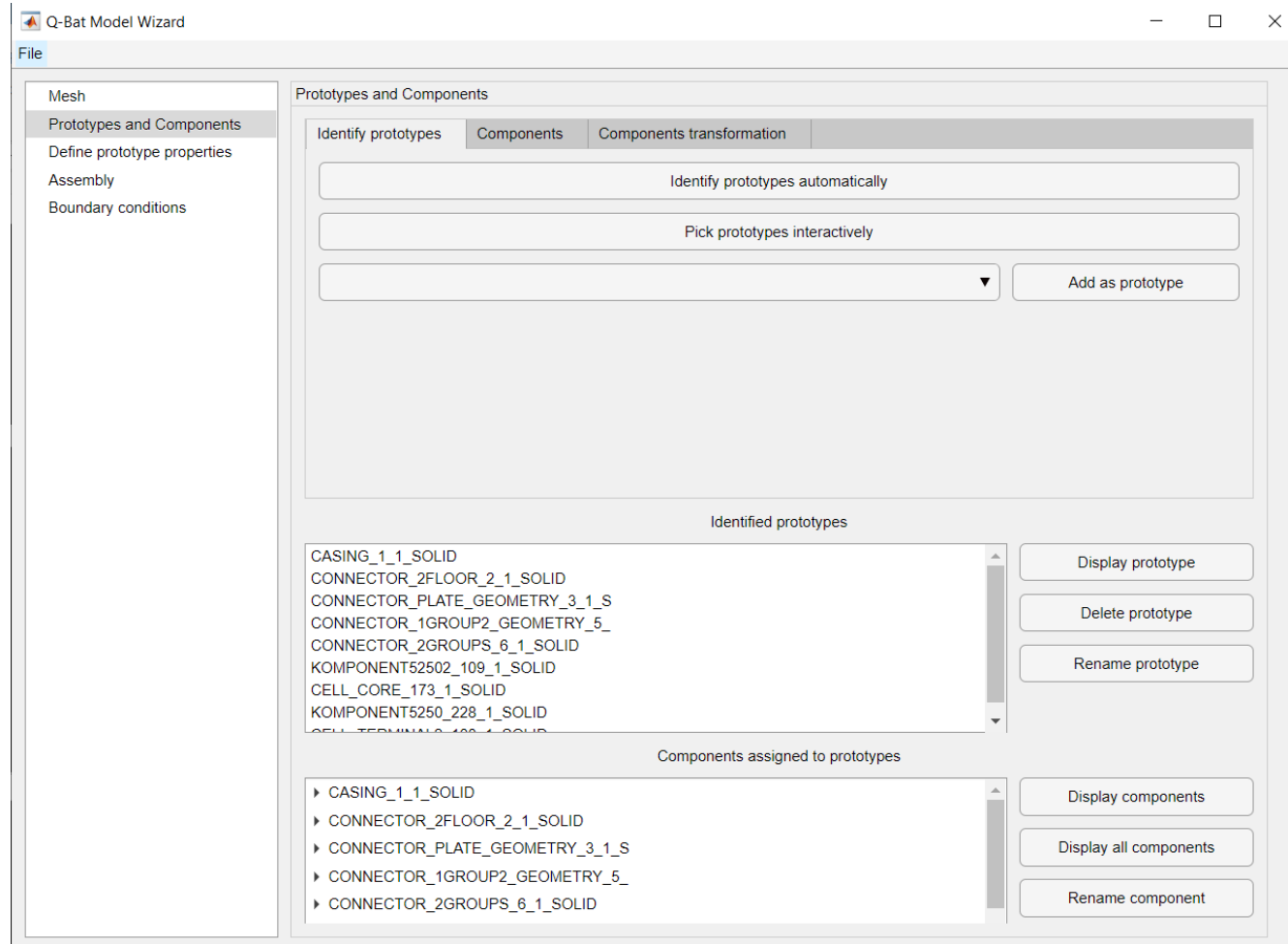
Let's start by enabling Q-Bat GUI. Open the MATLAB command window and use function `wizard.WizardGUI`.

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

Now you can display the mesh and check mesh information.



