Battery Pack Temperature Distribution Simulation with COMSOL and MATLAB®

M.Masomtob*[1,4], C.Schaeper[1], W.Zhou[1] and D. U. Sauer[1,2,3]
[1]Electrochemical Energy Conversion and Storage Systems Group, Institute for Power Electronics and Electrical Drives (ISEA), RWTH Aachen University, Germany
[2] Juelich Aachen Research Alliance, JARA-Energy, Germany
[3] Institute for Power Generation and Storage Systems (PGS), E.ON ERC, RWTH Aachen University, Germany
[4] National Metal and Materials Technology Center (MTEC), National Science and Technology Development Agency (NSTDA), Thailand
*Corresponding author: Jaegerstr. 17/19, 52066 Aachen, batteries@isea.rwth-aachen.de

Abstract: This work is to present combined application of COMSOL with MATLAB® [1] and to test it for the liquid cooling system of a battery pack based on physical principles in COMSOL. The *m.file of Battery Module* from COMSOL 4.2 is exported and modified to run in MATLAB®, as well as is managed by using *m.file for Running Management*. The advantage of running the model of COMSOL in MATLAB® is that each part of the system built in COMSOL 4.2 or MATLAB® can be combined together and simulated at the same time in MATLAB®. Moreover, thanks to MATLAB® functionality, the results are also easy to monitor and collect [2].

Keywords: COMSOL with MATLAB®, liquid Cooling System, Battery Cell, Battery Module, Battery Pack

1. Introduction

Nowadays, the batteries find their application in mobile devices, electric vehicles and so forth. While scientists have been trying to develop low-cost materials in order to increase the power density of battery cells, engineers have been trying to design and investigate battery packs for various strategies of Battery Management System (BMS). The aim of this paper is to present a technique where COMSOL is used in combination with MATLAB® to investigate the effect of liquid cooling system of a battery pack.

The source code of the heat conjugate model via COMSOL is generated in format of m.file that can be run in MATLAB® (the computer is installed with COMSOL 4.2 with MATLAB® module). There are several main reasons why COMSOL is used in this work. First of all, building geometries, setting boundary conditions and generating meshes in COMSOL are easier than making the same in MATLAB®. Moreover, if the geometries are very complicated, COMSOL is capable of importing the geometries from CAD software. Secondly, COMSOL includes many physical principles in term of Finite Element Method and Computation Fluid Dynamics. The physical principle used in this work includes heat transfer and fluid mechanics, collectively referred to heat conjugate. Lastly, m.file generated from COMSOL can be also modified for further works in MATLAB®/ SIMULINK for large system designs [3].

2. Cooling System

One of the limitations in assembling battery packs in an electric car is space. Consequently, the liquid cooling system has been chosen because it is compacted side and is more effective than air cooling system in term of space [4].

There are eighty battery modules in the battery pack. Twenty battery modules are assembled on each floor in the battery pack and each battery module has twenty six cells, as shown in Figure 1.

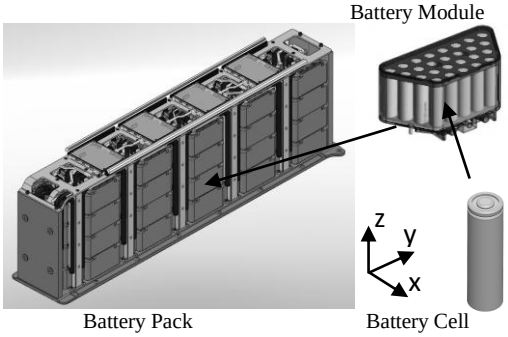

Figure 1. Three – dimensional model of battery pack, battery module and battery cell.

Figure 2 shows the diagram for the liquid cooling system assembled with the battery pack. The coolant flows through the main inlet channel, and then forks out equally onto two channels, A-side and B-side. In each side, coolant is again separated to four channels. According to flow direction onto each channel, 12.5 percent of the whole coolant flows through each battery module. Moreover, the heat generation removed of each floor and each side is assumed equal. Consequently, the cooling system analyzed in this work is simplified because of the symmetric flow direction, as shown in Figure 3.

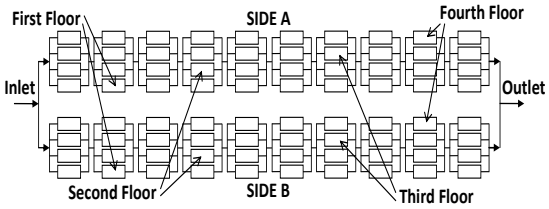

Figure 2. The diagram of the cooling system.

