Industrial Training

MSc Numerical Methods in Engineering

Master in Numerical Methods in Engineering, at the
Universitat Politècnica de Catalunya.
Campus Nord (CIMNE, UPC)

Internship at Fraunhofer Institut für Solar Energiesysteme (ISE)
Freiburg im Breisgau, Baden Württemberg, Germany

L. Millet

July 2016

Contents

Outline **3**
Introduction to Electrochemical Cells **4**
- Electrochemical cell and battery 4
- Operation of a cell 4
  - Energy and power 4
  - Capacity 5
  - State of health (SOH) and state of charge (SOC) 5
  - Memory effect 6
  - Equilibrium state: Open circuit 6
  - Charge, discharge, polarization 6
  - Discharges curves and C-rate 7
- Lithium-ion 8
  - Battery of study 9

Thermal simulation literature review **10**
- Introduction 10
- Thermal-Electrochemical cell physics 10
  - Molecular scale coupled electrothermal approach 11
  - Macroscopic thermodynamic balance approach 12

Introduction to Comsol FEA Software **14**
- Introduction 14
- Getting started 15
  - Comsol Desktop: Getting started. Building a model 15
  - Documentation and tutorials 16
- Comsol for Thermal Modeling of Li-ion Batteries at ISE 17
- Acquired knowledge, tips and tricks 18
  - Geometric features 18
  - Expressions definition 19
  - Model couplings 23
  - Specifying and defining materials 23
  - Meshing 24
  - Modeling with Java code 25
  - Customized physics 26
  - Others/ Suggestions 28

Calorimeter Instrument Introduction and Workshop **31**
- Instrument 31
- System components 32
- Calibration 34
  - Calibration procedure 34
  - Calibration check 35
- Operation and data analysis 36
  - Loading a Sample procedure 36
  - Experiment Setup Software 37
  - Data analysis 37

References **39**

Outline

This report is intended to review the tasks developed during the 7 weeks of internship at Fraunhofer Institut für Solare Energiesysteme (ISE), in Freiburg im Breisgau, Baden-Württemberg, Germany, for the completion of the Master of Science in Numerical Methods in Engineering, taught by the International Center of Numerical Methods in Engineering (CIMNE) at Universitat Politècnica de Catalunya (UPC), Campus Nord, Barcelona.

The internship is performed at the Electric Energy Systems (EES) team, from ISE, and is linked to the future development of the master thesis, which is contextualized within one of the tasks of the Jospel project, under EU Horizon 2020 research and innovation programme, regarding advances in EV efficiency of acclimatization systems, within the field of thermal simulation of lithium-ion battery systems for EV applications.

For that reason, the main part of the internship time has been dedicated to formation tasks for the future development of the thesis, and the main tasks have been: literature research on general battery operation, terminology, and State of the Art of thermal modeling of lithium-ion batteries; getting familiarized to the FEA software (Comsol) in which it is intended to develop the thermal modeling; and, finally, to formation –workshop– for the usage of the available isothermal calorimeter at the laboratory of the company, which is going to be used for validating the results of the thermal models. The contents of this report are presented in the previously mentioned order.

It should also be pointed out that the development of the two last chapters of the present report has also been conceived for future use and consultancy of workers at ISE, that will also use Comsol FEA software for any modeling purpose, or the isothermal calorimeter Netzsch IBC 284.

Introduction to Electrochemical Cells

Electrochemical cell and battery

Electrochemical power sources convert chemical energy into electrical energy. The characteristic feature of an electrochemical cell is that the energy of the reaction from the chemical process during its operation, that undergo at least two reaction partners, is available externally as electric current at a defined voltage and time. Depending on the nature of the chemical reactions and of the materials of an electrochemical cell, they can be classified either as an irreversible or *primary* cell, which are non-rechargeable single-use cells; or a reversible or *secondary* cell, which can be recharged several times after discharge.

The name *battery* is usually given to the system conformed by an arrangement of a certain number of electrochemical cells, normally assembled in parallel to increase its capacity. Assembles in series are limited because of reaching dangerous voltages between battery terminals.

Batteries come in many shapes and sizes. Mainly, there exist in the market three different kinds of physical battery arrangements for power applications, whose difference relies on how the electrochemical cells –usually layered structures– are grouped and packed:

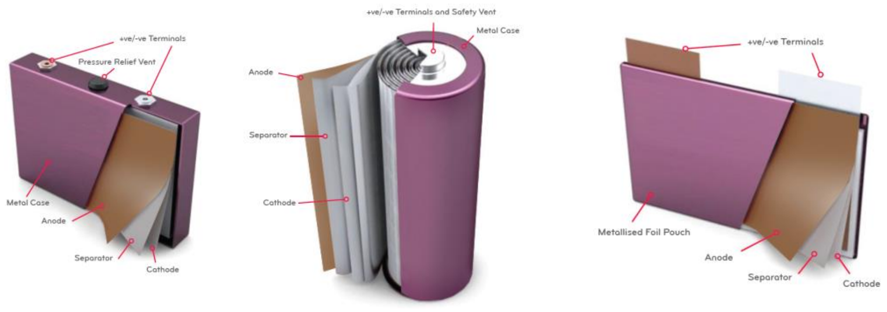


Fig. 1: Scheme of the main cell assemblies that can be found in the market. Left, a prismatic cell. In the center, a cylindrical cell. Right picture, a pouch cell.

A *battery pack* is usually conformed of an arrangement of parallel and series batteries, which is designed in order to obtain the desired energy and power, or amperage and voltage.

Operation of a cell

Energy and power

Specific energy, or gravimetric energy density, defines the ratio between the battery capacity and the weight (Wh/kg). Energy density, or volumetric energy density reflects the ratio of capacity on volume in liters (Wh/l).

Specific power, or gravimetric power density, indicates loading capability. Power density is related to the speed of energy delivery. Batteries for power tools are made for high specific power and come with reduced specific energy (capacity).

Capacity

The *capacity* of a battery is the amount of electric charge it can deliver at the rated voltage. The theoretical capacity of a cell may be calculated as $C = x(nF)$, where x is the number of moles of reaction associated with the complete discharge of the cell, n is the number of moles of electrons exchanged, and F is the Faraday's constant. Thus, note that the more electrode material within the cell, the greater its capacity is. The capacity is measured in amper-hours (Ah) in SI units.

Due to electrode kinetics–explained more in detail in the following section–, the fraction of the stored charge that a battery can deliver is not constant but depends on many factors, i.e., the rate at which the current is supplied or delivered, the storage period, or the ambient temperature. In general, the higher the discharge rate is the lower the discharged capacity, and similarly occurs for the charging process.

The *nominal* or *rated capacity* provided by the manufacturers is obtained by constant current charge and discharge procedures within the full range of operating voltage, by trying different current magnitudes until the current which is set charges or discharges the cell completely in one hour.

State of health (SOH) and state of charge (SOC)

The *state of health* (SOH) is a figure of merit of the condition of an electrochemical cell, battery, or battery pack, which indicates it as a percentage, taking 100% as the battery conditions that match the battery's specifications. Typically, the battery manufacturers provide the products at 100% SOH, and this will decrease over time due to storage time (calendric aging), and use of the batteries (cyclic aging).

Conventionally, it is taken that at 80% SOH the battery life has arrived to its end, since most of the commercially available batteries show very fast degradation behaviour after this point. However, there is no consensus in how SOH should be determined or measured, and usually it might be derived from some parameters of the battery, such as the internal resistance, the capacity, the voltage, the self-discharge, and the number of charge-discharge cycles.

The *state of charge* (SOC) is the equivalent to a fuel gauge for a battery or a battery pack. It indicates the current state of a battery in use, in percentage, where 100% SOC is a fully-charged state, where the energy stored inside the battery is equal to its capacity; and 0% SOC is empty state, where the energy stored is zero.

In general there are two methodologies to determine the state of charge: first, using direct measurements (Coloumb-counting) of the charge that is flowing inside and/or outside the battery; and second, using a set of indirect methods that can avoid direct measurements of SOC, or improve its results for SOC estimation due to a better robustness, such as Kalman filtering.

Memory effect

Memory effect, also known as lazy battery effect or battery memory, is an effect observed in some rechargeable batteries that cause them to hold less charge. It describes the specitic situation in which NiCd batteries gradually lose their maximum energy capacity if they are repeatedly recharged after being only partially discharged. The battery appears to “remember” the smaller capacity. Some other battery chemistries also show this behaviour in a less pronounced fashion.

Equilibrium state: Open circuit

Under equilibrium conditions, the potential difference of the terminals of a lithium-ion cell corresponds to the so-called metal ion potential. This equilibrium voltage, or *open circuit voltage*, exists due to the coexistence in balanced phases of a metal and its ions. It can either be measured or calculated from thermodynamic data of the cell reaction, and usually depends, for a given chemistry, on the temperature and the pressure of the cell.

The open circuit potential is one of the main characteristics of the state of charge, since it has been demonstrated that the open-circuit voltage is (current) rate independent [1], but it only depends on the SOC and the temperature.

Charge, discharge, polarization

During the discharge process, electrons are released at the anode from the active material due to oxidation. At the same time, cathodic substances are reduced obtaining electrons, which arrive to the cathode through the external circuit transport. The charge process is the opposite process, and usually requires between the cathode and the anode a potential slightly higher than the equilibrium voltage.

When a current flows, for example when discharging the battery, a shift in the potential of the cell can be observed. This deviation is called *overpotential*, and is originated in general in electrochemical cells from different physical sources:

- Charge transfer overpotential: the speed of the charge transfer, or the current flow speed, through the phase-boundary of the electrode and electrolyte is limited. A higher overpotential is generated with higher charge transfer speed, following the *Butler-Volmer* and the Tafel equations [2].
- Diffusion overpotential: When high current densities at the electrodes exist, depletion of the reacting substances is possible, resulting in a concentration polarization. In this case, the reaction kinetics is determined only by diffusion processes within the electrode and electrolyte boundary –the so-called Nernst layer.
- Reaction overpotential: Both the overpotentials mentioned above are normally of greater importance than the reaction overpotential, but it may happen that other phenomena such as adsorption and desorption in the electrolyte are also rate-limiting factors.

- Crystallization overpotential: This can occur as a result of the inhibited intercalation of metal ions into their lattice, especially important during metal deposition at the negative side when the batteries are charged [3].

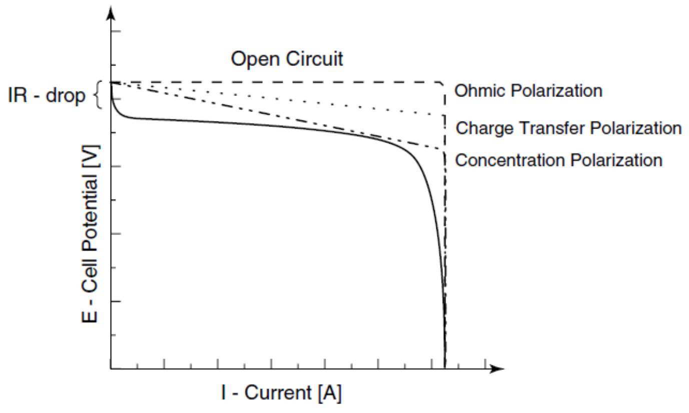


Fig. 2: Diagram of the cell polarization as a function of the current.

Information on the characteristic behaviour of cells under load and at various stages of discharge can be conveyed by graphs.

