MathWorks
AUTOMOTIVE CONFERENCE 2022
India

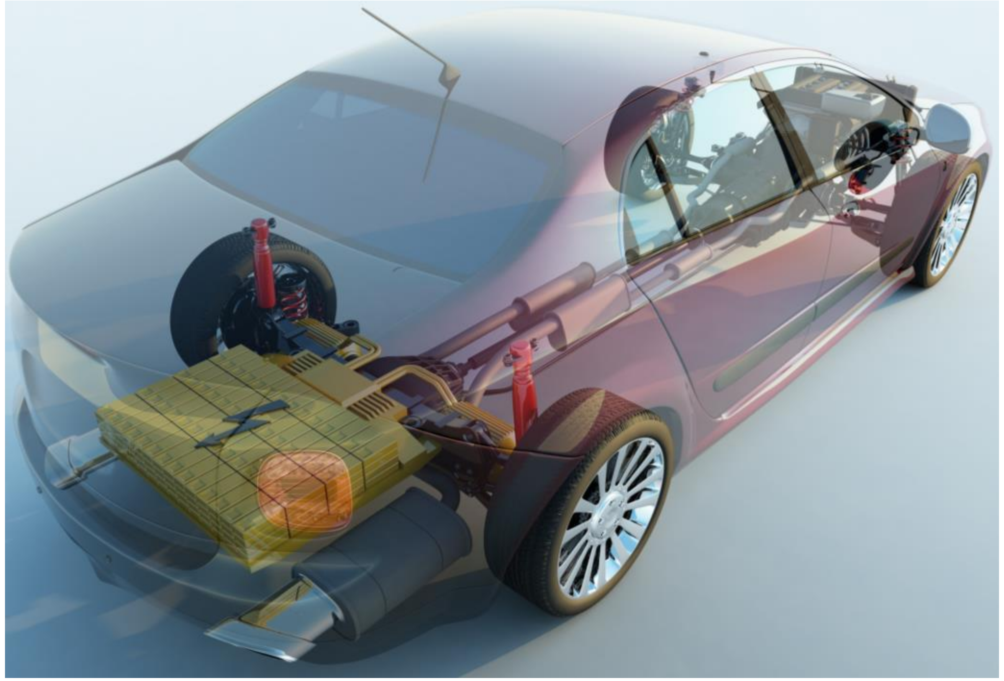
Virtual Development of Battery and BMS

Abhisek Roy, MathWorks

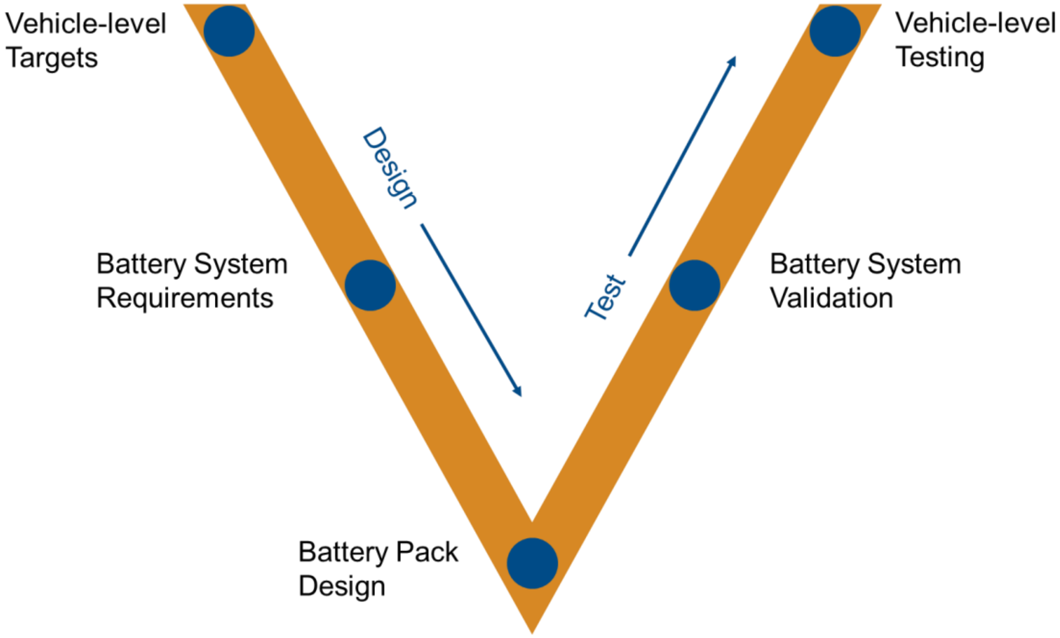
MathWorks

Context

- Common challenges for EV Battery pack design
  - How to size the battery pack?
  - How to manage thermal loads?
  - How to control the battery pack?
  - How to do predictive maintenance?



Goal for today is demonstrate how MathWorks tools support battery design and controls development throughout the V-cycle



Agenda

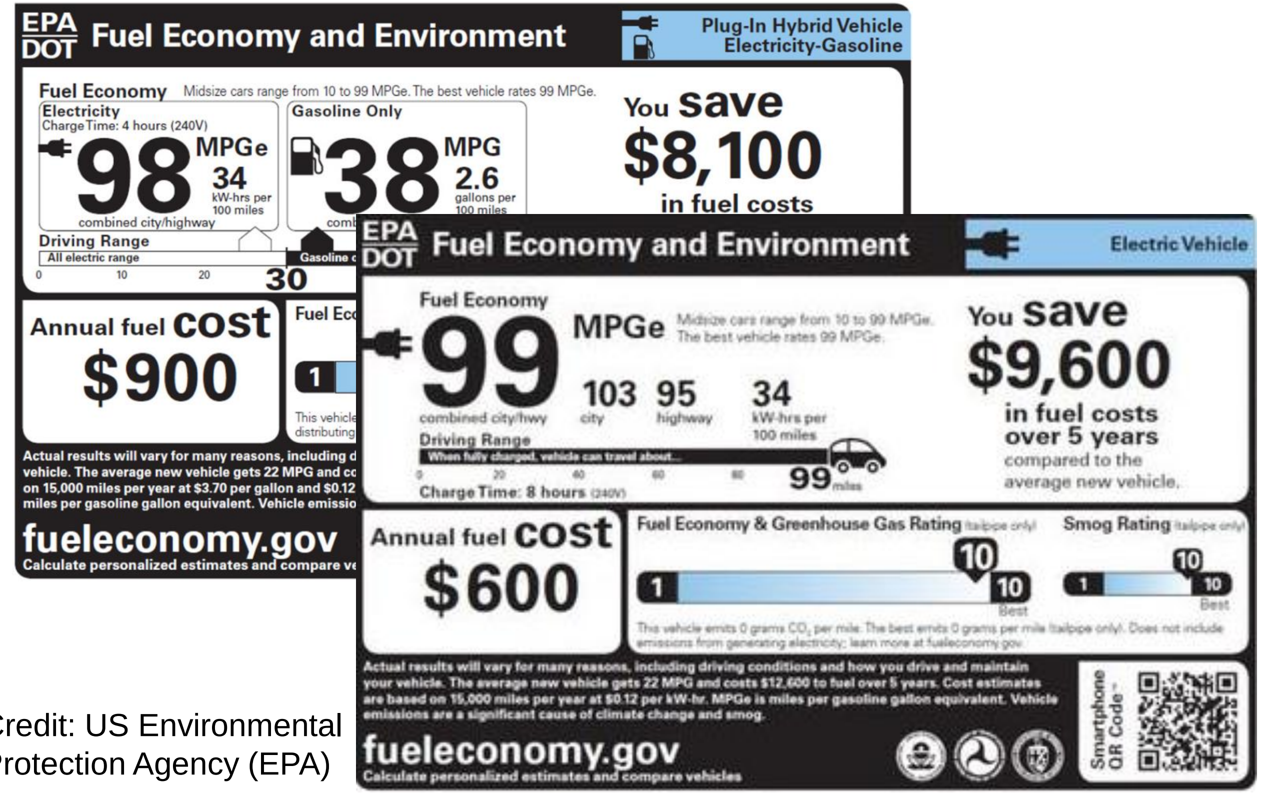
- Determine battery pack size to meet system-level targets
- Design and analyze thermal management systems
- Develop control systems
- Realize digital twin and predictive maintenance applications

Agenda: Determine battery pack size to meet system-level targets

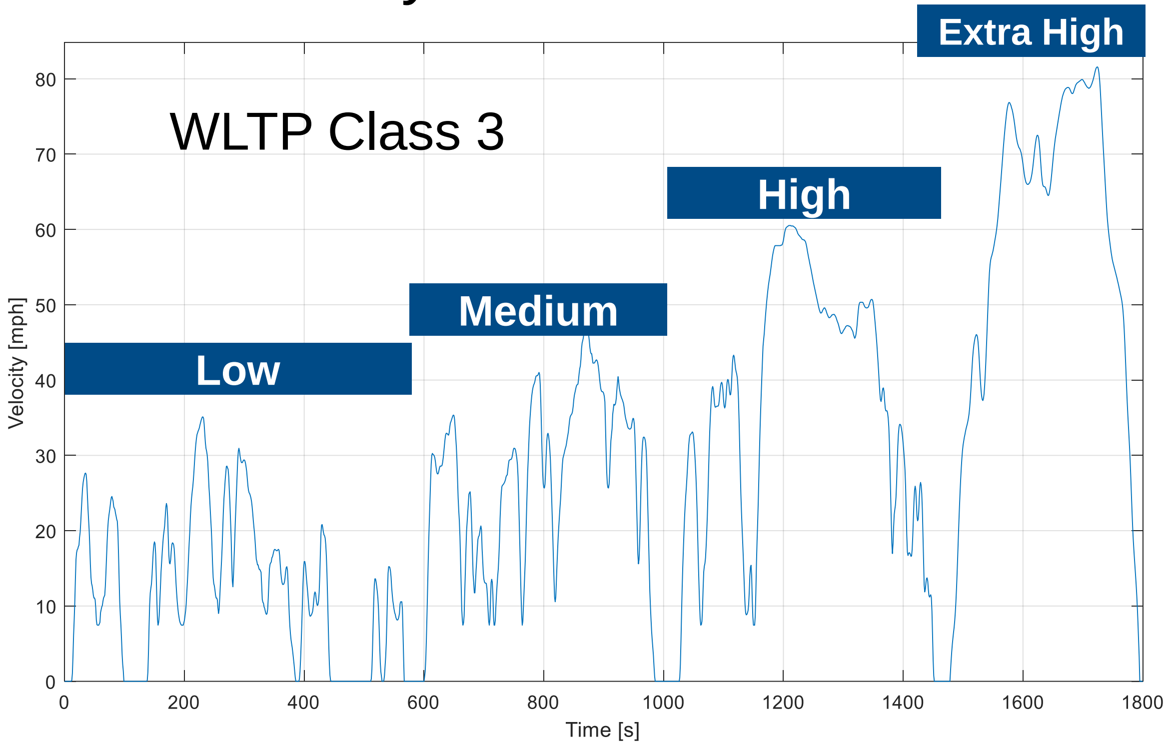
- **How to perform system level analysis?**
- How to evaluate battery efficiency and sizing?

Vehicle-Level Targets

- Government agencies rate conventional, HEV and EV’s using different standardized tests (US city / highway cycle, WLTP, etc.)
- Different metrics to define energy efficiency (MPGe, Wh/km, etc.)
- Vehicle program sets targets → requirements for subsystem teams



Credit: US Environmental Protection Agency (EPA)

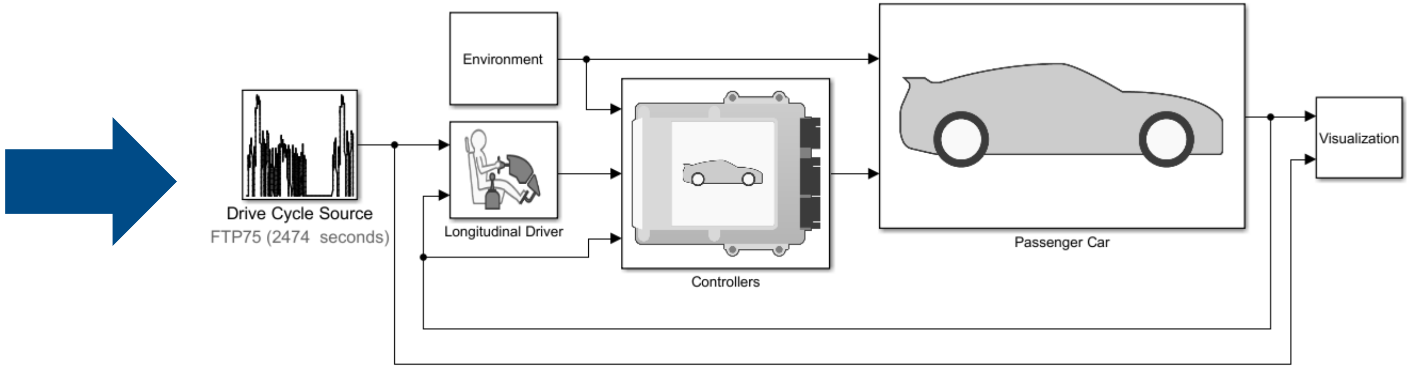


World harmonized Light-duty vehicles Test Procedure

Use System-Level Models to Evaluate System-Level Targets

Target	How to evaluate
Fuel economy	Perform drive cycle test
Range	Perform drive cycle test
Acceleration	Perform Wide Open Throttle (WOT) test
Cost	Assume $ / kWh

Credit: 4x4 Dynamometer by Adam Navrotny / CC BY-SA 3.0

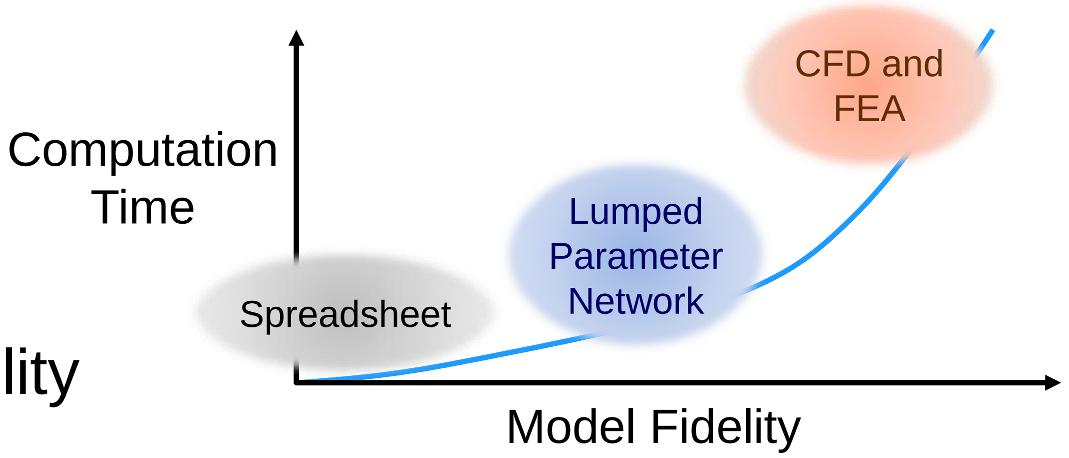


Simulations used to frontload design / save money

Right-Level Modeling

- We can answer system-level questions using system-level models, but what level of fidelity is appropriate for the task?

- Initial estimates use simplifying assumptions
  – Fast running 1D models
  – Neglect thermal / spatial effects
  – Simplified controls

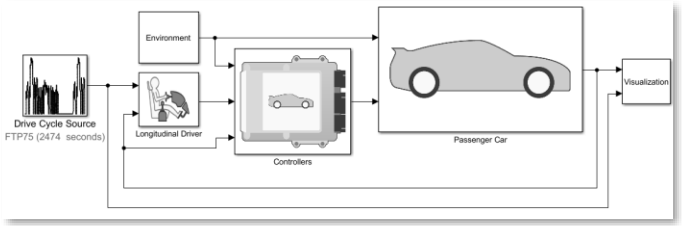


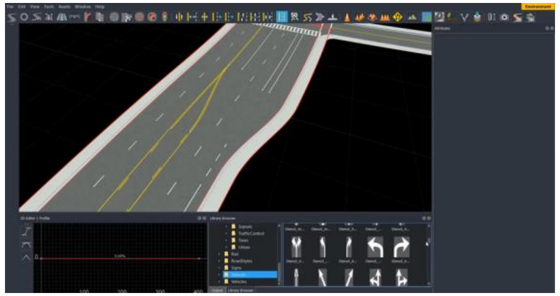
- Design-oriented tasks require higher fidelity
  – Slower running multidomain models
  – Include thermal / spatial effects
  – Production-oriented controls

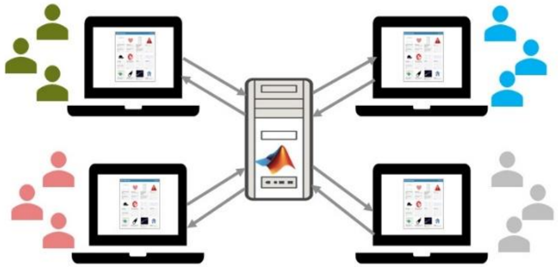
MathWorks Offering for Virtual Vehicle Simulation

Engineering Tools + Application Expertise

Create Vehicle	Integrate Software	Author Scenarios	Simulate & Analyze	Deploy Simulation
Vehicle Templates Subsystem Libraries Modeling Guidelines	C/C++ Interface Reduced Order Models FMU Integration	Scene & Scenarios Open Standards Drive Cycles	Visualization Data Analysis Report Generation	Cloud Integration Datalake Integration HIL Deployment







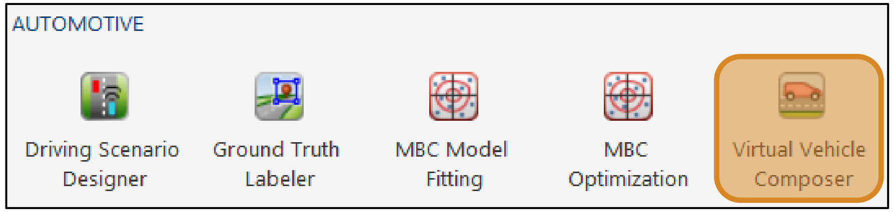
Value proposition:

- Proven tools for modeling of physics and software
- Reference applications for reduced time-to-simulation
- Common platform for model reuse
- Solutions for large-scale modeling and simulation
- Flexible platform for growth / new use cases

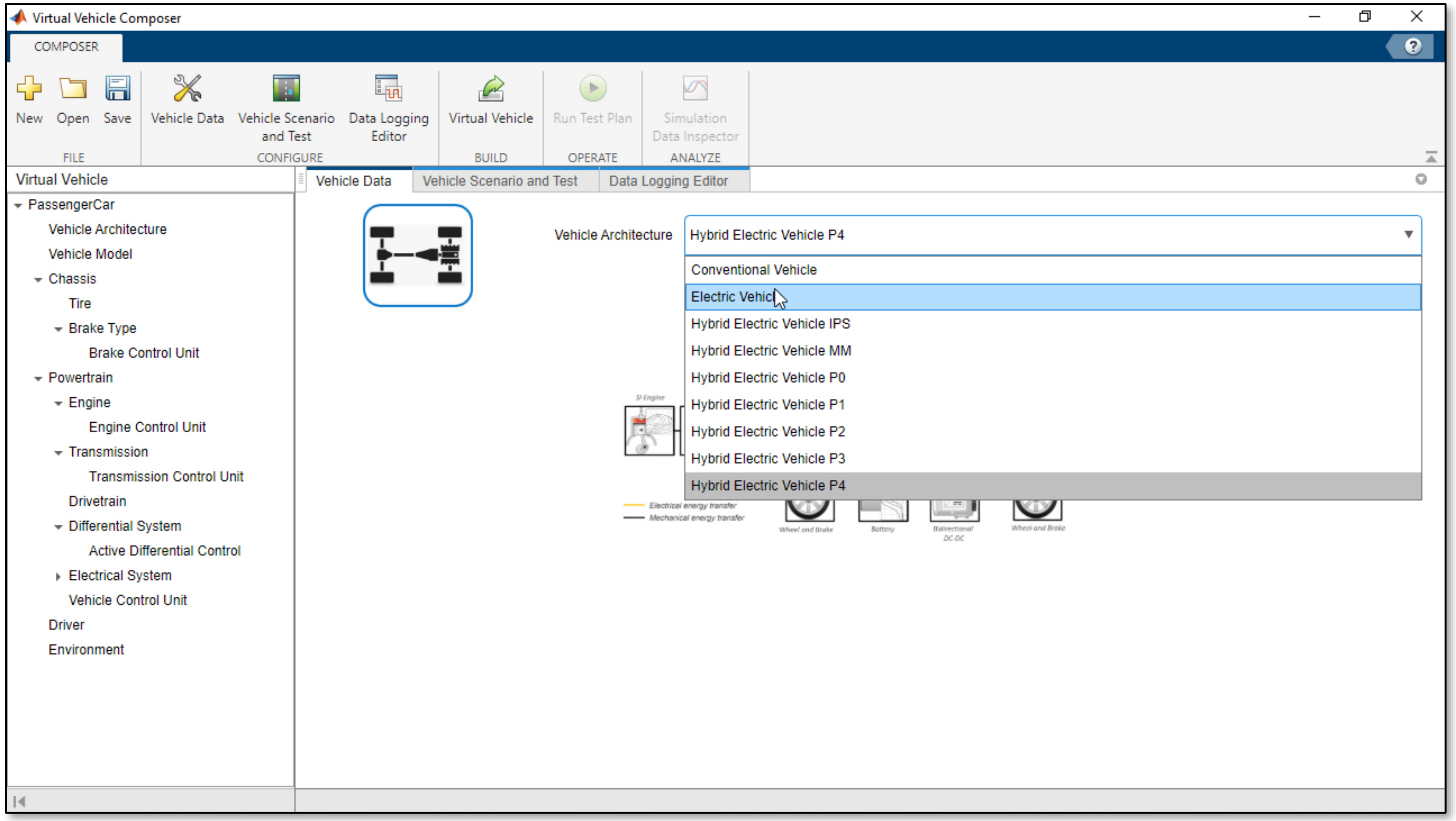
Learn more:
Virtual Vehicle

Virtual Vehicle Composer App

New in R2022a



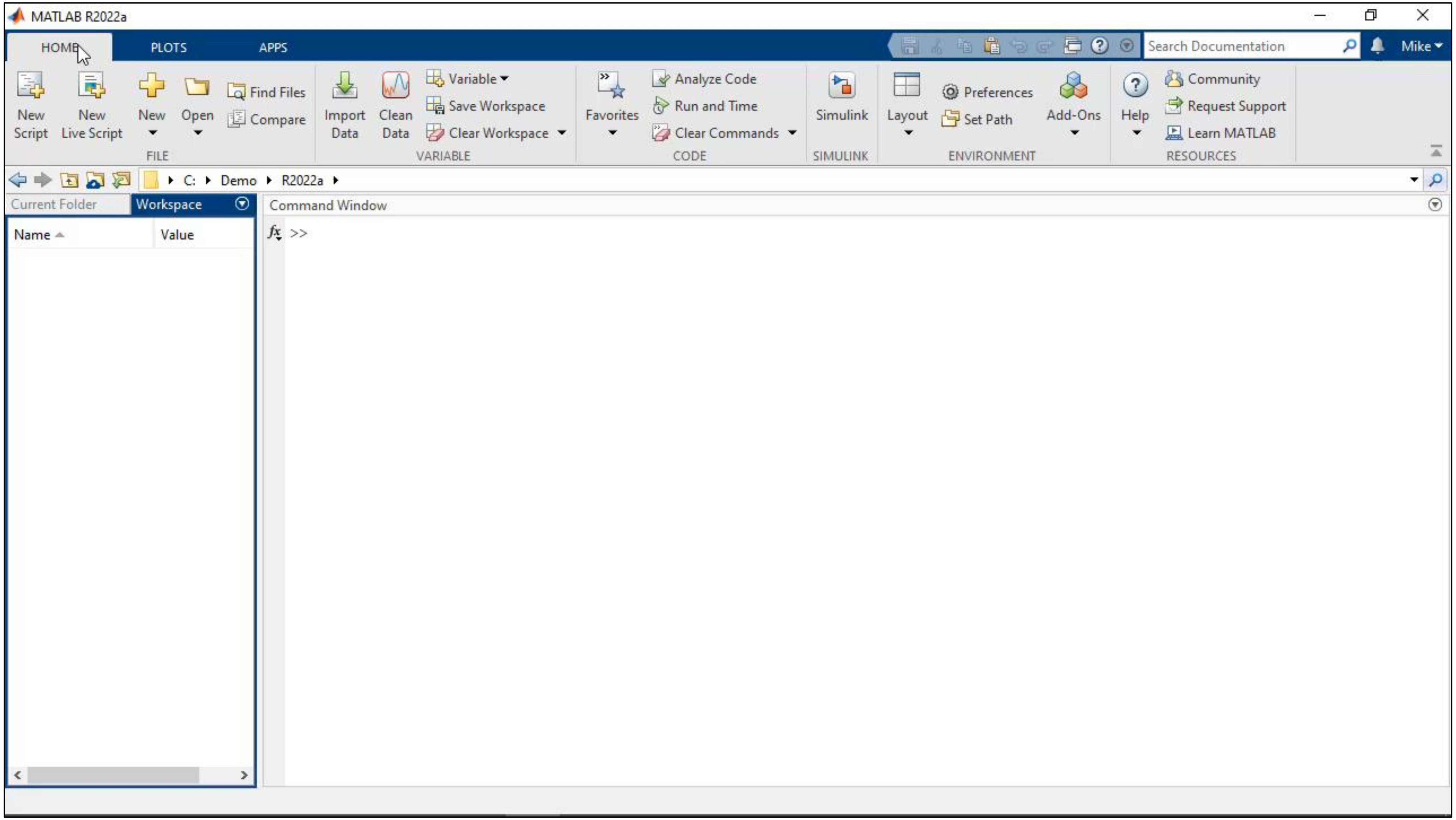
- Unified interface to quickly configure a virtual vehicle model, select test cases and review results
- Available with **Powertrain Blockset** and / or **Vehicle Dynamics Blockset**
- Includes detailed powertrain models, vehicle dynamics and closed-loop controls



