



Developing Battery Systems with Simulink and Simscape

MathWorks & CES

CES is the exclusive partner of MathWorks in the Middle East

Table of Contents

Battery System Development Workflow	2
Battery Pack Design	3
Battery Thermal Management System	4
Battery Management System Algorithms	8
Desktop Simulation	12
Real-Time Simulation of Battery Systems	15
Hardware Implementation	17

Battery System Development Workflow

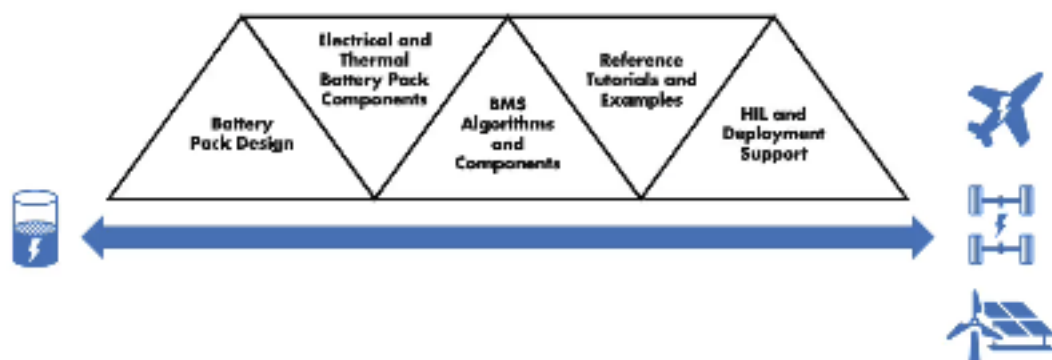
Electrification is driving the use of batteries for a range of applications, including electric vehicles (e.g., cars, buses), ships, electric aircraft, grid-tied energy storage systems, and photovoltaic systems. These applications have different requirements for battery system design in terms of cell selection, power/energy density, volume, weight, and lifetime.

Simulation of the battery system design before testing provides insight into the dynamic behaviour of the battery pack. It also lets you:

- Explore and compare software algorithms.
- Expand operational test cases.
- Shorten the technology development cycle from battery cell to battery system.

The workflow for battery system development begins with building the battery cell. Five major tasks build a bridge from battery cell design to a battery system. Those steps include:

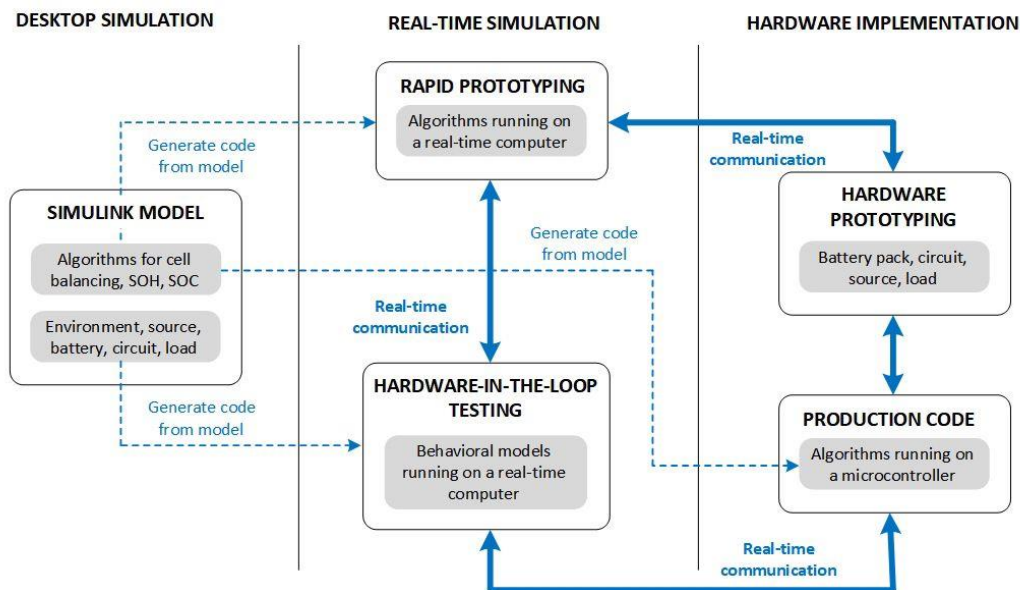
- Battery pack design
- Electrical and thermal battery pack component design
- Battery management system (BMS) algorithm development
- Integrating components to run desktop simulations
- Hardware-in-the-loop (HIL) testing and deployment



Building a bridge between battery cell and battery system.

Using Simulink® and Simscape™, the battery system development workflow begins with integration of the system components so that you can perform desktop simulation to validate the component designs and algorithms (see [Desktop Simulation](#)). The next step is real-time simulation of models using rapid prototyping and hardware-in-the-loop

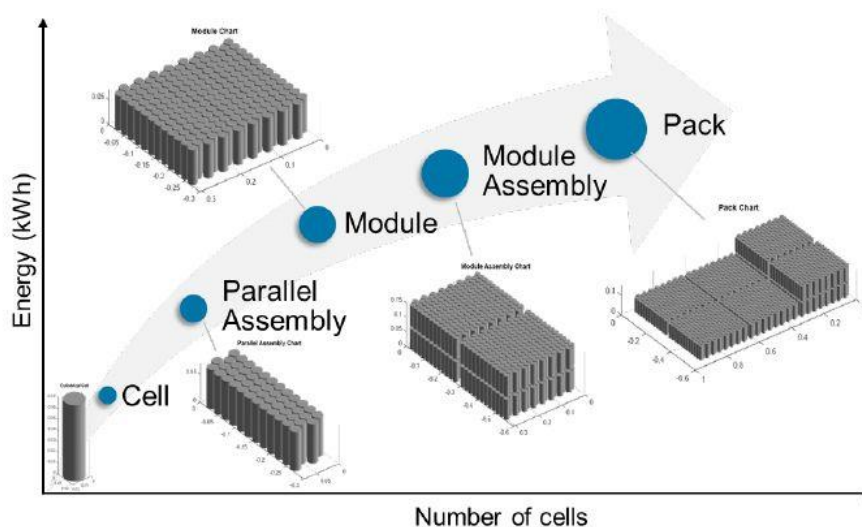
testing (see [Real-Time Simulation of Battery Systems](#)). The final stages of development involve hardware implementation, deployment, and testing (see [Hardware Implementation](#)).