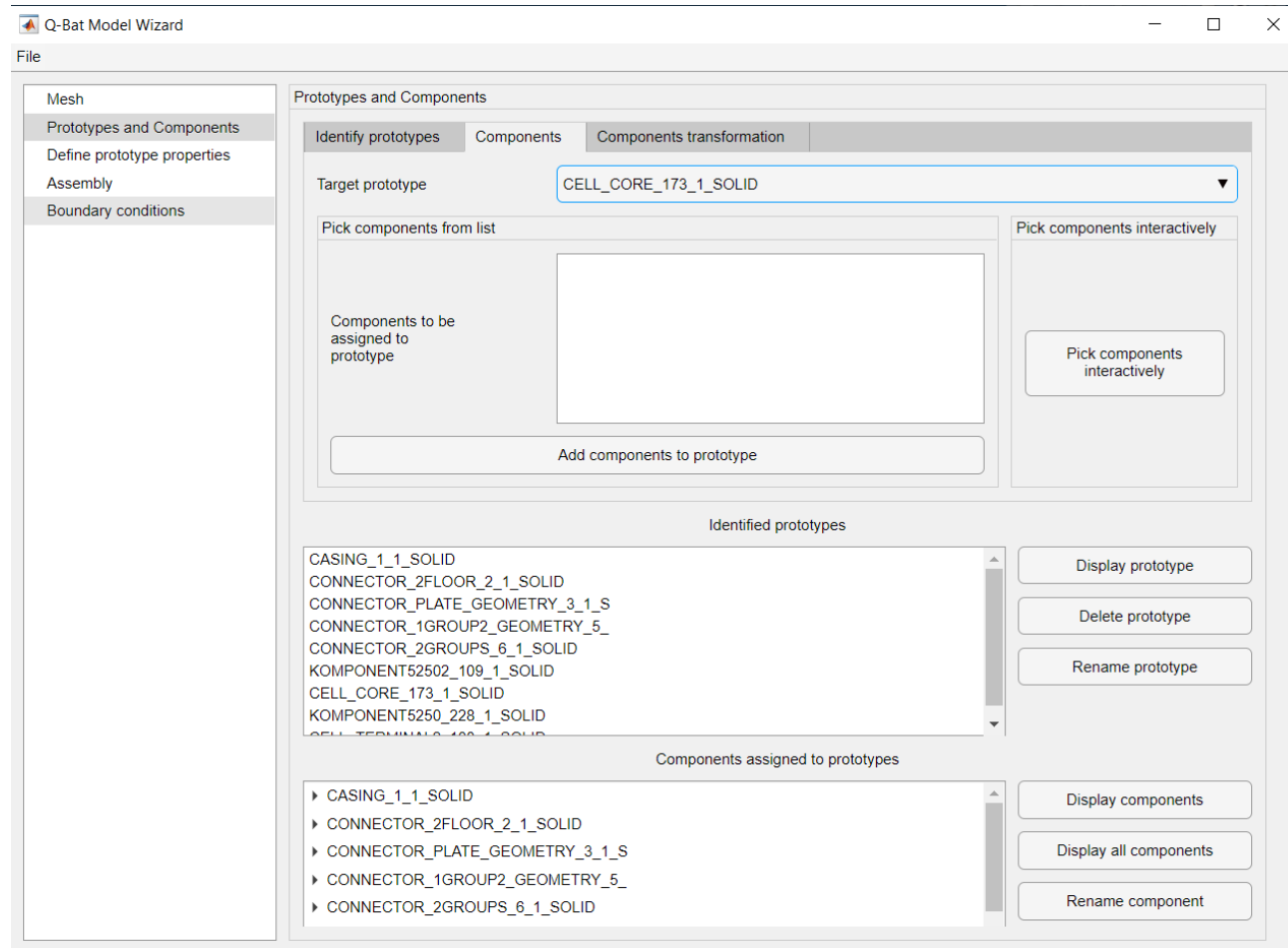
# Identify prototypes



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

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

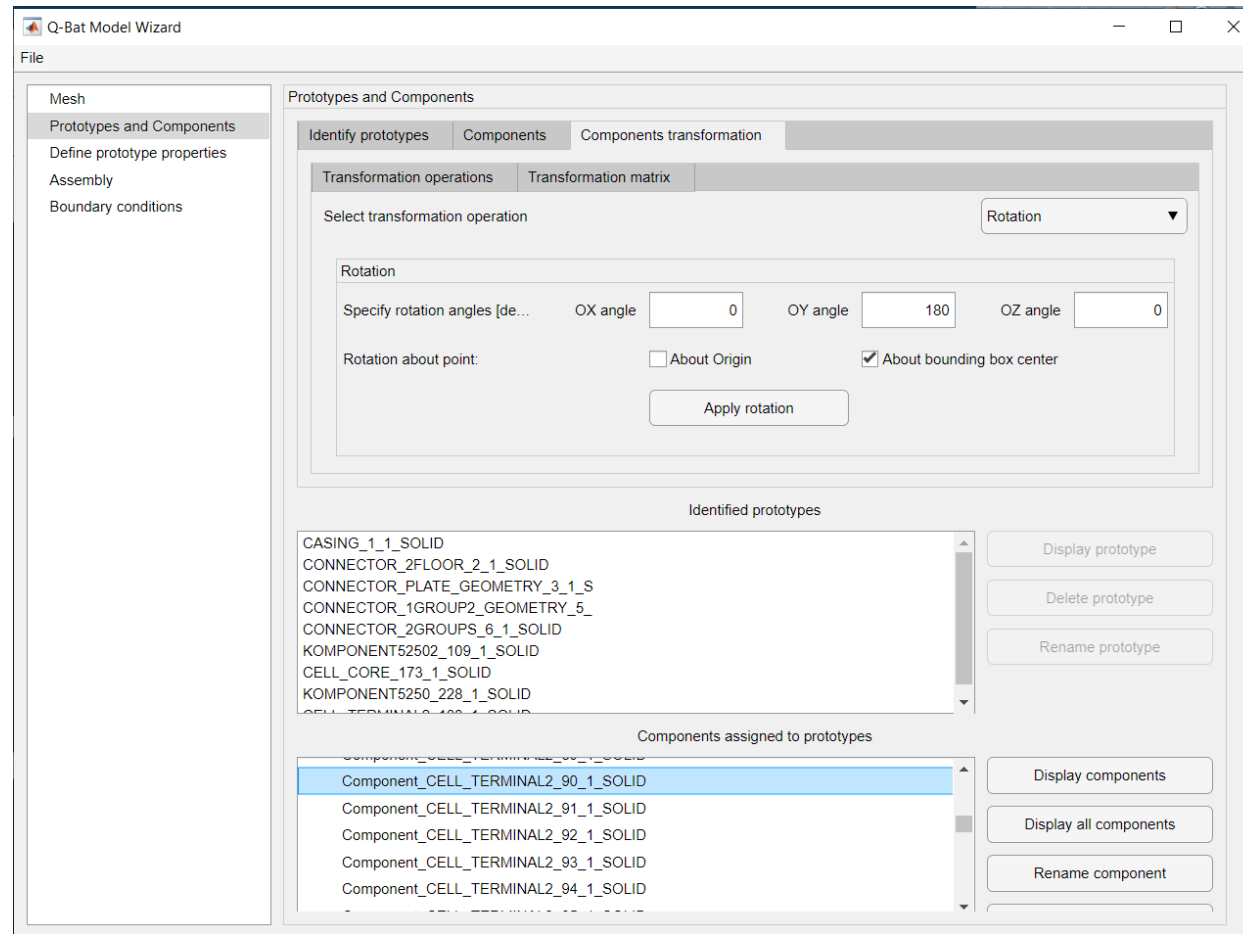
# Define components



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

The selected components can be displayed and renamed.