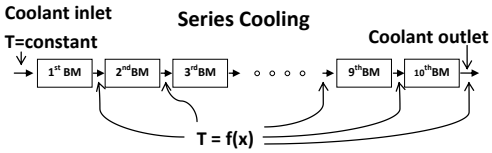

Figure 3. Ten battery modules connected in series.

Temperature sensors are installed in 1st, 4th, 7th and 10th battery modules; the signals will be sent to BMS.

3. Simulation Process

The CAD file in this work, built in SolidWorks® 2011 and exported to COMSOL 4.2 as shown on the top of Figure 4, is a 3D CAD file including solid and fluid parts. The heat conjugate module in laminar flow and the transient solver are used in this work because heat generation varies over time of each battery cell. After material properties, principles, boundary conditions, meshes and the solver are set up properly, the model in COMSOL will be run in transient mode from 0 to 2 seconds using 0.1 seconds time step to prove that the model can be run in COMSOL. Thereafter, the model in COMSOL is exported in format of m.file, which has then to be modified so that it can be called from another m.file. Although, there is only one *m.file of the Battery Module* from COMSOL, it can also be applied to other battery module models because of the same CAD geometry, running time, time step, principles and solver. Only the inlet temperature of each battery module is changed.

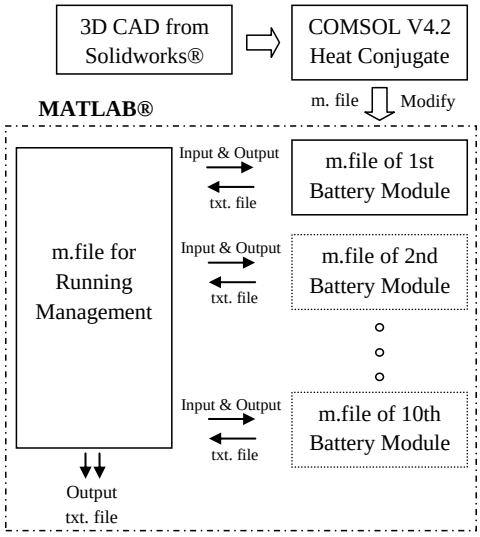

Figure 4. The flow chart for simulation process.

The *m.file for Running Management* block, as shown on the left hand side of Figure 4, is built to control parameter input of the *m.file of each Battery Module* such as running time, mesh quality, heat generation of each battery cell, flow rate of coolant, inlet temperature and *m.file of Battery Module* number running in serial. Moreover, it is also used to collect the results from each of the m.file battery modules in form of txt.file. When the *m.file for Running Management* is run, it will send all parameters in form of txt.file and values. In the same time, the *m.file of 1st Battery Module* is also started by the *m.file for Running Management.* When the *m.file of 1st Battery Module* finishes running, the results will be sent back to the *m.file for Running Management* to collect and send new parameters to the *m.file of 2nd Battery Module*. The procedure will continue until the *m.file of 10th Battery Module*. The advantage of saving the data in txt.file during run time is that we can examine the results of a battery module running to completion while another *m.file of Battery Module* is running. Moreover, the results of the battery module have already been saved even in the case of a power failure.

Table 1: Properties of Materials

Material	Phase	Thermal Conductivity [W/(m.K)]	Heat Capacity [J/(kg.K)]	Density [Kg/m^3]	Heat Ratio	Viscosity [Pa.s]
Battery Cell	*Solid*	{1.01,1.01,30.22}	2690	750	-	-
TIM*	*Solid*	5	500	3100	-	-
Copper	*Solid*	400	385	8700	-	-
Aluminium	*Solid*	155	893	2730	-	-
Coolant	*Liquid*	0.405	3300	1078	1	0.00429

4. Battery Module Model

The model was made in both SolidWorks® for CAD file and COMSOL for CAD import, physical setting, boundary condition setting, mesh generation and solver setting. After that, the model was exported and modified to run in MATLAB®.

4.1 Geometry and Materials

The CAD file in this work, made in SolidWorks®, is a 3D model including solid and fluid parts. The solid parts in this work consist of twenty six battery cells, one thermal interface material (TIM) plate, two copper plates and one aluminium plate. Some solid parts such as battery module housing, insulator, positive common plate and aluminum wiring are not considered in the model because of their low heat transfer.