Learn more:
[Virtual Vehicle Composer]

Virtual Vehicle Composer App

New in R2022a



Workflow steps:

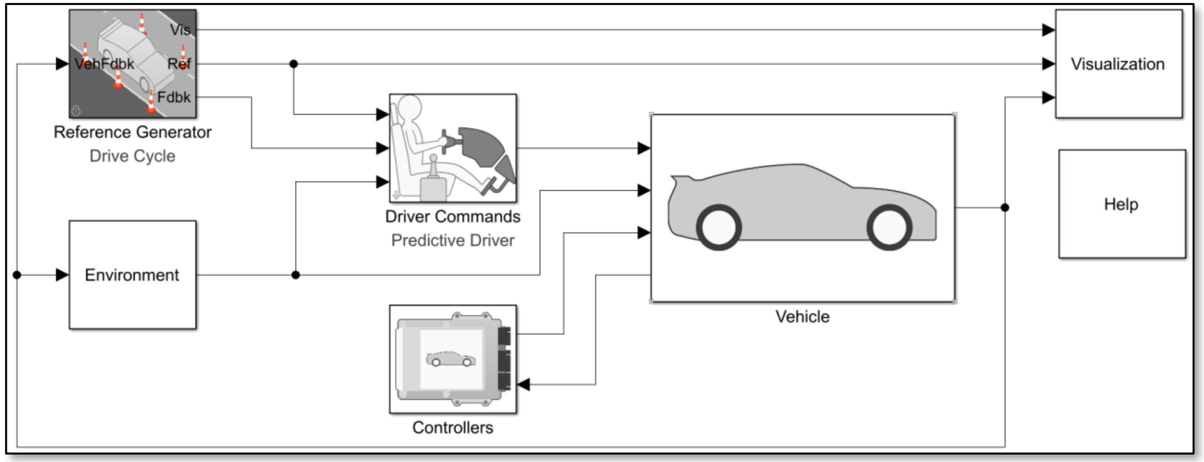
1. Start new session
2. Select powertrain
3. Select data
4. Select scenarios
5. Select signals to log
6. Generate model
7. Run test suite
8. Review results

Model Customization

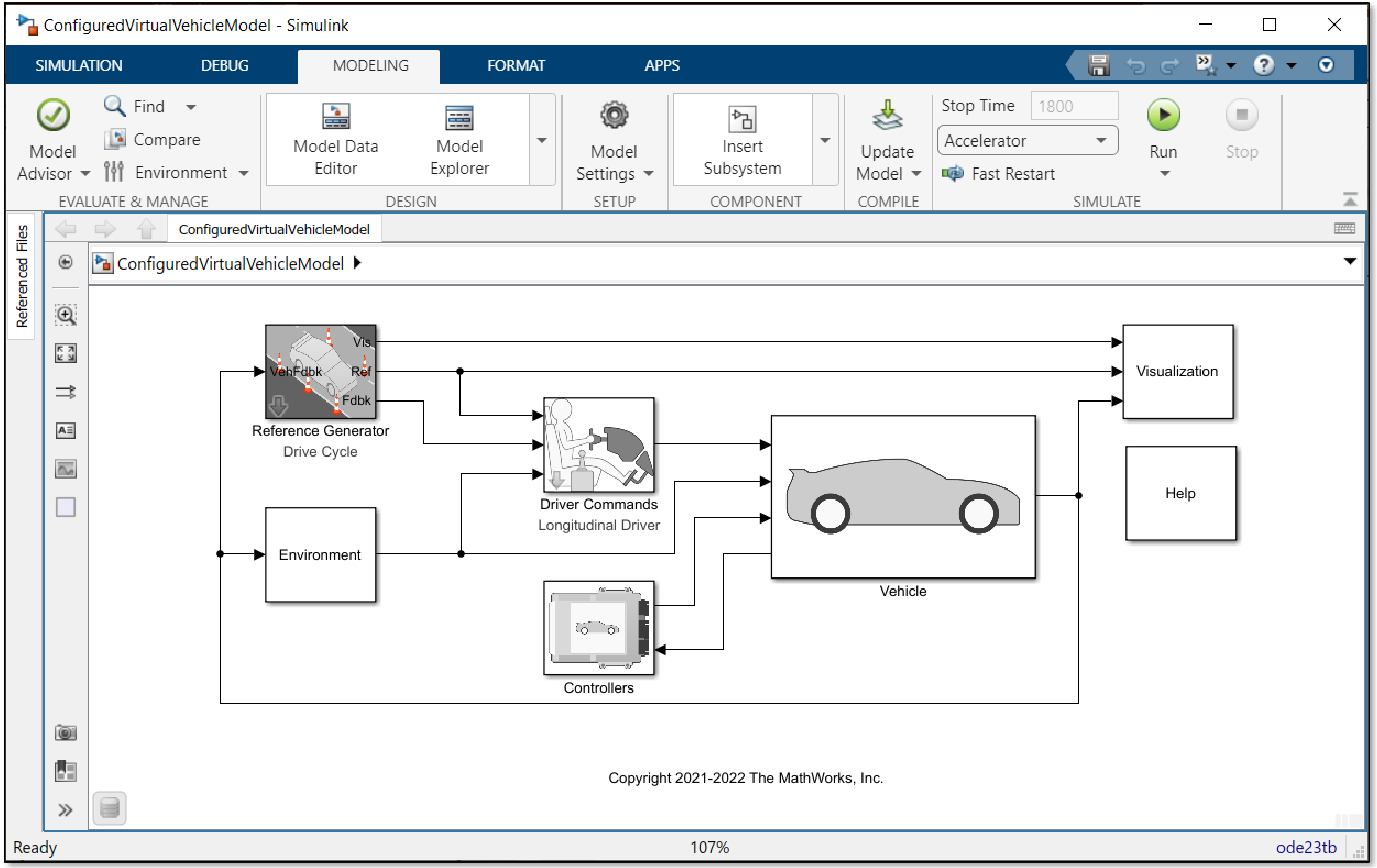
- Virtual Vehicle Composer app gets you a good starting point quickly

- Generated models are open, so you can customize it
  - Add new plant, controller or sensor model features
  - Create custom test scenarios

- Leverage Simulink platform
  - Integrate C code, S-functions, FMU, etc.
  - Perform large scale studies
  - Deploy model (HIL, cloud, etc.)



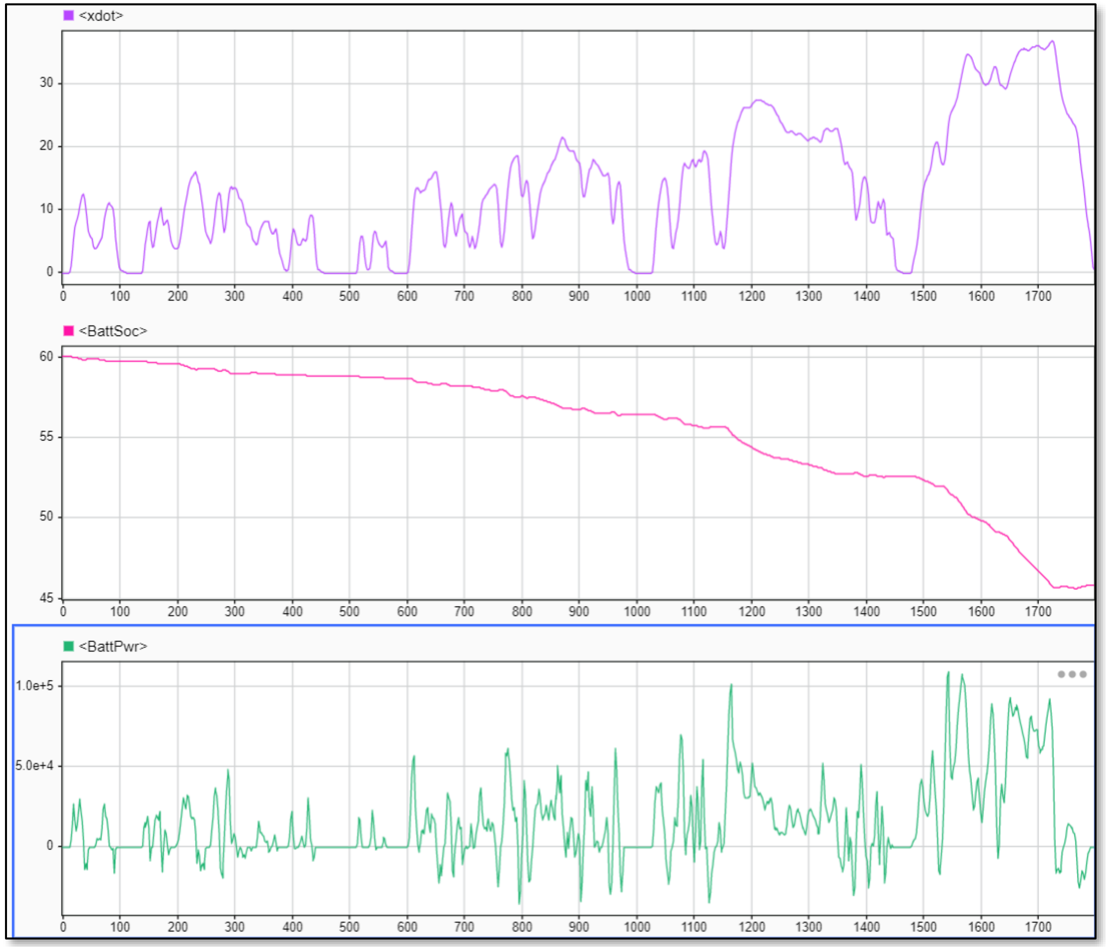
Generated Model



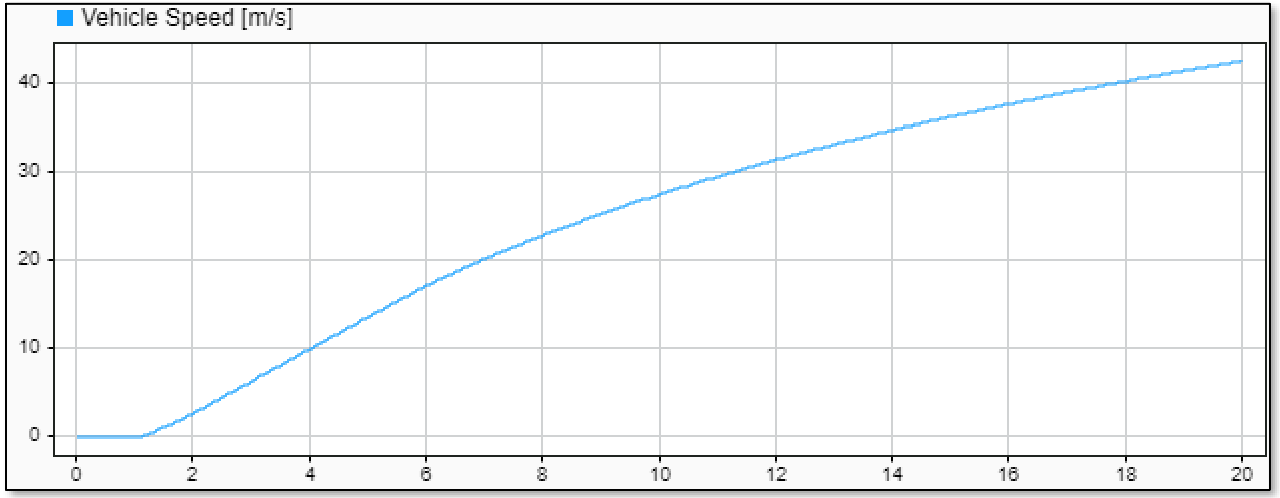
System-Level Results

Default component sizes don’t achieve system-level requirements. Time for a redesign!

Metric	Target	Results
Efficiency [Wh/km]	≤ 175	179.3
Battery cost [$]	≤ 7000	6428
Range [km]	≥ 300	286.8
$t_{0\text{-}100}$ [s]	≤ 8.0	8.3



WLTP test



WOT test

Summary: System-Level Simulation

- Key take-away
  - MathWorks provides system-level simulation tools to evaluate trade-offs early / quickly

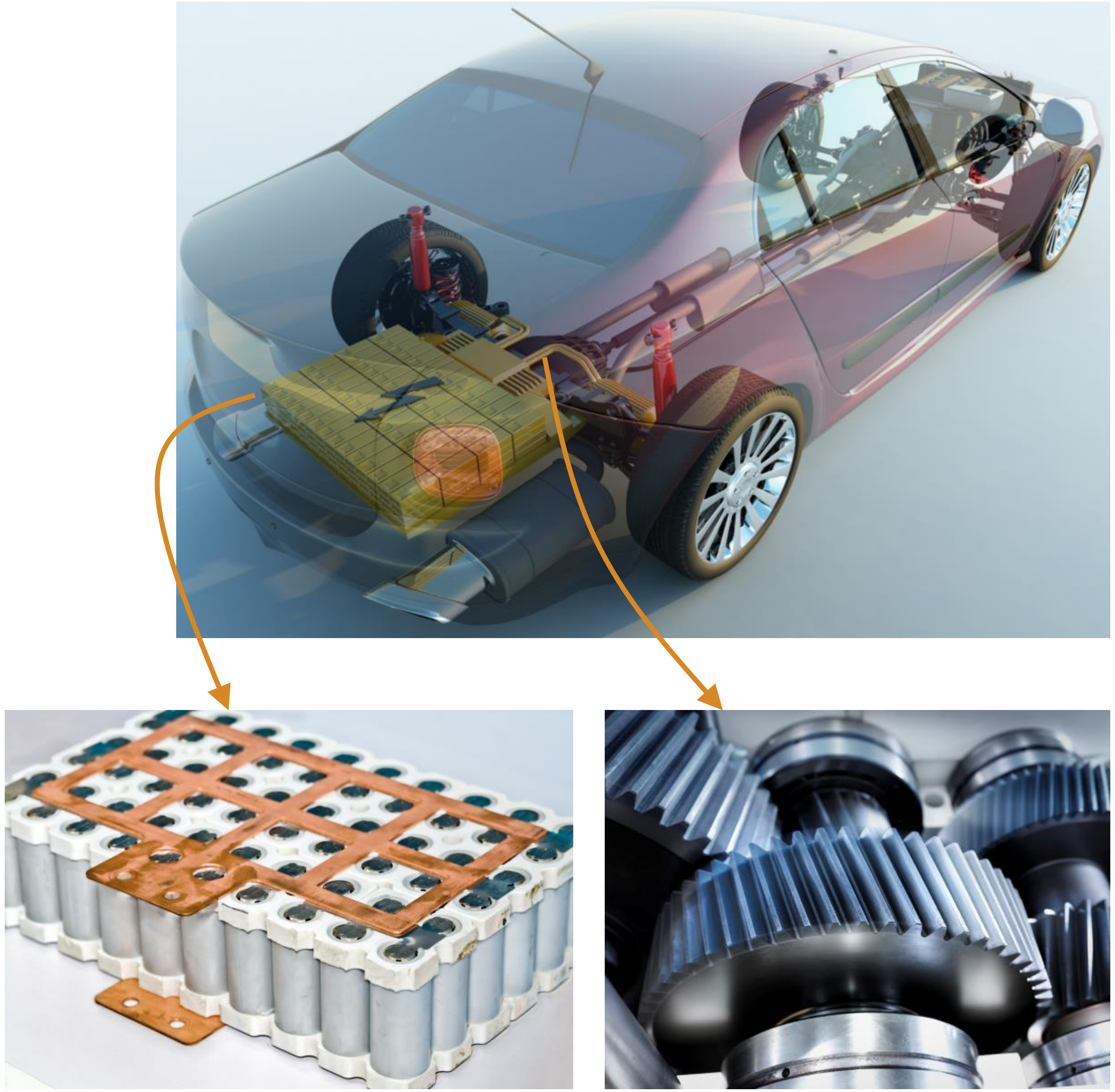
- Tips discussed
  - WLTP provides a single drive cycle to capture both city and highway driving
  - Virtual Vehicle Composer can quickly generate model of interest
  - Generated models can be customized as needed

Agenda: Determine battery pack size to meet system-level targets

- How to perform system level analysis?
- **How to evaluate battery efficiency and sizing?**

Component Sizing Problem Statement

- Goals:
  - Find battery size & gearing that provides good efficiency at a reasonable price
- Constraints:
  - Meets typical driving demands
  - Reasonable EV range
  - Reasonable acceleration
- Design Variables:
  - Number of battery cells in parallel (Np)
  - Number of battery cells in series (Ns)
  - Final drive ratio (Nd)



Component Sizing Problem Statement

$\min f(\mathbf{x}) = w_1 * ECR + w_2 * Cost$

subject to:

g_1: DriveCycleFault ≤ 0

g_2: Range $\geq$ 300 km

g_3: $t_{0\text{-}100} \leq 8$ sec

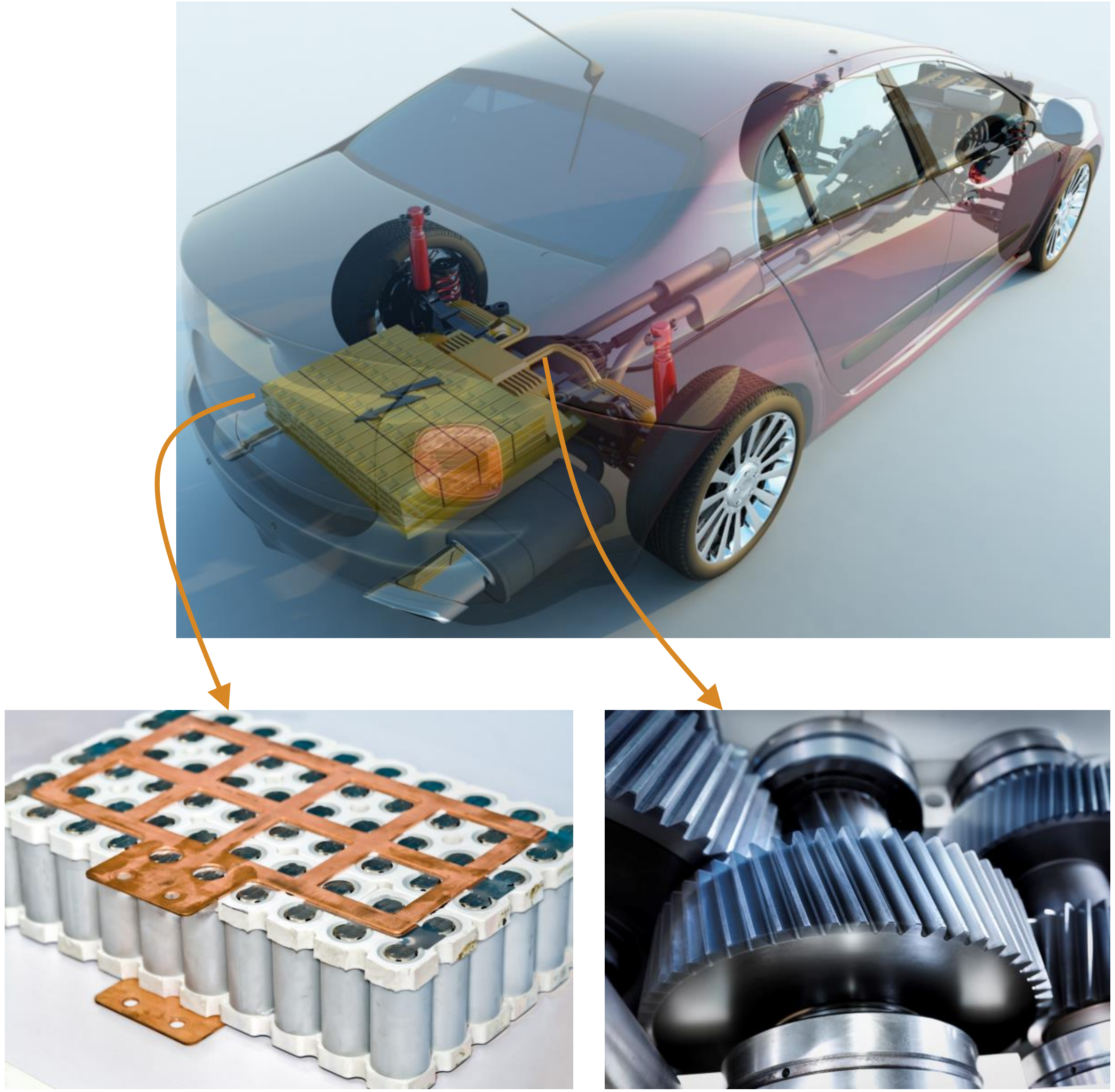
Where:

x_1: $10 \leq Np \leq 50$ *(Integer)*

x_2: $32 \leq Ns \leq 160$ *(Integer)*

x_3: $2 \leq Nd \leq 10$ *(Continuous)*

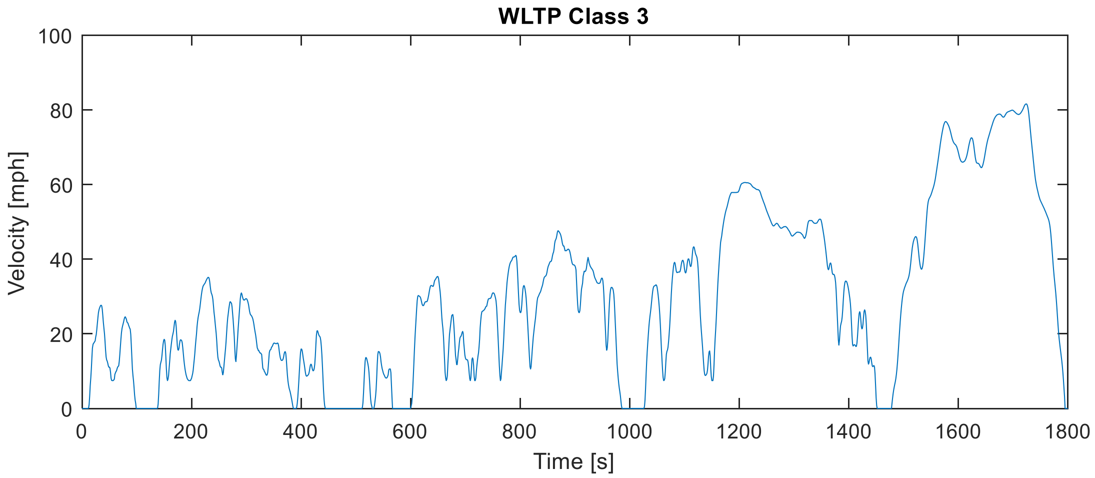
ECR = Energy Consumption Rate [Wh/km]