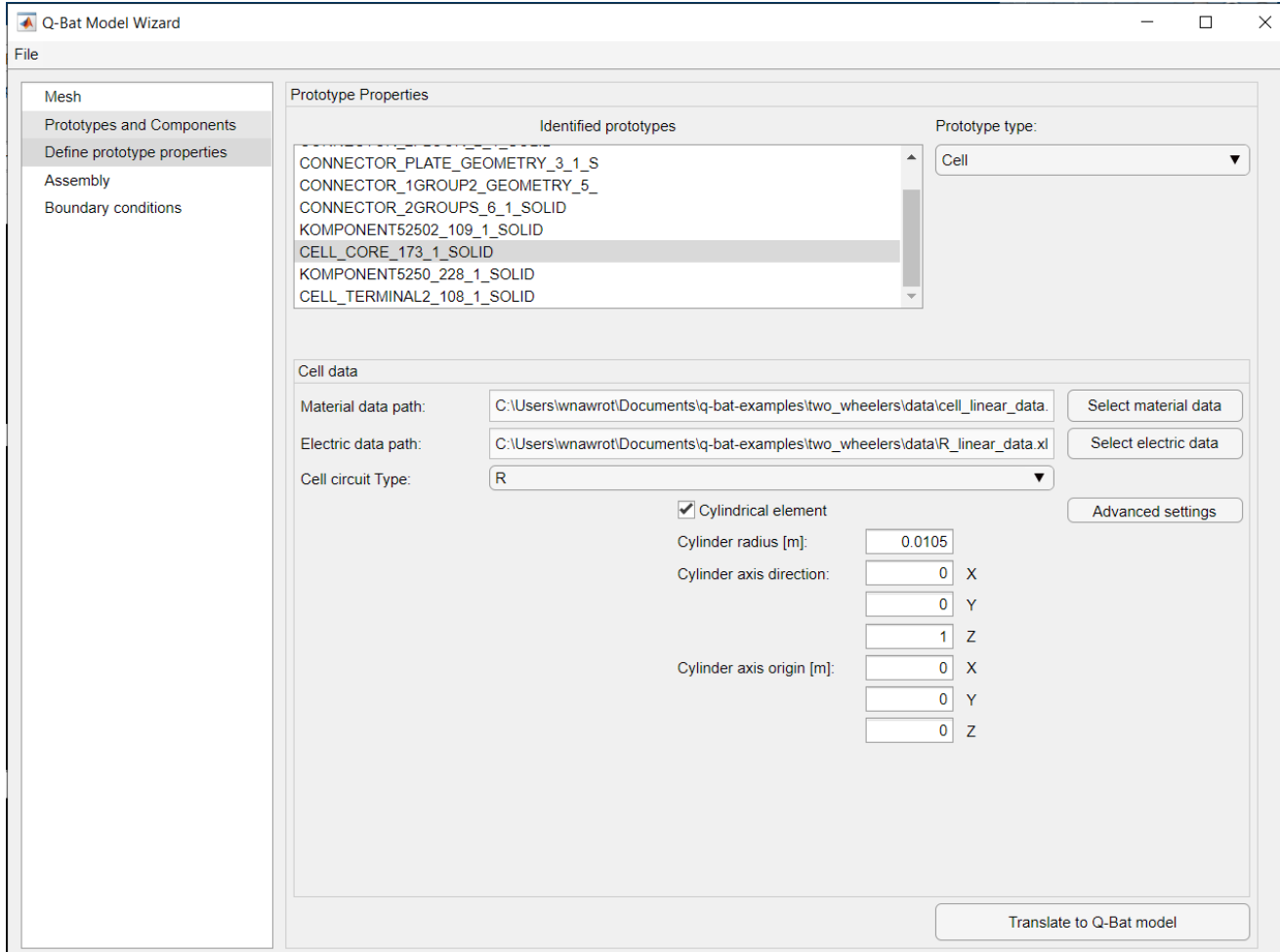
# Components transformation



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



The screenshot shows the 'Q-Bat Model Wizard' window with the 'Prototype Properties' tab selected. The left sidebar contains a tree view with 'Mesh', 'Prototypes and Components', 'Define prototype properties' (selected), 'Assembly', and 'Boundary conditions'. The main area is divided into two sections: 'Identified prototypes' and 'Cell data'.

**Identified prototypes:** A list box containing the following items: CONNECTOR\_PLATE\_GEOMETRY\_3\_1\_S, CONNECTOR\_1GROUP2\_GEOMETRY\_5\_, CONNECTOR\_2GROUPS\_6\_1\_SOLID, KOMPONENT52502\_109\_1\_SOLID, CELL\_CORE\_173\_1\_SOLID (highlighted), KOMPONENT5250\_228\_1\_SOLID, and CELL\_TERMINAL2\_108\_1\_SOLID. To the right of this list is a 'Prototype type:' dropdown menu set to 'Cell'.

**Cell data:** This section contains several input fields and buttons:

- Material data path:** C:\Users\wnawrot\Documents\q-bat-examples\two\_wheelers\data\cell\_linear\_data. [Select material data]
- Electric data path:** C:\Users\wnawrot\Documents\q-bat-examples\two\_wheelers\data\R\_linear\_data.xl [Select electric data]
- Cell circuit Type:** R [Advanced settings]
- ☒ Cylindrical element
- Cylinder radius [m]:** 0.0105
- Cylinder axis direction:** 0 X, 0 Y, 1 Z
- Cylinder axis origin [m]:** 0 X, 0 Y, 0 Z

At the bottom right of the dialog is a button labeled 'Translate to Q-Bat model'.