Last but not least, self-discharge must be mentioned. This phenomenon explains that the batteries reduce the stored charge without any connection between the electrodes due to internal chemical reactions. How fast self-discharge in a battery occurs is dependent on the type of battery, state of charge, charging current, ambient temperature, and other factors.

Discharges curves and C-rate

Charge and discharge curves show either the open circuit voltage of a cell as a function of the fraction of discharge completed, or the cell voltage during a deep discharge –usually at a constant current. The current-voltage charge and discharge curves are one of the most important characteristic of batteries which are available experimentally. Plots of cell voltage against current are usually referred to al *polarization curves*, whereas graphs of cell voltage as a function of the fraction of discharge completed are known as *discharge curves*.

A useful method of characterizing the charge and discharge current-voltage rates, either for recording discharge curves or for determining practical capacities is to standardize the current in terms of the nominal cell capacity.

The *C-rate* is defined as the current to discharge the nominal capacity in one hour. Thus, a C-rate of $\mathfrak{n}$ implies that the nominal capacity of the cell, in Ah, is delivered in $1/\mathfrak{n}$ hours, e.g. for a 2Ah cell, discharge at C/5-rate signifies a current of 0,4A.

Lithium-ion

Several types of rechargeable battery systems are available on the market, such as lead-acid (LA), nickel-cadmium (NiCd), nickel-metal hybride (NiMH), lithium-ion, and lithium-ion polymer batteries. Due to their high specific energy density, lighter weight, lower self-discharge rate and good cycle-life, lithium-ion has emerged as the prime candidate as a power source for electric and hybrid electric vehicles (Evs and HEVs) [4-5].

The first non-rechargeable lithium batteries appeared in the early 1970s. Due to safety problems because of inherent instability of lithium metal, especially during charging, it was not until 1991 when the first secondary lithium-ion batteries were commercialized by Sony[6].

Lithium-ion batteries are common in home electronics, and one of the most popular types of rechargeable batteries for portable electronics, with a high energy density, tiny memory effect, and low self-discharge.

A lithium-ion cell consists of five regions –from left to right in Fig.3–: A negative electrode current collector made of copper, a porous composite negative insertion electrode, a porous separator, a porous composite positive insertion electrode, and a positive electrode current collector made of aluminium.

Many studies have been done on complex oxides of lithium and a transition metal, such as $LiCoO_2$, $LiNiO_2$, $LiMn_2O_4$, and $LiFePO_4$, for the positive electrode materials.

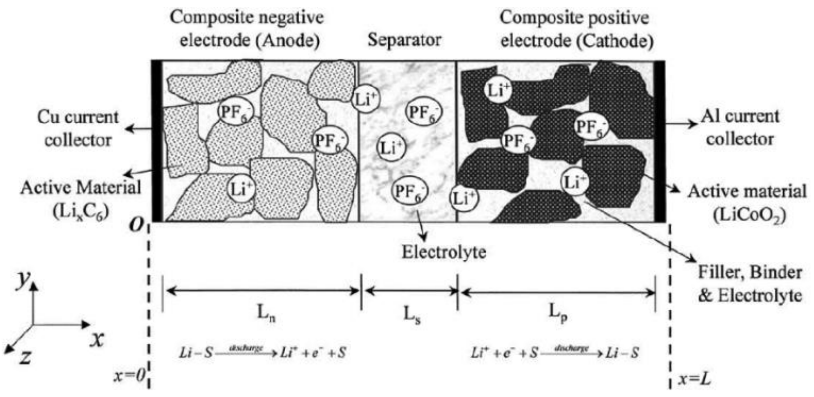

Composite negative electrode (Anode)
Separator
Composite positive electrode (Cathode)
Cu current collector
Active Material (Li_xC_6)
Al current collector
Active material ($LiCoO_2$)
Filler, Binder & Electrolyte
Electrolyte
Li^+
PF_6^-
O
y
x
z
L_n
L_s
L_p
$Li-S \xrightarrow{discharge} Li^+ + e^- + S$
$Li^+ + e^- + S \xrightarrow{discharge} Li-S$
$x=0$
$x=L$

Fig. 3: Internal structure and components of a lithium-ion cell.

Lithium-ion batteries can be dangerous under some conditions and can pose a safety hazard since they contain, unlike other rechargeable batteries, a flammable electrolyte and are also kept pressurized.

Pure lithium is highly reactive. It reacts vigorously with water to form lithium hydroxide and hydrogen gas. Thus, a non-aqueous electrolyte is typically used, and a sealed container rigidly excludes moisture from the battery pack.

Lithium ion batteries are more expensive than NiCd batteries but operate over a wider temperature range with higher energy densities.

Industry produced about 660 million cylindrical lithium-ion cells in 2012, and one of the major demands of the product is from Tesla car manufacturer. Production is gradually shifting to higher-capacity 3+ Ah cells.

Battery of study

The battery that will be studied, characterized and modeled in this project is the A123 Systems AMP20m1HD-A, whose chemistry is based on the use of a lithium iron phosphate patented cathode material (Nanophosphate®). The lithium iron phosphate battery, also called LFP battery (standing for “lithium ferrophosphate”), is a type of rechargeable lithium-ion battery which uses $LiFePO_4$ as a cathode material.

$LiFePO_4$ is a natural mineral of the olivine family (triphylite). Its use as a battery electrode was first described in published literature by John Goodenough's research group at the University of Texas in 1996 [7]. With somewhat lower energy density than common $LiCoO_2$ design found in consumer electronics, they offer longer lifetimes, better power density and are inherently safer [8].

Moreover, lithium iron phosphate (LFP) excels the others due to its flatness of discharge curve profile, as we can see in the following figure, Fig. 4.

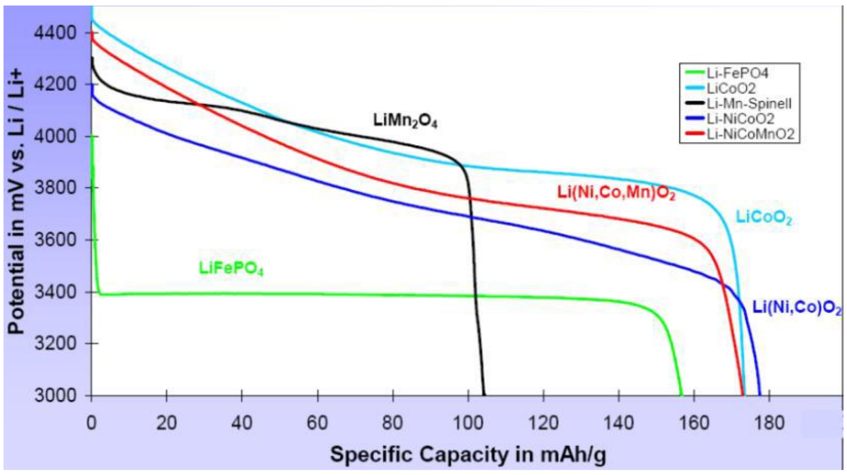


Fig. 4: Comparison of the open circuit potential curves between different lithium-ion commercially available chemistries.

Thermal simulation literature review

Introduction

Currently, most research into Li-ion batteries focus on the material aspect –essentially focused in the cathode material– to improve the energy, power, and life-cycle performance, with relatively less attention paid to thermal issues. However, some of the most important facts for which lithium-ion batteries have not been widely deployed commercially in EVs and HEVs yet are safety, cost, and poor low temperature performance, which are all challenges related to battery thermal management [9].

Main thermal issues in electrochemical storage systems for EDVs:

- Overheating can cause *thermal runaway*, which eventually leads to the fast destruction, or explosion, of the battery. This phenomenon occurs when too elevated temperatures are reached. Then, the high temperature trigger more and faster exothermic reactions inside the battery, raising the temperature further and potentially triggering more deleterious reactions.
- Capacity and power fading (aging of the battery) is commonly accelerated at elevated temperatures. The influence of temperature on degradation is complex, but the previous statement is true for nearly all chemistries. Low temperatures also accelerate degradation.
- At low temperatures, the internal resistivity of the cells grows significantly, causing the batteries not to operate efficiently –with high thermal losses–.

Thermal-Electrochemical cell physics

Accurate understanding of the characteristics of the battery heat generation is essential to the development and success of thermal management systems (TMS) [10]. The governing equations and descriptions of the electrochemical mechanisms to understand how is it the heat generated during charge or discharge processes are presented via different approaches in the literature, and a variety of assumptions are applied to computational schemes, from simple 1D models to detailed and complex 3D coupled electrochemical-thermal models.

Heat is produced in batteries from three fundamental sources [9, 11-12]: activation or polarization –related to the interfacial kinetics–, reaction heat –related to the concentration and transport of different active materials–, and Joule effect –ohmic losses in the current collectors, due to the movement of charged particles–. Note that most of these factors are sensible to temperature changes.

In the literature, there are mainly two prevalent focuses to characterize the heat generation in electrochemical cells [5]:

- Models integrating thermal and electrochemical models (*molecular level*): The most well-known electrochemical model which can be coupled with a thermal model is referred to as pseudo-two dimensional model (P2D). Its derivation was presented by Newmann et al. [13-14], and it is the reference model for most of the available software to simulate, usually in 1D, the voltage distributions along the thickness of different electrochemical cells.

- Models derived from global thermodynamical balances (*cell level*): The efforts to derive the heat effects have to be attributed mainly to Sherfrey et al.[15], and Bernardi, Newmann, et al.[16], which have proposed similar practical and compact equations to estimate the heat generation rate for battery systems.

Both approaches are briefly described in the following subsections, trying to point out their limitations or weaknesses, and the stronger features between each other.

<u>Molecular scale coupled electrothermal approach</u>

The chemical reaction for the discharge of the LFP battery is described by the following reactions [17]

At the negative electrode: $LiC_6 \xrightarrow{discharge} x_{s,n}Li^+ + x_{s,n}e^- + Li_{1-x}C_6$

And at the positive electrode: $x_{s,p}Li^+ + x_{s,p}e^- + Li_{1-x_{s,p}}FePO_4 \xrightarrow{discharge} LiFePO_4$

However, the quantify the exact cell reaction rates and its process, requires lying on the *material properties* within the used electrochemical components –the cathode material structure, the particles size, the active material concentrations, the different electrolyte phases, etc.

The LFP cell under study pertains to the so-called dual lithium-ion insertion cell or "rocking-chair" cell, that are characterized for having two insertion electrodes –versus the other possibility, to have solid lithium as one electrode. For the insertion electrodes, we can also consider whether the system uses a porous electrode geometry versus a flat, nonporous geometry, which will also affect the governing equations.

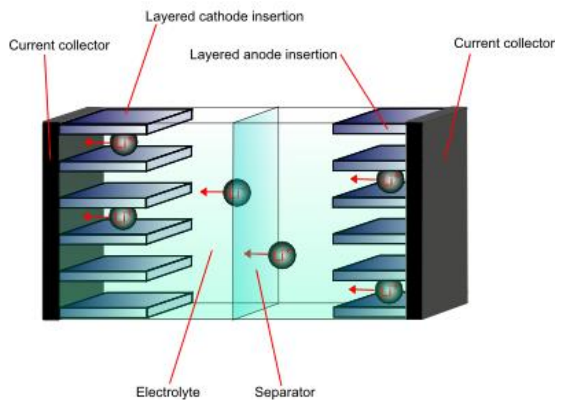


Fig. 5: Dual lithium-ion insertion cell, like the A123 AMP20m1 LFP cell [18].