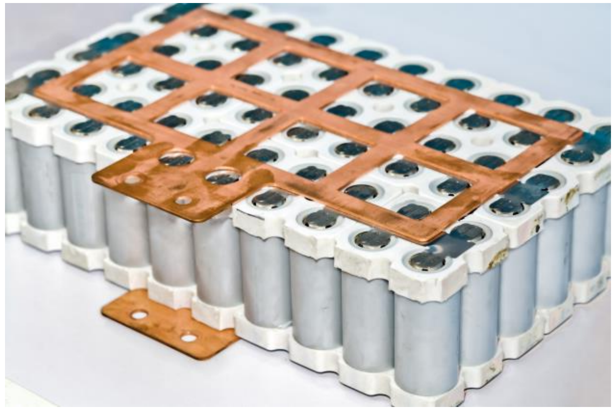


{Np, Ns} {Nd}

Assumptions

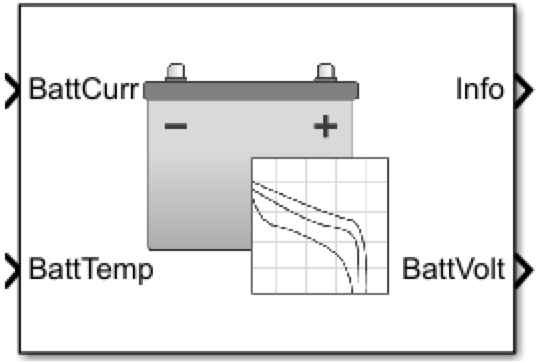
- System level metrics
  - ECR = battery power consumed over WLTP Class 3 / distance travelled
  - Range = battery capacity / ECR
  - Cost = battery capacity * $125 / kWh
- Battery
  - Cell characteristics: 4.8 Ah, 3.6 V (comparable to Tesla Model 3)
  - Energy density: 145 Wh / kg
- What's out of scope?
  - Packaging / geometry
  - Thermal cycling / aging
  - Component selection options (catalog)
  - Etc.

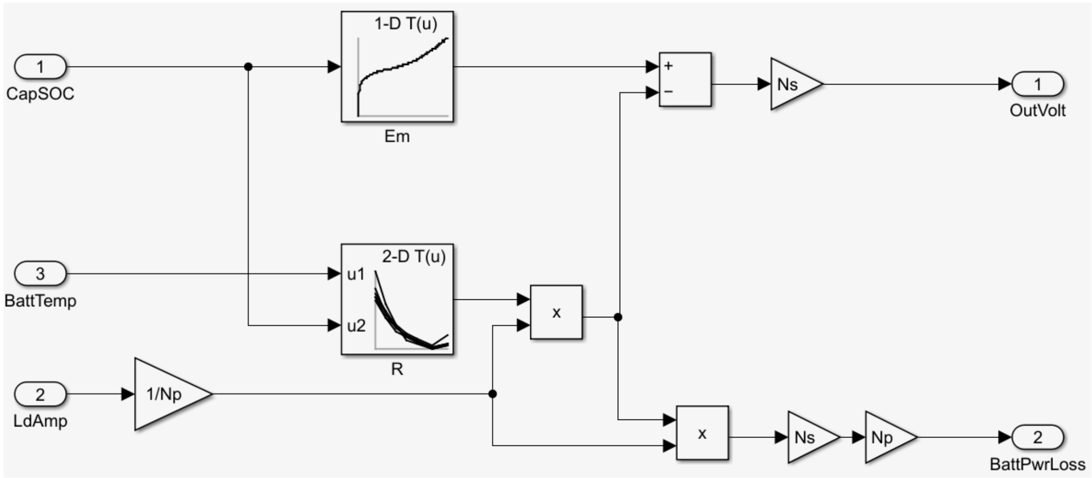




Battery Model

- Datasheet Battery block
  - Simple lumped, but fast model for system-level studies
  - Accounts for changes in Ns and Np
  - Temperature treated as external signal





$$E_m = f(SOC)$$

$$R_{int} = f(T, SOC)$$

$$V_T = E_m + I_{batt} R_{int}$$

$$I_{batt} = \frac{I_{in}}{N_p}$$

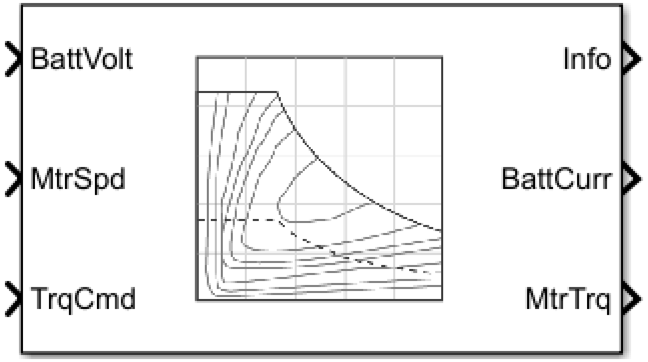
$$V_{out} = \begin{cases} N_s V_T & \text{unfiltered} \\ \dfrac{V_{out}}{\tau s + 1} & \text{filtered} \end{cases}$$

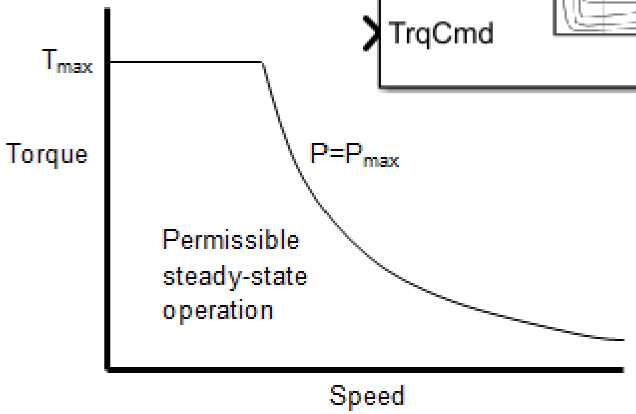
$$SOC = \frac{1}{Cap_{batt}} \int_0^t I_{batt} dt$$

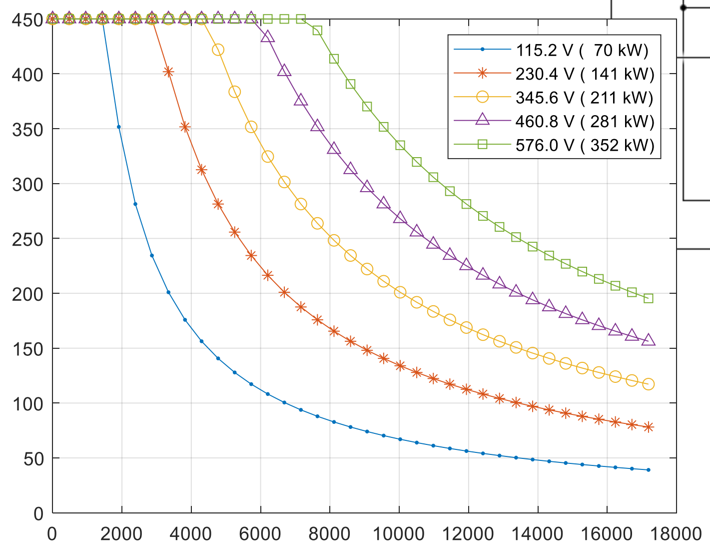
$$Ld_{AmpHr} = \int_0^t I_{batt} dt$$

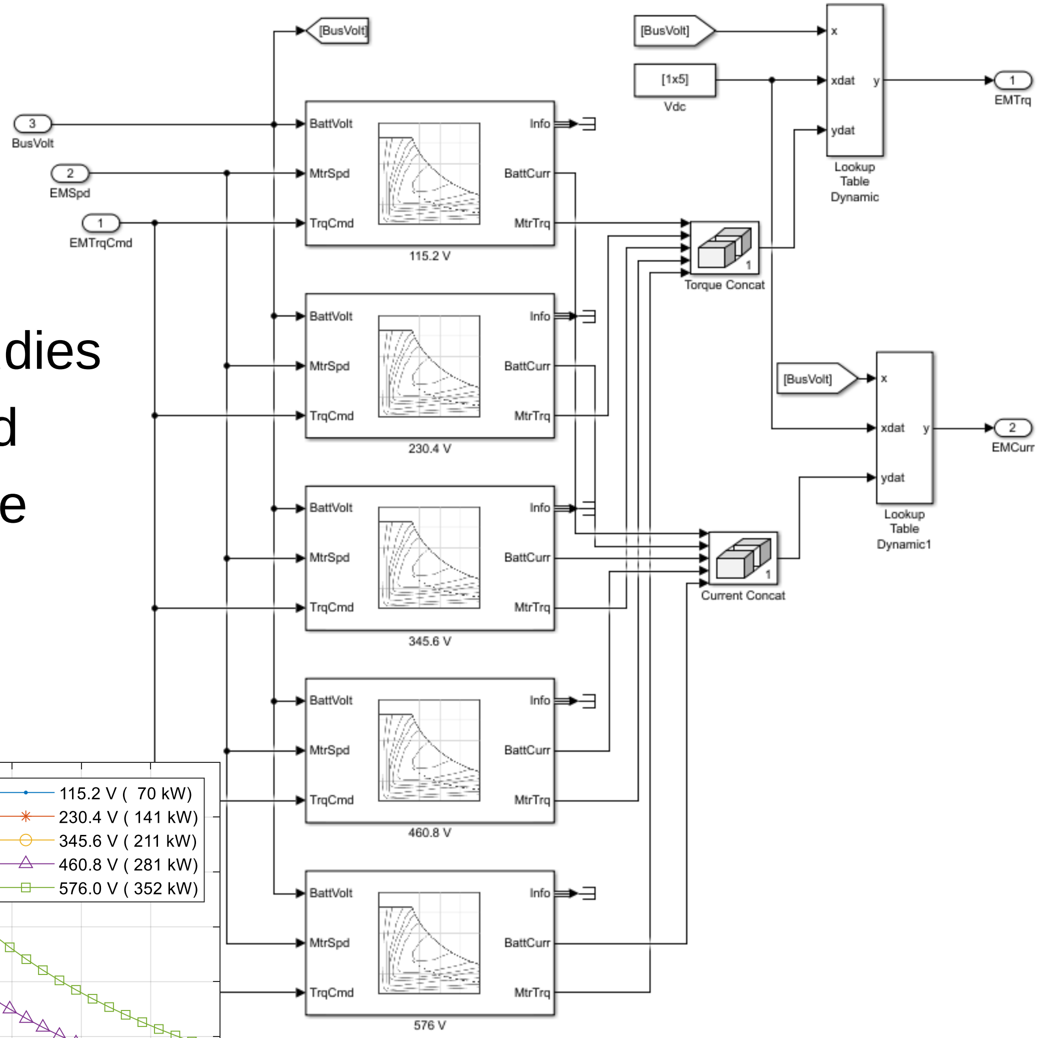
Motor Model

- Mapped Motor block
  - Simple lumped, but fast model for system-level studies
  - Neglects impact of bus voltage (Ns) on base speed
  - Used motor maps at 5 bus voltage levels to capture effect of Ns on max motor torque



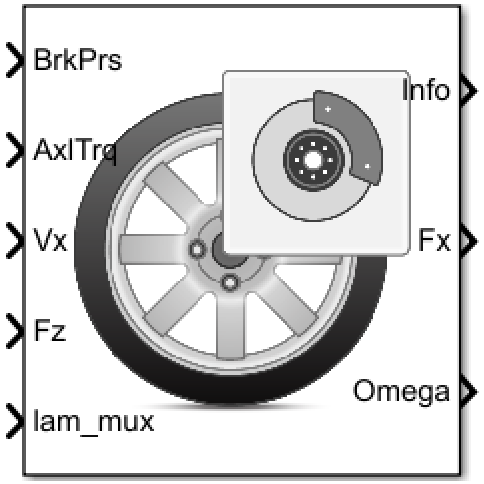


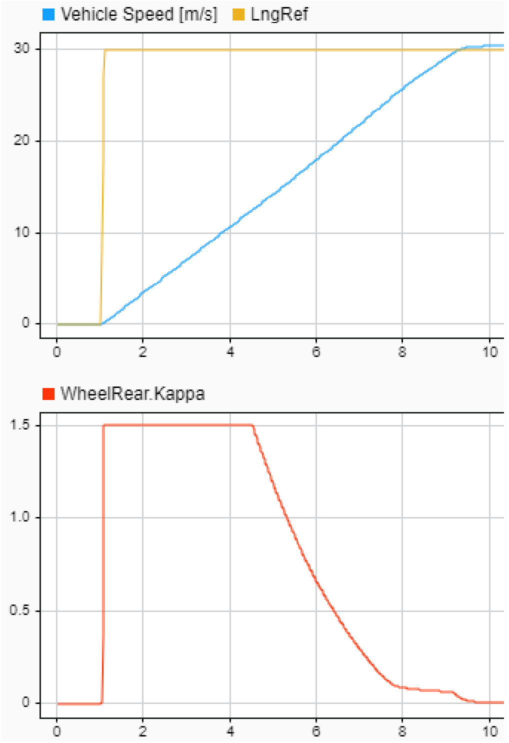




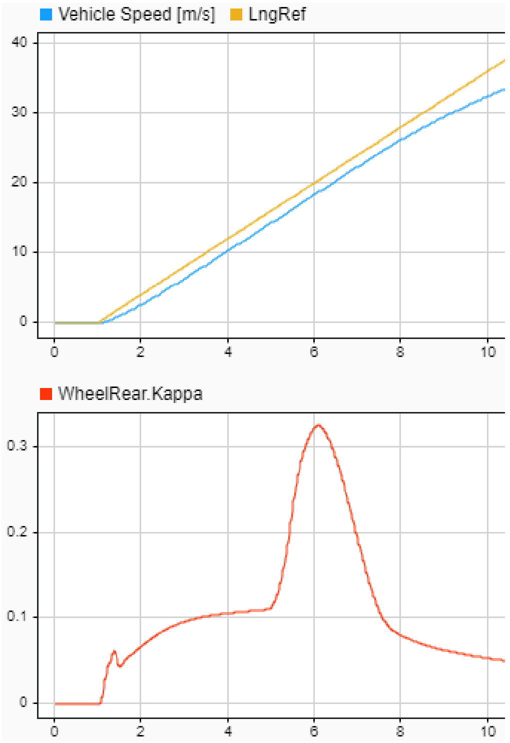
Tire Model

- Longitudinal Wheel block
  - Uses Magic Formula tire equations
  - Allows for scaling ground friction
  - Long list of parameters, often fit from tire test data
- Tire slip
  - Without good launch controls, WOT test can lead to excessive tire slip
  - Results in slower than expected $t_{0\text{-}100}$ performance
- Work-around
  - Apply torque gradually to minimize slip



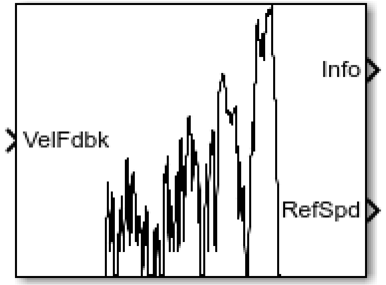


Tire slip saturates
$t_{0\text{-}100}$ = 7.6 s



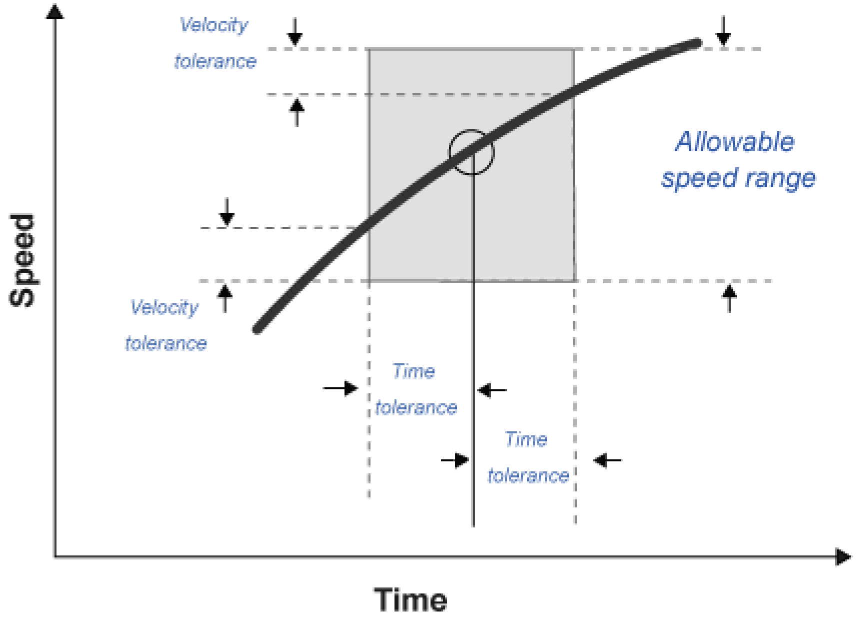
Tire slip in good range
$t_{0\text{-}100}$ = 7.3 s

Drive Cycle Faults

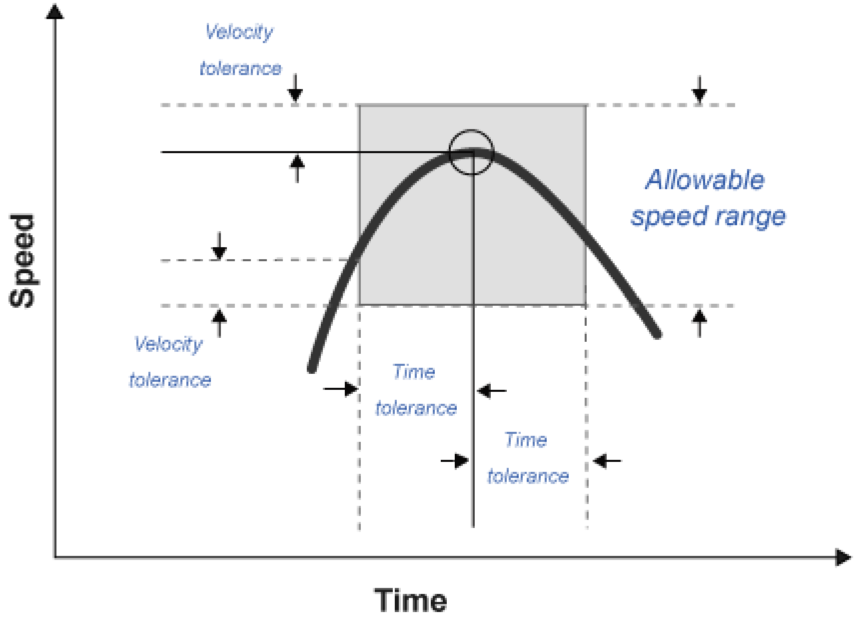


Drive Cycle Source

- Drive Cycle Source block
  - Includes options for fault tracking
  - WLTP allows for:
    - Velocity tolerance = 2 kph
    - Time tolerance = 1 s
    - Max faults = 10
    - Max single fault time = 1 s



- When simulations exceed allowances
  - Track cumulative time spent outside tolerance window
  - Provides more continuous measure of how “infeasible” design is



