Fulfill range, acceleration and cost targets using battery sizing

October 20, 2022 | Stuttgart

Gernot Schraberger, Lorenzo Nicoletti

MathWorks®

Agenda

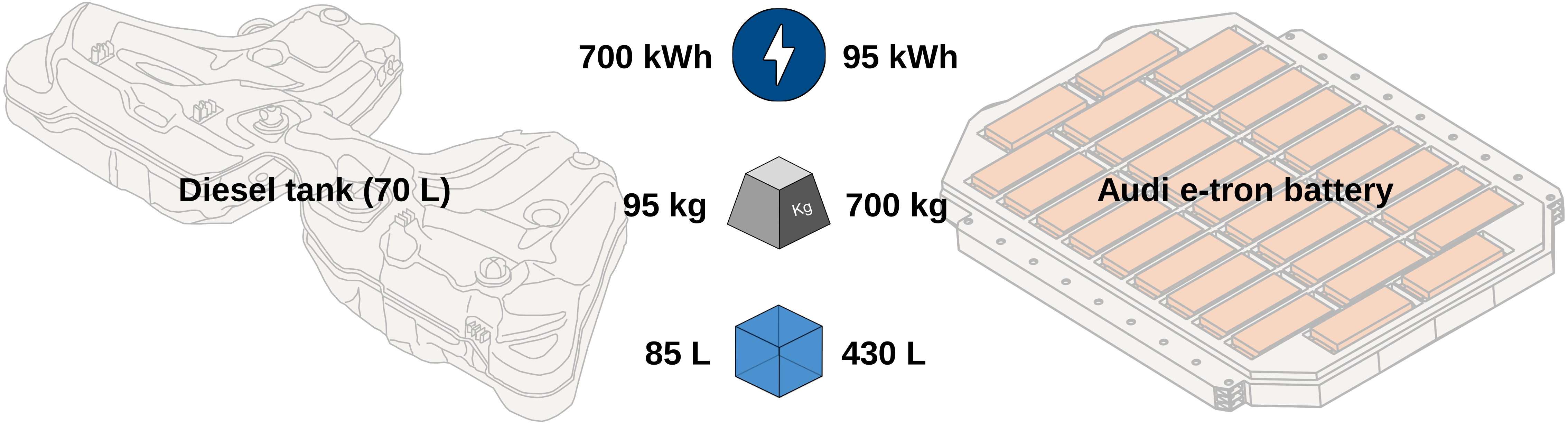
- Problem description
- Modeling of a Battery Electric Vehicle with the Virtual Vehicle Composer
- Parameter sweep
- Numerical optimization of the battery size and additional parameters
- Summary and outlook

Agenda

- **Problem description**
- Modeling of a Battery Electric Vehicle with the Virtual Vehicle Composer
- Parameter sweep
- Numerical optimization of the battery size and additional parameters
- Summary and outlook

The electrification of the powertrain

- Global effort to reduce human-induced CO_2 emissions
- The automotive sector focuses on reducing the fleet emissions
- Battery Electric Vehicles (BEVs) are a promising long-term solution since they do not cause any local CO_2 emissions