The governing equations ought to account for several physical phenomena:

- Concentrated solution theory: the salt concentrations used in lithium batteries are generally large. Thus, transport of the electrolyte should be treated rigorously by using *concentrated solution theory*, as it has been shown that the assumption of dilute-solution theory are especially poor for the polymer electrolytes [19].
- Porous electrode theory: normally treated as a superposition of continuous electrode and solution phases. Usually, the detailed pore geometry is not considered but, instead, describe the porous electrode in terms of its specific interfacial area and the volume fractions of each phase. Includes also charge conservation and Ohm's law. *Insertion electrodes* are described by some modifications and simplifications of this model.
- Solid state diffusion: describes the insertion process of lithium into the oxide lattice [20]. Normally spherical particles are considered of a given average diameter.
- Electrode kinetics: describe the kinetics of the reactions, and characterizes the exchange current. They are usually modeled by Butler-Volmer equation.

All the governing equations accounting for those physical phenomena are coupled with thermal energy balance in the P2D model, giving a system of numerous partial differential equations that described the electrochemical-thermal system.

However, the most limiting factor of this approach is that a huge number of parameters pertaining to the molecular scale are needed to characterize. In between others, some of the variable inputs are: maximum and minimum concentrations of active species in both cathode and anode, solid phase diffusivities and diffusion activation energy, reaction rate constants, thermal conductivities of every material, characteristic particle radius of cathode materials, transfer coefficients for anodic and cathodic currents, and ionic or electronic conductivities of every material.

To characterize all of these parameters is a task which would require a huge amount of resources and time (and even, some of them can only be approximated roughly), which are not available for this project.

<u>Macroscopic thermodynamic balance approach</u>

By utilizing the first law of thermodynamics for an isobaric system, Bernardi et al. [23] proposed a general energy balance equation for battery thermal models in which the heat generation rate can be compactly expressed as [12]:

$$q = -\sum_l I_l T \frac{dU_l^{avg}}{dT} + \left(\sum_l I_l U_l^{avg} - IV \right) + mixing\ term + phase\ change\ term$$

where U_l^{avg} is the average open-circuit potential for reaction l evaluated at the average compositions.

By reviewing the literature, it can be seen that the previous equation is the *most prevalent heat generation model* used for simulating the heat behavior of batteries at the cell level [5]. Generally, the mixing and phase-change terms are ignored due to the low contribution that they provide (less than 1%) to the total heat generation. Moreover, in the case of the LFP chemistry it can be assumed that only one reaction is occurring. Hence, the simple form described by Gu et al. [20] can be adopted. The equation yields

$$q = I(U_{OC} - V) - IT\frac{dU_{OC}}{dT}$$

The first term accounts for ohmic loss inside the battery cell, the charge transfer overpotential at the interface, and the mass transfer limitations, where U_{OC} is the open-circuit potential of the whole cell, and V the operating potential. The second term accounts for the entropic effects, or reversible heat term, where the term $\frac{dU_{OC}}{dT}$ is known as the entropic coefficient. Doing an analogy with an electrical circuit, the overpotential can be expressed as an internal resistance, such as:

$$q = I^2R_{int} - IT\frac{dU_{OC}}{dT}$$

This is the model that is going to be employed, as the open-circuit potential, U_{OC}, the operating voltage, V, and the entropic coefficient, $\frac{dU_{OC}}{dT}$, can be characterized under several different conditions in the lab facilities of Fraunhofer ISE for the cells under study. Note that *all those variables depend on the working state*, which is usually characterized by the State-Of-Charge (SOC), the applied current –usually expressed in terms of C-rate–, the temperature, and the time (due to the transient responses of the cells).

Introduction to Comsol FEA Software

Introduction

Comsol Multiphysics is an interactive environment for modeling and solving all kind of scientific and engineering problems.

Multiphysics refer to simulations that involve multiple physical models or multiple simultaneous physical phenomena, and this program is especially conceived for those kinds of problems. Multiphysics simulations typically involve solving coupled problems of partial differential equations, which may combine finite element methods with molecular dynamics, or different physics such as chemical kinetics and fluid mechanics.

Comsol Multiphysics is one of the commercially available software packages to do so, whereas other alternatives could be Ansys, LS-Dyna, Abaqus, or Altair, and also the open source packages of Elmer, MOOSE and Advanced Simulation Library. Several physical equations and mathematical models are already implemented in the physics interfaces of Comsol, and using those built-in interfaces, it is possible to build complex models by only taking care of defining appropriately the physical quantities, such as material properties, loads, constraints, fluxes, and sources.

Comsol assembles and solves models using SOA numerical methods for either linear or nonlinear problems, and, meanwhile different discretization options are used in some of the add-on modules –including finite volume method, boundary element method, and particle tracing methods– the emphasis is put on the finite element method. Many types of finite elements are available, and fully coupled elements are automatically generated “on-the-fly” by the software at the time of solving. This allows for unlimited Multiphysics combinations, and might be the essential distinguishing characteristic of this software in comparison to its competitors. Moreover, error control and adaptive meshing, in between other advanced numerical algorithms, can also be added to the simulations for stationary, transient, or modal or frequency domain studies.

The problem or system of equations, or PDEs, that the software solves depends on the physics interfaces that are included in the model definition. Comsol has more than 30 add-on products to choose from, with which the simulation platform can be expanded with dedicated physics interfaces and tools for electrical, mechanical, fluid flow, and chemical applications.

There are also some packages for interfacing Comsol with other CAD or analysis programs, such as Matlab. Every one of the add-on products requires a license, and might include different modules, physics interfaces, or functionalities.

Getting started

Comsol Desktop: Getting started. Building a model

The physical models are usually built through the Comsol desktop GUI, in which all the tools for creating, analyzing and visualizing the models are easily accessible by the user. The possibility to build and modify the models from coding is available through the LiveLink for Matlab, which we do not have license for. On the other hand, the software is based on Java language, and the models can alternatively be directly modified from their Java source code and then compiled using a Java Developer Kit (JDK) compiler before running them in Comsol.

When the program is started, the Model Wizard is opened, and here the user must select the most basic features of the model: the space dimension of the model (3D, 2D Axisymmetric, 2D, 1D Axisymmetric, or 0D), the physic interfaces to be included (one or more predefined physic equation nodes), and the study type (stationary, time dependent, ...). After doing so, the Comsol desktop GUI is opened –the main modeling tool–, a screenshot of which is included hereafter:

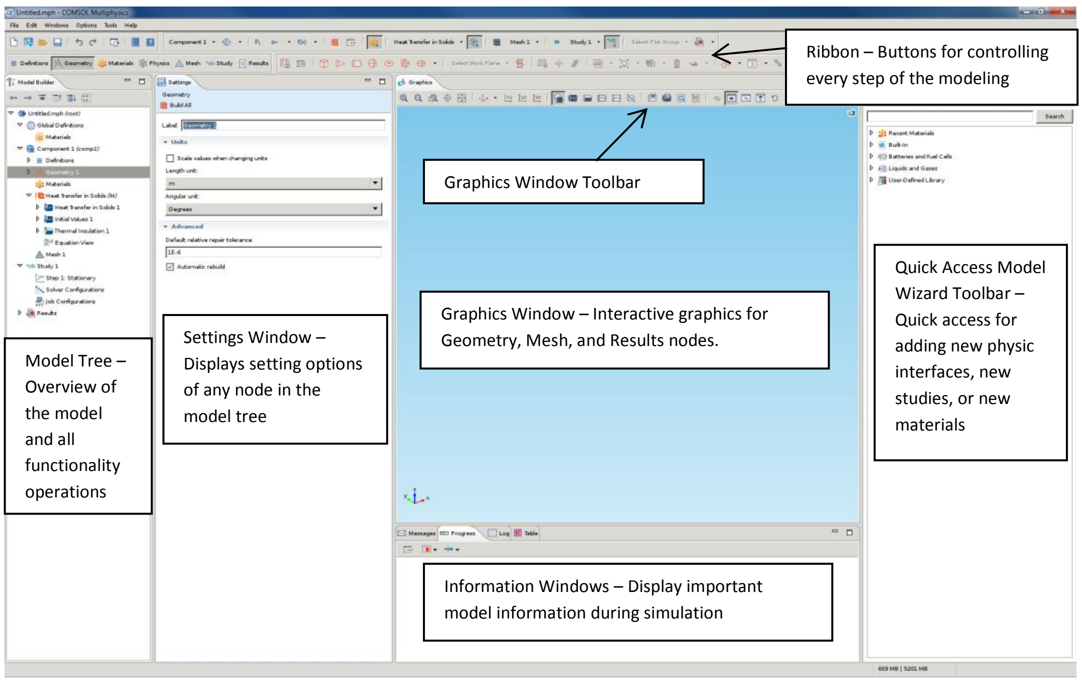

Ribbon – Buttons for controlling every step of the modeling
Graphics Window Toolbar
Quick Access Model Wizard Toolbar – Quick access for adding new physic interfaces, new studies, or new materials
Graphics Window – Interactive graphics for Geometry, Mesh, and Results nodes.
Settings Window – Displays setting options of any node in the model tree
Model Tree – Overview of the model and all functionality operations
Information Windows – Display important model information during simulation

Figure. 6: Screenshot of the Comsol Desktop GUI environment.

A model tree always has a root node, with the name of the model file (initially *Untitled.mph*); a global definitions node, where global parameters, variables, and functions can be defined; and a results node, where the results of the simulation can be analyzed, stored, displayed and exported.

In addition to the three general nodes previously described, there are two additional top-level, or parent, node types: component node, and study node. Both are usually created by the Model Wizard, before the Desktop being opened, and a model or simulation file can include one or several of these nodes. For example, this can be useful if we want to study both the stationary and the frequency domain behaviors of the same physical model, or if a model component wants to be coupled with other components, or submodels, to form a more sophisticated simulation.

Within the component node, the structure includes, and by this order from up to down: local definitions, geometry, materials, physic interfaces (such as heat transfer in solids, or any other), and meshing features. Note that the workflow and the general model tree structure are unique, regardless of the application area and the add-on products which are engaged to every model, resembling other available software packages, such as Ansys (Workbench), or Abaqus.

Building a model is equivalent to building a model tree, or a set of sequences that contain all the geometry, mesh, studies, and settings definitions. This is usually done by modifying the default model tree –adding or subtracting nodes, and editing the node settings–, the structure of which mimics the user workflow and gives and overview of the functionality available for each modeling step. Every node in the model tree can have various children nodes that can be added beneath them, by selecting them from the list displayed when we right click on the parent nodes. When we click a child node, then the settings window is displayed and the feature settings can be edited.

Documentation and tutorials

In order to get started with the software usage, workflow and capabilities, a lot of documentation has been consulted, and several simple model examples have been developed to familiarize with the physics interfaces which might be necessary during the project –described above.

Official written documentation which has been consulted includes:

- *Introduction to Comsol Multiphysics (5.1)* provides a quick overview of the Comsol environment, including simple examples to get initialized with the GUI workflow. This guide might be the first lecture for new users, for its simplicity and didactical structure; meanwhile quite advanced topics are also introduced.
- *Comsol User's Guide (version 4.3)* is the most extended and complete documentation guide.
- *Comsol Reference Guide (version 4.3a)* is the detailed guide that includes the detailed information about all the general tools of the software –geometric modeling, meshing features, definitions, and results analysis and plots–, files format, but also gives details on its implementation principles –the finite element method, stabilization techniques–, working sequences, solver features, and advanced solver topics –such as preconditioning for iterative solvers–.
- *Comsol API. For use with Java* is the reference manual for the Java language of Comsol. All the information regarding the internal language that every model has, and all the available commands, is included in this guide. It is only interesting in case one might make some modifications in the Java model files.