Battery system development workflow with Simulink and Simscape.

Battery Pack Design

Using the [Simscape Battery™](#) application programming interface (API) in MATLAB®, you can [design a battery pack](#). Foundational elements of the design include cell design, parallel assembly, module, module assembly, and battery pack design.

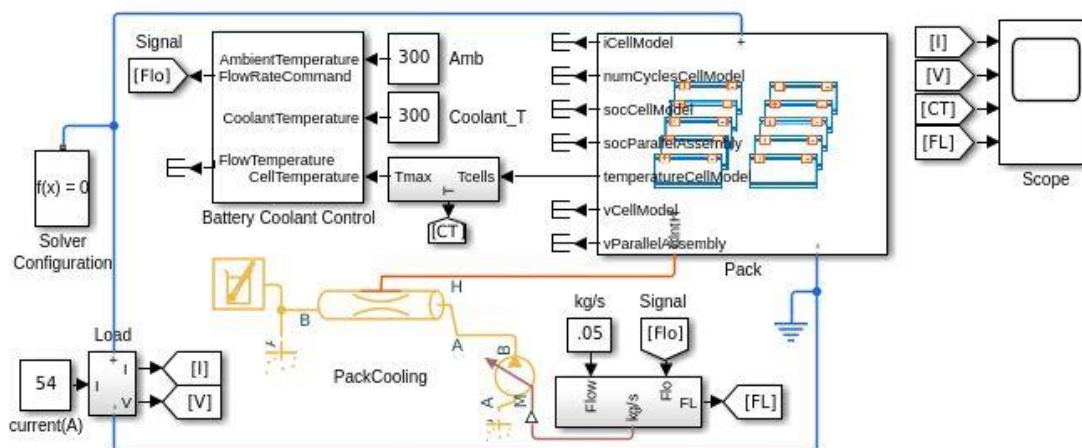


Battery pack design from cell to pack.
Using Simscape Battery, you can:

- Model electrothermal behaviour and include charge dynamics, aging, thermal, and heat transfer effects in battery cell models.
- Parameterize cells based on manufacturer data sheets.
- Build and visualize battery models with different geometries and topologies, from cell to module and from module to pack.
- Model cooling plates with customizable fluid paths and thermal connections to the battery pack.
- Explore cell-to-cell temperature variation and measure cooling efficiency.
- Generate a custom Simulink library model for your battery pack design.
- Set a suitable model resolution to strike a balance between model fidelity and simulation speed.

Battery Thermal Management System

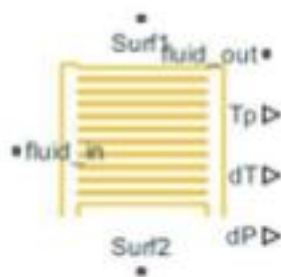
Engineers can use MATLAB and Simulink to design a battery thermal management system to regulate battery pack temperature within specifications and ensure it delivers optimal performance for a variety of operating conditions.



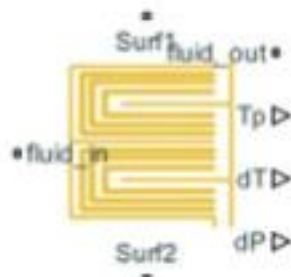
Thermal analysis comparison of a new and aged lithium-ion battery using Simscape Battery.

▼ Capture Battery Thermal Behavior

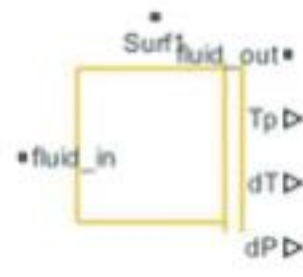
Using [Simscape](#) and [Simscape Battery](#), you can create models starting at the battery cell level and then add ambient temperature effects, thermal interface materials, and [cooling plate connections](#) to create a more representative model. Heat transfer can be considered from cell-to-cell, cell-to-plate, and cell-to-environment perspectives by defining the thermal paths to the ambient, the coolant, and the location of the cooling plate. Simscape Battery provides prebuilt cooling plate blocks that support different flow configurations, including [parallel channels](#), [U-shaped rectangular channels](#), and [edge cooling](#).



Parallel channels block in Simscape Battery.

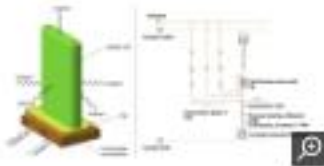


U-shaped channels block in Simscape Battery.

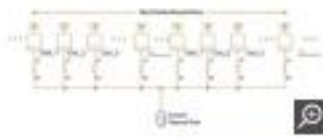


Edge Cooling block in Simscape Battery.

A pack-level thermal model can be built by [assembling cells into modules with thermal effects](#) and arranging modules inside a pack. You can perform [thermal performance analysis](#) on battery packs with different levels of aging to meet warranty criteria at end-of-life (EOL) time.