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:

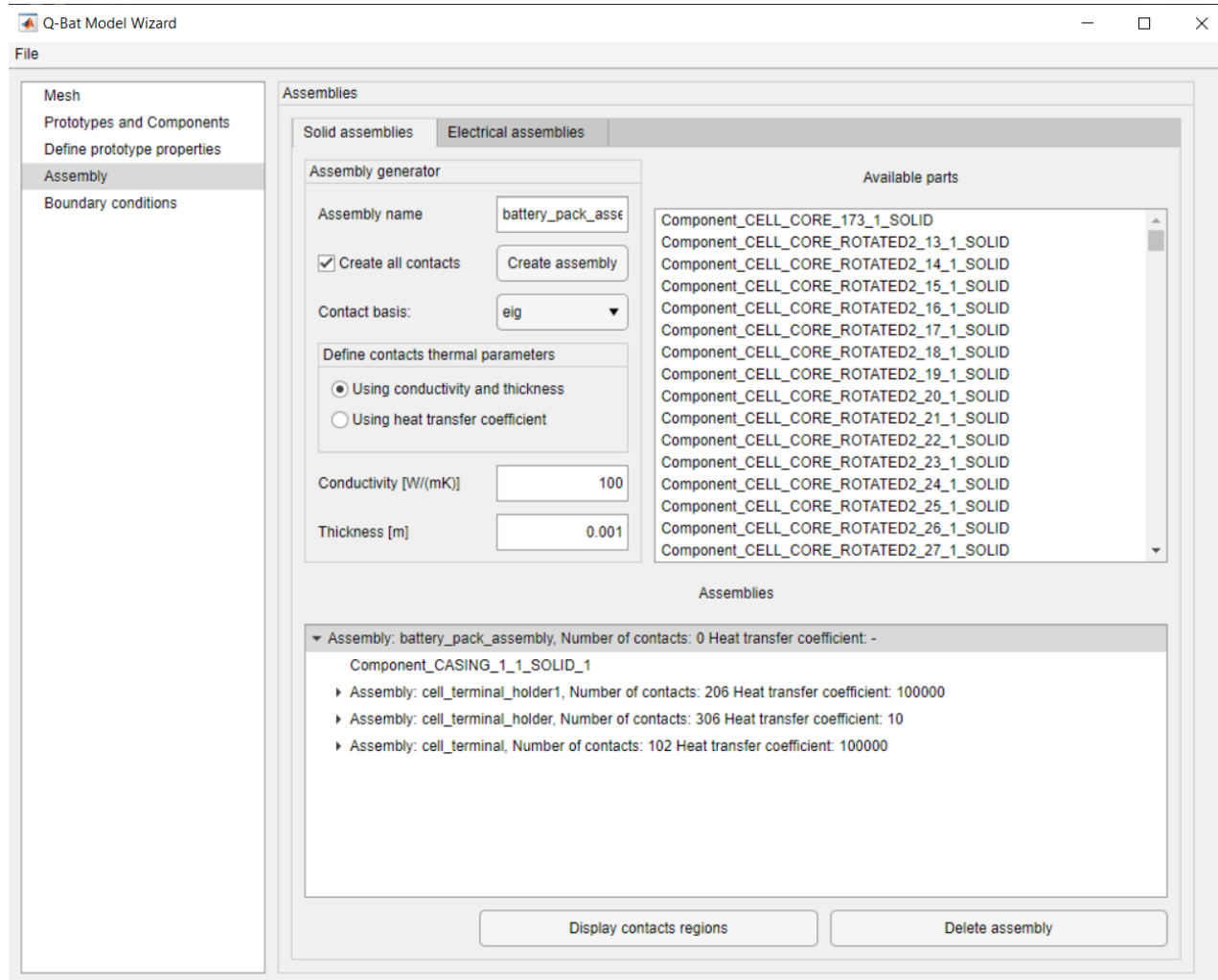
- density,
- specific heat capacity,
- thermal conductivity in axis OX, OY and OZ.

electric data for cells:

- for R cricuit type:
  - capacity,
  - voltage,
  - resistance  $R_0$ .



