MATLAB EXPO 2019

Developing Battery Management System using Simulink

Romain LACHAUX
Battery: a good answer to energy storage across industries…
... with some risks to keep under control
CHALLENGE: Design and verify battery management functions
Monitor Cell Voltage and Temperature

Isolate battery pack from source and load

Balance Battery Cell

Estimate State-of-charge (SoC)

Current and Power Limits (Derating)

Control the charging profile

Battery Management System (BMS)
SOLUTION: Perform **system-level simulations** with **Simulink**
BMS Development Workflow with Simulink and MBD

**Desktop Simulation**

**Simulink Model**
- Controller: Algorithms for cell balancing, State-of-Charge
- Plant: Environment, source, battery, circuit, load

**Real-Time Simulation**

**Rapid Prototyping**
- Algorithms running on a real-time computer

**HIL Testing**
- Behavioral models running on a real-time computer

**Hardware Implementation**

**Hardware Prototyping**
- Battery packs, circuit, source, load

**Production Code**
- Algorithms running on an embedded microcontroller

---

MATLAB EXPO 2019
Agenda

- **BMS Model Demo**
  - Physical Modeling
  - BMS Algorithms

- **Deployment on Hardware**
  - Code Generation
  - Real-Time Testing
BMS Model Overview

- **System Model**
  - Controller: BMS Algorithms
  - Plant: Physical Modeling

- **Advantages of System-Level Simulation:**
  - Quick design iterations
  - Early results in the development workflow
  - Possible to test each part alone or together in the same model (Closed-loop testing)
Simulation Results Overview

- **Early results** during design process
  - Possible to refine or add missing requirements

- Example:
  - Temperature differences
  - Potential impact on cells ageing

  ➔ Need of a cooling system?
PLANT: Battery Physical Modeling with Simulink and Simscape
Battery Cell Modeling

▪ **Thevenin Model** (1st Order) to represent electrical behavior of battery cell

▪ Model based on Simscape Foundation Library components…

▪ ... with dependance upon SoC and temperature by **modifying source code**
Battery Cell Modeling

- What about **thermal behavior**?
  - Cell heat up under load
  - Convection heat flux between cells
  - Thermal exchange with environment

⇒ **Thermal component** from Foundation Library

---

**Multidomain**

**Simscape**

- Electrical
- Mechanical
- Magnetic
- Thermal
- Custom equations
- Hydraulic
- Thermal
- Two-Phase
- Fluid
- Gas
- Moist
- Air
Battery Pack

Possible to model **different architecture or technological choice** of a subsystem in the same model.

**Variant Model**

1 Module of 6 cells

OR

16 Module of 6 cells
Tuning a Lithium Battery Model to Match Measured Data

Cell Physical Model:

\[ E_m \]

\[ R_0 \]

\[ R_1 \]

\[ C_1 \]

Lithium Cell Characteristic Measurement:

<table>
<thead>
<tr>
<th>Temperature</th>
<th>SOC 1</th>
<th>SOC 0.9</th>
<th>SOC 0.8</th>
<th>...</th>
<th>SOC 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>5°C</td>
<td>4.20 V</td>
<td>4.12 V</td>
<td>4.05 V</td>
<td>...</td>
<td>3.09 V</td>
</tr>
<tr>
<td>20°C</td>
<td>4.18 V</td>
<td>4.09 V</td>
<td>4.01 V</td>
<td>...</td>
<td>3.05 V</td>
</tr>
<tr>
<td>40°C</td>
<td>4.15 V</td>
<td>4.02 V</td>
<td>3.97 V</td>
<td>...</td>
<td>3.01 V</td>
</tr>
</tbody>
</table>

OBJECTIVE: Match model behavior to tests measurements
1. **Import** measurement datas

2. **Identify** parameters and set range

3. **Perform estimation**
Advantages of Physical Modeling

- With this physical model, you can
  - Evaluate your architecture
  - Optimize your design
  - Refine and Validate your requirements

+ Enable **Closed-loop testing of your control algorithms** to verify and validate it
CONTROLLER: BMS Algorithms with Simulink and Stateflow
Battery Management System Functions

- Battery State
- Fault Management
- Battery isolation control
- Derating Calculations
- State-of-Charge Estimation
Fault Management

- Monitoring three physical channels:
  - Battery Pack **Current**
  - Cell **Voltage**
  - Cell **Temperature**

- Broadcasting Fault Presence to other BMS subsystems
  → Contactor Opening (SAFETY)
  → BMS_State == FAULT
State-Of-Charge Estimation

- Two methods:
  - Coulomb Counting
    ✓: Simple to implement / low computational needs
    ✗: Accuracy and robustness
  - Kalman Filtering*
    ✓: High accuracy by including a nonlinear battery model which uses current and voltage measurement
    ✗: Slightly higher computational effort

*: ready to use block available in Control System Toolbox or System Identification Toolbox
Agenda

- BMS Model Overview
  - Physical Modeling
  - BMS Algorithms

- Deployment on Hardware
  - Code Generation
  - Real-Time Testing
Generate C/C++ Code From BMS Models

- Generate target optimized C/C++ code
- Fine-tune code optimizations, package and build generated code
Real-Time Testing of Battery Management System

- Testing BMS with Battery Cells
  - Longer test cycles
  - Difficult to test fault conditions
  - Difficult to reproduce results
  - Limited test automation

→ Costs (Hardware prototype, possible failure, several people to perform tests, etc)
Testing BMS with Emulated Battery Cells
- Reduce testing time
- Test fault conditions safely
- Automate testing

Battery Emulation

Main Controller

Measurement & Diagnostics

Automatic Code Generation
BMS Development with Simulink

Reduce Design Iteration Time

Collaborate Across Domains

Gain Confidence in Design

MATLAB EXPO 2019