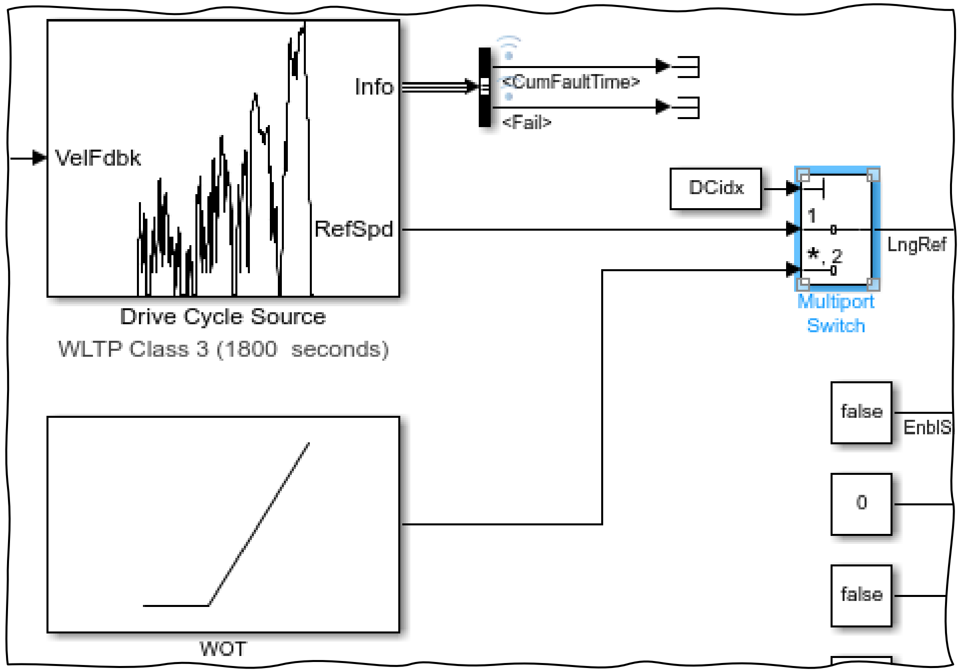
Design Trade-offs

	Metric improves as…		
Metric	**Ns**	**Np**	**Nd**
Mass	▼ (fewer cells)	▼ (fewer cells)	
ECR	▲ (higher max torque) ▼ (less mass)	▲ (lower resistance) ▼ (less mass)	▼ (more efficiency)
Cost	▼ (fewer cells)	▼ (fewer cells)	
Range	▲ (more energy) ▼ (less mass)	▲ (more energy) ▼ (less mass)	
Acceleration	▲ (higher max torque) ▼ (less mass)	▼ (less mass)	▲ (more wheel torque)

Numerical optimization provides a rigorous method to balance competing objectives

Preparing Models for Optimization Studies

- Simulation settings
  - Use “Accelerator” mode to compile model for faster execution time
  - Use “Fast Restart” to avoid recompiling in between sims
- Parameter handling
  - Use parameter-based Multiport Switch to change drive cycle source without recompile
  - Remove parameters from data dictionary / model workspace for simple script overrides




```
in = Simulink.SimulationInput(model);
in = in.setVariable('PlntVehMass', mass);
in = in.setVariable('PlntBattNumCellPar', Np);
in = in.setVariable('PlntBattNumCellSer', Ns);
in = in.setVariable('PlntDiffrntlRatio', Ndiff);
```

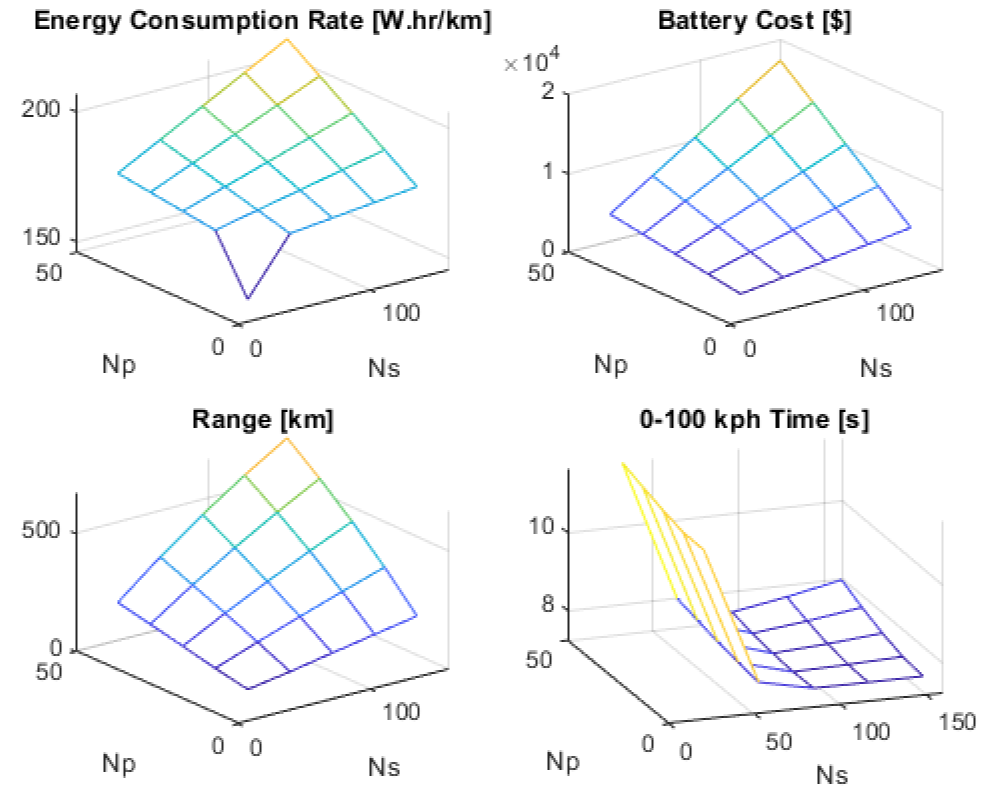
Running Simulations as a Function Call

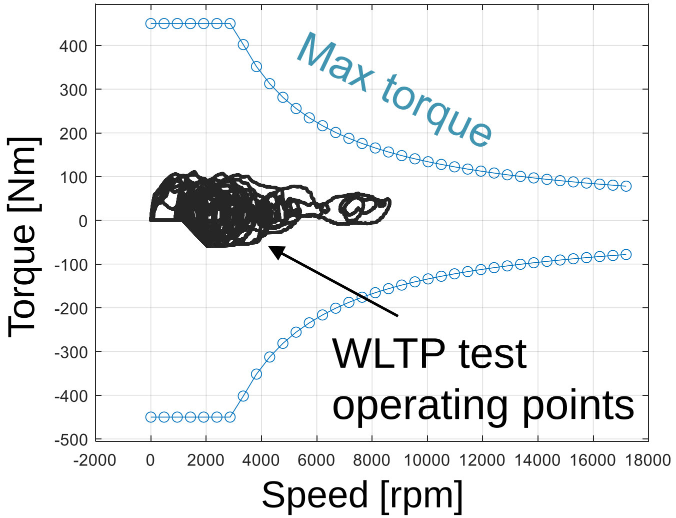
```
function [f, g, ECR, cost] = RunEV(Np, Ns, Ndiff, model)
% RunEV runs a series of EV sims for a given set of parameters

% Do some internal calculations
batt_energy = (4.8*Np)*(3.6*Ns)/1000; % 4.8 Ah/string, 3.6 V/cell
mass = 1250 + batt_energy/145*1000; % 145 Wh/kg
cost = 125 * batt_energy; % assume cell cost of $125/kW.hr

% Create Simulation Input object to store temporary parameter overr
in = Simulink.SimulationInput(model);
in = in.setVariable('PlntVehMass', mass);
in = in.setVariable('PlntBattNumCellPar', Np);
in = in.setVariable('PlntBattNumCellSer', Ns);
in = in.setVariable('PlntDiffrntlRatio', Ndiff);

% Run WLTP drive cycle
in = in.setVariable('DCidx', 1);
in = in.setModelParameter('StopTime','1800'); % 23.266 km long test
simout = sim(in);

% Post-process WLTP result
logsout = simout.get('logsout');
DCerror = logsout.get('DCFaultTime [s]').Values.Data(end);
DCfail = logsout.get('DCFail').Values.Data(end);
bp = logsout.get('Battery Power [W]').Values;
v = logsout.get('Vehicle Speed [m/s]').Values;
% Energy consumption rate in [W.hr/km]
ECR = trapz(bp.Time,bp.Data)/3600/(trapz(v.Time,v.Data)/1000);
range = batt_energy / ECR * 1000; % range in km
```

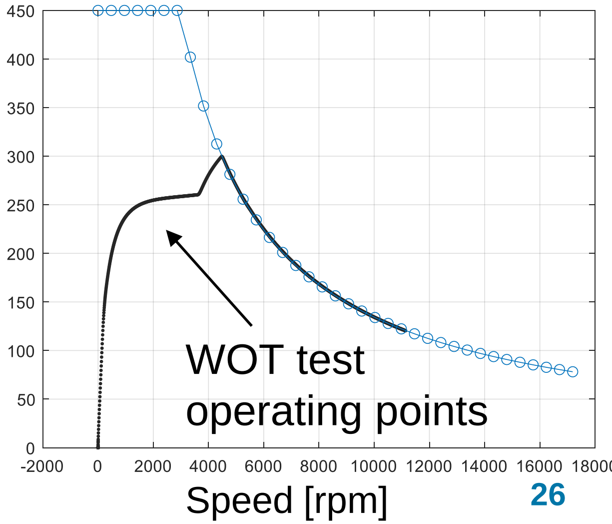
```
% Run 0-100 kph test
in = in.setVariable('DCidx', 2);
in = in.setModelParameter('StopTime','20');
simout = sim(in);

% Post-process WOT result
logsout = simout.get('logsout');
v = logsout.get('Vehicle Speed [m/s]').Values;
try
    id = find(v.Data>0.1,1,'first');
    t0 = interp1(v.Data(id-1:id),v.Time(id-1:id),0.1);
    id = find(v.Data>27.778,1,'first');
    t100 = interp1(v.Data(id-1:id),v.Time(id-1:id),27.778);
    t0_100 = t100 - t0;
catch
    t0_100 = 100;
end

% Assemble results into objective and constraint values
f=[]; g = struct();
w = [0.5, 0.5]; % relative weights for the objectives
s = [150, 6250]; % scale factor to normalize objective terms

f = ECR*w(1)/s(1) + cost*w(2)/s(2);
g.DCerror = DCerror*DCfail; % total drive cycle fault time (or 0 if passed)
g.range = 300 - range; % range > 300 km
g.t100 = t0_100 - 8.0; % t0_100 < 8.0 sec

end
```

Initial Assessment

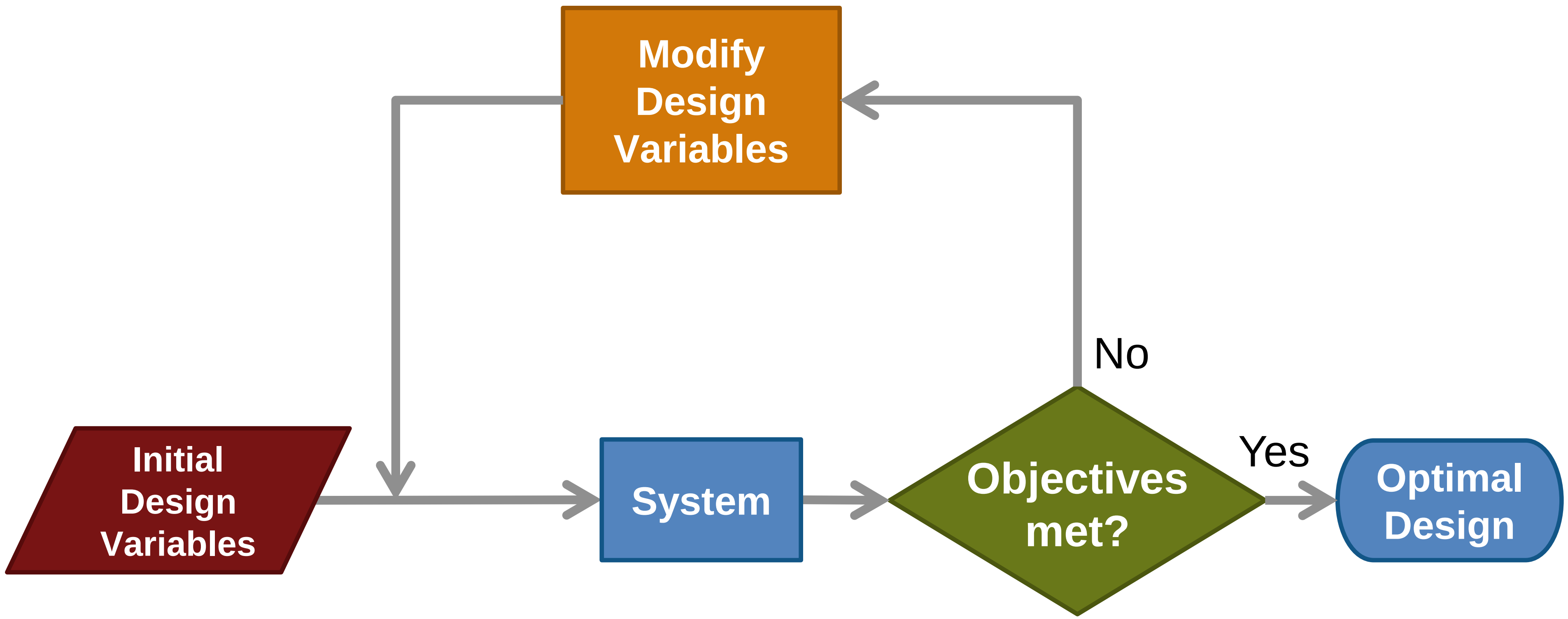
- Performed initial parametric study
  - Sweep of Np, Ns for fixed Nd
  - Study problem statement before launching long optimization study
- Lessons learned
  - ECR trend was unexpected
  - Dominated by mass penalty, not benefits of more power (Ns) / lower losses (Np)
  - WLTP never pushed motor to max torque / power limits





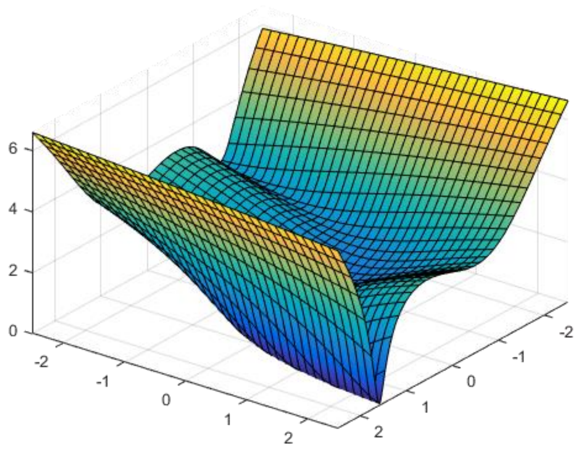


Design Process



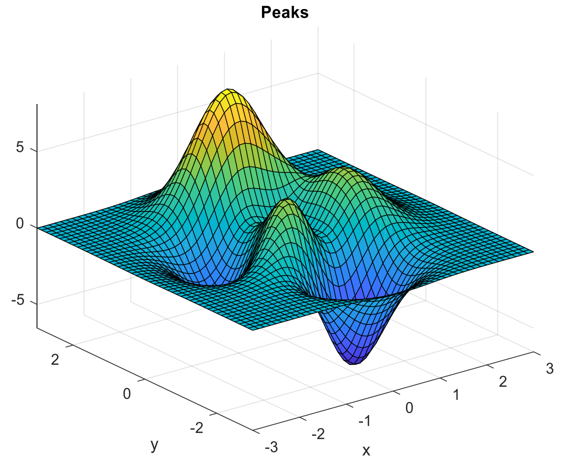
MathWorks Optimization Products

- **Optimization Toolbox**
  - Functions for finding parameters that **minimize or maximize objectives** while **satisfying constraints**



Objective with single minimum

- **Global Optimization Toolbox**
  - Functions that **search for global solutions** to problems that contain **multiple maxima or minima** on **smooth or nonsmooth** problems *(requires Optimization Toolbox)*

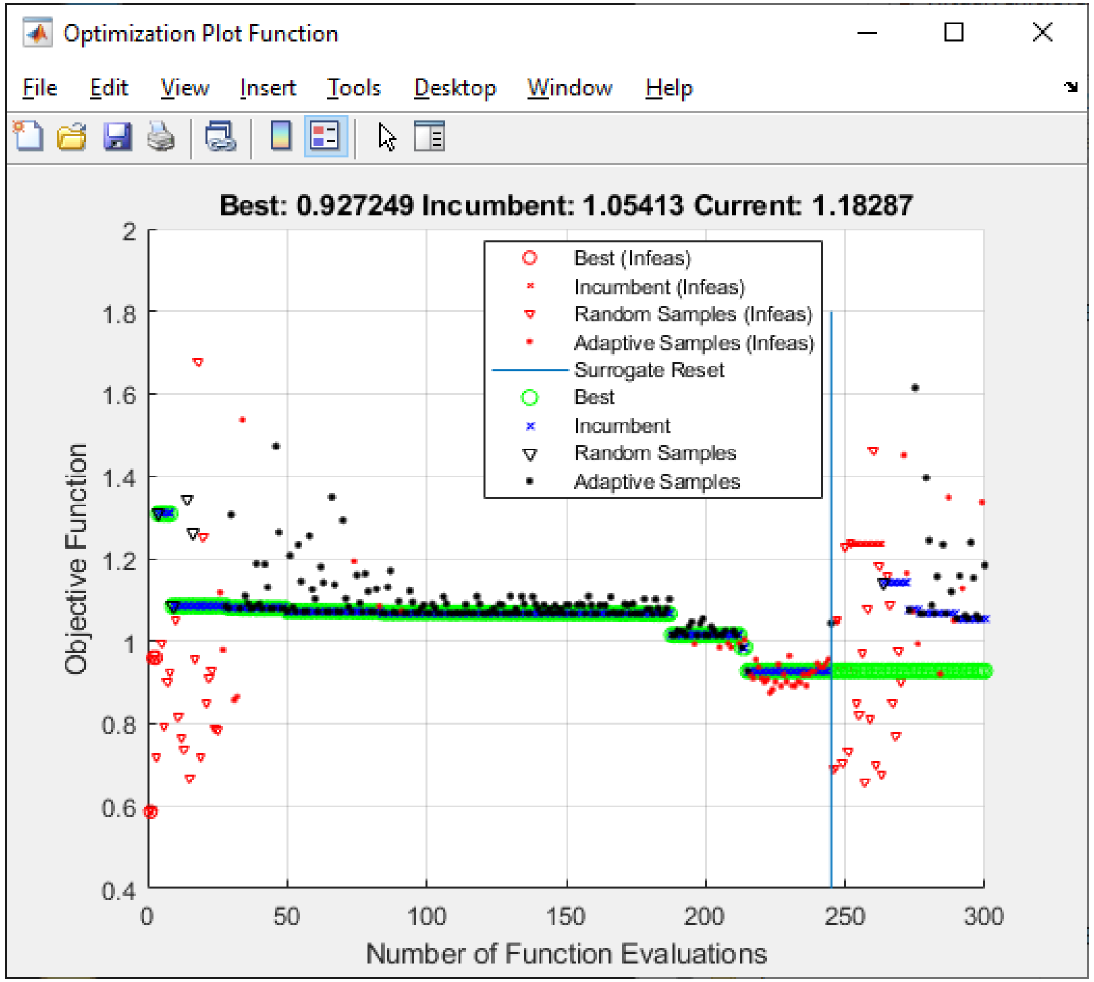


Objective with multiple minima

Learn more:
[Optimization Toolbox]
[Global Optimization Toolbox]

Optimization Results

Metric	Baseline	Optimized (% improvement)
ECR [Wh/km]	179.3	172.5 (-3.8%)
Cost [$]	6428	6484 (+0.9%)
Range [km]	286.8	300.6 (+4.8%)
$t_{0\text{-}100}$ [s]	8.3	8.0 (+3.6%)
Nd	7.97	4.88
Battery cells	96s31p	158s19p
Bus voltage [V]	345.6	568.8
Capacity [kWh]	51.4	51.9



Performed 300 function calls (~3 hours)

Pareto Optimization Studies

- Reassess problem with tradeoff between range and cost (direct competition for battery size)

$$\min f(\mathbf{x}) = -w_1{*}Range + w_2{*}Cost$$

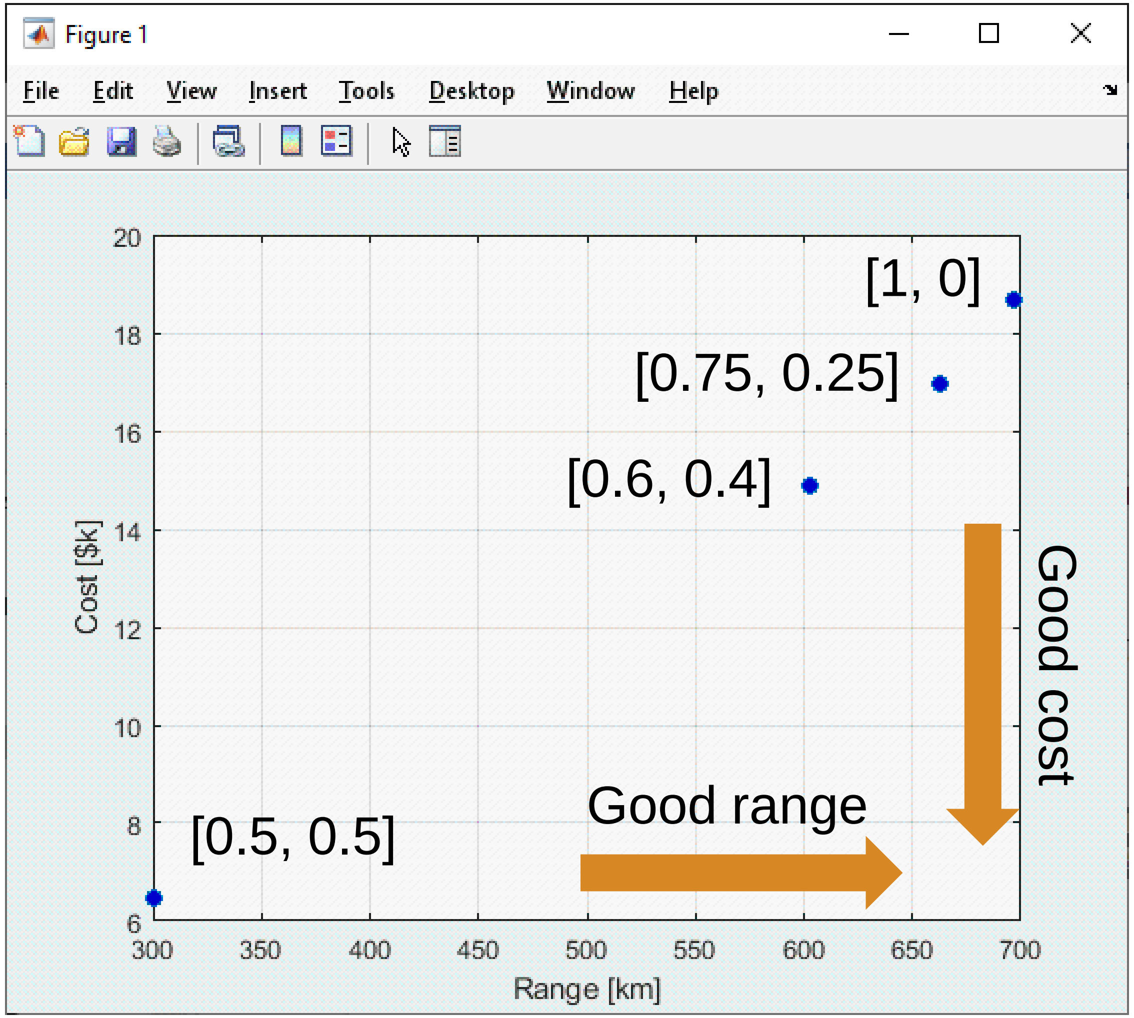
subject to:

g_1: DriveCycleFault ≤ 0

g_2: Range $\geq$ 300 km

g_3: $t_{0\text{-}100} \leq 8$ sec

- Sweep weights to quantify tradeoffs
  - $w_2 < 50\%$: range constraint is active
  - $w_2 \geq 50\%$: cost objective gets outweighed



Summary: Component Sizing

- Key take-away
  - Optimization is a rigorous means to identify best parameter values

- Tips discussed
  - Customize model to enable fast-restart / accelerator mode
  - Watch for tire slip during aggressive acceleration (e.g., WOT test)
  - **Specifying the right problem statement can be iterative process**
  - Additional constraints can be added (e.g., limit selection to parts catalog)
  - Optimization studies can **quantify tradeoffs** between competing objectives

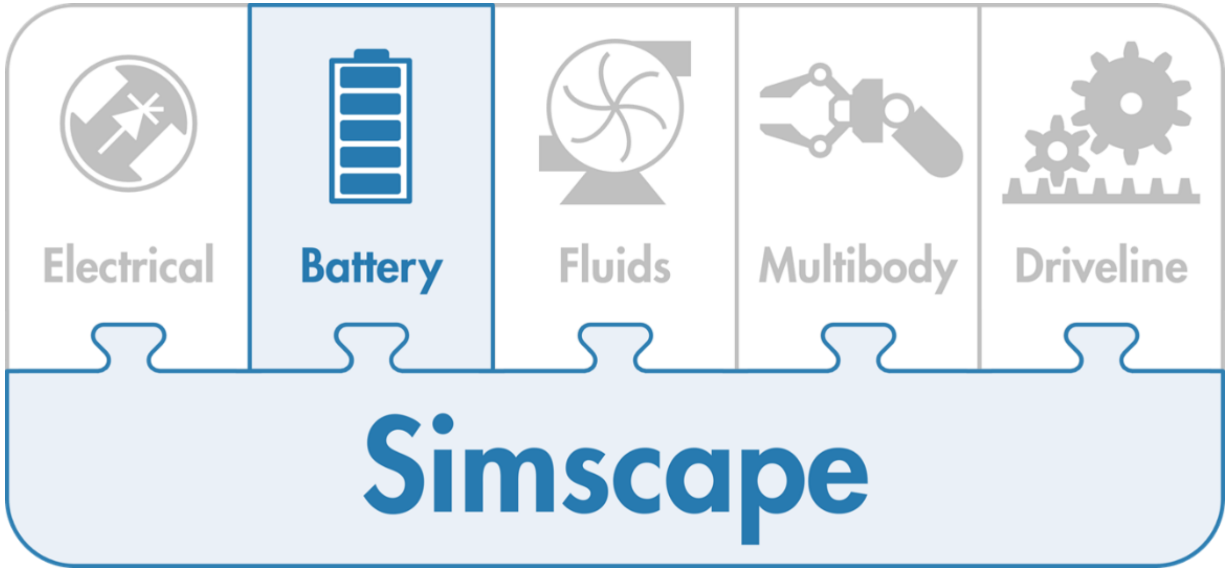
Things to consider for creating the battery pack?