Detailed 1D thermal modeling of a single battery cell with Simscape using the [Thermal Elements library](#).



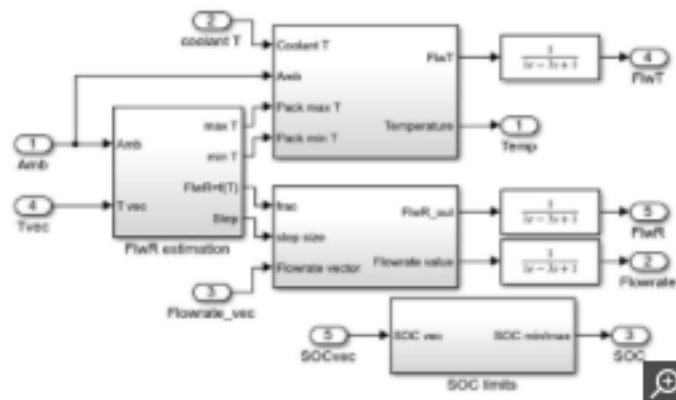
Defining the [coolant thermal path](#) for the battery module with Simscape Battery.



Connecting a [cooling plate](#) to a battery module and parallel assembly.

- Design Controls for Battery Thermal Management

Simulink makes it easy to design closed-loop controls that combine feedforward and PID techniques for circulation system controls—such as feed stream (valve) control, mass flow (pump) control, and heat exchange path selection control. With Simscape Battery, you can use prebuilt blocks, such as [Battery Coolant Control](#) and [Battery Heater Control](#), to build battery thermal management control algorithms. With [Stateflow](#)[®], you can also design supervisory control logic for switching between different [operating modes](#)—such as heating versus cooling—based on the environmental temperature and the battery temperature.

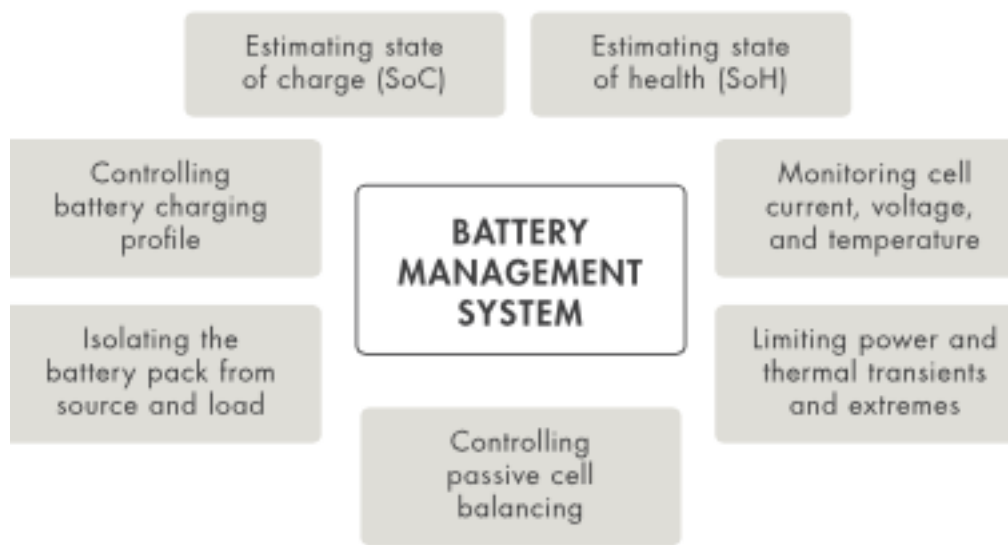


A Simulink model of a coolant control system that calculates the flow rate based on temperatures among the battery cells as well as the ambient temperature.

Battery Management System Algorithms

A well-designed battery management system (BMS) ensures maximum performance, safe operation, and optimal lifespan under diverse charge-discharge and environmental conditions. Simulink and Simscape enable you to gain insight into the dynamic behaviour of the battery pack, explore software architectures, test operational cases, and begin hardware testing early, reducing design errors. Engineers can use built-in BMS control blocks in Simscape Battery to evaluate the designed battery pack performance, develop a thermal management system, and run system-level simulations.

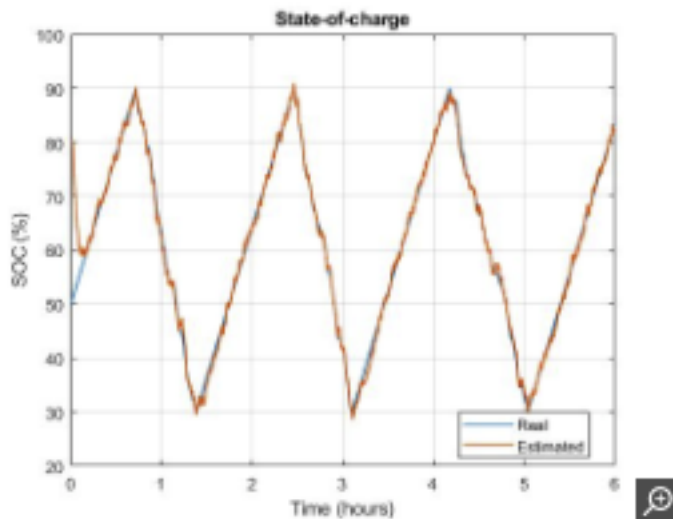
To meet these objectives, the BMS consists of algorithms that control the behaviour and performance of the battery pack.