# Create assembly

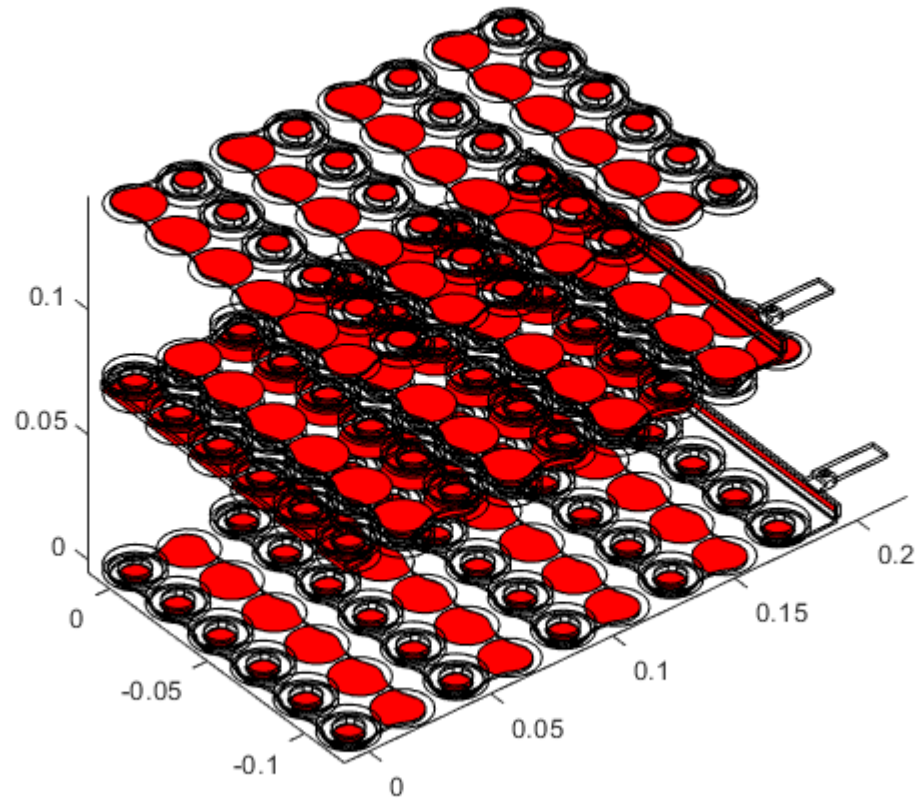


The screenshot shows the 'Q-Bat Model Wizard' window. On the left is a sidebar with a 'File' menu and a list of steps: 'Mesh', 'Prototypes and Components', 'Define prototype properties', 'Assembly' (which is highlighted), and 'Boundary conditions'. The main area is titled 'Assemblies' and has two tabs: 'Solid assemblies' and 'Electrical assemblies'. The 'Solid assemblies' tab is active. It contains an 'Assembly generator' section with the following fields: 'Assembly name' (set to 'battery\_pack\_asse'), a checked 'Create all contacts' checkbox, a 'Create assembly' button, a 'Contact basis' dropdown (set to 'eig'), and a 'Define contacts thermal parameters' section with two radio buttons: 'Using conductivity and thickness' (selected) and 'Using heat transfer coefficient'. Below these are input fields for 'Conductivity [W/(mK)]' (set to 100) and 'Thickness [m]' (set to 0.001). To the right of the generator is a list of 'Available parts' including 'Component\_CELL\_CORE\_173\_1\_SOLID' through 'Component\_CELL\_CORE\_ROTATED2\_27\_1\_SOLID'. At the bottom, there is a section titled 'Assemblies' showing a tree view of the current assembly structure: 'Assembly: battery\_pack\_assembly, Number of contacts: 0 Heat transfer coefficient: -' which contains 'Component\_CASING\_1\_1\_SOLID\_1', which in turn contains three sub-assemblies: 'Assembly: cell\_terminal\_holder1, Number of contacts: 206 Heat transfer coefficient: 100000', 'Assembly: cell\_terminal\_holder, Number of contacts: 306 Heat transfer coefficient: 10', and 'Assembly: cell\_terminal, Number of contacts: 102 Heat transfer coefficient: 100000'. At the very bottom are two buttons: 'Display contacts regions' and 'Delete 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.