- *Essentials of Postprocessing and Visualization in Comsol Multiphysics* is a very useful guide for dealing with the results analysis, evaluation and plots, and data export.

On the other hand, different kinds of tutorials can be performed, as:

- *Application libraries*, which we can access from the same Desktop of the software, through File > Application Libraries. These libraries are ordered by the physics branch they belong to, i.e., AC/DC Module, CFD Module, or Heat Transfer Module, and consist basically of a set of explanations of different models which are pre-built and previously solved with Comsol. Every example includes a PDF description of the model objectives, physics contextualization, and results analysis and observations, and a complete set of step-by-step instructions to build and analyze the existing model. Additionally, some pre-built model sequences and .mph files are included, sometimes, and we can open or import from the Desktop the already built-in model. This is one of the most powerful learning tools for the software, as it includes dozens of examples for every physics interface module.
- Webinars, which can be accessed for free from the main website of Comsol. This is also a powerful learning tool, especially for introduction to the software and for learning basic features, but every webinar –which lasts for approximately 40 to 60 minutes– also includes, nearly always, some complex concepts, special tricks and/or important recommendations.

Moreover, the website provides other support services, such as the Comsol community, which mainly consists of a Discussion Forum in which a wide variety of problems and modeling issues are discussed. It is linked to a Model Exchange Forum, where the models can be easily shared with other users to facilitate the discussions. A lot of good practice tips are explained in this forum, although it is sometimes quite difficult and messy to find appropriate recommendations for concrete issues. For personalized doubts, a Technical Support service through mailing is included with each license, which is proved to be a very fast and efficient service.

In order to summarize this huge amount of information, some of the most interesting features and capabilities of Comsol, or, more precisely, the ones that have been most useful in my project development, are resumed in the next section *Acquired knowledge, tips and tricks*.

Comsol for Thermal Modeling of Li-ion Batteries at ISE

The products which are covered by our licenses are the CFD Module and the Batteries and Fuel Cells Modules, which add diverse physics interfaces to the core and common physics which are provided by default with the general software license.

For thermal modeling of electrochemical cells and battery packs, which has been the main assigned task, the most adequate available physics interfaces to study and get familiarized with are selected to be:

- Heat Transfer physics branch will surely be used. Heat transfer in solids, in fluids, and in porous media is included, as well as conjugate heat transfer (for laminar and turbulent flows). Interestingly, electromagnetic heating is also implemented and provided.

- Electrochemistry branch includes several physic interfaces for simulating the transport of active species, electrode kinetics, and current balances in the electrolyte and electrodes, as well as other minor physical events that are behind the working principles of any electrochemical cell or that can be related with its simulation.

- AC/DC might be used for electrical currents, which might be simulated within the battery cells or battery packs.

- Fluid Flow branch might be used in case liquid or air cooling studies would be included in the scope of the work. This branch includes laminar and turbulent single phase flow formulations, but also multiphase and thin film flows, porous media and subsurface flow, and high Mach number flows.

All of those physic interfaces consist basically of the already implemented conservation equations, but include besides a lot of capabilities and built-in boundary conditions definition possibilities. The objective is to help the user to define complex physical problems without needing a deep knowledge on the mathematical formulation of the problem.

Acquired knowledge, tips and tricks

After reading most of the documentation, and the completion of some of the model tutorials, some of the features and capabilities of the software are in this section highlighted, in order to try to improve the learning procedure of future workers. For that reason, in this section the most relevant concepts, features and methods of the software that have been acquired during the project development are summarized.

Geometric features

Since the licenses for interfacing CAD programs to Comsol (named *LiveLink* products) are not available at the moment, the geometrical definitions of the models will need to be defined in the included geometry builder. Geometry sequences are created by adding and building geometry features (cubes, cylinders, spheres, etc.), editing them, and using geometric transforms and conversions.

Some external geometry files can be imported in the geometry node, with formats such as .stl, .wrl, or .vrml. However, this is not recommended, for when we import geometries in such a manner the resulting geometry is limited to consist of only a single domain. Even if the part is built as a junction of bodies or solids –in any CAD software, such as Autodesk Inventor–, Comsol will interpret the imported object as a single object or domain. This might not be adequate in most of the cases, since different materials, or different initial values, could not be distinguished from one part to others of the domain, since the whole imported object is considered as a single domain. Partitioning the imported object would be an option, by making use of the Partition geometric operation of Comsol, but requires building a Work Plane for every desired partition.

Similarly, we can directly import a meshed object, in .stl, .wrl, .vrml, .nas, or .bdf formats, but the similar inconvenients arise. Moreover, only Nastran formats can be imported as a volumetric mesh, while .vrml and .stl files are only used to import a surface mesh. It can be concluded, thus, that really useful CAD import functionalities are exclusive for licenses including LiveLink interfaces.

When building any geometrical model, it is highly recommended to specify appropriately the units, as well as to work with parametrized geometrical models since it makes the models much more versatile and robust for any later modification. The parameters ought to be defined under Global Definitions > Parameters, and can be saved or exported as a .txt file, for editing it externally, or for importing them in other models.

Similarly, any geometrical sequence can be saved or exported as a binary file, which will adopt the format *.mphbin*, by right clicking on geometry node, and selecting the export option. Afterwards, the exported geometrical sequence can be imported in any other model.

Tips for geometry selection: it might be difficult to get along with the GUI tools to efficiently create geometric selections. Wireframe rendering is suggested. Then, *Explicit selections* are very useful, if we have several groups or geometry objects which can be grouped, and recall that domains, boundaries, and edges can be typed in by using their numbers, by using a spare *Explicit selection*, and clicking on the *Paste* icon.

Expressions definition

There exist basically three different kinds of expressions that can be user-defined and linked to our simulations, namely parameters, variables, and functions. However, their definition has some limitations and differences, which are explained hereafter.

Parameters are defined under the Global Definitions node, and are global in nature. Important common uses of parameters include: definition of constant material properties, geometrical dimensions, or mesh sizes; or preparing the model for further parametric sweep studies. Parameter expressions may depend on: numbers, other parameter values or built-in constants (i.e., *G_const*, which takes the value of the gravity, or *pi*), built-in functions of parameter expressions, such as *cos*, for the cosine, or unary and binary operators. However, parameters cannot depend on global variables as the time, *t*, or the space coordinates, *x,y,z*, nor on the dependent variables the model solves for.

Variables can be defined as global or local, in the latter case, under the Component > Definitions node. Typical uses of variables are material properties definition, or boundary conditions and interactions between dependent variables specification.

Unlike parameters, variables can depend on other variables, also on time and spatial variables, as well as on dependent variables, such as, i.e., the temperature *T*, if we are simulating a heat transfer problem. On the other hand, variables cannot be used in Geometry, Mesh, or Study nodes, with the only exception of “Stop Conditions”, under Study node.

Functions can also be globally or locally defined, and allow the same dependencies as the parameters, with the exception that functions might also include argument as an input of the expression. Those arguments can be any kind of variable in the model, including time, and spatial variables. Functions can be analytically defined, or, alternatively, can be built from several already built-in templates that are included, such as step functions –in which we can tune the initial and final values, the abscises of the step, and the regularity of it–, piecewise functions, ramps, waveforms, random, and Gaussian pulses in between others.

An important general remark, for any expression definition, is that only scalar expressions are permitted. Vector expressions are not allowed.

i. Interpolations and look-up tables

Interpolations are one of the most interesting functions that can be incorporated to our model, by adding an *Interpolation* function node and defining the table values. By the use of this feature, look-up tables can be added as interpolations to our model. There are two ways of defining interpolations: first, by manually entering the data, keeping *Data source* –in the settings window– as *Table*; or second, by importing the interpolation data from an external file, by selecting the *Data source* to be *File*. Accepted format files are .txt, .csv, or .dat.

When trying to import the interpolation from a file, Comsol requests for the *Data format.* There are three file types that Comsol can read for interpolations: grid, sectionwise, and spreadsheet, whereas the latter is the predefined and the easiest format to use. A remarkable fact is that interpolations or tables are limited to a maximum of three independent variables, and the only way to define interpolations of more than one variable is by importing a numeric file, as explained just before. When the data format is specified, the path and name of interpolation data must be specified in Comsol, by clicking on *Browse*. Finally, click on import.

The Spreadsheet format (documented in the File Formats chapter in the *Comsol Reference Guide*) must contain the arguments in the first columns and one or more functions in the following columns. For example, for defining a two variable function:

```
x1 y1 f(x1,y1)
x2 y2 f(x2,y2)
       ...
```

Remark: The grid data must be ordered in increasing order, likewise it is common in Boolean algebra tables: from the first to the last dependent variable, each variable ordered from small to big values. Otherwise, an error will arise.

The fastest way to create this format files from an Excel sheet table is to copy the desired columns and rows into Notepad or any other text editor. The desired format is automatically obtained, and it can be saved as a .txt, or .dat file.

Alternatively to the spreadsheet, the grid format can be employed, and its structure is:

```
%grid
x1 x2 x3
y1 y2
%data
f1 f2 f3 f4 f5 f6
```

where `f1=f(x1,y1), f2=f(x1,y2),f3=f(x2,y1)`, and so on.

Once the interpolation data has been introduced, different kind of built-in interpolation –nearest neighbor, linear, piecewise cubic, or cubic spline– and extrapolation methods –constant, linear, or nearest neighbor– can be selected to obtain and/or plot the function that we can later call and evaluate within the computational model.

If we want to save or export interpolation data that we have introduced manually in the program, we can use .txt or .dat formats, which will then save the data as

```
x1 f1
x2 f2
```

Otherwise, if we save it as .csv the space between columns will be replaced by commas. Note that units and decimals are separated in Comsol with ".", so the .csv exported format might not have the perfect formatting for reading it in Excel, and thus requires some preprocessing.

ii. Using Units

All the expressions described just above –parameters, variables, and functions– accept "units" definition. Appropriate "units" definition is a good practice, first of all because the physics interfaces inputs request for proper units, and it points out a Warning (definition becomes yellow) when the unit input is not the expected; secondly, because the software does not allow us to add or subtract magnitudes of different units; and finally, because unappropriated "units" definition might yield to errors in the model solution, since the software makes, silently, unit conversion operations which might not be desired in some cases, and which might be difficult to find out.

The units ought to be defined using the appropriate format, which is typing the appropriate unit name in between claudators, "[ ]", i.e., defining the value of the parameter *length_one* as *20[mm]*. Comsol supports a number of consistent unit systems, and uses by default the SI unit system. For a complete list of the supported units, and its symbol representation, see *SI Base, Derived, and Other Units* chapter in the *Comsol User's Guide*. However, most of the symbol representations are intuitive and can be found rapidly by quick check. Example: meter, centimeter, and millimeter are written as[m], [cm], [mm].