Estimating State of Charge

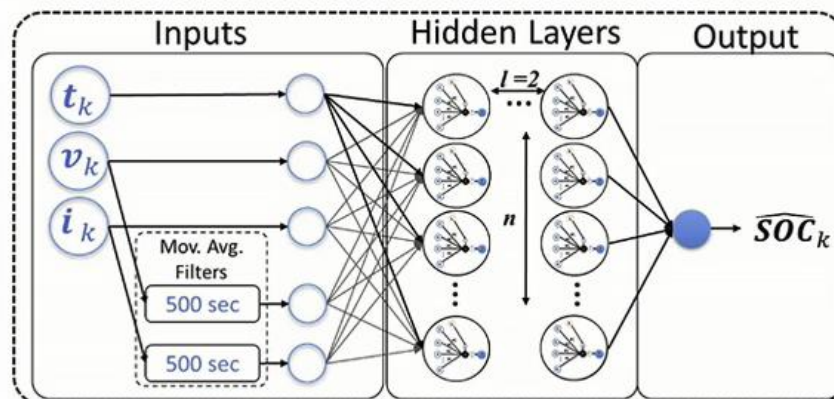
Accurate battery models are vital in the development of algorithms for state-of-charge (SOC) estimation. Traditional approaches to SOC estimation, such as open-circuit voltage (OCV) measurement and current integration (Coulomb counting), are reasonably accurate in some cases. However, estimating the SOC for modern battery chemistries that have flat OCV-SOC discharge signatures requires a different approach. The extended Kalman filter (EKF) and unscented Kalman filter (UKF) are approaches that have been shown to provide accurate results for a reasonable computational effort.

Simscape Battery contains three SOC Estimators: [Coulomb counting](#), [adaptive Kalman filter](#), and [Kalman filter](#). Compared to the Kalman filter SOC estimator, the adaptive Kalman filter SOC estimator has terminal resistance as an additional state. Both the adaptive Kalman filter SOC estimator and the Kalman filter SOC estimator have the options to select EKF or UKF to develop an observer for estimating SOC. Such observers typically include a model of the nonlinear system of interest (the battery), which uses the current and voltage measured from the cell as inputs, as well as a recursive algorithm that calculates the internal states of the system (SOC among them) based on a two-step prediction/update process.



Real SOC and estimated SOC using EKF from Simscape Battery.

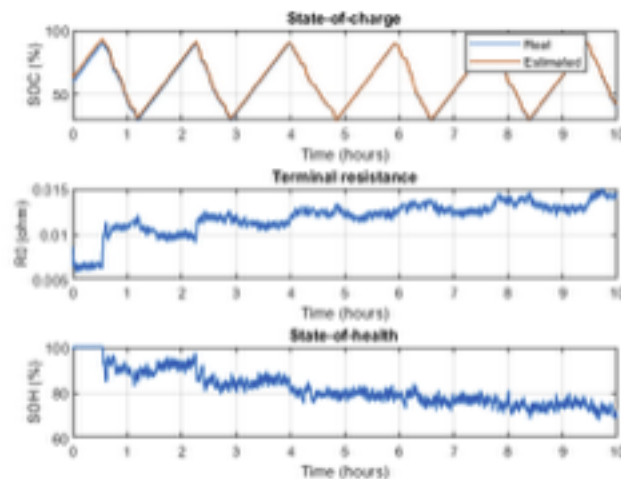
In contrast to a Kalman filter, using a neural network to develop an SOC estimator does not require extensive information about the battery or its nonlinear behavior. Instead, the network is trained with current, voltage, and temperature data and SOC as a response.



Feedforward neural network for calculating SOC using temperature, voltage, and current.

▼ Estimating State of Health

All batteries, including those that meet performance specifications at time of manufacture, degrade over time due to calendar life and cycling, suffering a gradual loss in reserve capacity and an increase in internal resistance. While the latter is relatively straightforward to estimate using short time measurements, the former requires a full charge or discharge excursion for an accurate calculation, which is not always practical. This challenge has led to growing interest in state-of-health (SOH) estimation as well as the development of adaptive Kalman filter formulations augmented to include battery parameters in addition to states. An accurate estimation of the instantaneous internal resistance is very helpful for the BMS to establish power limitations.



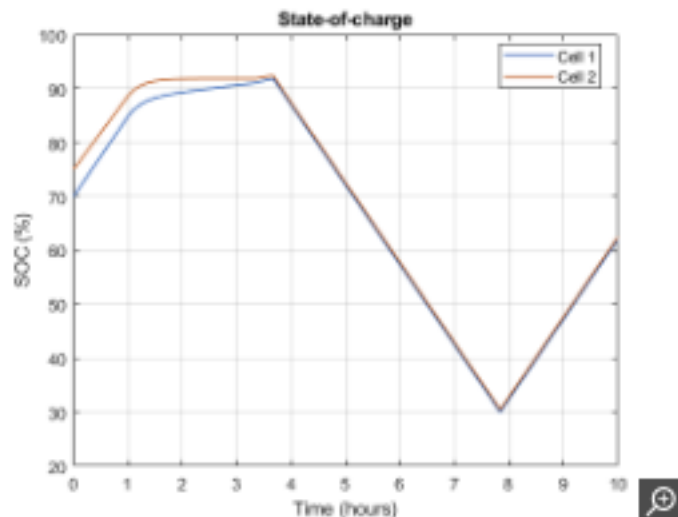
Battery SOH estimation using an adaptive Kalman filter in Simscape Battery.

SOH estimation is more subjective than SOC estimation; there is no universal agreement on how SOH is to be defined. As a result, each organization may have its own specific method for quantifying an SOH estimate, making it impossible to use general-purpose, off-the-shelf solutions. Using Simulink and Simscape, you can develop and simulate custom SOH estimation algorithms that are in line with your organization's specific interpretation of battery health.