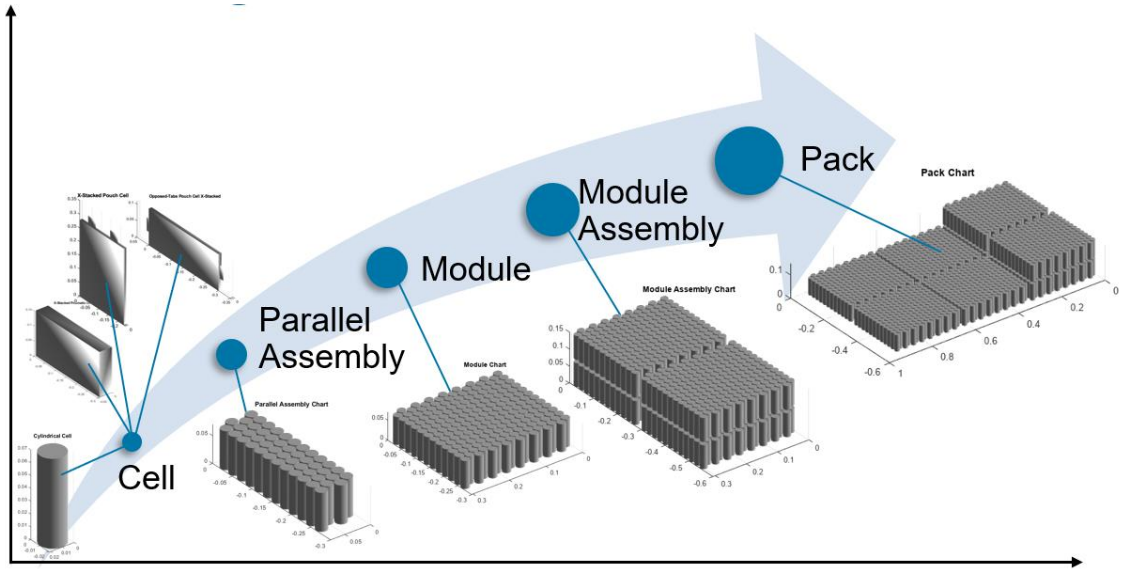
- Till now –
  - We choose Ns and Np suitable for our targets

- We need to address –
  - How to assemble the cells?
  - How to organize the modules?
  - What kind of cooling to use? How should be place the thermal barriers, etc.

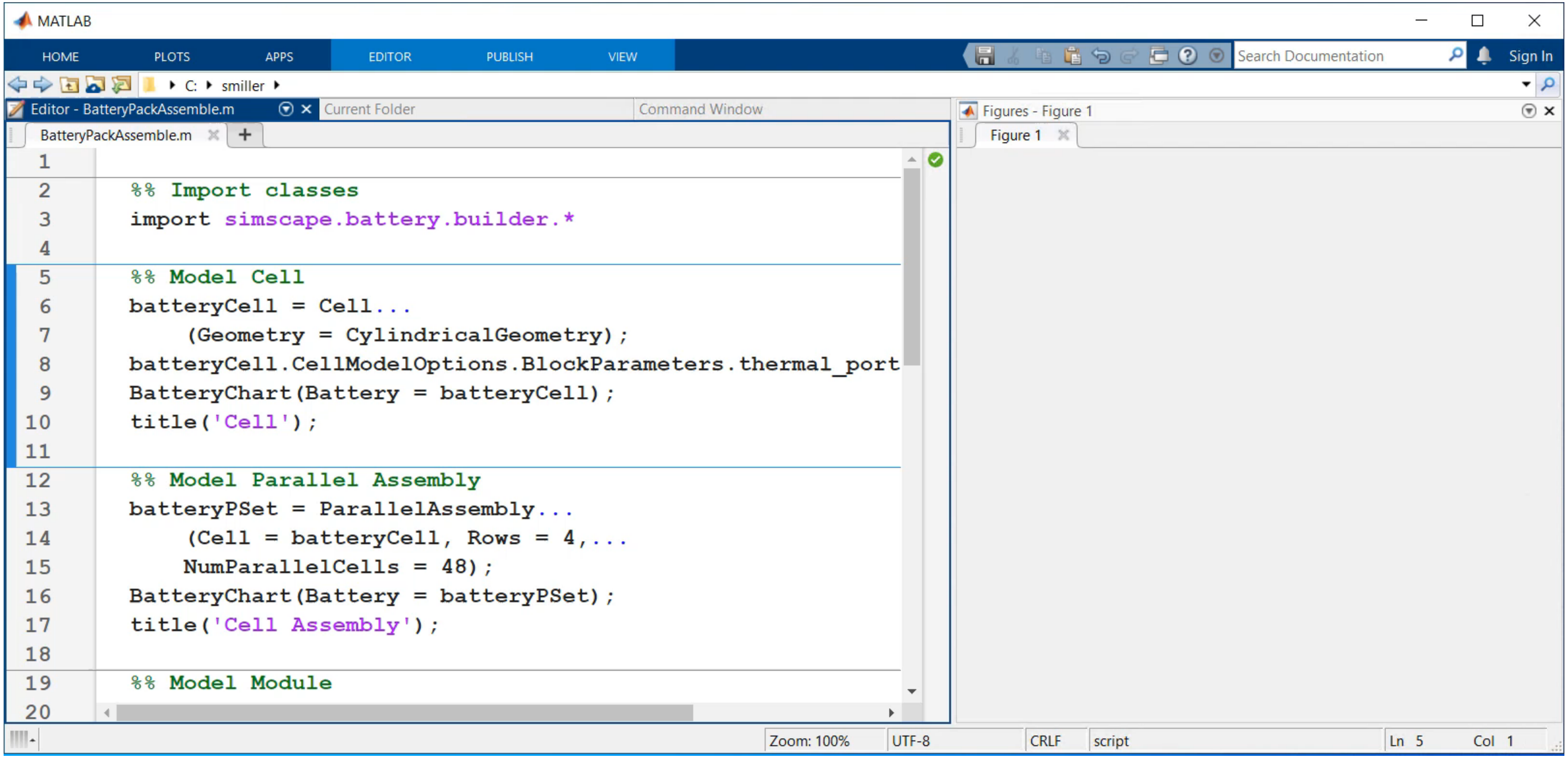
Simscape Battery

New product launched in R2022b





Simscape Battery Pack Builder



Various Modelling Approaches

Battery Models

System-Level Behavior	Component Validation	Component Design

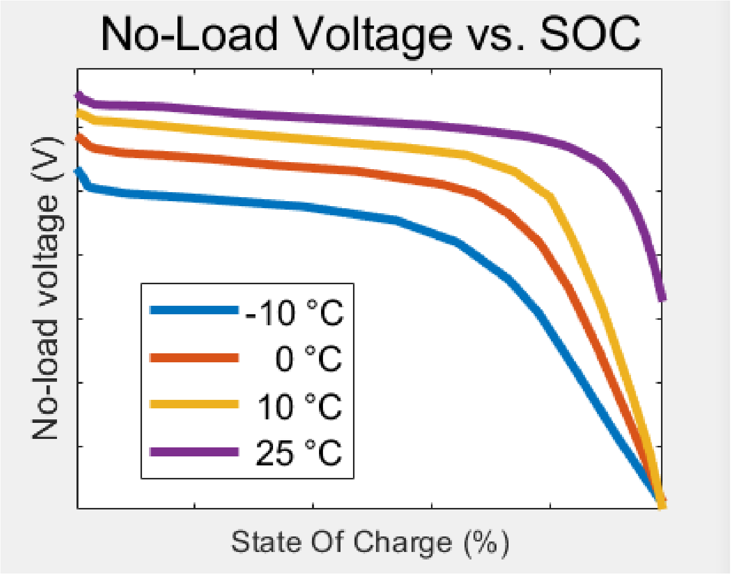
Low Fidelity **Charge/Discharge Characteristic** High Fidelity

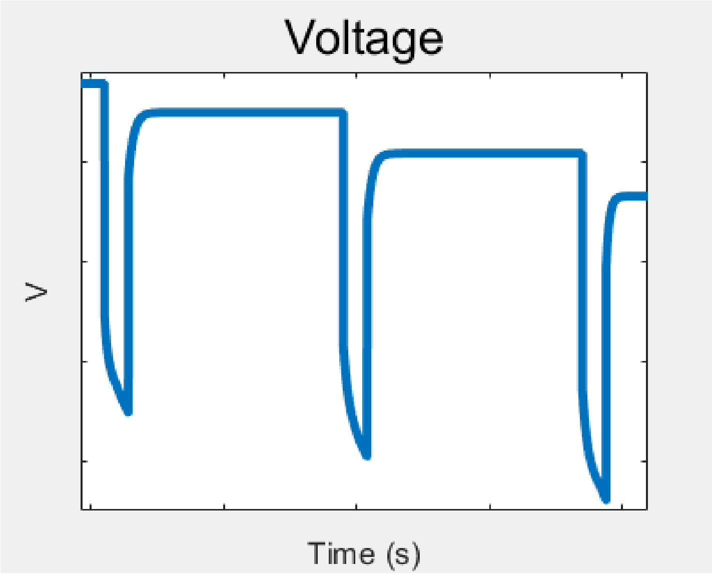
Infinite Charge

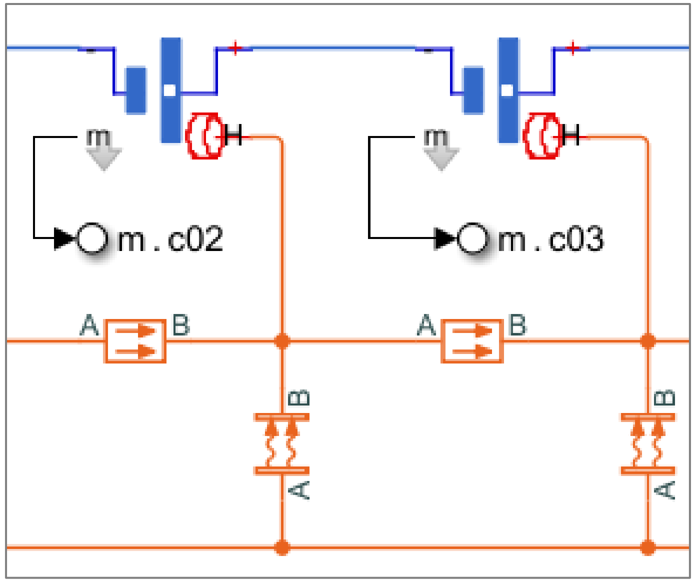
Finite Charge, Fade, Temperature Dependent

Charge Dynamics

Cell Behavior



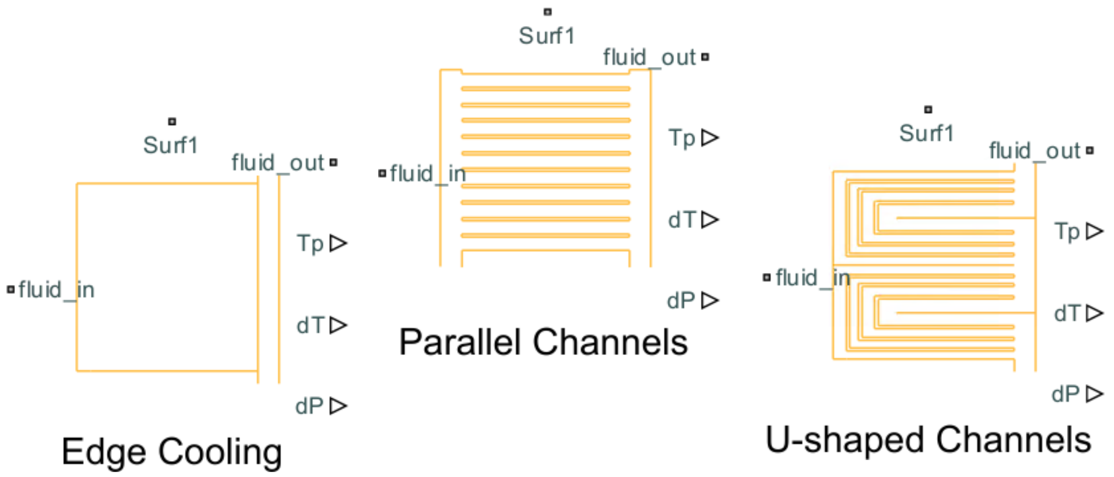




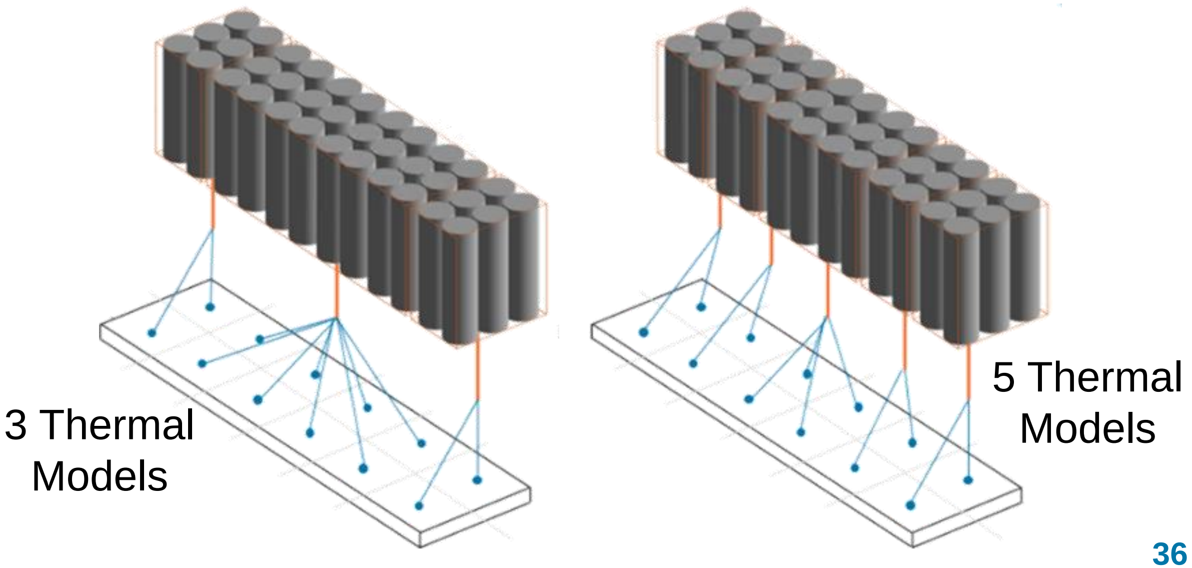
Simscape Battery Capabilities

Modeling API, Cooling Plates, Battery Management Algorithms

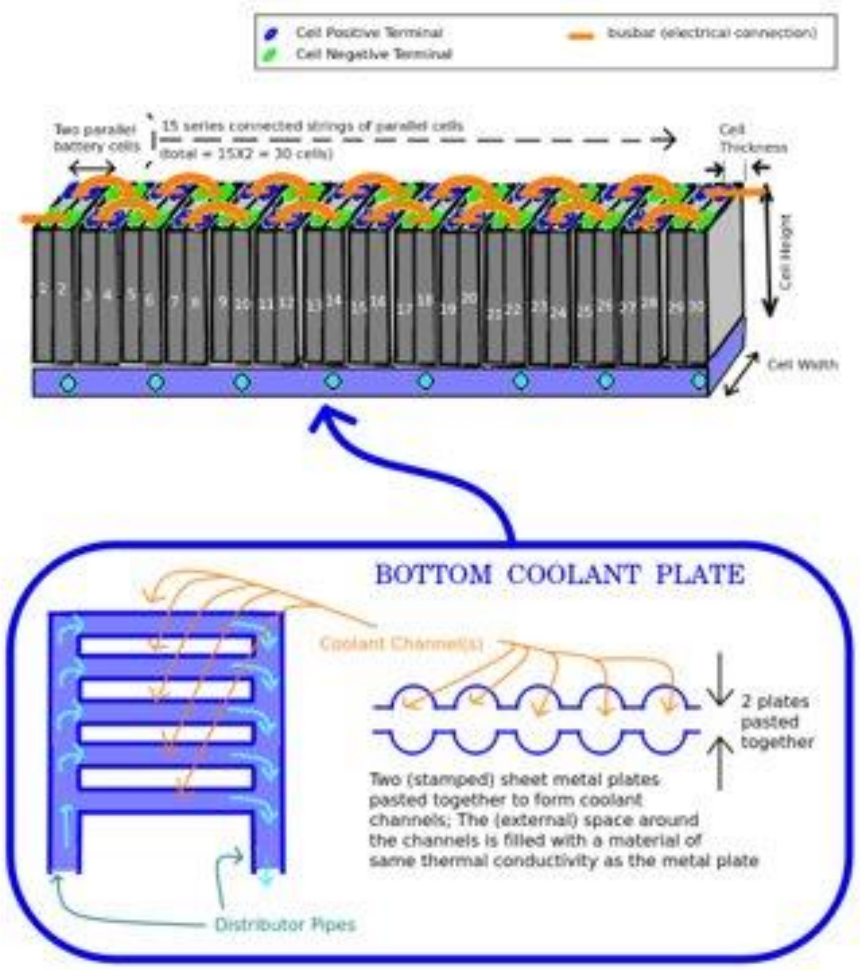
- Model heat transfer between battery, liquid cooling system, and environment
  - Control cell-to-cell temperature variation
  - Tradeoff of pumping costs and cooling efficiency



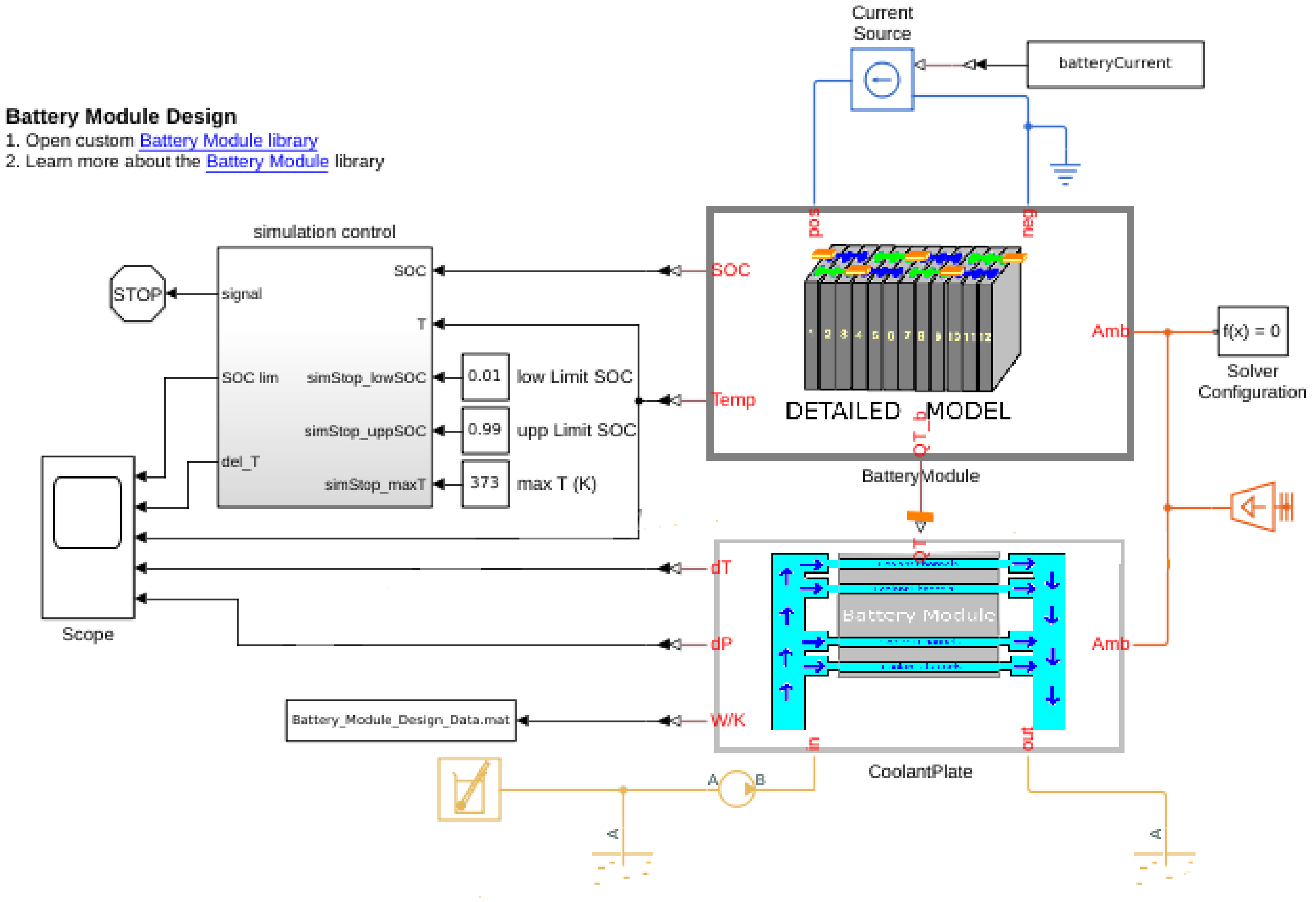
- Different cooling plate topologies
  - Edge, parallel channel, U-shaped channel
  - Single- and double-sided plates
- Adjust resolution of thermal model
  - Define quantity and placement of nodes