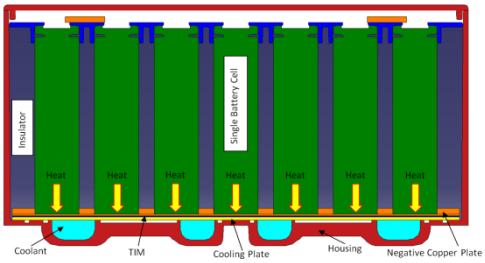

Figure 5. The cross section area of the battery module.

Figure 5 shows the cross section area of the battery module. Each of the battery cells is assumed to be single property. Copper plate is used as the negative common plate for conducting electricity and fixing every single cell. Before the heat from every single cell is removed by a coolant, it has to pass through copper, TIM and aluminum plates, respectively. The aluminum plate is a cooling plate which is directly in contact with the coolant. TIM is assembled between the aluminum cooling plate and the negative common plate to avoid electric currents passing through the coolant, but the allowing heat transfer. The fluid part is the coolant, a mixture of water and GLYKOSOL N [5].

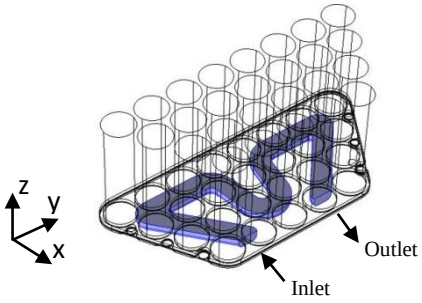

Figure 6. The geometry of the flow channel.

Figure 6 shows the geometry of the flow channel of the coolant. There is one liquid flow channel under the cooling plate designed to remove the heat from every single battery cell.

Table 1 shows the properties of each material. The properties are set by using constant values independent of temperature change. However, thermal conductivity of the battery cell is only set for the three canonical directions. The thermal conductivity of the battery in z-axis cell is higher than in x-axis and y-axis cells. The heat ratio and the viscosity value are set only for the coolant due to the fact that is liquid phase.

4.2 Physical Principle

The conjugate heat transfer model for laminar flow pattern is used in this work. The heat generation as a function of time is used for each of the twenty six battery cells; it is transferred through the copper plate, TIM and

the aluminum plate in the form of heat conduction before it is moved out from the aluminum plate by the coolant through heat convection. According to heat generation as a function of time, the model used in this work is also a transient model, as shown in Figure 7.

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u}\cdot\nabla)\mathbf{u} = \nabla\cdot\left[-p\mathbf{I} + \mu\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right) - \frac{2}{3}\mu(\nabla\cdot\mathbf{u})\mathbf{I}\right] + \mathbf{F}$$

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\mathbf{u}) = 0$$

$$\rho C_p\frac{\partial T}{\partial t} + \rho C_p\mathbf{u}_{trans}\nabla T = \nabla\cdot(k\nabla T) + Q$$

Figure 7. Equations used in the mode based on heat conjugate heat module in COMSOL 4.2.

4.3 Assumption

There are several assumptions used in this work to reduce many complications of the simulation. First of all, a few parts in the battery module are not considered such as the insulator between battery cells, small aluminium conducting wires (fuse) as well as the positive common plate, and the battery module housing because little heat can be transferred to these parts. Secondly, the heat generation both as the results of heat losses from Joule effect heating and from electrochemical reactions of the battery is combined into only one term in a function of time [6]. Thirdly, each battery cell is treated as a homogenous model. Finally, the coolant hoses connected between the battery modules are not considered**.**

4.4 Boundary Condition

The main boundary conditions are stated as follows;

- Heat generated from each single battery cell as a function of time.
- The inlet temperature of the 1st battery module is kept constant at 20^{o}C.
- The inlet temperature of the other battery modules as function of time.
- The flow rate of the coolant is 1 L/min.

4.5 Mesh and Calibration

The precision of the results depends on the number of elements. Lengthy runtime and high memory usage are required if the number of elements is increased. In this work, although a number of elements is generated at the computer performance limit, the results are not sufficient in accuracy. Therefore, the calibration is used to find coefficient values for adjusting the result of simulation adjacent to be as exact as possible. For the method of calibration, the temperatures, 290, 295, 300, 305, 310, 315 and 320K, are set up at the cooling inlet of the battery module and heat generation in each battery cell is neglected. Therefore, the temperature at outlet channel should have the same value as the temperature at inlet channel; in contrast, the results after running showed that the outlet temperature from simulation is a little bit lower than the inlet temperature as shown in Figure 8. Therefore, calibration is very important before the temperature at outlet channel will be set at the inlet channel of the next battery module. The temperature at outlet channel has to be calibrated in the *m.file for Running Management* using a coefficient value, slope in Figure 8.