Cell Balancing, Battery Charging, and Battery Monitoring

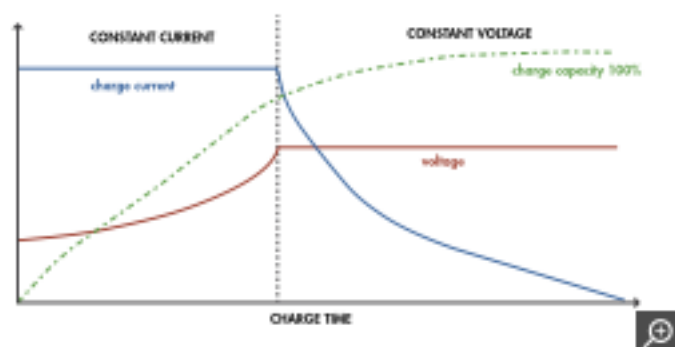
Simscape Battery provides prebuilt library blocks for cell balancing, battery charging, and battery protection.

Cell balancing capabilities keep a similar SOC in all cells by dissipating the excess charge through a bleed resistor. Balancing is necessary to prolong the life of cells that can be damaged by overcharge or discharge. Multiple cells improve capacity when cells are balanced.



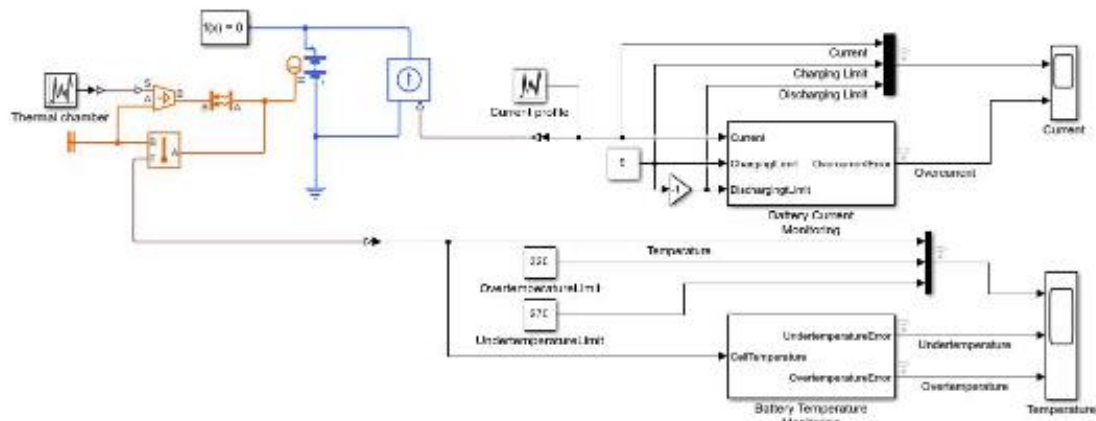
Cell balancing in Simscape Battery as a function of SOC and time.

Battery charging is controlled with a constant-current/constant-voltage (CC-CV) algorithm. This is a common approach that maintains battery life by preventing overcurrent and overvoltage when charging. Current is limited during charging until a preset voltage is reached.



Battery charge vs. time using a CC-CV algorithm in Simscape Battery.

Protecting the battery requires monitoring current, voltage, and temperature. Simscape Battery provides blocks that you can incorporate into your BMS algorithms.



Battery monitoring blocks implemented into a BMS model.

Desktop Simulation

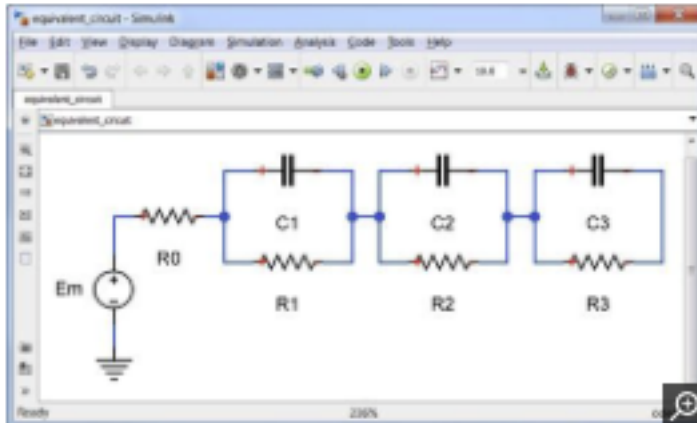
Desktop simulation in Simulink enable you to verify functional aspects of the battery system design. On the desktop, the battery system, environment, and algorithms are simulated using behavioural models. For example, you can explore active versus passive cell balancing configurations and algorithms to evaluate the suitability of each balancing approach for a given application. You can use desktop simulation to explore new design ideas and test multiple system architectures before committing to a hardware prototype. You can also perform requirements testing in desktop simulations; for example, you can verify that contactors are prevented from opening or closing when an isolation fault is detected.

Modeling BMS Software

The ability to perform the realistic simulations that are central to the development of BMS control software starts with an accurate model of the battery pack. Batteries are often designed using finite element analysis (FEA) models that account for the physical configuration of the batteries and capture their electrothermochemical properties. Although these models are excellent for designing and optimizing a battery pack's chemistry and geometry, control engineers need models that are better suited for system-level design and software development.

Modeling and Characterizing the Battery Cell

When developing BMS algorithms in Simulink, you can use equivalent circuits to simulate the thermoelectric behavior of the battery cell. The equivalent circuit typically comprises a voltage source, a series resistance, and one or more resistor-capacitor pairs in parallel. The voltage source provides the open circuit voltage while the other components model the internal resistance and time-dependent behavior of the cell. These equivalent circuit elements are, in general, temperature and state-of-charge (SOC) dependent. Because these dependencies are unique to each battery's chemistry, they need to be determined using measurements performed on battery cells of the same type as those for which the controller is being designed. You can use optimizations in Simulink and MATLAB to parameterize an equivalent circuit via [model correlation with experimental data](#) (9:54).