Battery Pack Design
Electrical & Thermal



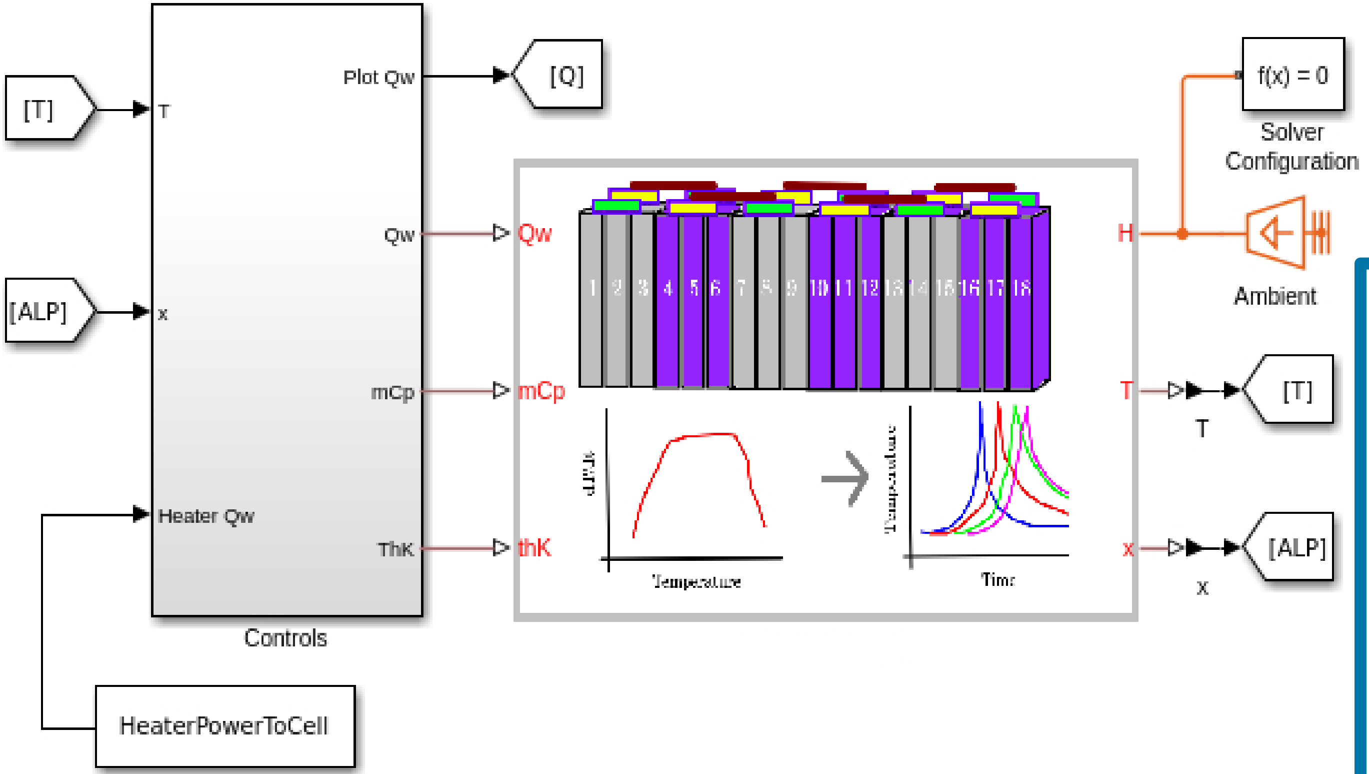
Physical Layout



Simscape Implementation

Example – Thermal Runaway

Simscape Component



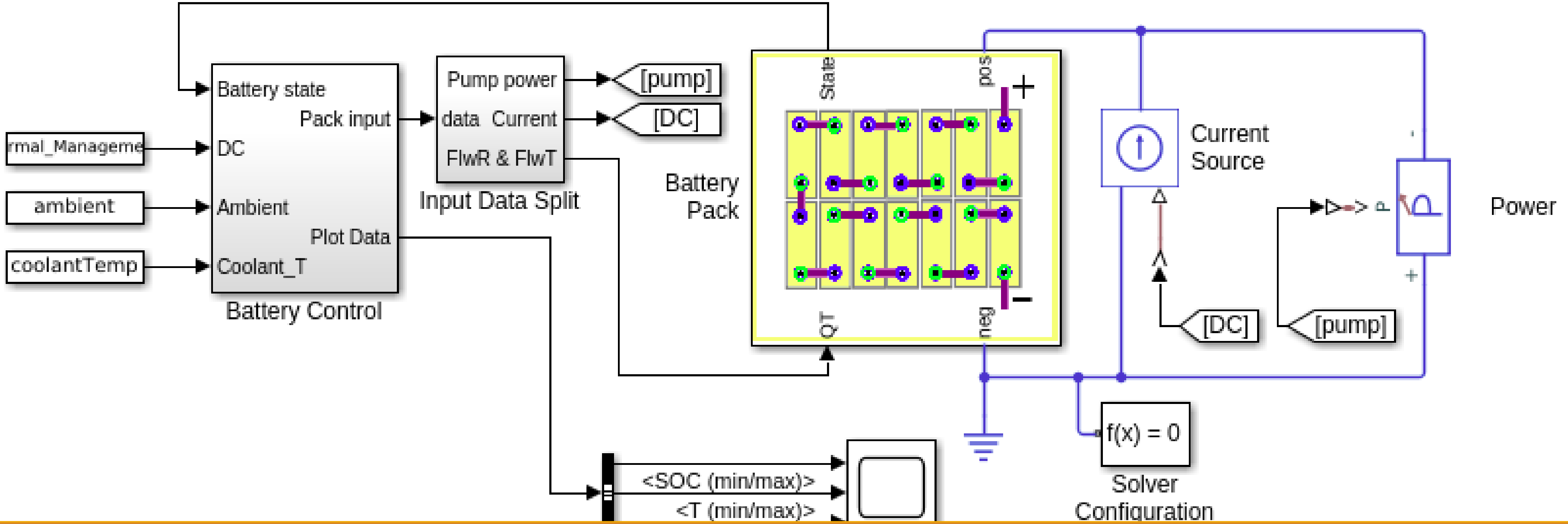

```
let
    % Check if temperature limit for thermal runaway initation reached
    % If reached, calculate cell abuse heat
    triggerTcrossed = (T>=triggerT); % 0 or 1 (Qabuse valid)
    Qabuse = triggerTcrossed*mass*sp_heat*tablelookup(cell_Tvec,...
        cell_dTdt,T,interpolation=linear,extrapolation=nearest);
    % Check if all reactants in cell have been consumed
    % If yes, no more abuse heat generation
    % No burning or phase change of cell material considered
    cellDead = (alpha>=maxAlpha);
    alphaRate = (unityDummy-cellDead)*(Qabuse/(rxnHeat*activeMass));
in
    % Calculate extent of reaction and cell abuse heat
    alphaRate == alpha.der;
    Q + (unityDummy-cellDead)*Qabuse == frac * mass * sp_heat * T.der;
end
```

Energy Balance Equations

Calculate heat load due to cell abuse reactions

Battery Pack Design Examples
Thermal Management

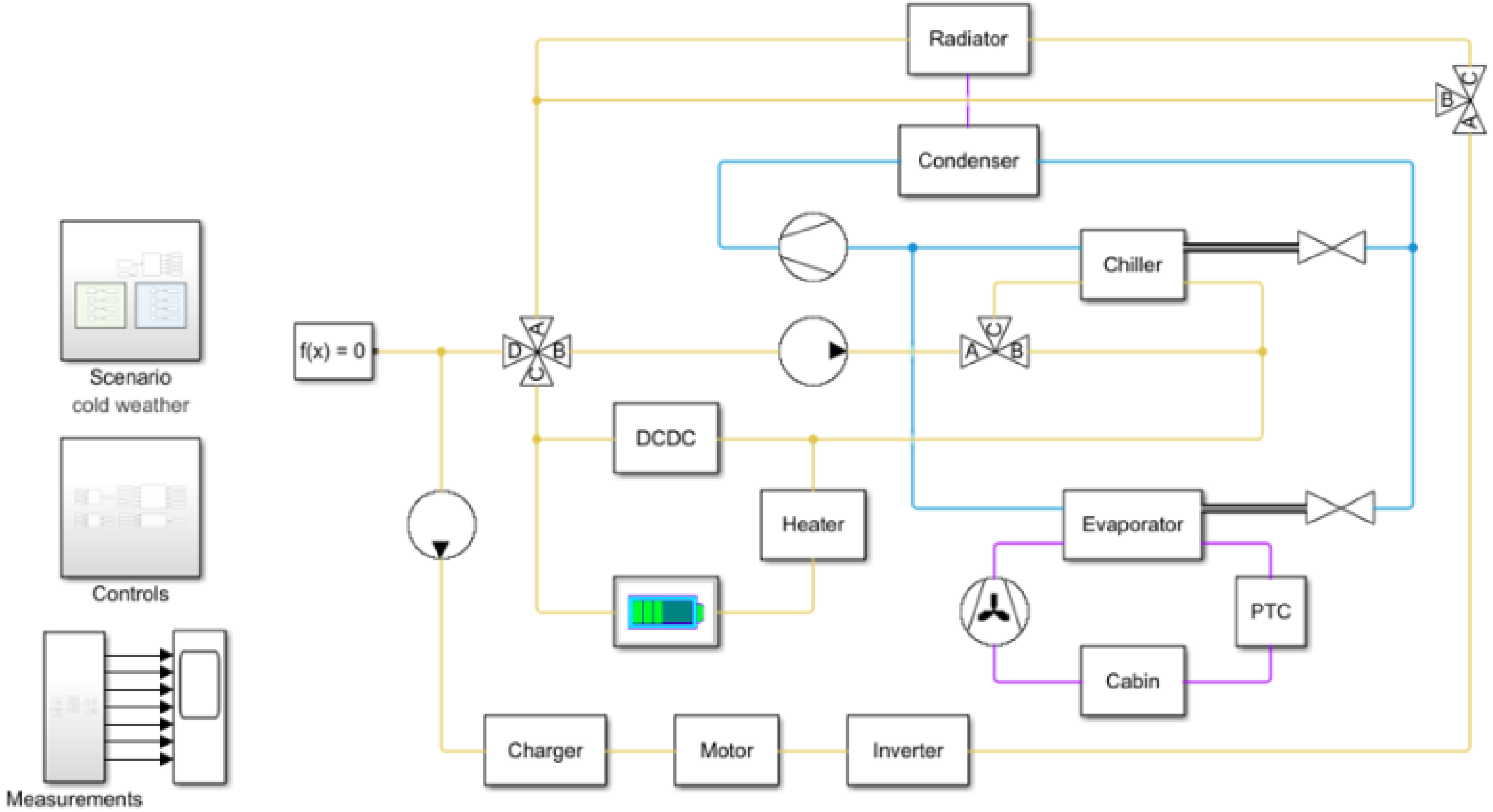
- Analyze cell-to-cell temperature gradient and devise thermal management strategies, robust BMS



Ability to track different weak/strong cells in the entire pack and design robust strategies for managing temperature, electrical safety, and pack utilization from a range perspective.

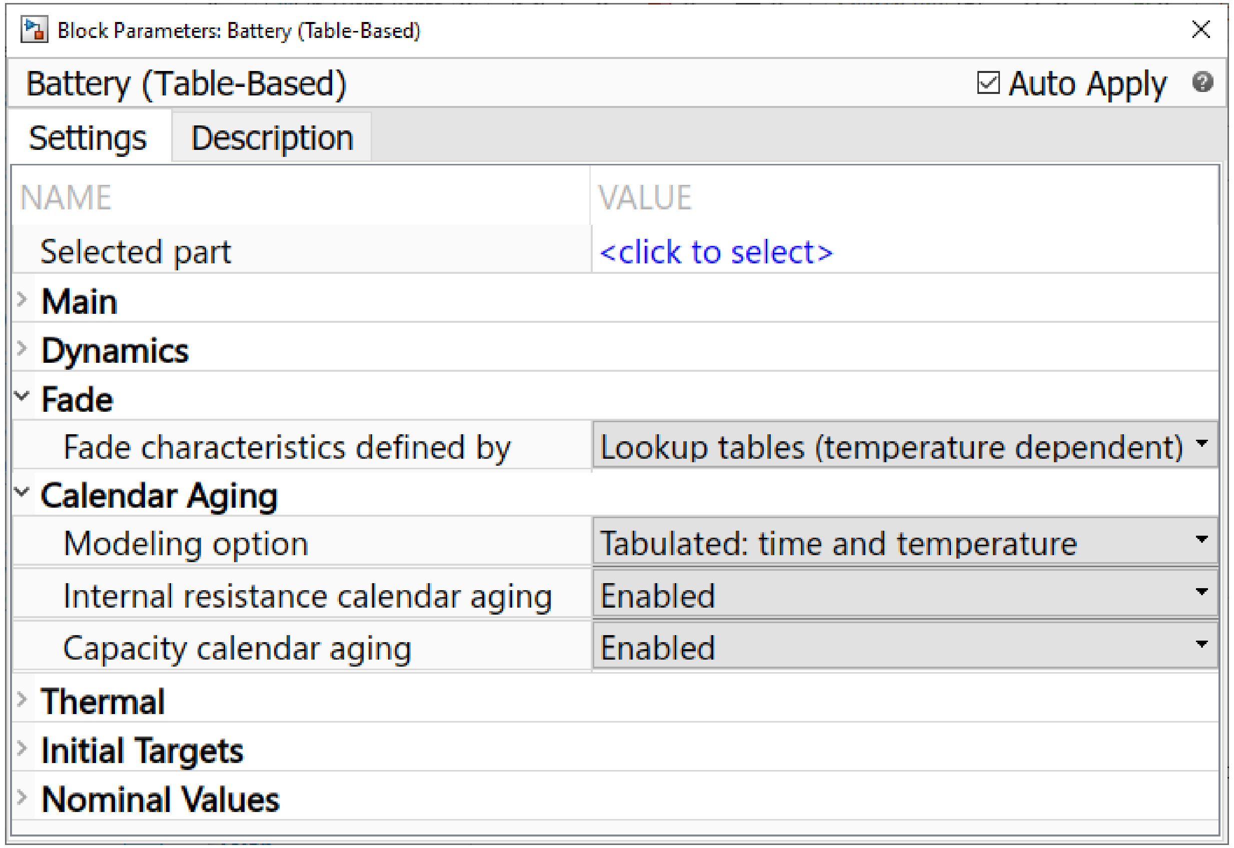
Battery Pack Design Examples
Thermal Management for BEVs

- Battery pack cooling strategy – single or separate cooling systems



Model Degraded Battery Behavior

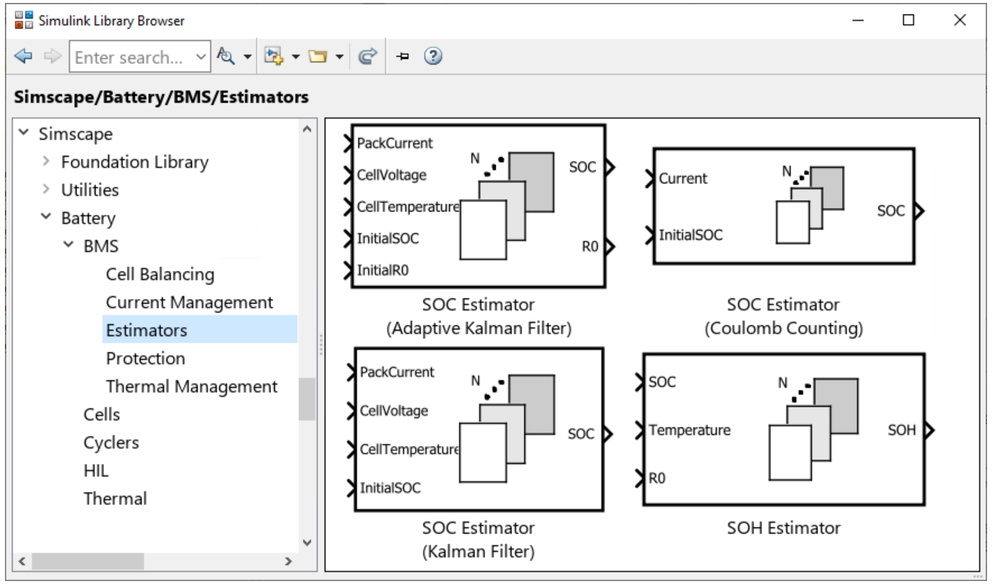
- Model age-related degradation of battery performance
  - Specify dependence of other battery parameters on the charge-discharge history
  - Specify calendar aging

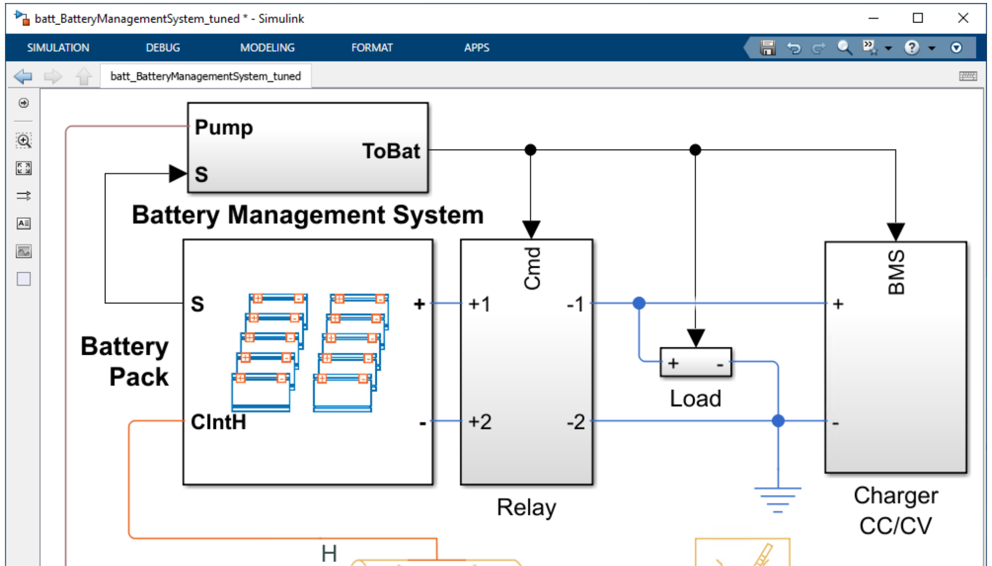


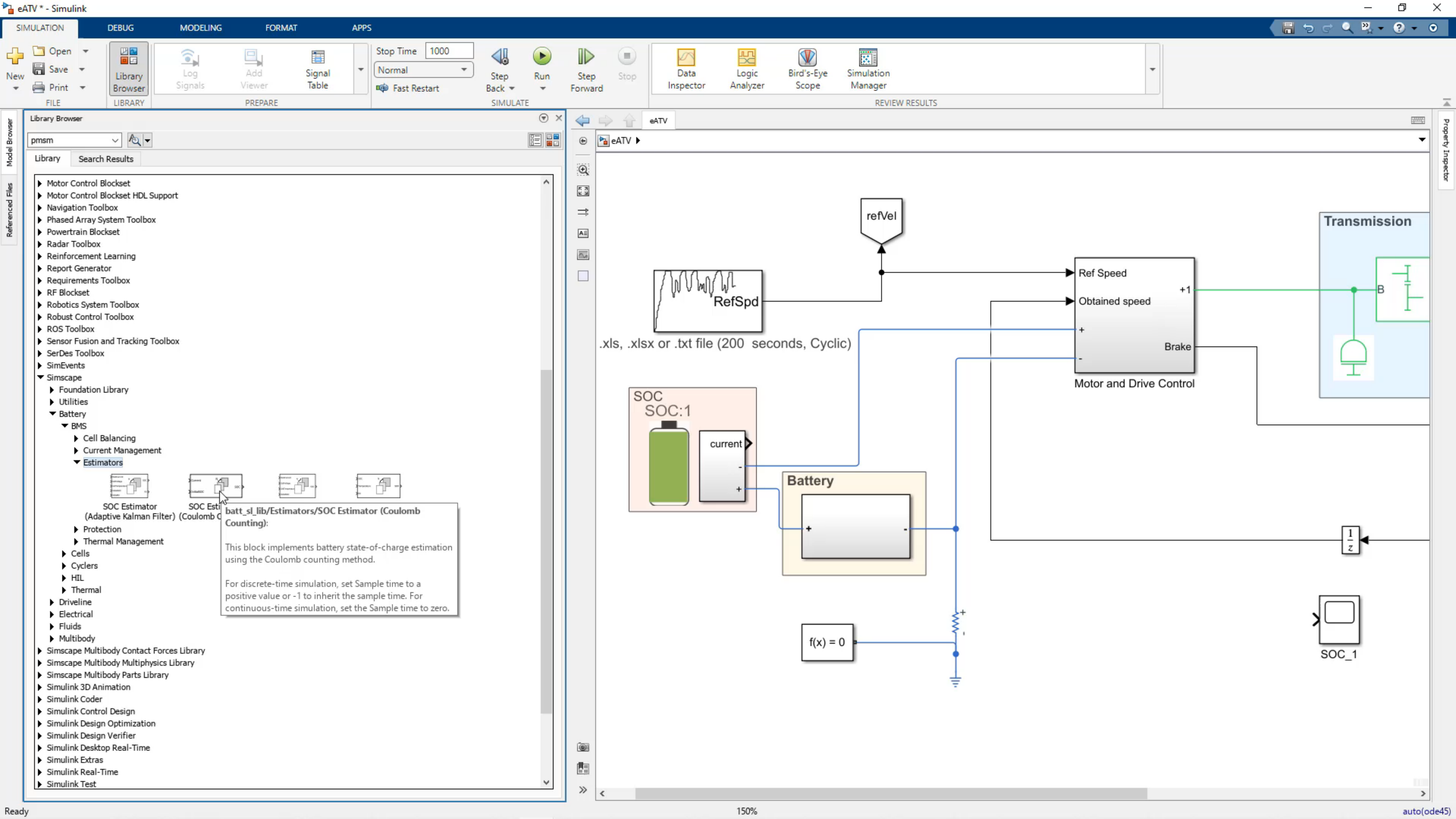
Simscape Battery Capabilities

Modeling API, Cooling Plates, Battery Management Algorithms

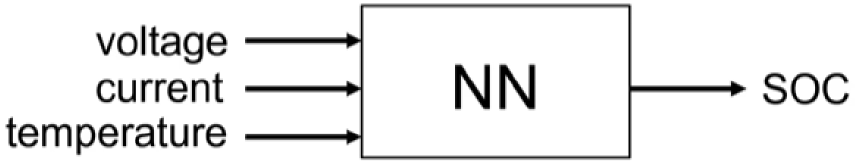
- Charge and discharge
  - CC-CV, current limits
- Estimators
  - SOC, SOH
- Protection
  - Current, voltage, and temperature monitor
  - Fault qualification
- Thermal management
  - Coolant and heater control
- Support for C-code generation