If a physics interface input requires some units, but the input we want to introduce does not have the same dimensional units, we can convert the units by adding some units definition to the expression. Example: Let's suppose that we have defined the parameter *q* as a power of *10[W]*, and we want to use this variable for a surface heat flux boundary condition, which requires for a number or an input expression in W.m^{-2} units. If we do not want to change the units of the variable *q* definition, the simplest alternative is to type in: *q[m^-2]*, which will be read and interpreted by Comsol as 10 W.m^{-2}.

iii. Check variable names and units

For checking the units of any variable we don't know what physical quantity it refers to, for any reason, probably the quickest way is to try to plot this unknown variable, in the postprocess *Results* node. This checking procedure is also useful if we do not know, for example, how the heat flux on a surface is called in the Heat Transfer physics interface, but we want to use it in some expression definition.

Add some plot group (surface, slice, contour,…), in any possible dimension (3D Plot Group, 2D Plot Group,…), and click on its generated node. Then, in the Settings Window, press the button and explore through the pop-up menu until you find the appropriate variable. When we select any variable, the Comsol internal variable name and the variable units are automatically added in the Settings Window. By doing so, we can quickly verify that the convective heat flux in z direction is defined internally in Comsol as *comp1.ht.dfluxz*.

iv. Interesting operators

In between others (extended documentation can be found at the *Comsol User's Guide,* chapter *Global and Local Definitions*), some of the most interesting built-in functions for expression definitions are:

- The `up`, `down`, and `mean` operators, to evaluate an expression on the upside or downside of the boundary, or to evaluate the mean value at the boundary. If the variable or expression to evaluate is not defined in one of the boundary sides, the operation returns 0.
- `dest`, for use in integration coupling expressions to evaluate its input on the destination points instead of on the source points.
- The `with` operator, for accessing any solution during results evaluation, i.e., at different time steps, for any parameter value, or any eigensolution; and the `at(t0,u)` operator, which can access a solution `u` of a time-dependent problem at any time, `t0`.
- The `timeint` and `timeavg` functions can be used to integrate or average at any time a function or expression within a defined time interval.

- The `if(cond,expr1,expr2)` operator to implement a conditional expression.
- The `reacf` operator, which is available for calculating integrals of reaction forces or fluxes. For example, in structural mechanics, `reacf(u)` and `reacf(v)` give the x and y reaction forces.

Model couplings

Model couplings (detailed in chapter 4 of the *Comsol Reference Guide v4.3a*) can be used to couple and map quantities across spatial dimensions, between different parts of a model or between different models. They are defined by *coupling operators* – which can be found and defined by right-clicking on Definitions node > Component Couplings–, which take an expression as an argument. When the operator is evaluated at a point in the destination, the value is computed by evaluating the argument on the source.

This allows easily setting cross-dimensional simulations. For example, a 2D solution can be mapped to a 3D surface or extruded throughout a 3D volume. This feature is really interesting for simulation of batteries, because of its nature, and because most of the SOA simulation approaches, i.e., the so-called multi-physics multi-domain approach developed at NREL [21], include different scale, different dimension spaces, and different physics which interact between each other. The model couplings can also be very interesting for postprocessing aims, if, for example, a solution wants to be plotted on a mapped surface.

Specifying and defining materials

How to save the custom materials, how to define them, and to add properties?

All the material databases are accessed from the Material Browser, which incorporates a large number of built-in material definitions. When different licenses are added-on, some specific materials are added to the Material Library.

When including a material to the model, it is automatically set to all domains. However, different entities can have different material properties, and we must take care of a proper assignment of the materials to the geometric entities. For evaluating and plotting material properties, you can access them like other variables using the appropriate naming conventions. For example, `root.material.rho` is the density, as defined by the materials in each domain in the geometry.

We can customize the existing materials by adding or modifying the predefined material properties, or even define complete new materials (from Blank Material), by using parameters or defined expressions to give value to the material properties. The resulting user-defined materials can be stored for later use in the *User Defined Materials* library. We can also create new Material Libraries in the Material Browser, and they are usually stored as .xml files.

If anisotropic properties are desired, recall that the anisotropic coefficient is a tensor of at most four components in 2D and nine components in 3D. At any coefficient, or material property of a material that we want to define, we can select *Isotropic,* then we must only enter one value; *Diagonal,* then we must enter the diagonal components of the anisotropic property with the main axes aligned with the model's coordinate system; *symmetric,* then we must enter a symmetric matrix using the diagonal components and the upper off-diagonal components; or *Anisotropic*, which will require that we enter the full 2-by-2 (2D) or 3-by-3 (3D) matrix of the anisotropic tensor.

Meshing

A mesh node is added by default when a model tree is generated. However, more meshing sequences can be added to a model, and they will be gathered under a parent *Meshes* node.

By default, physics-controlled mesh, if available with the physics interface which is being used, creates a mesh that is adapted to the current physics settings in the model. This is the fastest way to generate a mesh: using physics-controlled mesh, we only have to adjust the element size –which sets the approximate element size of the mesh generator– to some value.

Tetrahedral elements (tets) are the default element type in Comsol, since they are a *simplex* in the sense that they are the only element type that can mesh any arbitrary 3D volume. They are also the only element type with which adaptive mesh refinement can be applied.

However, other element type meshes can also be created, and also tets custom meshes. Customized meshes are a user-defined sequence of steps, built using the buttons in the mesh toolbar or by selecting the available features that appear when right-clicking in the mesh node. The meshing algorithm usually requires some more user input to create such a mesh, so before going through this effort, you need to ask yourself if it is motivated. While the pyramids are only used when creating a transition in the mesh between bricks and tets, there exist many motivations behind the use of brick and prism elements.

First, bricks and prisms can sometimes reduce a lot the number of dofs, since they can have very high aspect ratios (ratio of longest to shortest edge), whereas the algorithm to create a tet mesh will always try to keep the aspect ratio close to unity. It is reasonable to use high aspect ratio brick and prism elements when you know that the solution varies gradually in certain directions or if you are not very interested in accurate results in those regions because you already know the interesting results are elsewhere in the model. The other significant motivation for using brick and prism elements is when the geometry contains very thin structures in one direction, such as an epitaxial layer of material on a wafer, a stamped sheet metal part, or a sandwiched structure, such as an electrochemical cell. Whenever your geometry has layers that are about 10^{-3} or so times thinner than the largest dimension of the part, the usage of bricks and prisms becomes very highly motivated.

In order to create prismatic meshes in 3D volumes, the easiest and most common procedure is: first, create a *Free Quad* on a boundary or surface; then, add a *Swept* operation which extrudes or sweeps the previously created quadrilateral surface into the desired volume.

If we want to mix prism and tet meshes, we have to add a *Convert* feature to the interface between both kinds of element types, in order to partition the surface elements on the prismatic mesh into triangles that can, then, form part of a tets mesh.

Modeling with Java code

A very powerful and quite advanced way of modeling is through direct manipulation of the code that Comsol generates with every model that is built. Comsol is Java-based, and so it generates a Java command for every edit within a model that we make from the model builder GUI.

We can save the models in .java format and inspect the code of the model. For doing so, we should a priori reset the history of the model, by doing: File > Reset history. This will reorder all the commands that we may have chaotically generated through the GUI within the model development. Afterwards, we can save the final state of the model, with the steps logically grouped, by: File > Save as Model Java-file.

Once exported the Java code, we can edit the file in any text editor, and add entries or comments to the generated code. Finally, for allowing the manipulated model code to be run in Comsol, we must compile the file by the use of a Java Developer Kit (JDK) compiler, available for free from Oracle, by using the command

```
sh /…/comsol compile /…/model.java –jdkroot /pathJDKcompiler/
```

The compiled file will then adopt a .class format, which we can import and open from Comsol GUI. Alternatively, we can also run the compiled code in batch mode.

This way of working can save a huge amount of time in some cases, since it allows to copy and paste anything from one model to another, change dimensions of a model, and access to some advanced tools and features which might be more difficult to manipulate from the GUI. Java syntax is not needed to work with Comsol generated files, since they are logically ordered, and can be modified by following some examples.

For more information on the usage and control of Comsol vy using the application programming interface with Java programming language, all the available commands and features can be consulted in the *Comsol API for use with Java* guide.

Runing a simulation in batch mode

For running a model from Linux shell we have to use the command:

```
sh /…/comsol batch -inputfile /…/InFile.mph -outputfile
/…/OutFile.mph -batchlog /…/OutLog.log
```

where the InFile.mph is the model we want to run, and OutFile.mph is the file that Comsol will generate after running the simulation, and which will include the results of the simulation. OutLog.log contains some registrations regarding running information, which might be important in case some problems occur or simulation times get too large.

Take into account that we should generate a Batch node in the model before running the simulation in batch mode, and specify the study that we want to perform, and the settings for it. We can run either .mphbin and java code compiled .class files.

Other very useful additional flags than we can use, apart from `-batchlog`, and the mandatory input/output paths specification, are:

- For specifying the number of processors and computational nodes: use, i.e., the flag `-np 6` to specify that six processors must run the simulation; and for setting `-nn 2` to set the number of computational nodes to two. Normally Comsol assigns the simulation to the maximum number of computational nodes and processors that are available, because of convenience, but it might be interesting to set this in some cases.
- For running parametric sweeps from batch: we first have to make sure that the parametric sweep node is inserted within the model (through GUI modeling). Then we can use the flags `-pname` and `-plist` and run the parametric sweep from batch for any parameter which is included in the parametric sweep node, and with any value or list of values we want to specify, e.g.: `-pname Param1,Param2 -plist Value1,Value2`

For the sake of completeness, one can access to a brief description of all the available flags for Comsol command by typing: `sh /…/comsol -h`.

One of the interesting capabilities that can offer running simulations in batch mode is that we can automatically export results, such as tables, figures, or derived values after running a simulation without needing to do it through the GUI, which is time-consuming, by adding the appropriate lines to the java code file before compiling it.

Customized physics

Any predefined physics interface might be customized to particular simulation objectives by adding some additional terms or dependencies to the preset equations. For example, nonlinearities might be added, under local definitions node, by the inclusion of equations that depend on the dependent variables. Furthermore, global, domain, or boundary ODEs and DAEs can be added to influence on any feature of the model, by the addition of Mathematics > ODE and DAE interfaces.

Another feature for customizing any physics interface is the possibility to add weak contribution expressions on domains, boundaries or edges. The weak contributions will be automatically added to the weak form of the governing equations.

In more customized studies, or in the case of arbitrary mathematics or physics simulations, where a preset physics is not available, it is included with the general Comsol license a set of physics interfaces, under the branch of Mathematics. These physics interfaces consist of templates through which the user can set up simulations by defining the PDE equations from the principles.

There are basically three different ways to specify customized equations and boundary conditions, under Mathematics > PDE Interfaces, from simpler to more complexes, namely:

- the Coefficient Form, where the problem is specified through the setting of the coefficients $e_a, d_a, c, \alpha, \gamma, \beta, a$, and f in the general system of PDEs:

$$e_a \frac{\partial^2 u}{\partial t^2} + d_a \frac{\partial u}{\partial t} + \nabla \cdot (-c\nabla u - \alpha u + \gamma) + \beta \cdot \nabla u + au = f$$

- the General Form, in which the equation is specified through the definition of a flux function, Γ, in the so-called conservation-law formulation of the same general PDE equation, written as:

$$e_a \frac{\partial^2 u}{\partial t^2} + d_a \frac{\partial u}{\partial t} + \nabla \cdot \Gamma = f$$

- the Weak Form, the most general form to specify equations, where the PDE is defined through the definition of integrands that conform the equation in variational or weak form.