Equivalent circuit of a battery with three time constants, internal resistance, and open-circuit potential.

Modeling the Power Electronics and Passive Components

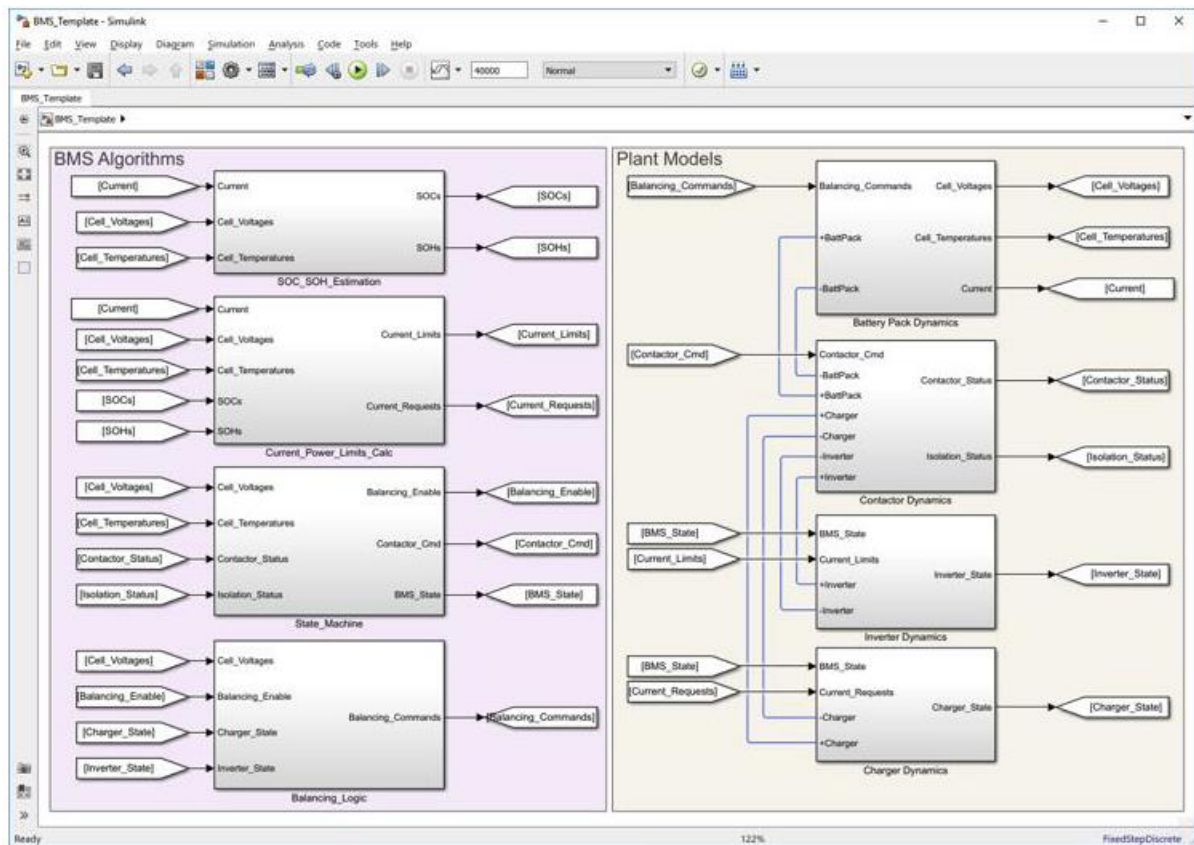
In addition to the battery pack model, realistic battery system simulations require accurate models of the circuit components connecting the battery system to the power source and load. Simscape Electrical, an add-on product for Simulink, provides complete libraries of the active and passive electrical components needed to assemble a complete battery system circuit, such as the analog front end for cell balancing. The charging source can consist of a DC supply, such as a photovoltaic (PV) system, or an AC source, for which the current is rectified.

System-level simulation with Simulink and Simscape lets you construct a sophisticated charging source around the battery and validate the battery system under various operating ranges and fault conditions. The battery pack load can be similarly modeled and simulated. For example, the battery pack may be connected through an inverter to a permanent magnet synchronous motor (PMSM) in an electric vehicle (EV). With simulation, you can vary the operation of the EV through drive cycles and evaluate the effectiveness of the BMS in changing operating conditions.

Testing with Desktop Simulation

When you incorporate desktop simulation into your testing procedures, you can author and execute test cases to exercise the battery system along all possible branches of logic and closed-loop control—a level of coverage rarely available when testing with hardware. With this approach, the simulation model serves as an executable specification driving the design and testing of the battery system. Simulink supports testing via desktop simulation with a range of features that enable you to:

- Incorporate requirements, such as limits, tolerances, logical checks, and temporal conditions, into the model with traceability to the original specifications
- Construct complex sequences of simulation-based tests to perform functional, baseline, equivalence, and back-to-back testing
- Track industry-standard metrics such as decision, condition, and modified condition/decision coverage (MC/DC), as well as relational boundary coverage
- Generate test inputs to achieve complete model coverage and custom objectives
- Use formal methods to identify hidden design errors that result in integer overflow, dead logic, and division by zero



BMS algorithms and plant dynamics, including battery pack, contactor, inverter, and charger, modeled in Simulink.