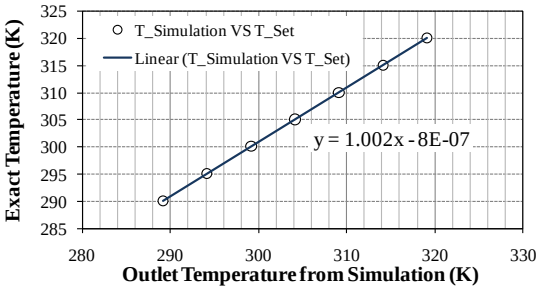

Figure 8. Comparison between outlet temperature from simulation and exact temperature.

The command for generating meshes in MATLAB®, FreeTet is Triangular Elements. MESHQ which describes the mesh resolution is set to 5 (Extremely fine, 1 – Extremely coarse, 9) because of the limit of the computer used in the simulation. The number of elements in this work is 856824 elements:

```
    model.mesh('mesh1').feature.create('ft
et1', 'FreeTet');
    model.mesh('mesh1').feature('size').se
t('hauto',MESHQ);
    model.mesh('mesh1').feature('size').se
t('table', 'cfd');
    model.mesh('mesh1').run;
```

4.7 Solver

The heat conjugate for laminar flow module is considered to build the model of battery module. The running time is set from 0 to 2 seconds only and the time step is 0.1 seconds. A few parameters in Fully Coupled block are set such as *Automatic* for damping method and *20 iterations* for maximum number of iterations, as shown in Figure 9.

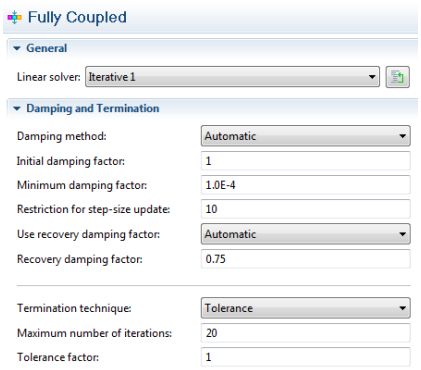

Figure 9. Setting some parameters in Fully Coupled block in COMSOL.

4.8 m. file of Battery Module modify

After exporting m.file from COMSOL, the m.file of the Battery Module is modified in order to support the commands from *m.file for Running Management*.

For example, the command for importing a parameter as a function of time in form of txt.file, the name of "Heat'' represents the total heat generation from every battery cell and "Heat_Loss.txt" is the txt.file of the heat generation values over time:

```
   model.func.create('int1',
'Interpolation');
   model.func('int1').model('mod1');
   model.func('int1').set('source',
'file');
   model.func('int1').set('filename',
'D:\running
model\Version1\Heat_Loss.txt');
   model.func('int1').setIndex('funcs',
'Heat', 0, 0);
   model.func('int1').importData;
```

The code using the parameter from txt.file in source code of COMSOL looks as follows,

```
   model.physics('nitf').feature('init1')
.set('T', 1, '293');
   model.physics('nitf').feature.create('
hs1', 'HeatSource', 3);
   model.physics('nitf').feature('hs1').s
election.set([6 8 11 13 15 17 19 21 23 25
27 29 31 33 35 37 39 41 43 45 47 49 51 53
55 57]);
   model.physics('nitf').feature('hs1').s
et('heatSourceType', 1, 'TotalPower');
   model.physics('nitf').feature('hs1').s
et('Ptot', 1, 'Heat(t)');
```

In case, the parameters are constant value (*Flow* and *Area*), the variable can be inserted directly in source code of COMSOL:

```
   model.physics('nitf').feature.create('
inl1', 'Inlet', 2);
   model.physics('nitf').feature('inl1').
selection.set([337]);
   model.physics('nitf').feature('inl1').
set('U0in', 1, 'Flow/Area');
```

5. m. file for running management

The *m.file for Running Management* includes three main loops. The first loop is used for *1st Battery Module* simulation. It can input the parameters, start running the *m.file of 1st Battery Module* and collect to save the result when the *m.file of 1st battery module* finishes. The second loop is used to represent that the simulation is finished. The main reason why the second loop has to be between first and third loops because in case the last battery module is finished, the *m.file for Running Management* can stop the procedure. The third loop is used for simulating the m.files of other Battery Modules apart from 1st battery module because the inlet temperature of 2nd to 10th battery module is not constant, it needs some command to retrieve the outlet temperature from the previous battery module. Moreover, when each of the m.files of the battery modules is finished, each loop will show its runtime.