These equation-based tools can further be combined with any of the existing available physics modules, allowing for customized and fully-coupled analyses, and they include interesting features to define moving or deformed meshes or interfaces, and sensitivity analysis, in between others. This physics interface might also be used, and, thus, studied, in case the existing or available preexisting modules do not fit with the simulation objectives.

Customizing the physics interfaces with complex expressions, weak contributions, or by the addition of newly defined PDE interfaces requires some knowledge on the finite elements method and the mathematical formulation or derivation of the variational form. However, one should also get familiarized with the language that must be used in the definition of expressions and operators in Comsol, so that the software can interpret the expressions adequately. For doing so, a look at the complete list of mathematical and other operators included in the Appendix C of the *Introduction to Comsol Multiphysics (5.1)* ought to be taken.

For convenience, partial derivatives syntax is here also highlighted:

For any stated dependent variable, all the partial derivatives with respect to spatial and temporal coordinates, up to the second derivative –included–, are generated as built-in variables. Example: T is the name for the temperature, in a heat transfer model. For computing spatial derivatives one can write: *Tx, Tt, Txx, Txy, Txt, Txtt, Tyytt,* or alternatively, *d(T,x), d(d(T,y),t),* etc.

In addition to what appears in the previous appendix, it is necessary for writing weak form expressions to know the syntax for test functions, which will be explained throughout an example:

The common integrand of a variational form expression

$$\int_{\Omega} \nabla v \cdot \nabla u \, d\Omega$$

with $u \in H^1(\Omega)$ being the dependent variable and $v \in H_0^1(\Omega)$ the test function for u, would be written in Comsol syntax (within a Weak Form expression) as

```
test(ux)*ux
```

Another useful built-in function for modeling using the weak form, is the `nojac`, which makes sure that any expression that it operates on is excluded from the Jacobian computation.

Useful Shortcuts

F1	Display help of selected node
F2	Rename selected model tree node
F3	Disable selected model tree node
F4	Enable selected model tree node
F8	Build geometry and mesh, compute solver sequence, update results and plots
Supr (Entf)	Delete selected node
Ctrl+A	Select all domains, boundaries, edges or points; select all cells in a table
Ctrl+F	Find concurrences of variables through the full model tree. Double click on them to jump to the relevant node
Ctrl+P	Print the contents of the plot window: labels, titles, and legends can be included in the exported image, or not. Export formats: .pdf, .ps, .svg
Ctrl+S	Save the model
Ctrl+Z	Undo
Ctrl+Y	Redo

Other shortcuts can be found in the documentation in the Appendix B of *Introduction to Comsol Multiphysics 5.1*.

Others/ Suggestions

i. Effective Memory Management

Especially in 3D modelling, extensive memory usage dictates some precautions. First, check that an iterative linear system solver is selected. Normally, the choice of physics interface carries a proper solver selection so the user does not have to care much, but revising this might be necessary in some cases. The MUMPS and PARDISO out-of-core solvers can make use of available disk space to solve large models that do not fit in the available memory.

Second, if large models drive to out-of-memory error messages, try to simplify or reduce the model geometry as much as possible, finding symmetry planes. Try to avoid small geometry objects, avoid geometries with sharp, narrow corners, use linear elements if possible, and make sure that the mesh elements are of a high quality. Mesh quality is important for iterative solvers: convergence is much faster and more robust with a good quality mesh.

ii. Analysing convergence and accuracy

It is important that the finite element model accurately captures local variations in the solution, such as stress concentrations in a solid mechanics problem. If a model has not been verified by comparison to other established results, a *convergence test* is useful for determining if the mesh density is sufficient. For doing so, refine the mesh and run the study again, and then check if the solution is converging to a stable value as the mesh is refined. If the solution value changes while mesh refining, then the solution is mesh dependent and either the model requires a finer mesh (adaptive mesh might be used, which refines the mesh in specific locations using numerical error estimates), or there exist one or more singularities in the geometry.

iii. Facing nonlinear problems

In nonlinear problems might exist more than one solution. Comsol makes use of a Newton iterative method to solve nonlinear systems of PDEs, and this method might be sensible to initial solution estimates for converging to a proper solution. Different things can be done to improve the chances for finding the relevant solutions to complex nonlinear problems:

- Provide as good as possible initial values
- Solve sequentially and iterate between single-physics equations. Finish by solving the fully coupled multiphysics problem when you have obtained better starting guesses.
- Ensure that boundary conditions are consistent with the initial values
- For convection-type problems, introduce artificial diffusion to improve the problem's numerical properties (see *Stabilization Techniques*, in *Comsol User's Guide*). Most fluid flow interfaces and chemical species transport provide artificial diffusion as default settings.
- Turning a stationary nonlinear problem into a time-dependent problem generally results in smoother convergence. Make the time interval span enough to reach a steady state solution

Use parametric solver and vary the material properties or PDE coefficients starting from values that make the equations less nonlinear, and keep varying them progressively to the desired values, using every parameter step solution as the initial value for the next one.

iv. Equivalent Circuit Modeling would require License extension

SPICE Circuit Netlist file types could be imported if we had one of the AC/DC, RF, MEMS, Plasma, or Semiconductor Modules. When importing a circuit (format .cir), it would be automatically converted to a series of lumped circuit element nodes under an *Electrical Circuit* node. In any case, having the possibility of using the *Electrical Circuit* node would also open the possibility of simulating the batteries as equivalent circuits, without needing to wonder about finding closed solutions for those circuits and implement them as a function or a group of functions. Recall that equivalent circuit networks can also be used for many other applications, i.e., for modeling a complex thermal system by a Foster's network, which may be interesting because of its simplicity and low computational cost in comparison to other alternatives.

Calorimeter Instrument Introduction and Workshop

Instrument

Calorimetry is the science of measuring the heat of chemical reactions or physical changes of a specimen.

The IBC 284 is a testing equipment tool for studying high power and large battery packs for application in electric drive vehicles or airplanes. It was co-developed by Netzsch and the National Renewable Energy Laboratory (NREL), one of the leading research organizations in the development of analytical tools for designing optimized battery thermal management systems. For measuring inhomogeneous samples, such as batteries or entire battery packs, isothermal calorimetry –or calorimetry at a defined temperature–, is ideal, since the isothermal regime simplifies the kinetics within the results, and the analysis of them.

The IBC 284 consists of a large volume analysis chamber submerged in an isothermal bath of a glycol-water mixture that permits lower temperatures to be reached without freezing. For cycling experiments, a commercial battery tester might be employed to charge and discharge the battery pack. The instrument offers the possibility to characterize the efficiency of batteries over different temperature ranges and states of charge, with a high accuracy, thanks to a very stable bath temperature control of ±0.01°C, and with a high heat flux sensitivity detection of 30mW.

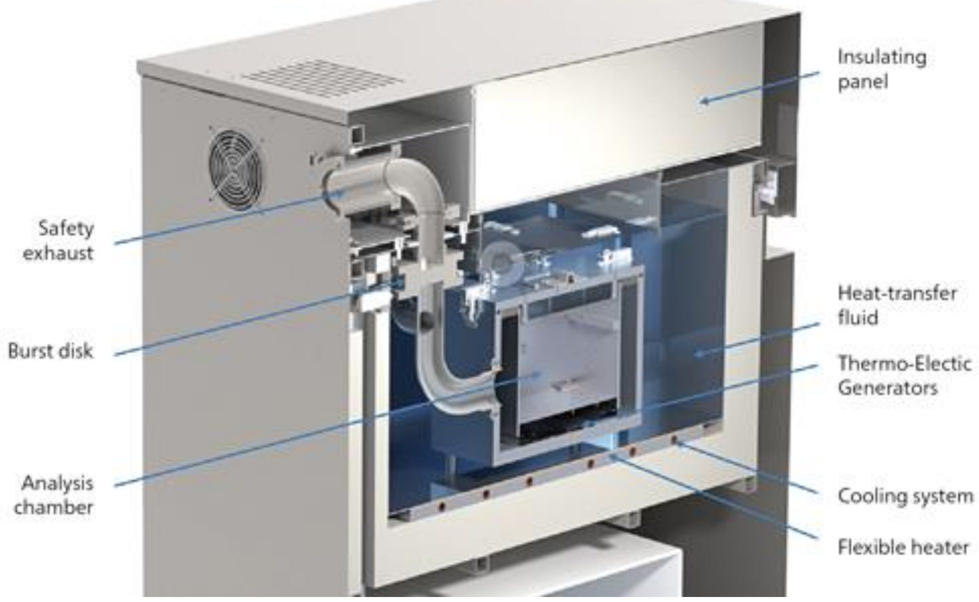


Fig. 7: Internal components of the calorimeter

The other alternative, regarding calorimetric instruments for thermal studies related to batteries are the so-called Adiabatic Reaction Calorimeters (ARC). However, ARC instruments, which try to copy or reproduce the temperature of the sample in the outer part of the sample container, for providing adiabatic conditions at any stage of the study, may have problems to deal with large samples –i.e., large pouch cells, or battery packs– since they might not be able to produce heat in the same speed as the samples. Another important disadvantage of ARC instruments with respect to isothermal calorimetry is that in ARC the generated heat is indirectly obtained, whereas in isothermal instruments the direct measurement of the dissipated heat (due to the heat flux sensors) avoids the necessity of knowing or estimating the heat capacity of the sample. On the other hand, the major advantage is that isothermal calorimeters have a much smaller temperature range (the IBC 284 working range is from -30°C to 60°C) than the ARC, which can work up to 200°C. However, this is not that disadvantageous in battery research since the interesting operating temperatures of batteries and battery packs are often comprised within the admissible range of the isothermal calorimeters.

System components

The instrument was commissioned and set up by a customer service engineer at our laboratory. A battery analyser and a computer system, which is to be held on a separate table, next to the IBC, are required for the operation of the instrument.

The main external connections to the machine are:

- Computer system is connected through two USB ports.
- Two gas network connections are needed: inert gas (Nitrogen or Argon) for the calorimeter chamber, and compressed air for level management of the liquid within the calorimeter bath.
- The battery analyser might be connected by an Anderson SB175 port.
- Safety exhaust of the IBC must be connected to an evacuation vent duct. The gases of the calorimeter inner chamber will be evacuated when the pressure exceeds 5psi.

The fluid of the thermal bath surrounding the calorimeter inner box is a mixture of Ethylene/Glycol with deionized water in a 50/50 ratio. However, since water is slightly evaporated in time due to the heating elements, the tank level must be checked every month and re-filled with deionized water (the other components do not evaporate).

The temperature control of the thermal bath consists of a cooling plate, which is placed underneath the thermal fluid bath container and connected to a refrigerating system and which has only two operating positions (high and low), and a silicone/rubber heating element whose power delivery control, via a PID temperature controller, ensures the stable bath temperature. The temperature of the bath is homogenised by convection with the use of some stirring motors that constantly mix the thermal bath. Draining or filling of the thermal bath container is done by an air compressor, and the reserve liquid tank is held under the bath container.

The box of the calorimeter, depicted in the following figure Fig. 8, is the core of the instrument. The inner container (2) is made of thin aluminium walls and it is very-well isolated to the thermal bath. The box is closed by a well-insulated lid (1) that ensures calorimeter tightness using 16 latches. In the lower part of the calorimeter inner box, a thick aluminium plate (4) is placed, which makes contact with all the heat flux gauges (7) just underneath. The electrical power is provided in the calorimeter box by two copper bus bars (6), which are rated for 350A currents, and several voltage and temperature sensors are also available inside the box, for a more detailed data acquisition. The connection from the external battery analyser to the bus bars is made in the electrical connection boxes (5). The signal wires get inside the box through the snorkel tube (3), which also serves for pressure control and for pressure safety, thanks to a rupture disk.