Real-Time Simulation of Battery Systems

Once validated via desktop simulation, Simulink models can be used to generate C and HDL code for rapid prototyping (RP) or hardware-in-the-

loop (HIL) testing to further validate the BMS algorithms in real time. With RP, instead of handwriting control software code for real-time testing, you generate code from your controller model and deploy it to a real-time computer that performs the functions of the production microcontroller. With automatic code generation, algorithm changes made in the model can be tested on real-time hardware in hours rather than days. Further, you can interact with real-time control hardware from within Simulink to change algorithm parameters and log test data.

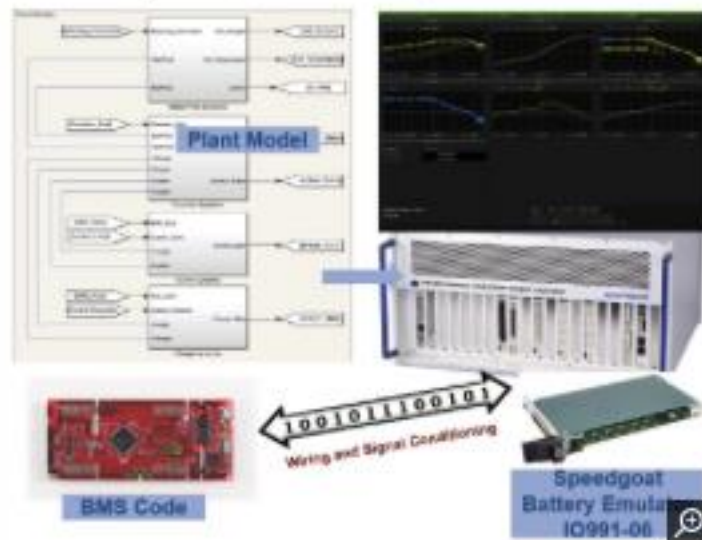
As with rapid prototyping, HIL testing involves generating code from a Simulink model and deploying it to a real-time computer. In the case of HIL testing, code is generated from the battery system models rather than the control algorithm models, providing a virtual real-time environment that represents battery pack, active and passive circuit elements, loads, charger, and other system components. This virtual environment lets you validate the functionality of the BMS controller in real time before developing a hardware prototype and in an environment where hardware will not be damaged.

Tests developed during desktop simulation can be carried over to HIL testing to ensure that requirements are met as the BMS design progresses. Though HIL testing is employed primarily to test code running on a microcontroller or FPGA, you can instead use a rapid prototyping system, such as Simulink Real-Time™ and Speedgoat® target hardware, and connect it to the HIL setup before production controller hardware is selected.

✓ Validating BMS Software

Real-time simulation—encompassing both rapid prototyping and HIL testing—provides power electronics control engineers with additional insights into how a BMS design will perform on hardware. In both RP and HIL, the objective is to emulate in hardware one aspect of the overall design: the BMS controller in RP and the balance of the battery system in HIL. Real-time simulation offers several significant advantages in BMS design, letting you:

- Conduct RP to start validating algorithms before the final controller hardware is selected.
- Exploit the flexibility of a real-time test system for rapid design iteration and testing.
- Conduct HIL testing before the battery system prototype hardware is available.
- Use a combination of RP and HIL testing to exercise BMS algorithms for test cases that may be difficult, expensive, or destructive if you were to use the actual hardware.



Hardware-in-the-loop (HIL) testing of battery management system software. The BMS code is generated from BMS algorithms modeled in Simulink and deployed to a Texas Instruments® C2000 microcontroller. The plant model (battery pack, contactor, inverter, charger) is modeled in Simulink. Code is generated and deployed to run on Speedgoat real-time machine with battery emulator.

By reusing desktop simulation models in Simulink to generate code for real-time simulation, you can shorten overall development time. You can generate C/C++ and HDL code that executes on computers optimized for real-time performance. Code generated from Simulink models for real-time simulations includes interfaces that enable you to adjust control parameters while the real-time simulation is running.

▼ Performing Rapid Prototyping

During hardware testing, making changes to controller code can cause delays and additional risks. Modifying the code by hand, recompiling it, and deploying it to the microcontroller or FPGA takes time—potentially a long time if you are a control algorithm developer who relies on a software or hardware engineer to make the changes. Depending on the extent of the changes required, you also risk introducing new problems into the implemented code.

Instead of handwriting code updates to controller software, you can use Simulink to generate code that executes in real time on a dedicated computer and uses high-speed I/O to communicate with test hardware. In addition to eliminating manual coding and its associated delays, another advantage of this RP approach is that you can validate changes to the BMS software by running the simulation model on the desktop first to verify that no other problems were introduced.

▼ Testing with Hardware-in-the-Loop

Because hardware prototypes for a battery system can take considerable effort to build and modify, and because they are often costly to repair, it is not always feasible to test such prototypes against the electrical system in which the battery pack will operate. Given these limitations, even small design changes can threaten development schedules, and BMS designs tend to evolve slowly because teams consider radical departures from the previous design as too risky.

With Simulink, you can generate C/C++ and HDL code from the model of the hardware in your battery system and the greater system of which it is part, including the supply and the load. Once you deploy this code to a real-time computer, you are able to run real-time simulations of the hardware against your controller code before testing the controller in a battery system prototype. As a result, you can find and correct control design errors before they potentially damage expensive and difficult-to-replace prototype hardware. You can also uncover hardware design errors, such as incorrect component sizing.

Many HIL real-time systems, including Speedgoat target hardware, incorporate battery emulators, letting you emulate portable battery power supplies, emulate battery stacks for electric vehicles, or sink current to simulate batteries under charge.