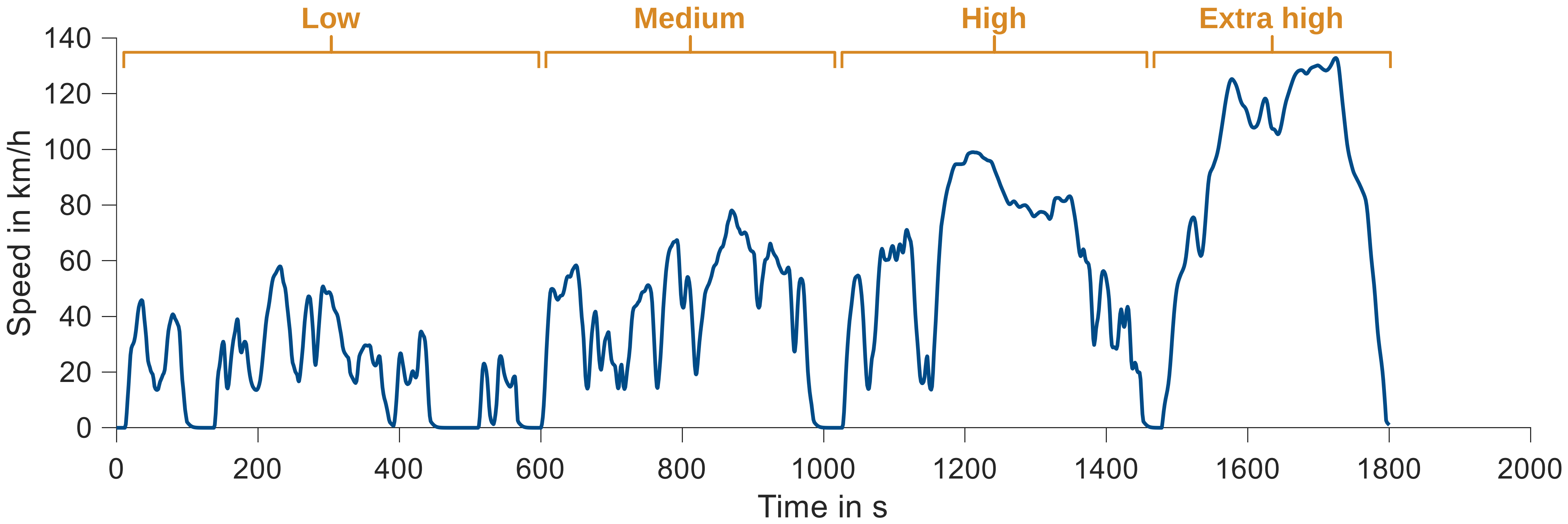


The electrification of the powertrain

- The battery impacts system-level specifications
  - Vehicle mass
  - Energy consumption
  - Vehicle range
- The battery’s impact represents a major challenge for BEVs development
- Today’s goal is to demonstrate how MathWorks tools support:
  - Configuring and building BEV model for drive cycle / performance analysis
  - Sizing components (such as the battery)
  - Performing component- and system-level optimizations

System-level targets

System-level target	How to evaluate
Acceleration time $t_{0\text{-}100}$ in s	Perform Wide Open Throttle Test (WOT)



Agenda

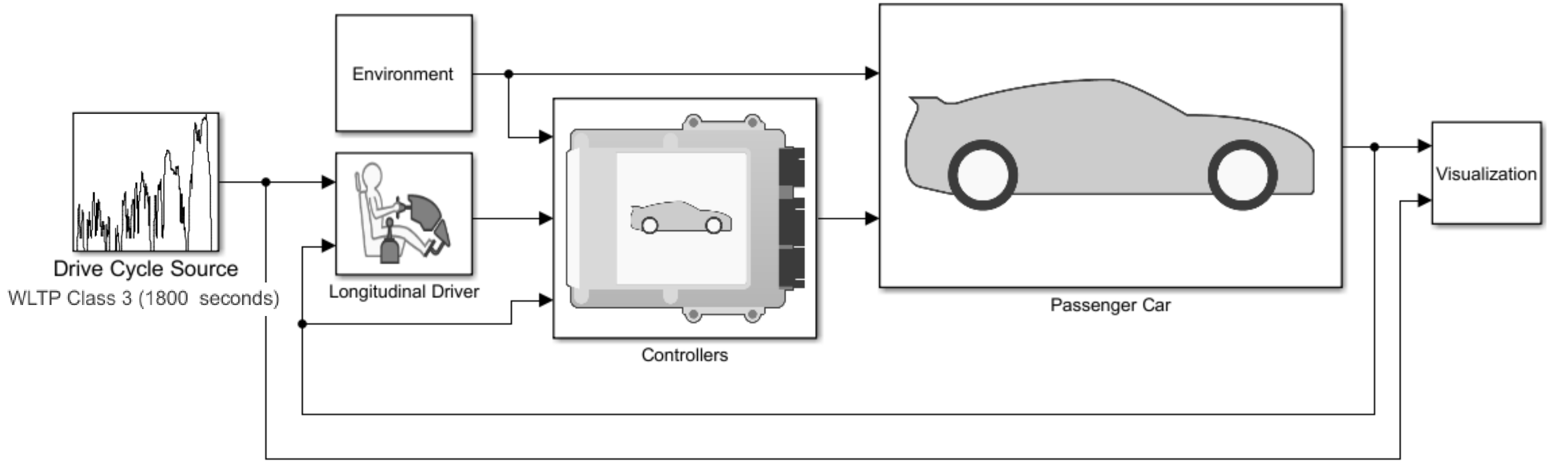
- Problem description
- **Modeling of a Battery Electric Vehicle with the Virtual Vehicle Composer**
- Parameter sweep
- Numerical optimization of the battery size and additional parameters
- Summary and outlook