## Create assembly



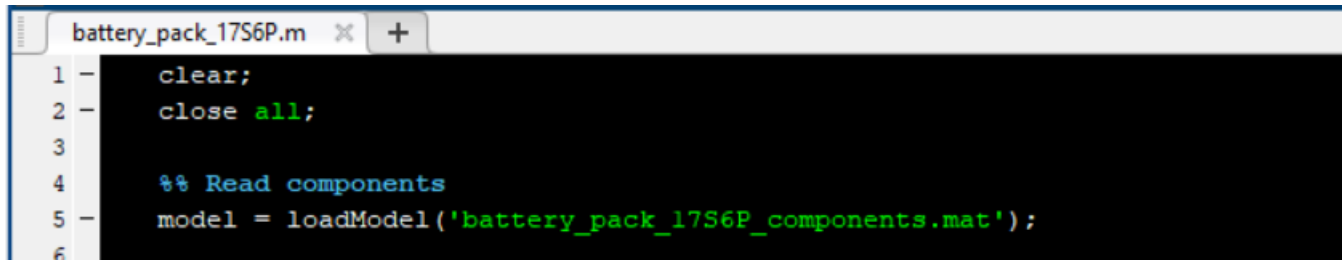
You can display created contacts regions.

## 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).



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

# Create assembly

```

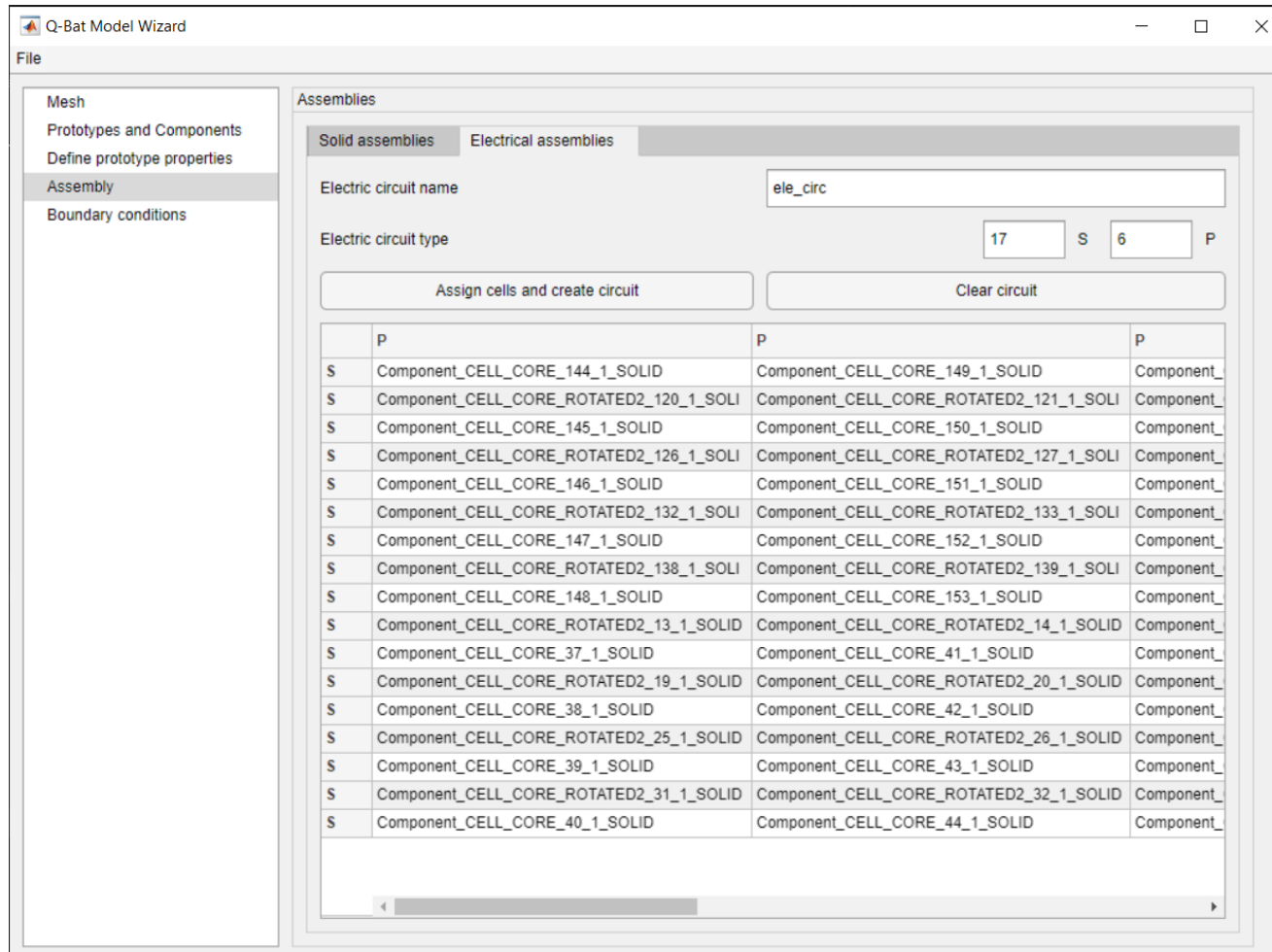
battery_pack_17S6P.m  x  +
7  %% Create assembly
8  -  tic;
9
10 -  model.config.contact_radius_treshold = 0.01;
11
12 -  model.createAssembly('cell_terminal_assembly', ["Component_CELL.*"], ...
13      'create_all_contacts', true,...
14      'contact_thickness', 1e-4, 'contact_conductivity', 100, ...
15      'number_of_max_contact_modes', 1, 'basis_type', 'eigs',...
16      'search_for_cylindrical_contacts', false);
17
18 -  model.createAssembly('cell_terminal_holder_assembly', ...
19      ["cell_terminal_assembly","Component_KOMPONENT.*"], ...
20      'create_all_contacts', true,...
21      'contact_thickness', 1e-3, 'contact_conductivity', 0.01, ...
22      'number_of_max_contact_modes', 1, 'basis_type', 'eigs',...
23      'search_for_cylindrical_contacts', true);
24
25 -  model.createAssembly('cell_terminal_holder_connector_assembly',...
26      ["cell_terminal_holder_assembly","Component_CONNECTOR.*"], ...
27      'create_all_contacts', true,...
28      'contact_thickness', 1e-3, 'contact_conductivity', 100, ...
29      'number_of_max_contact_modes', 1, 'basis_type', 'eigs',...
30      'search_for_cylindrical_contacts', false);
31
32 -  model.createAssembly('battery_pack_assembly',...
33      ["cell_terminal_holder_connector_assembly","Component_CASING.*"], ...
34      'create_all_contacts', false);
35
36 -  assembly_time = toc;

