Leveraging MATLAB-Simulink in Building Battery State-of-Health Estimation Pipelines for Electric Vehicles

Nilesh Kulkarni, Director, Artificial Intelligence
Matthew Daigle, Principal Scientist, Artificial Intelligence

NIO USA
Artificial Intelligence Team Overview

Team

• About 25 full-time employees
• Average experience 12 years

Projects

• Autonomous Driving: Trajectory Planning, Decision Making
• Vehicle AI: Smart Controls, Diagnostics & Prognostics
In electric vehicles, understanding battery State-of-Health (SOH) is critical:

- Powertrain performance
- Range estimation
- Fleet management
- Service operations
Challenges

- In the product design phase, battery data is available only under laboratory and limited driving conditions
  - No existing fleet
  - Limited in-vehicle data collection
  - Data for only specific driving conditions
- However, to build an analytics stack focused on monitoring battery SOH and predicting battery life, we need lots of data

**Solution:** Scalable simulation-based data generation deployed in the cloud
Cloud-based Architecture

Vehicle Data

Data Store

Battery Analytics

Insights

Vehicle Data & Insights

Simulated Fleet

Visualizations, Alerts, Annotations
Outline

Vehicle Simulation
- Li-ion Cell Models
- Powertrain
- Vehicle Dynamics
- Driving Conditions

Battery Analytics
- SOH Estimation
- SOH Prediction

Cloud Deployment
- Code Generation
- Simulation as a Service
- Batch Simulations
- Analytics
- Dashboards
Vehicle Simulation Summary

- **Electrical**
  - Cell models and battery configuration
  - Cell/battery health degradation
  - Motor and inverter efficiency and losses
  - Torque/current limits
  - Regenerative braking
  - Charging
  - HVAC
  - Thermal management
- **Mechanical**
  - Drag
  - Road type
  - Brakes
  - Driver model
- **Sensors**
  - Speed, brake pressures, voltages, currents, etc.
- **Fault injection via parameter changes**
Simulation Architecture

- **Docker Container**
  - **Driver Route Following**
  - **Lat. & Long. Des. Speed**
  - **Driver Model**
    - **Driver Controls**
  - **Vehicle Model**
    - **Vehicle Outputs**

- **Route**
- **Driver Settings**
- **Environmental Conditions**

- **Scenario Specification**
- **C++**
- **Simulink → C++**
Battery Analytics
System gets input and produces output

Estimation module estimates the states and parameters, given system inputs and outputs

• Must handle sensor noise
• Must handle process noise

For some event \( E \), e.g., end-of-discharge or end-of-life, prediction module predicts \( k_E \)

• Must handle state-parameter uncertainty at time of prediction
• Must handle future process noise trajectories
• Must handle future input trajectories (battery loads)

In model-based approaches, require a dynamic model of the battery
**Discharge**
Positive electrode is cathode, negative electrode is anode
Reduction at pos. electrode:
\[ \text{Li}_{1-n}\text{CoO}_2 + n\text{Li}^+ + n\text{e}^- \rightarrow \text{LiCoO}_2 \]
Oxidation at neg. electrode:
\[ \text{Li}_n\text{C} \rightarrow n\text{Li}^+ + n\text{e}^- + \text{C} \]
Current flows + to −, electrons flow − to +, lithium ions flow − to +

**Charge**
Positive electrode is anode, negative electrode is cathode
Oxidation at pos. electrode:
\[ \text{LiCoO}_2 \rightarrow \text{Li}_{1-n}\text{CoO}_2 + n\text{Li}^+ + n\text{e}^- \]
Reduction at neg. electrode:
\[ n\text{Li}^+ + n\text{e}^- + \text{C} \rightarrow \text{Li}_n\text{C} \]
Current flows − to +, electrons flow + to −, lithium ions flow + to −
Cell Discharge Modeling

- Lumped-parameter, ordinary differential equations
- Capture voltage contributions from different sources
  - Equilibrium potential \(\rightarrow\) Nernst equation with Redlich-Kister expansion
  - Concentration overpotential \(\rightarrow\) split electrodes into surface and bulk control volumes
  - Surface overpotential \(\rightarrow\) Butler-Volmer equation applied at surface layers
  - Ohmic overpotential \(\rightarrow\) Constant lumped resistance accounting for current collector resistances, electrolyte resistance, solid-phase ohmic resistances
Battery Aging Modeling

Aging results in two major qualitative effects on dynamics:

• Loss of capacity (due to diffusion stress, irreversible parasitic side reactions)
• Increase in internal resistance (due to solid electrolyte interface layer growth)

Capture with changes in three age-related parameters:

• $q_{\text{max}}$ (max available charge)
• $R_0$ (Ohmic resistance)
• $D$ (diffusion rate parameter)

Given a discharge cycle, can estimate age-related parameters and determine how they change over time

• Assume rate of change of age parameters is of form $w|\text{i}_{\text{applied}}|
• w$ is aging rate parameter, $\text{i}_{\text{applied}}$ is applied current

Models developed and validated in MATLAB
Defining EOL

Capacity is measured in Ah for a given discharge cycle

- But, EOD is dependent on the load, so capacity measurement will be different depending on how battery is used
- Only meaningful to measure capacity w/r/t reference conditions
- For given age parameters, can use model to simulate a reference discharge and compute corresponding capacity
State of Health Estimation
Battery Analytics

Health Estimation
- For each trip, take pack voltage, current, and temperature
- Estimate battery state of charge (SOC) using unscented Kalman filter (UKF)
- Estimate current values of aging parameters over the trip
- Map aging parameters to current battery capacity/state of health (SOH)

State of Health Prediction
- Obtain SOH estimates for all previous trips
- Determine expected future battery loads
- Fit aging model (e.g., linear regression)
- Predict time/miles at which SOH will fall below 80%

Other Metrics
- Powertrain efficiency, trip energy regenerated, etc.
Cloud Deployment
Simulation Deployment

Vehicle Simulator

Driver + Vehicle Model (C++ generated with Simulink Coder) 100x real time

REST API

Simulation Request:
- Single vehicle trip simulation
- Batch simulations for vehicle lifetime with battery degradation

Telemetry

Data Store

docker
Battery Analytics Deployment

- Algorithms are implemented in Python
- For each vehicle
  - Queries data for a trip from Elasticsearch
  - Runs analytics algorithms on the trip data
  - Pushes results back to Elasticsearch
- Results include time-based analytics (e.g., state-of-charge) and trip summary metrics (e.g., SOH)
- Implemented as a batch job that is Dockerized
Conclusions

Summary

• Simulations executed on request basis or in batch mode
• Dashboards combining vehicle- and fleet-level metrics built from Elasticsearch data source
• Automated pipelines running on test vehicles
• Deployment to production in progress

Next Steps

• Scalability
• Validation with customer data
• Connection to service operations