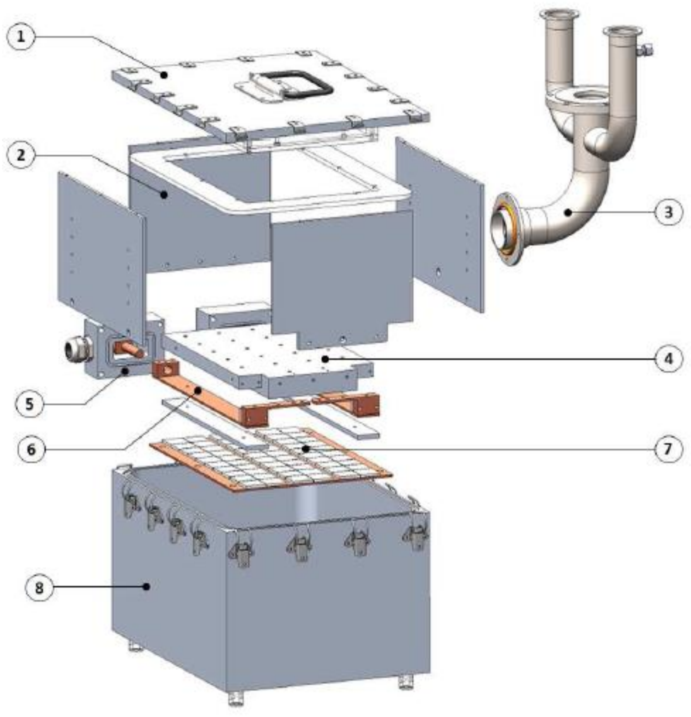


Fig. 8: Calorimeter box assembly scheme

The calorimeter has four different measurement modules; the data acquisition rate of all is of 10-20Hz:

- First, a measurement module for the heat flux gauges for measuring the heat flow from the inner box to the bath, or vice versa. The heat flux gauges are composed of 60 high sensitivity type K thermocouples that can measure from 100mW to 50W of heat. The heat flux measurement system can have a 5mW baseline noise and a stability of 30mW, and the final enthalpy measurements are ensured to have an accuracy of ±2%.
- A voltage measurement instrument is incorporated to measure the voltage at all the internal voltage sensors, and at the copper bus bars of the calorimeter box. The maximum voltage allowed is 50V and the accuracy of the data acquisition is of 1mV.
- The external input current is measured by a shunt resistor. The maximum current is 300A and the accuracy 5mA.
- Within the thermal bath there are two probes, one is sensed and read, with a readability of 0.01K and reproducibility of 0.1K. The other probe is used for the bath temperature control system, which ensures a constant ±0.01ºC temperature.

A LabView customized software is provided for operating the instrument, setting the bath temperature, and starting and managing the data acquisition and record. The software *Proteus* is employed for data analysis, and incorporates many tools specifically designed for electrochemical cells thermal analysis.

Calibration

Calibration procedure

The operation of any analysis with the instrument will be explained in this section through the calibration process and set-up.

The calibration of the IBC 284 consists in applying a controlled electrical current to a precision shunt resistance located inside the calorimeter chamber. The calibration factor can afterwards be obtained by the factor between the electrical energy spent by the resistance (100% converted into heat) and the measured heat by the sensors.

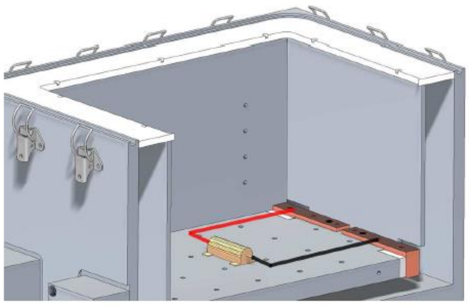


Fig. 9: Set up for Joule effect calibration with a high precision resistor.

$$Calibration\ Coefficient\ \left(\frac{mW}{V}\right) = \left(\frac{\text{Power area (J)}}{\text{HeatFlux area (V.s)}}\right) * 1000$$

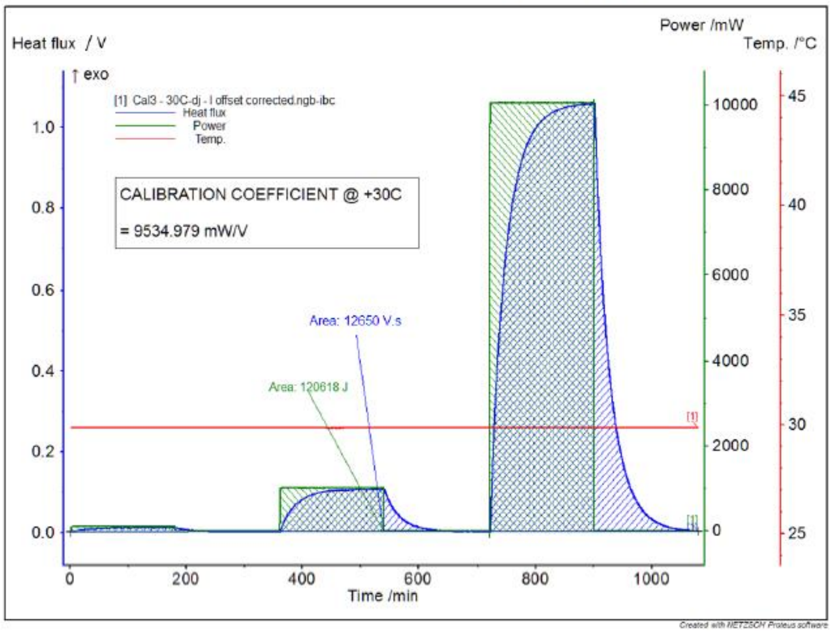


Fig. 10: Typical calibration results and calculation of the calibration coefficient.

Once the calibration power pulses has run, the electrical power –computed as $P = U.I$, using the voltage and current measurements from the input electrical power– and the heat power –from the heat flux gauges– curves are integrated in the time domain, to and the electrical and the thermal energy are compared to obtained the calibration factor. As it can be seen in the previous figure, the Joule effect calibration is repeated for three different magnitude power pulses, and finally the calibration coefficient is averaged. The power pulses have a duration of 3h, and the resting period in between every pulse must be sufficiently large so that the heat flux gauges return to an equilibrium at around 0mW baseline state (this usually requires approximately 3h, too).

The calibration factors are provided by the manufacturer at -30ºC, 0ºC, 30ºC and 60ºC, and are computed individually for every instrument. When all those temperatures have been calibrated, the calibration factor might be approximated by regression to any other temperature set up.

Calibration check

In order to verify the provided calibration factors, the calibration is suggested by the manufacturer to be checked at 40ºC before starting the instrument operation.

Before doing so, we realized that there was a small incongruence in the calibration method the manufacturer was applying. The electric power which is integrated for the calibration factor formula is computed as $P = U.I$, although it is known that the current measurement is less accurate than the voltage. For that reason, it would be better to compute the electrical power using the analogous expression $P = U^2/R$, where R is the high precision calibration resistance, since this expression might produce more accurate electrical power results. Due to that fact, finally the manufacturer decided to add and amplifier between the shunt resistor and the current measurement, to increase the accuracy of the current measurement.

Despite of that, the Joule calibration was performed at 40°C, and the results were found to be within a small error in the range of the predicted ones, so the calibration coefficient at 40°C was consistent with the approximation, by polynomial fitting, of the calibration factors at -30°C, 0°C, 30°C, and 60°C that were provided.

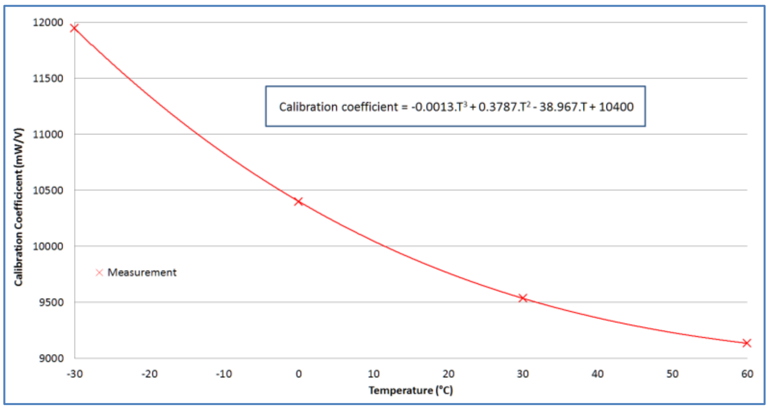


Fig. 11: Interpolation of the obtained (provided) calibration factors.

However, at ISE we stablish as a protocol to make a calibration before and after any cell experiment, even if most of the tests will be performed at 20°C, to see if the calibration coefficients –sensitivity of the Peltier elements– changes in time or if they are, as would be expected and desired, stable.

Operation and data analysis

Loading a Sample procedure

First, we need to open the top hood (2) of the instrument. For doing so, push down the cover, then pull out the panel latch (1) located behind the Netzsch logo.

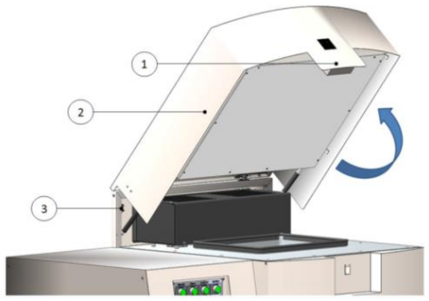


Fig. 12: Opening the hood of the IBC

Then, we have to open the calorimeter lid. Prior to that, we must make sure that the liquid level is in its lower position, by draining the liquid to the reserve tank (using the liquid lever switch), and that the pressure of the calorimeter box has been released (selecting Vent position on the pressure lever switch).

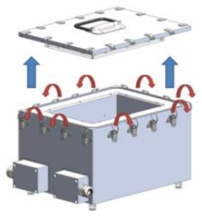


Fig. 13: Opening the calorimeter box

Then the sample can be loaded inside the calorimeter, after it has been correctly prepared for an easy electrical connection to the inner calorimeter bus bars. Always be careful to not create a short circuit by making any contact between the metallic walls and the connection wires of the battery. Isolating tape might be used to protect the battery connectors. As a last consideration, the cleanliness of the contact surface of the electrical connection between the battery and the buss bars might be ensured. Finally, when the sample is already introduced and properly connected, the temperature sensors and the additional voltage sensors can be positioned as desired.