```

Aggregate components in the assemblies and create contacts between them.

You can also set contact thickness and contact conductivity.

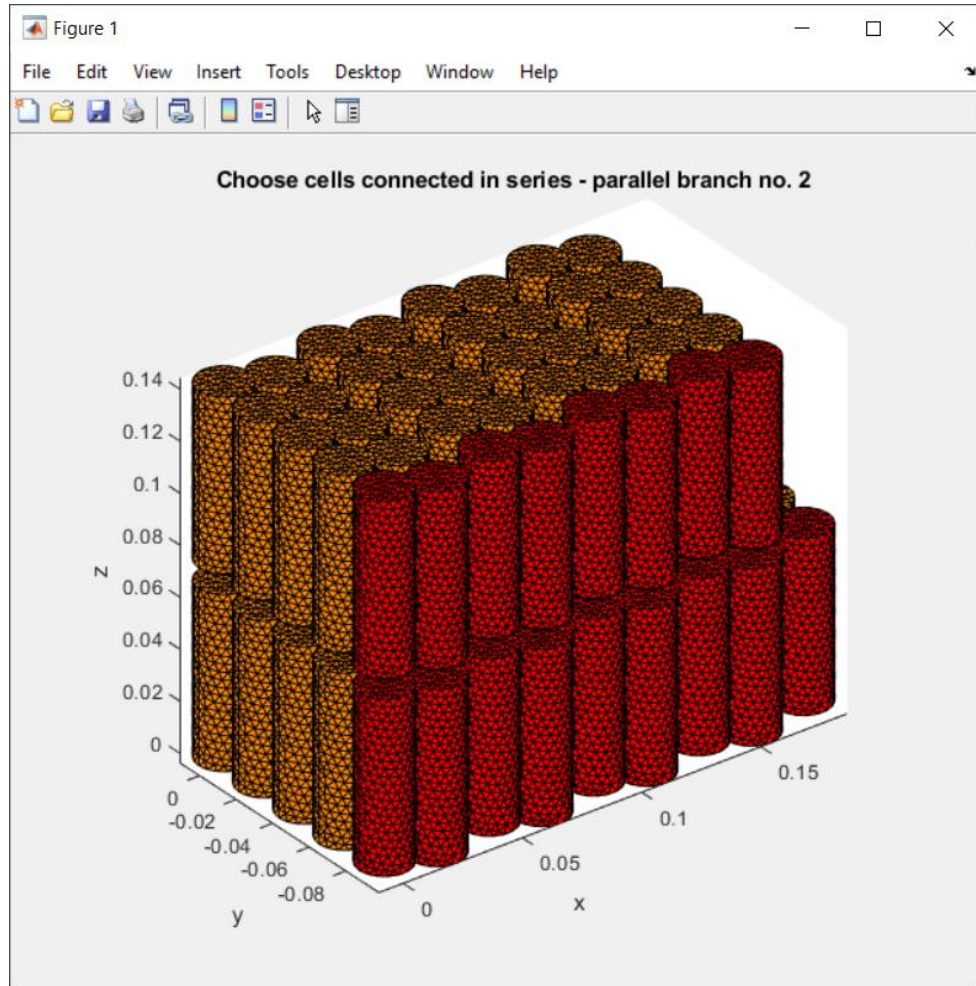
# Electric circuit



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

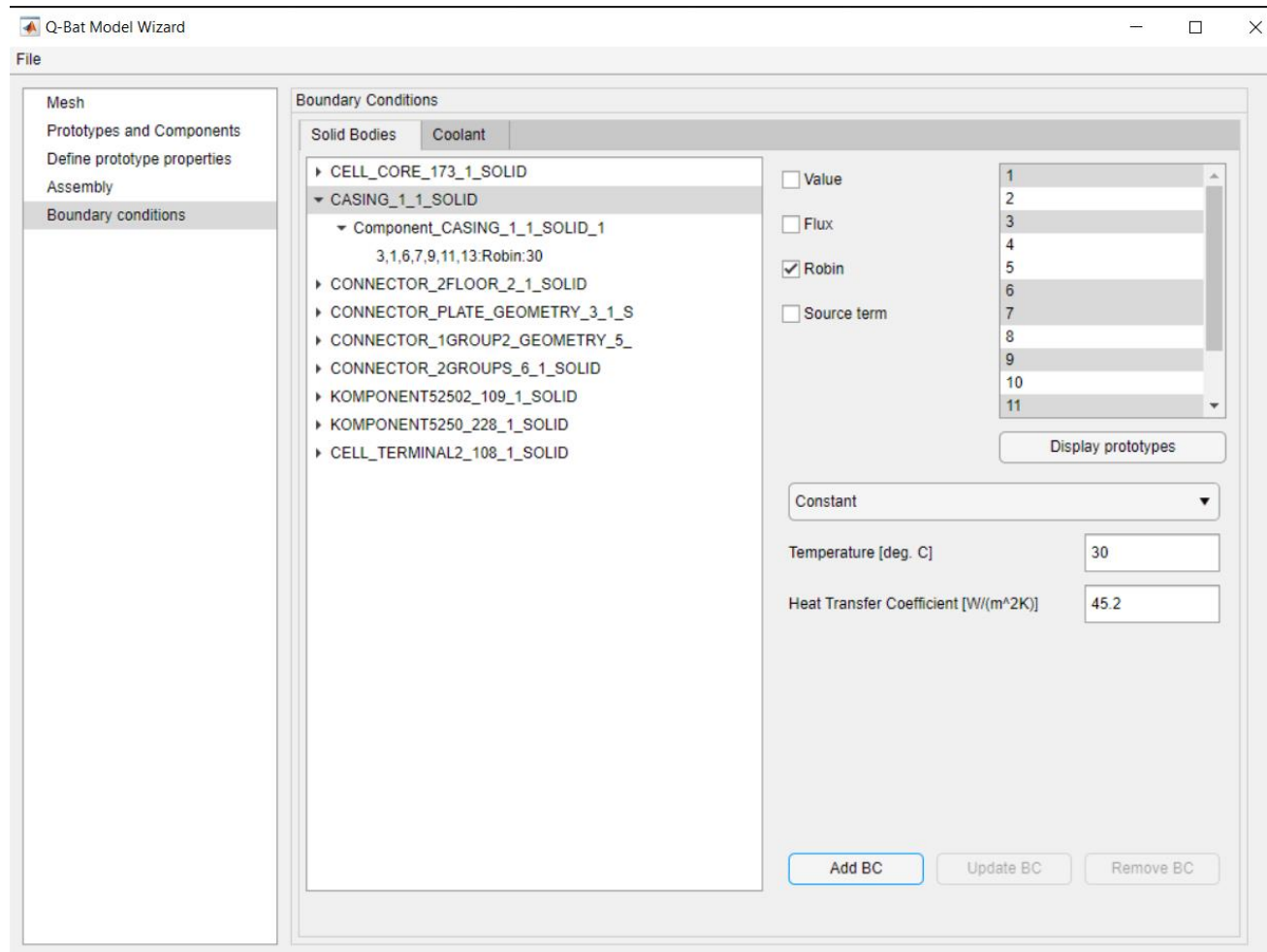
You can also create electric circuit from previously defined cell components in the script.

```

battery_pack_17S6P.m  x  +
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");
170

```

# Initial and Boundary Conditions



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

You can display prototypes to check the boundaries' ID.

Save the project and go to the script.



# Initial and Boundary Conditions

```

battery_pack_17S6P.m  X  +
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  x  +
192  %% Model Preparation and run
193  -  model.saveModel("battery_pack_17S6P_before_prep");
194
195  -  tic;
196  -  model.flag_is_parallel = false;
197  -  model_struct = model.prepare();
198  -  prep_time = toc;
199
200  -  model.saveModel("battery_pack_17S6P_after_prep");
201
202  -  current = readtable("data/current.xlsx");
203  -  current.Properties.VariableNames = {'t','current'};
204  -  model_struct.ele_circ_electro = current;
205
206  -  tic;
207  -  model.run(10, 100, 'electroSubsteps',100, 'signals_data', model_struct,...
208  -  'solver', 'Direct')
209  -  run_time = toc;
210

```

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.