The new running time for transient from 0 to 1200 sec, the inlet temperature of first battery module and the number of elements are set in the *m.file for running management* before running.

In addition, the *m.file for Running Management* is used for calibration before the results will be saved and used in the next battery module:

```
   [Tout unit_Tout] =mphint(model,'T/Area
*1.00262879243087','edim',2,'selection',1
81);
```

6. Results and Discussions

Figure 10 shows the inlet and outlet temperatures from the simulation in each battery module. The outlet temperature of the 1st battery module is lower than other battery modules because of its lower inlet temperature. The out temperature of each battery module increases quickly when the heat generation shoots up to 80W. On the contrary, the temperature will decline if the heat generation from battery pack goes down. After 600 seconds, the temperature has a continually downward trend because of small heat generation.

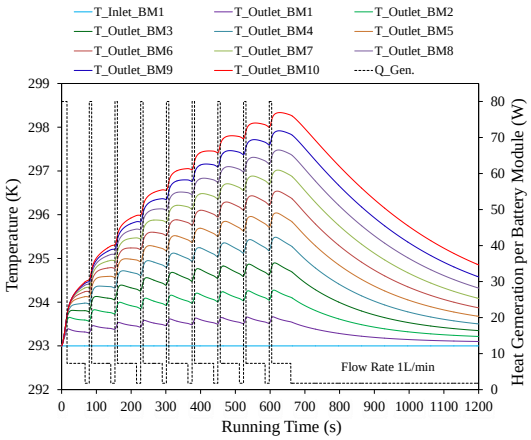

Figure 10. Inlet and outlet temperatures of each battery module.

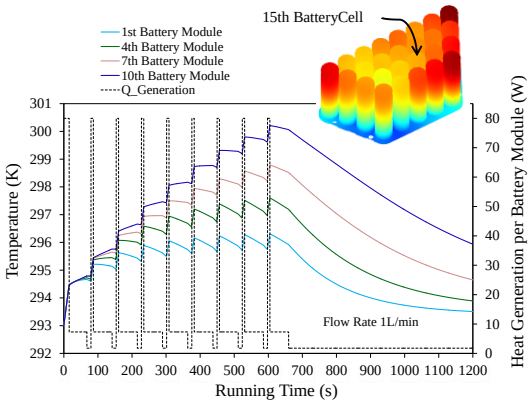

Figure 11. Temperature on the top of 15th battery cell in 1st, 4th, 7th and 10thbattery modules.

The average of the runtime of *m.file of each Battery Module* is 34-37 hours that depends on workloads of other applications running. However, the runtime will be large if the turbulent module is used in this work and the number of meshes is increased.

Figure 11 shows the temperature at the top of 15th battery cell in1st, 4th, 7th and 10th battery modules which are next to the temperature sensor. These temperatures are very important because they are data for BMS for controlling the flow rate of the coolant in order to save energy consumption of the cooling pump.

7. Conclusions

The COMSOL m.file of the battery modules is able to run with the *m.file for Running Management* in order to investigate the temperature distribution of the cooling system in the battery pack. Even though the results of the simulation still are not accurate, it is not significant if the number of elements is increased and performance of computer is higher. In addition, if the flow rate of the coolant is higher, the turbulent model will be used instead of laminar model; the runtime and high performance computer are required to simulate this work due to the complexity of the turbulent physical phenomenons

8. References

1. COMSOL Version 4.2, Reference Manual
2. MATLAB® Version R2011b, Reference Manual
3. Jos van Schijndel, Integrated Modeling using MatLab, Simulink and COMSOL with heat, air and moisture applications for building physics and systems
4. Ahmad A. Pesaran, Battery thermal models for hybrid vehicle simulations, Journal of Power Sources **110 (2002)** 377–382
5. The properties of GLYkosol_N_20, www.glykolundsole.com/Downloaddateien/ Glykosol_N_20seiten_deu.pdf
6. Heat generation combination for the battery pack, Institut für Kraftfahrzeuge, RWTH Aachen University

9. Acknowledgement

This work was kindly financed by the German Federal Ministry for Education and Research within the project ePerformance.