Vehicle modeling for sizing task

- Requirements
  - Fast vehicle model creation
  - Easy adaptability of model including component customization
  - Fast model execution for optimization at appropriate fidelity level

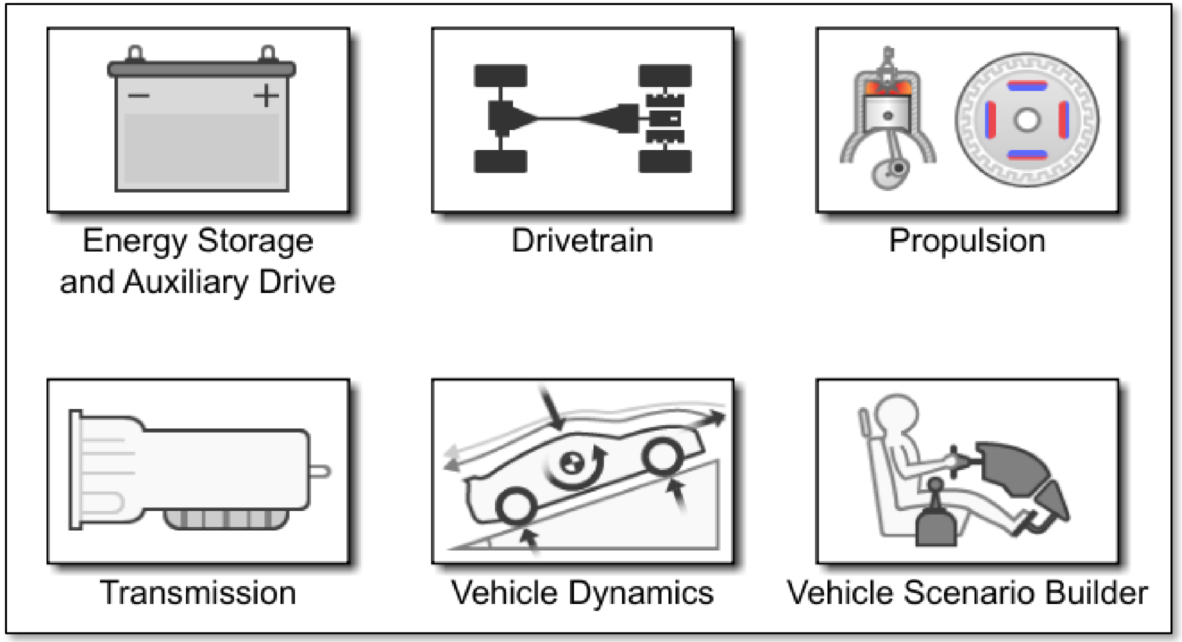
Powertrain Blockset

- Provide starting point for engineers to build **plant / controller models**
- Provide documented white-box models
- Provide **fast**-running models that also work with popular HIL systems