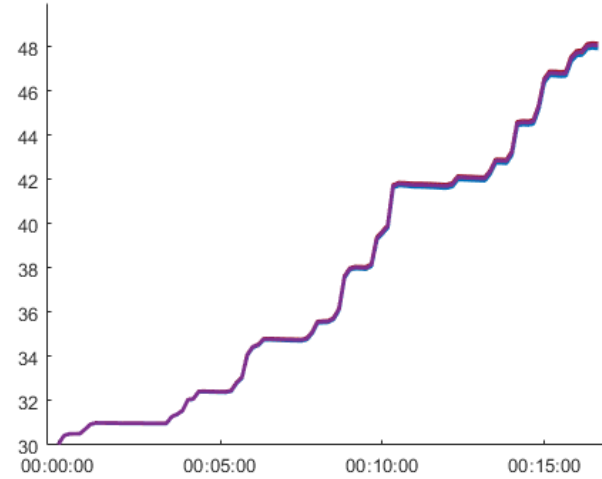
Lastly, you can export solution to extensions which are compatible with other software such as Paraview.

```

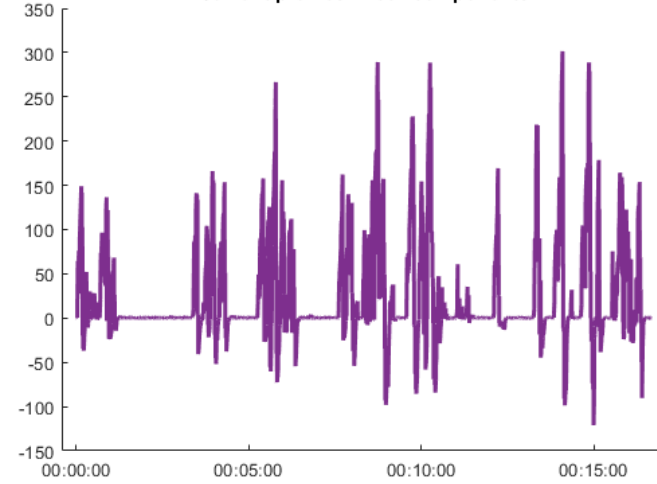
211 %% Post
212 % Elctro
213 - model.plotCircuitVoltage("ele_circ")
214 - model.plotSOCOverTime()
215 - model.plotCurrentOverTime()
216
217 %% Thermal
218 - model.plotMaxTempOverTime("Component_CELL_CORE.*")
219 - model.plotMinTempOverTime("Component_CELL_CORE.*")
220 - model.plotMeanTempOverTime("Component_CELL_CORE.*")
221
222 - model.plotSolution("cell_terminal_holder_connector_assembly", 3)
223 - model.plotSolution("cell_terminal_holder_connector_assembly", 50)
224 - model.plotSolution("cell_terminal_holder_connector_assembly", 100)
225
226 - model.exportSolutionToCGNS("battery_pack_assembly", "battery_pack_17S6P_result.cgns");
227 - model.saveModel("battery_pack_17S6P_results");
  
```

# Post-processing plot

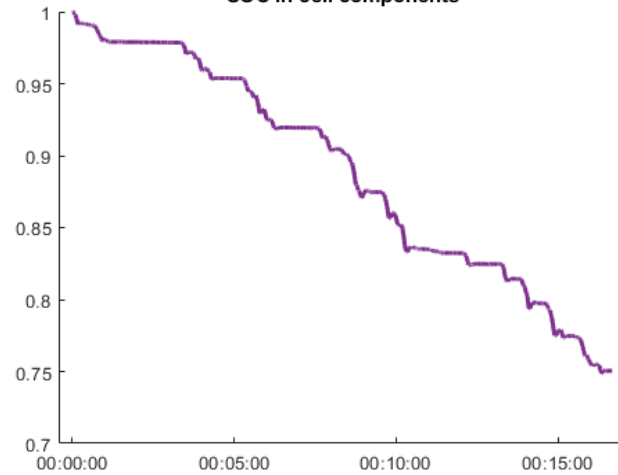
Maximum temperature in components



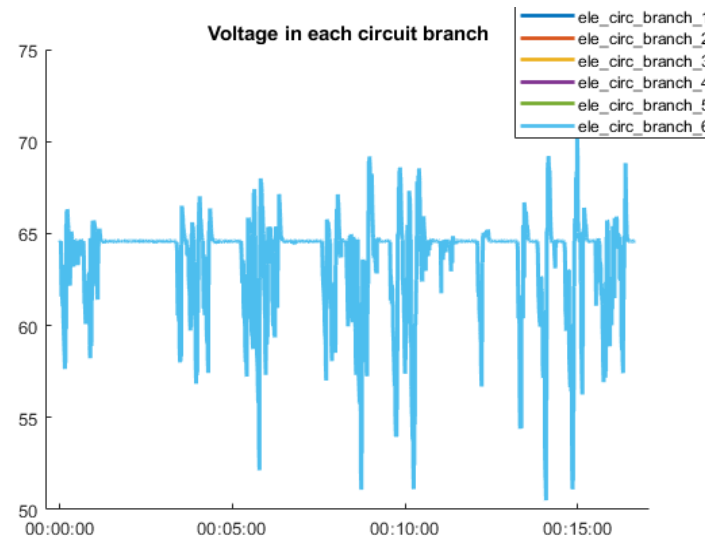
Current profiles in cell components



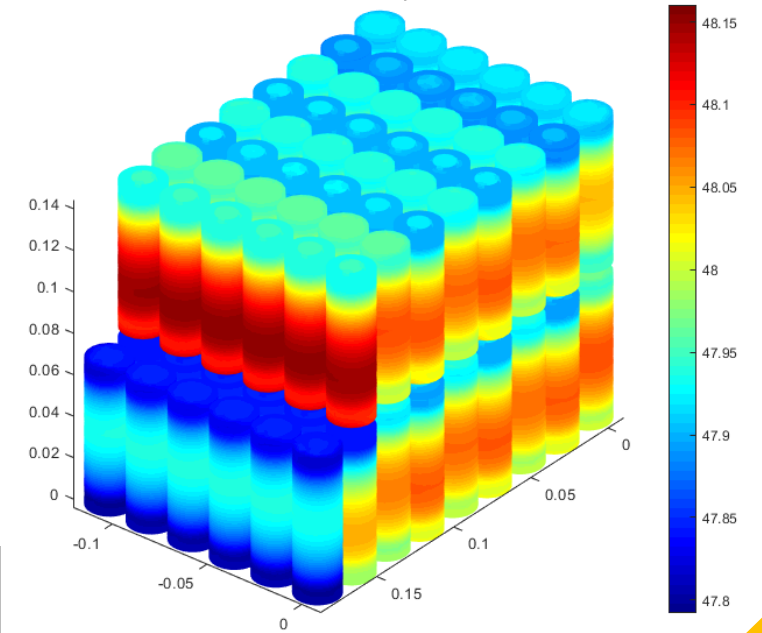
SOC in cell components



Voltage in each circuit branch



Solution at time step: 100



## 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](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](mailto:q-bat@quickersim.com)