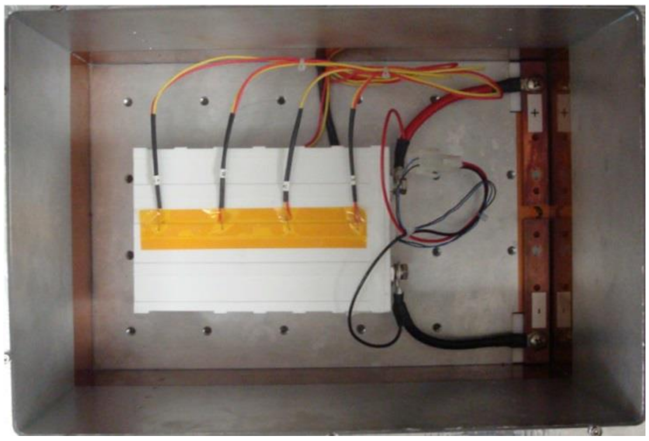


Fig. 14: Battery sample preparation. Electric and temperature sensors connections inside the box

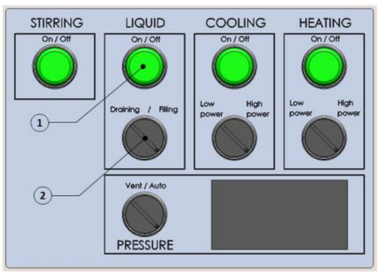


Fig. 15: Buttons in the main control panel of the IBC.

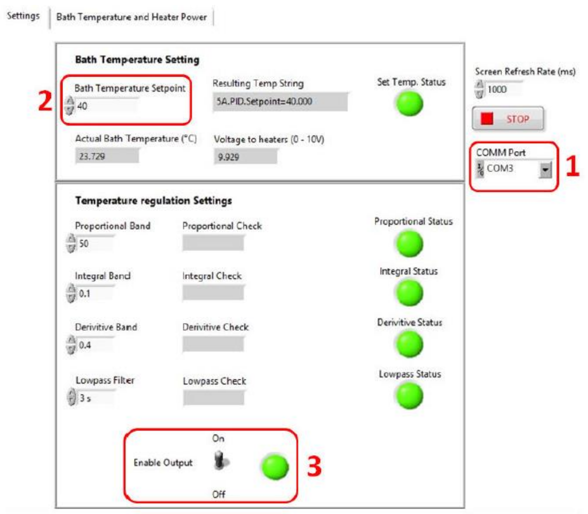
Finally, we have to close the calorimeter lid again, putting it back and closing all the latches. Then, pressurize again the calorimeter chamber, until it reaches a stable pressure of approximately 2psi, and fill the thermal bath by activating the liquid filling on the liquid lever switch.

Once all this steps have been completed, the physical setup is finished, and we only need to make sure that the Cooling and Heating modes are correct and ON, and that the Stirring (water mixing motors) is also ON for temperature homogeneity.

Experiment Setup Software

However, apart from the physical setup already described, before starting an experiment we need to set appropriately the desired bath temperature of study –using the Temperature Control Software– and set the desired path for recording all the experimental data –within the Data Acquisition Software.

For setting the desired temperature in the Temperature Control Software, first we need to verify that the communication port is the right one (named *COM3*). Secondly, we must specify in digits the bath temperature setpoint and press Enter. Finally, enable the output.



Settings | Bath Temperature and Heater Power

Bath Temperature Setting

Bath Temperature Setpoint: 40
Resulting Temp String: 5A.PID.Setpoint=40.000
Actual Bath Temperature (°C): 23.729
Voltage to heaters (0 - 10V): 9.929
Screen Refresh Rate (ms): 1000
COMM Port: COM3

Temperature regulation Settings

Proportional Band: 50
Integral Band: 0.1
Derivitive Band: 0.4
Lowpass Filter: 3 s

Fig. 16: View of the Temperature Control Software main window.

For setting the Data Acquisition Software, we need at first to define the desired file name of the experiment and browse the computer to define the location where the data should be saved. The file extension must be .ngb-ibc, otherwise *Proteus Analysis* software will not be able to process the information. The setup parameters should also be defined in this software –such as the Sample Name, the Identity, and the Operator name–, and finally we can start the data acquisition by clicking on the *Continuous Running* arrow button. When doing so, the program asks for a calibration file. Once the calibration file is introduced, the data record begins.

Data analysis

The recorded data can be analysed with *Proteus Analysis* software, which is provided with the instrument. In this software, we can plot all the recorded signals, such as:

- Heat flux: measured by the calorimeter, expressed in mW –by taking into account the selected calibration file.
- Voltage: on the battery (voltage sensors) or on the bus bars. Both measurements are provided to facilitate the data processing, since from their difference we can easily know which are the ohmic losses from the connections. Additionally, the voltage of the internal four voltage sensors can be plotted.
- Current: measured across the high precision shunt resistance.
- Power: electric power calculated from current and voltage signals.
- Battery temperature (TC1 to TC4): temperatures from the thermocouples, expressed in °C.
- Temperature of the thermal bath.

Among other analysis tools which are available, dedicated for battery analysis, the most important feature that the software can do is to calculate the time integrals of the heat and electric powers and compare both values. This feature has already been described in the Calibration chapter, and it is done by selecting *Evaluation > Efficiency* in the menu.

The following figure shows a typical response of a battery full discharge. The total amount of heat generation can be computed by integrating the curve of the heat flux in time, from the starting stable point to the steady-state relaxed point (usually after 4-6 hours of the pulse end), where the heat flux is stable at ≈0mW.

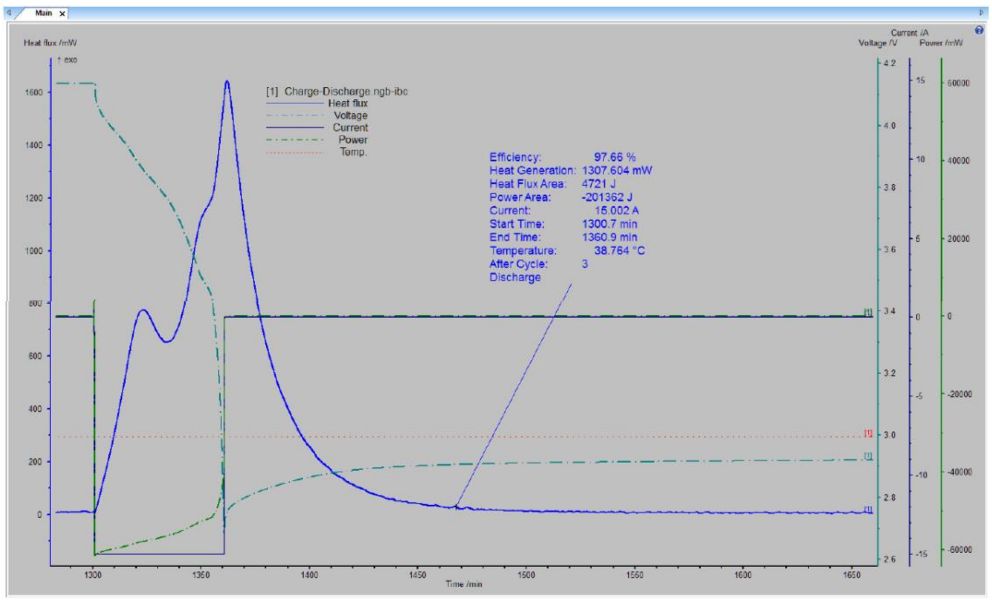

Efficiency: 97.66 %
Heat Generation: 1307.604 mW
Heat Flux Area: 4721 J
Power Area: -201362 J
Current: 15.002 A
Start Time: 1300.7 min
End Time: 1360.9 min
Temperature: 38.764 °C
After Cycle: 3
Discharge

Fig. 17: Typical results from a discharge current pulse in a battery.

References

[1] A. Barai, W. D. Widanage, J. Marco, A. McGordon, P. Jennings,
A study of the open circuit voltage characterization technique and hysteresis assessment of lithium-ion cells,
J. Power Sources 295 (2015) 99-107

[2] A.J. Bard, L.R.Faulkner
Electrochemical Methods: Fundamentals and Applications
John Wiley & Sons, Inc., London (2001)

[3] C. Daniel, J. O. Besenhard
Handbook of Battery Materials, Second Edition
Wiley-VCH Verlag GmbH & Co. KGaA (2008)

[4] S.C. Chen, C.C. Wan, Y.Y. Wang,
Thermal analysis of lithium-ion batteries,
J. Power Resources 140 (2005) 111-124

[5] R. Zhao, S. Zhang, J. Liu, J. Gu
A review of thermal performance improving methods of lithium ion battery: Electrode modification and thermal management system
J. Power Resources 299 (2015) 557-577

[6] V. Pop, H.J. Bergveld, D. Danilov, P.P.L. Regtien, P.H.L. Notten
Battery Management Systems: Accurate State-of-Charge indication for battery-powered applications
Philips Research Book Series Vol. 9, Springer

[7] A.K. Padhi, K.S. Nanjundaswamy, J.B. Goodenough
$LiFePo_4$: A novel cathode material for rechargeable batteries
Electrochem. Society Meeting Abstracts, 96-1 (1996) pp. 73

[8] Y. Nishi, Sony Corporation
Lithium ion secondary batteries; past 10 years and the future
J. Power Resources 100 (2001) 101-106

[9] T.M. Bandhauer, S. Garimella, T.F. Fuller
A critical review of thermal issues in lithium-ion batteries
J.Electrochem. Soc., 158 (2011) R1-R25

[10] K. Chen, G. Unsworth, X. Li
Measurements of heat generation in prismatic Li-ion batteries
J. Power Resources 261 (2014) 28-37

[11] D. Bernardi, E. Pawlikowski, and J. Newman
A general energy balance for battery systems
J. Electrochem. Soc., 132 (1985) 5

[12] L. Rao, J. Newmann
Heat generation rate and general energy balance for insertion battery systems
J. Electrochem. Soc., 144 (1997) 2697

[13] M. Doyle, T.F. Fuller and J. Newman
Modeling of galvanostatic charge and discharge of the Lithium/Polymer/Insertion cell
J. Electrochem. Soc., 140 (1993) 1526-1533

[14] T.F. Fuller, M. Doyle, and J. Newman
Simulation and Optimization of the Dual Lithium Ion Insertion Cell
J. Electrochem. Soc., 141 (1994) 1-10

[15] J.M. Sherfrey, A. Brenner
Electrochemical Calorimetry
J. Power Resources, 105 (1958) 665

[16] D. Bernardi, E. Pawlikowski, J. Newmann
A General Energy Balance for Battery Systems
J. Electrochem. Soc., 132 (1985) 5-12

[17] A. Greco, X. Jiang
A Coupled Thermal and Electrochemical Study of Lithium-ion Battery Cooled by Paraffin/Porous-Graphite-Matrix Composite
J. Power Sources 315 (2016) 127-139.

[18] University of Cambridge, Teaching and Learning Packages (ONLINE)
http://www.doitpoms.ac.uk/tlplib/batteries/figures/Lithiumion.png

[19] C.M. Doyle
Design and Simulation of Lithium Rechargeable Batteries
University of California, Ph.D. Thesis (1995)

[20] W.B. Gu, C.Y. Wang
Thermal-Electrochemical Modeling of Battery Systems
J. Electrochem. Soc., 147 (2000) 2910-2922

[21] G.H. Kim, K. Smith, K.J. Lee, S. Santhanagopalan, A. Pesaran
Multi-Domain Modeling of Lithium-Ion Batteries Encompassing Multi-Physics in Varied Length Scales
J. Electrochem. Soc., 158 (2011) A955-A969

COMSOL Multiphysics Documentation
Introduction to Comsol Multiphysics (5.1)
Comsol User's Guide (version 4.3)
Comsol Reference Guide (version 4.3a)
Essentials of Postprocessing and Visualization in Comsol Multiphysics

NETZSCH IBC 284 Documentation
Flyer Isothermal Battery Calorimetry IBC 284: Method, Instrumentation and Application
User's manual: Operating Instructions