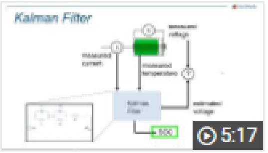



eATV * - Simulink
SIMULATION
DEBUG
MODELING
FORMAT
APPS
New
Open
Save
Print
FILE
Library Browser
LIBRARY
Log Signals
Add Viewer
Signal Table
PREPARE
Stop Time
1000
Normal
Fast Restart
Step Back
Run
Step Forward
Stop
SIMULATE
Data Inspector
Logic Analyzer
Bird's-Eye Scope
Simulation Manager
REVIEW RESULTS
Library Browser
pmsm
Library
Search Results
Motor Control Blockset
Motor Control Blockset HDL Support
Navigation Toolbox
Phased Array System Toolbox
Powertrain Blockset
Radar Toolbox
Reinforcement Learning
Report Generator
Requirements Toolbox
RF Blockset
Robotics System Toolbox
Robust Control Toolbox
ROS Toolbox
Sensor Fusion and Tracking Toolbox
SerDes Toolbox
SimEvents
Simscape
Foundation Library
Utilities
Battery
BMS
Cell Balancing
Current Management
Estimators
SOC Estimator (Adaptive Kalman Filter)
SOC Esti (Coulomb C
batt_sl_lib/Estimators/SOC Estimator (Coulomb Counting):
This block implements battery state-of-charge estimation using the Coulomb counting method.
For discrete-time simulation, set Sample time to a positive value or -1 to inherit the sample time. For continuous-time simulation, set the Sample time to zero.
Protection
Thermal Management
Cells
Cyclers
HIL
Thermal
Driveline
Electrical
Fluids
Multibody
Simscape Multibody Contact Forces Library
Simscape Multibody Multiphysics Library
Simscape Multibody Parts Library
Simulink 3D Animation
Simulink Coder
Simulink Control Design
Simulink Design Optimization
Simulink Design Verifier
Simulink Desktop Real-Time
Simulink Extras
Simulink Real-Time
Simulink Test
eATV
refVel
RefSpd
.xls, .xlsx or .txt file (200 seconds, Cyclic)
Ref Speed
Obtained speed
Brake
Motor and Drive Control
Transmission
SOC
SOC:1
current
Battery
f(x) = 0
SOC_1
Ready
150%
auto(ode45)

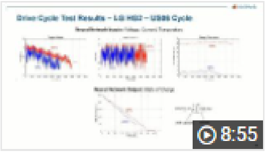
Additionally use neural networks to estimate SOC



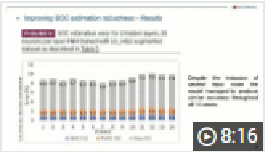
How to Estimate Battery State of Charge Using Deep Learning



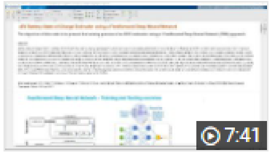
Part 1: An Introduction to Battery State of Charge Estimation
Get an introduction of battery state of charge (SOC) estimation, including a review of using neural networks.



Part 2: The Experiment Using Neural Networks
Discover the experimental process involved in training and testing the neural network.



Part 3: Neural Networks for SOC Estimation
Explore the theory and implementation of the deep neural network used in this study; motivation and tradeoffs for the utilization of certain network architectures; and training, testing, validation, and analysis of the network performance.

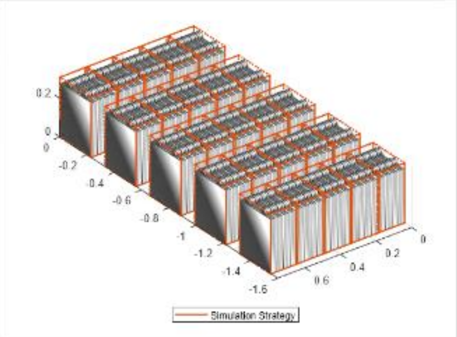


Part 4: Training and Prediction in MATLAB and Simulink Implementation
See the neural network training process and the Simulink implementation of the method.

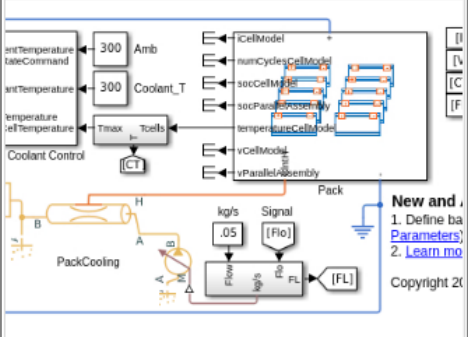
Simscape Battery

Examples

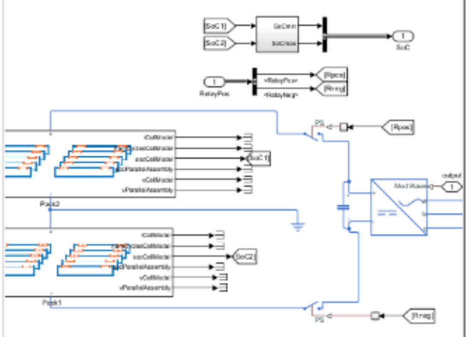
- Build Model of Battery Pack with Cell Aging (link)
- Thermal Analysis for New and Aged Battery Packs (link)
- Peak Shaving with Battery Energy Storage System (link)
- Build Model of Battery Pack for Grid Application (link)
- Protect Battery During Charge and Discharge for EV (link)
- Build Model of Battery Pack with Cell Balancing Circuit (link)



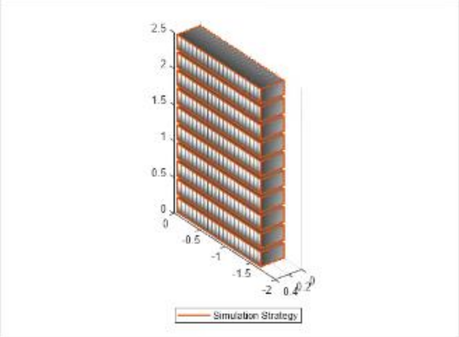
Build Model of Battery Pack with Cell Aging



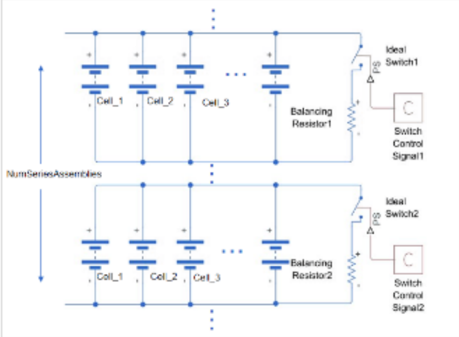
Thermal Analysis for New and Aged Battery Packs



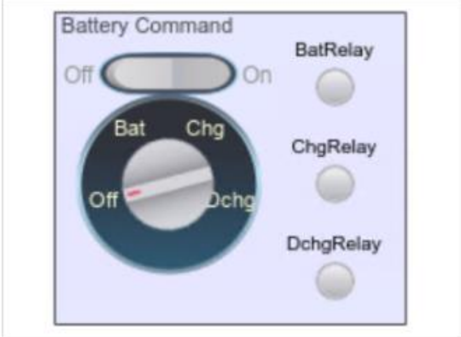
Peak Shaving with Battery Energy Storage System



Build Model of Battery Pack for Grid Application

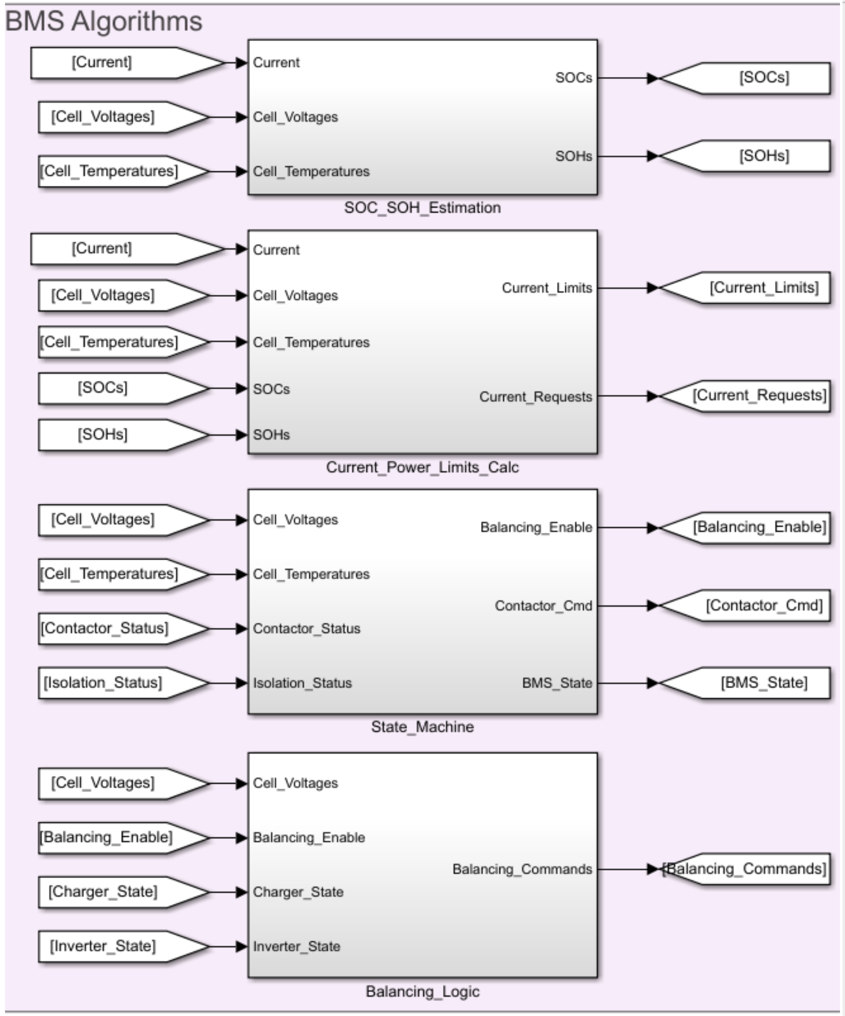


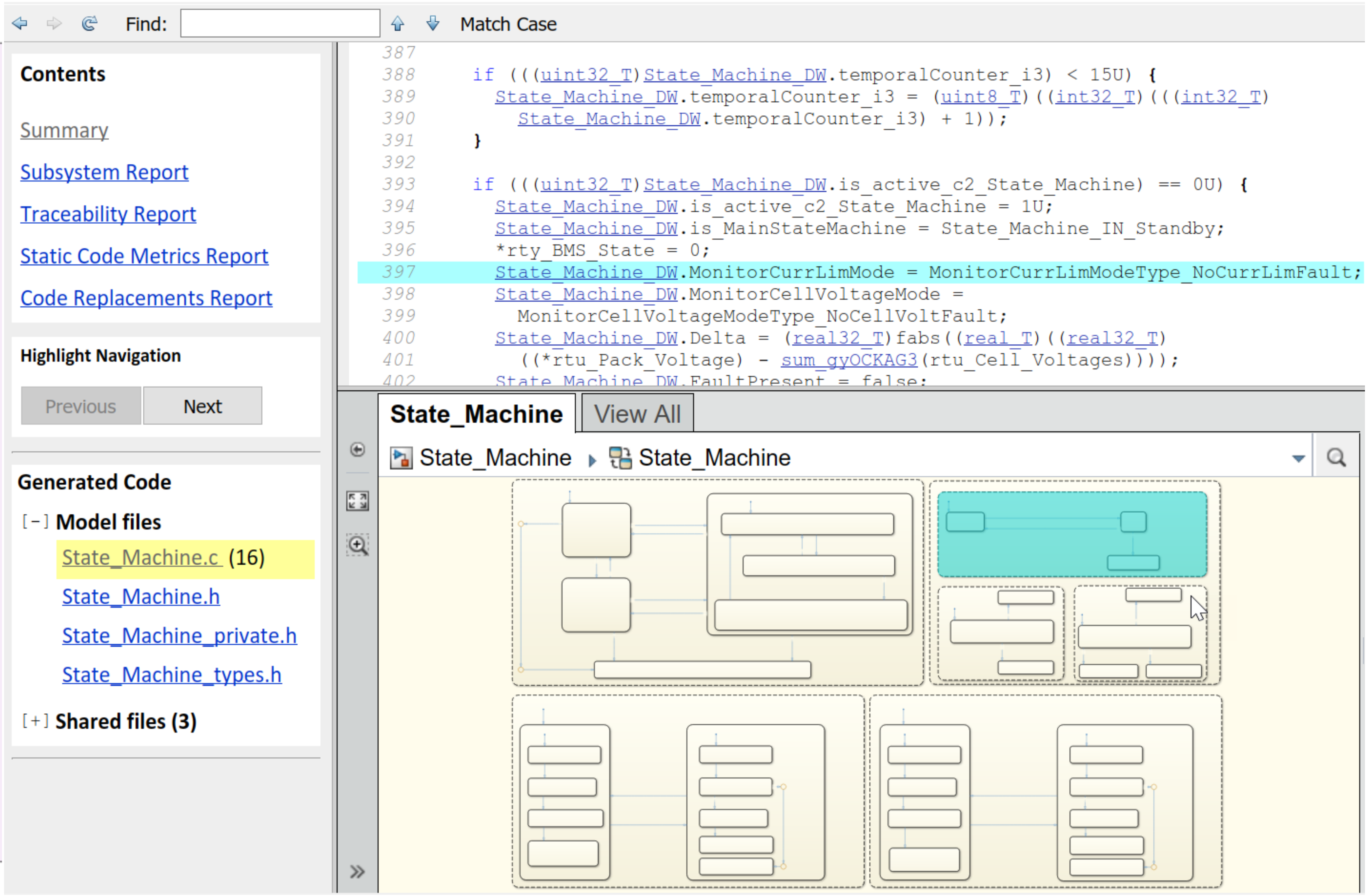
Build Model of Battery Pack with Cell Balancing Circuit



Protect Battery During Charge and Discharge for Electric Vehicle

Generate C/C++ Code From BMS Algorithm Models





Agenda

- Determine battery pack size to meet system-level targets
- Design and analyze thermal management systems
- Develop control systems
- **Realize digital twin and predictive maintenance applications**

How are Digital Twins used?

By logging data from the deployed assets

- Does the system perform as advertised?
  - **Operation**: must operate for 3-4 hours in the morning and 3-4 hours in the afternoon
  - **Charging**: battery must fully charge in 30 min (at lunch time)

- What is the effect of ambient temperature on the system?
  - Ambient temp ranges from -10 to 35°C over the year. How does this affect system performance?

- What is the actual duty cycle based on operational data?
  - Power used during operation vs. charging
  - Total number of charge / discharge cycles
  - etc.

Digital Twin?
A definition

*“A digital twin is an **up-to-date representation, a model, of an actual physical asset in operation**. It **reflects the current asset condition** and includes relevant historical data about the asset.*

*Digital twins can be **used to evaluate the current condition of the asset**, and more importantly, **predict future behavior, refine the control, or optimize operation**.”*

https://www.mathworks.com/discovery/digital-twin.html

Why Digital Twin?

Business value & motivating factors

- **Do things better**: Optimize your customer's experience
  - Anomaly detection
  - Predictive maintenance
  - Asset performance management
  - Operations optimization
  - Fleet management
  - Feedback to design

- **Do new things**: Evolve business models and opportunities

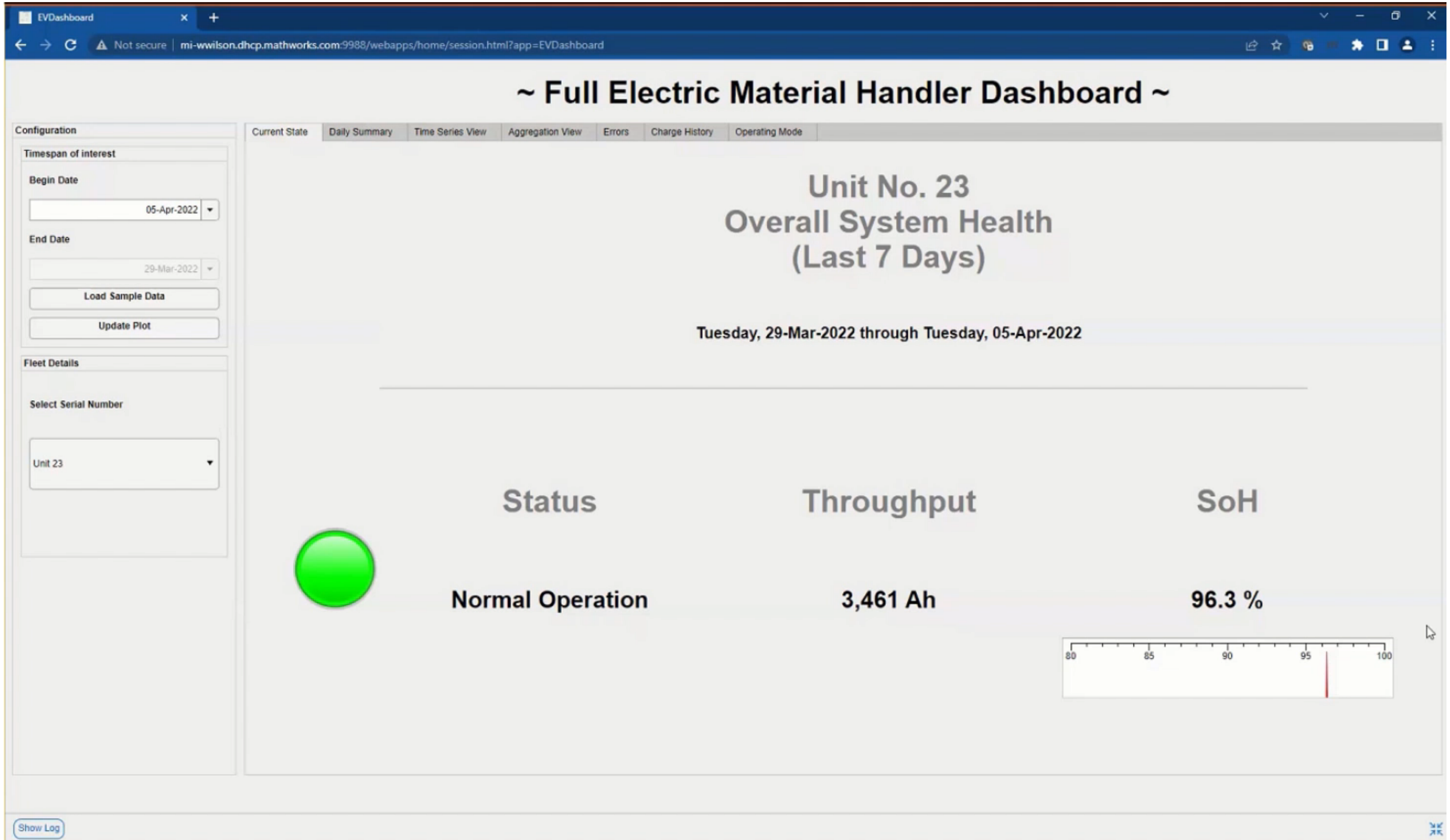
Current State

Sell a system

Future State

Selling a system's operation (capability as a service, etc.)

Sample use case:



Workflow for Digital Twin

Create a model with required physics or fidelity

Keep the model up-to-dated as per the real system (asset)

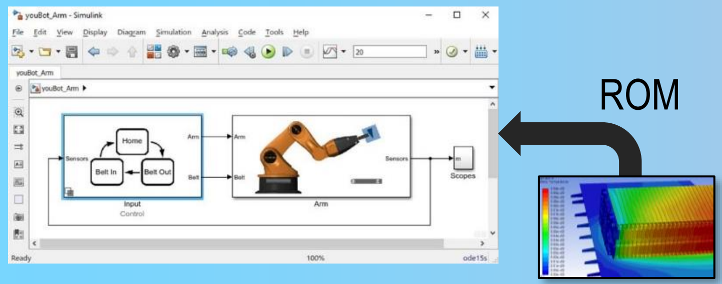
Use simulation to create/tune algorithms for predictive maintenance

Deploy on cloud or edge devices

Step 1: Create a model with required physics or fidelity

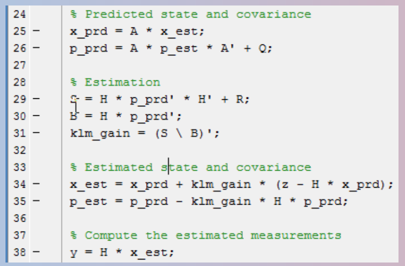
Choosing a model strategy is a function of what you **have** and what you **know**

Physics-Based



- Dynamic models of systems/components
- Electrical, mechanical, algorithms, etc.
- Can integrate models from other tools, e.g., FEM

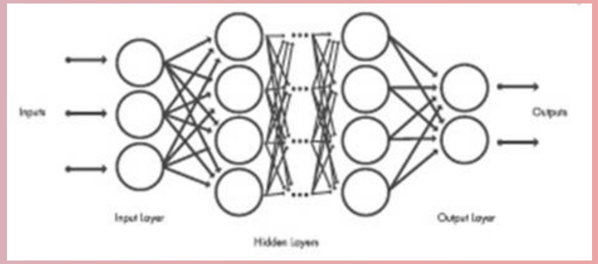
Data-Driven



```
% Predicted state and covariance
x_prd = A * x_est;
p_prd = A * p_est * A' + Q;

% Estimation
S = H * p_prd' * H' + R;
B = H * p_prd';
klm_gain = (S \ B)';

% Estimated state and covariance
x_est = x_prd + klm_gain * (z - H * x_prd);
p_est = p_prd - klm_gain * H * p_prd;

% Compute the estimated measurements
y = H * x_est;
```

- Kalman estimator
- System identification
- Regression

AI-Based



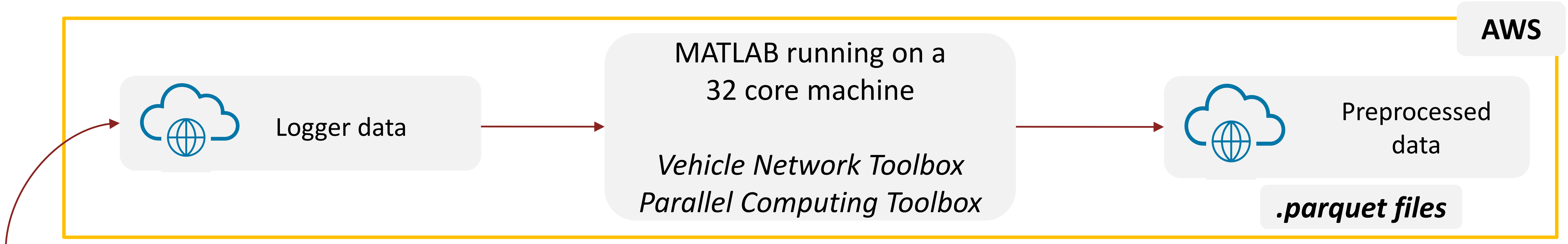
- Machine Learning
- Deep Learning
- Reinforcement Learning.