Hardware Implementation

In the hardware implementation stage, the Simulink control models that have been verified via desktop simulation, RP, and HIL are used to generate efficient, production-ready code for the BMS. If necessary, production code generation can be incorporated into workflows compliant with formal certification standards used in the automotive, aerospace, and other regulated industries.

▼ Production-Ready Code Generation

Simulink generates readable, compact, and efficient C/C++ and HDL code from controller models that is ready for implementation on production microcontrollers, FPGAs, and ASICs. Unlike code generated for RP, code generated for production use does not include the extra interfaces needed to support real-time monitoring, parameter tuning, and data logging. Optimization settings enable you to precisely control the generated functions, files, and data to improve code efficiency and facilitate integration with legacy code, data types, and calibration parameters.

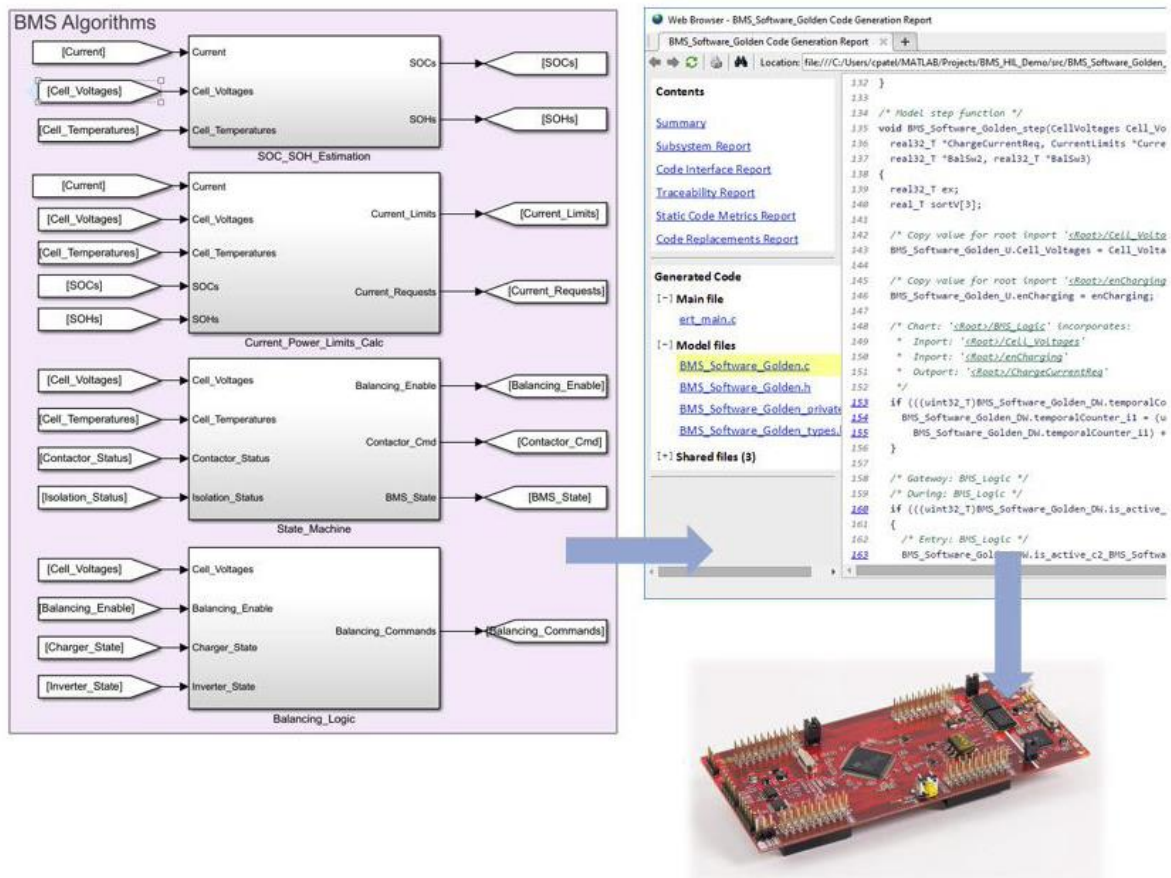
▼ Performing Processor-in-the-Loop Simulations

In processor-in-the-loop (PIL) simulation, the C/C++ or HDL code runs on the microcontroller or FPGA while the device is stepping in execution with a Simulink model of the BMS hardware, limiting the risk of damaging a hardware prototype during initial evaluations of the BMS code. Although PIL simulations are not executed in real time, they are bit-true, enabling you to verify your code under a range of conditions and build confidence that it will execute properly once deployed on the real system.

[Explore Simscape Battery](#)

Generating Production Code

Desktop simulation, RP, HIL, and PIL simulations all enable you to verify and validate the control algorithms for the BMS. With Simulink, you can use those same algorithms as the basis for generating production-ready code—either optimized and stable C/C++ code for implementation on microcontrollers or synthesizable HDL code for FPGA programming or ASIC implementation. Automatic code generation eliminates manual algorithm translation errors and produces C/C++ and HDL code with numerical equivalence to the algorithms you validated in Simulink. By simulating your control algorithms over all possible operating and fault conditions, you increase confidence that the generated code will handle those same conditions in the real system, even if you are unable to test for all of them. If hardware tests later indicate that algorithm changes are needed, you can simply modify the algorithms in your model, rerun simulation test cases to verify the correctness of the changes, and generate new, updated code. All generated C/C++ and HDL code is fully portable, optimizable with a range of options, and bidirectionally traceable to the Simulink model.



Automatically generated BMS production code from BMS algorithms modeled in Simulink. The code is deployed to a Texas Instruments C2000 microcontroller.

For more information



sales@cesimulations.com

+971 (4) 427 3663