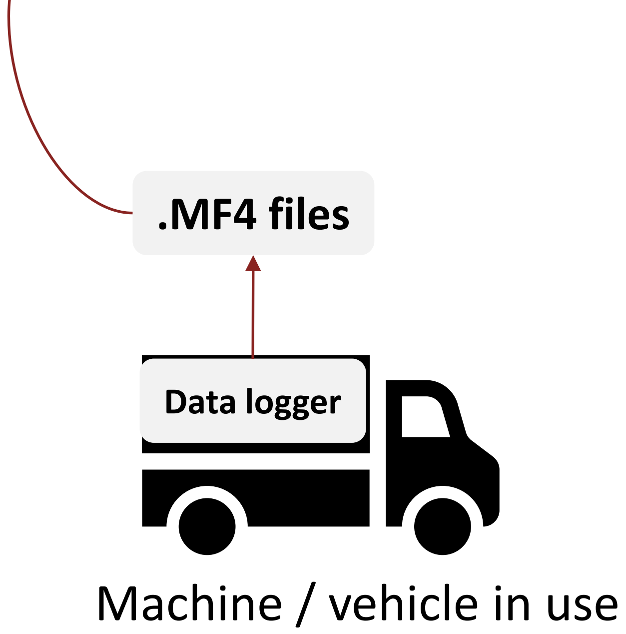
- Factors in selecting model strategy
  - What does your application need?
  - Do you have knowledge of system’s physics (or only historical data)?
  - Who has the expertise needed to build the model?

Step 2: Keep the model up-to-dated as per the real system (asset)

Raw Log Files

Cloud based data preprocessing pipeline

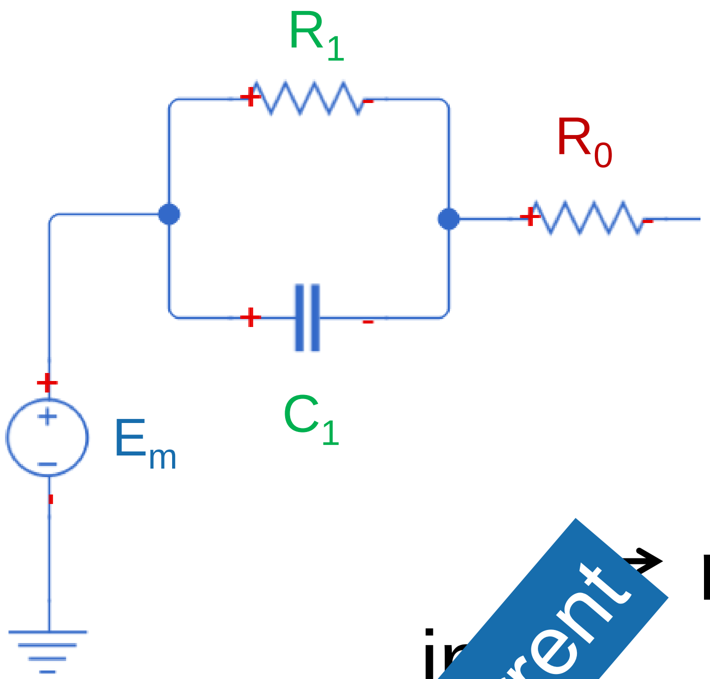


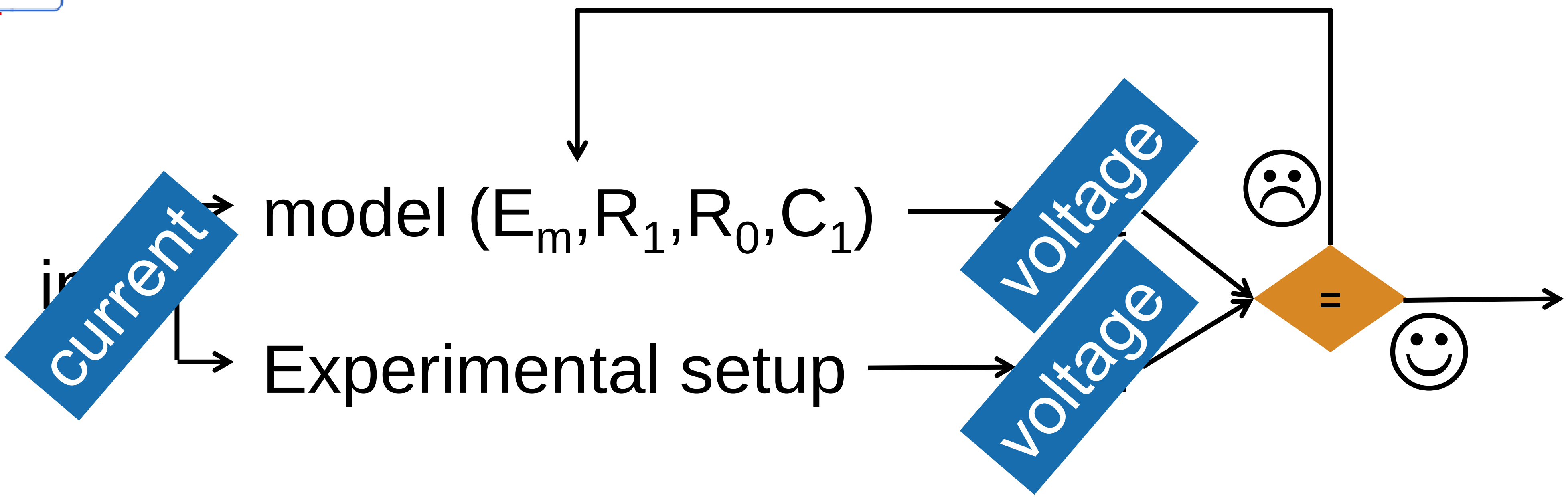


Implementation Details

- Source and destination s3 buckets are different
  Credential management
- Cloud based compute and parallel computing sped up the work
  Leverage compute when you need to
- Run MATLAB on a Windows machine
  Needed this for file specific functionality

Use Parameter Estimation to update the models

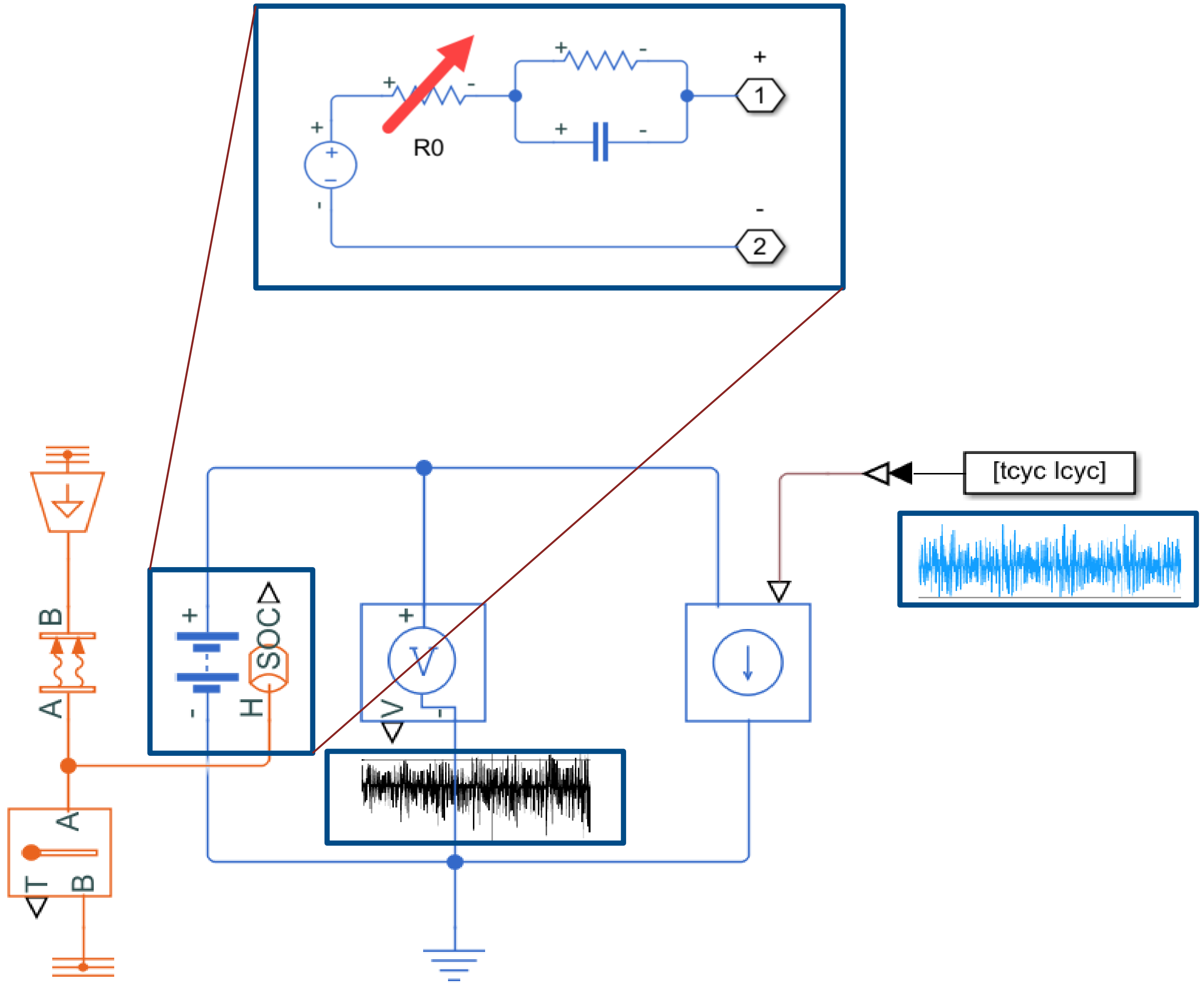




Step 3: Use simulation to create/tune algorithms for predictive maintenance

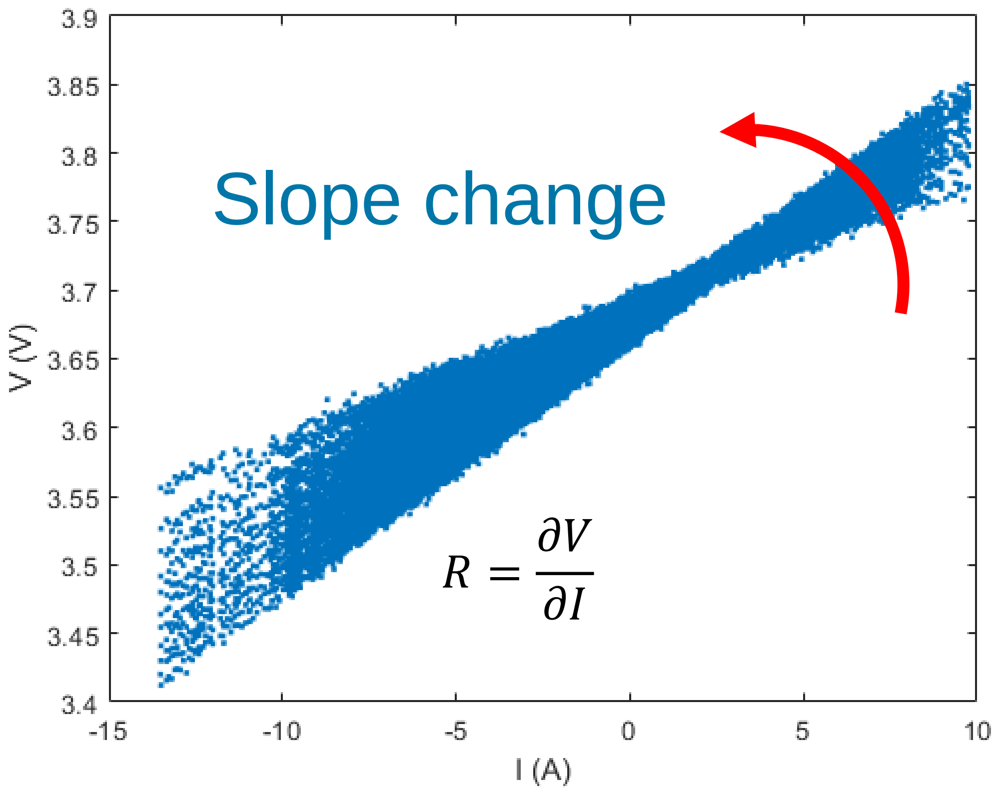
Simulate to Set Expectations

As internal resistance increases, what should we see?



Internal Resistance - $R = \frac{\partial V}{\partial I}$

Synthetic Data



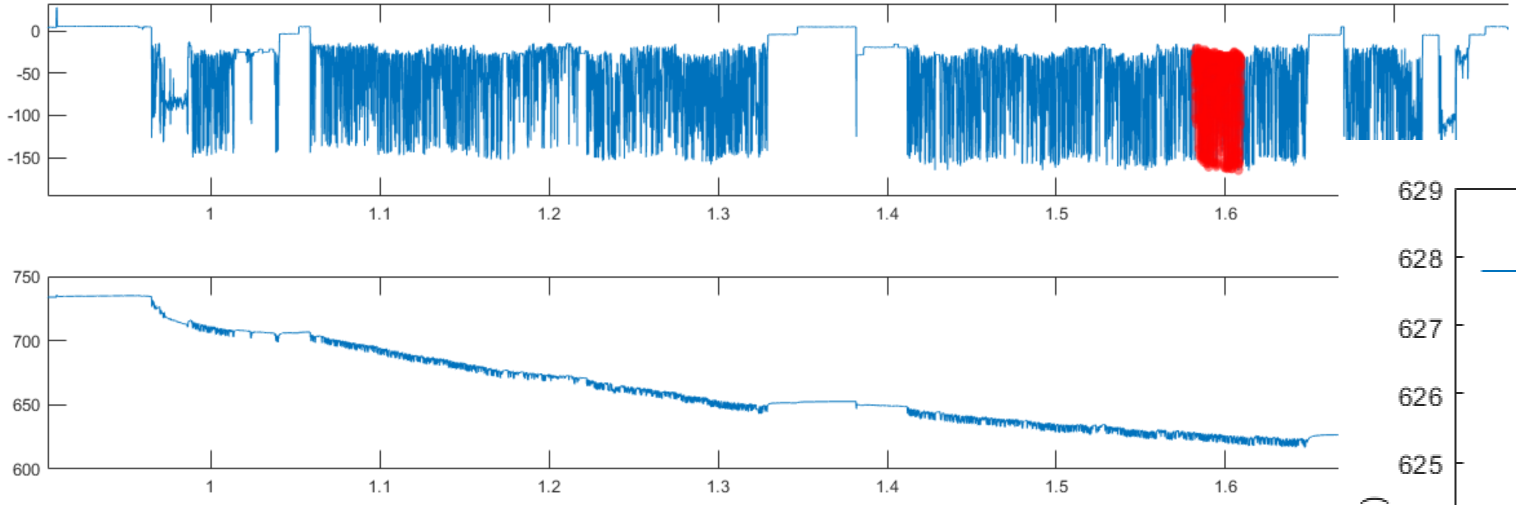
Incrementally fit data based on voltage values

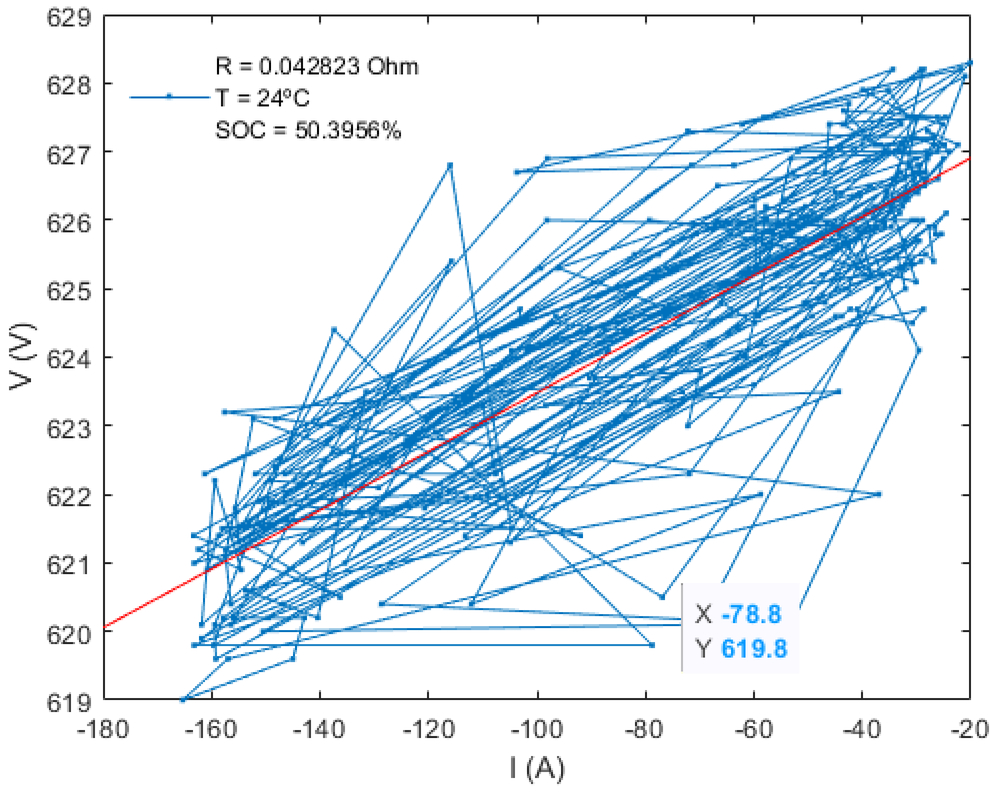
Bin data by SoC

selection

I(A)

V(V)





Internal Resistance - $R = \frac{\partial V}{\partial I}$

Discharge

Initial results on a subset of data

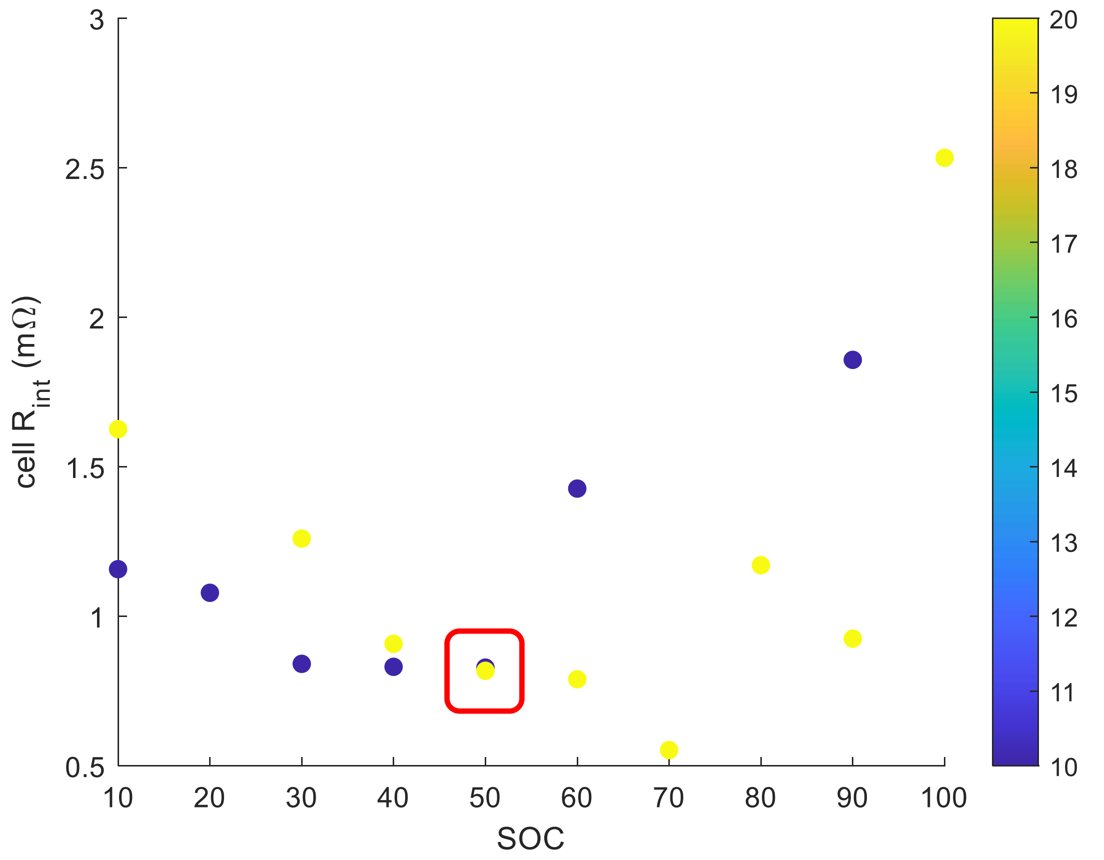
Internal resistance as a function of SoC and Temperature

Internal Resistance - $R = \frac{\partial V}{\partial I}$

Discharge

Convert from pack to cell

$\times \frac{Np}{Ns}$



Internal discharge impedance (10 sec DC pulse, 50% SOC, 25°C)	0.71 mΩ

Next Steps for Modeling Work

Strategy and planned next steps

- Understand system behavior over time
  - How does internal resistance change over time?
  - Can we detect degradation in power output over time?
- Battery cell performance parameters
  - Internal resistance so far (power), capacity next (energy)
  - Combine internal resistance and capacity learnings into a SoH story
- Feature Engineering + AI modeling & Automation
  - Cloud based parallel computing (“Thinking out loud on the cluster”)

Summary

- Determine battery pack size to meet system-level targets
- Design and analyze thermal management systems
- Develop control systems
- Realize digital twin and predictive maintenance applications

Join us for webinar on

Evaluating EV Charging Infrastructure with Simscape Electrical

November 18, 2022 | 6.00pm IST

MathWorks®