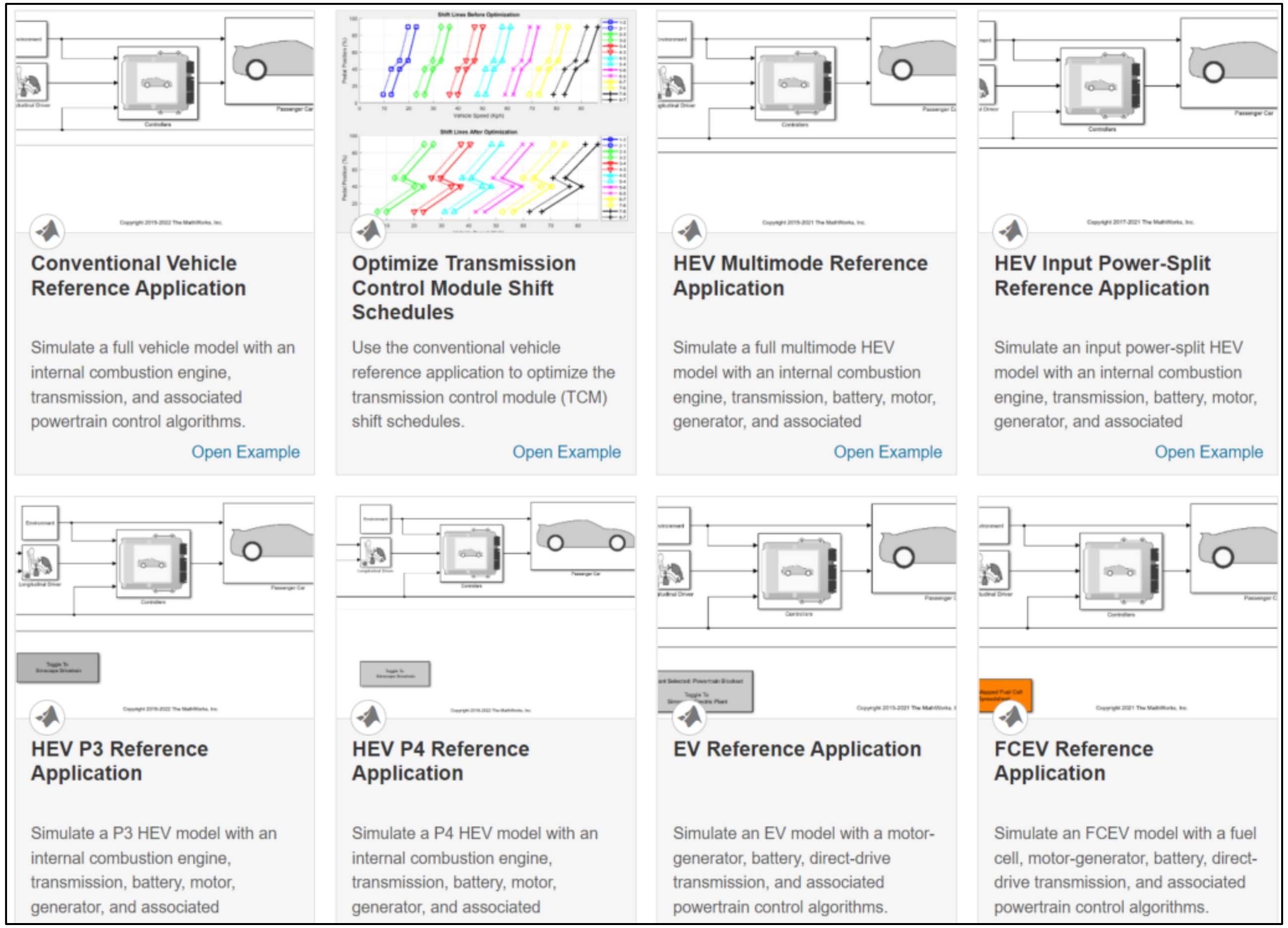


Starting with Powertrain Blockset

Library of blocks



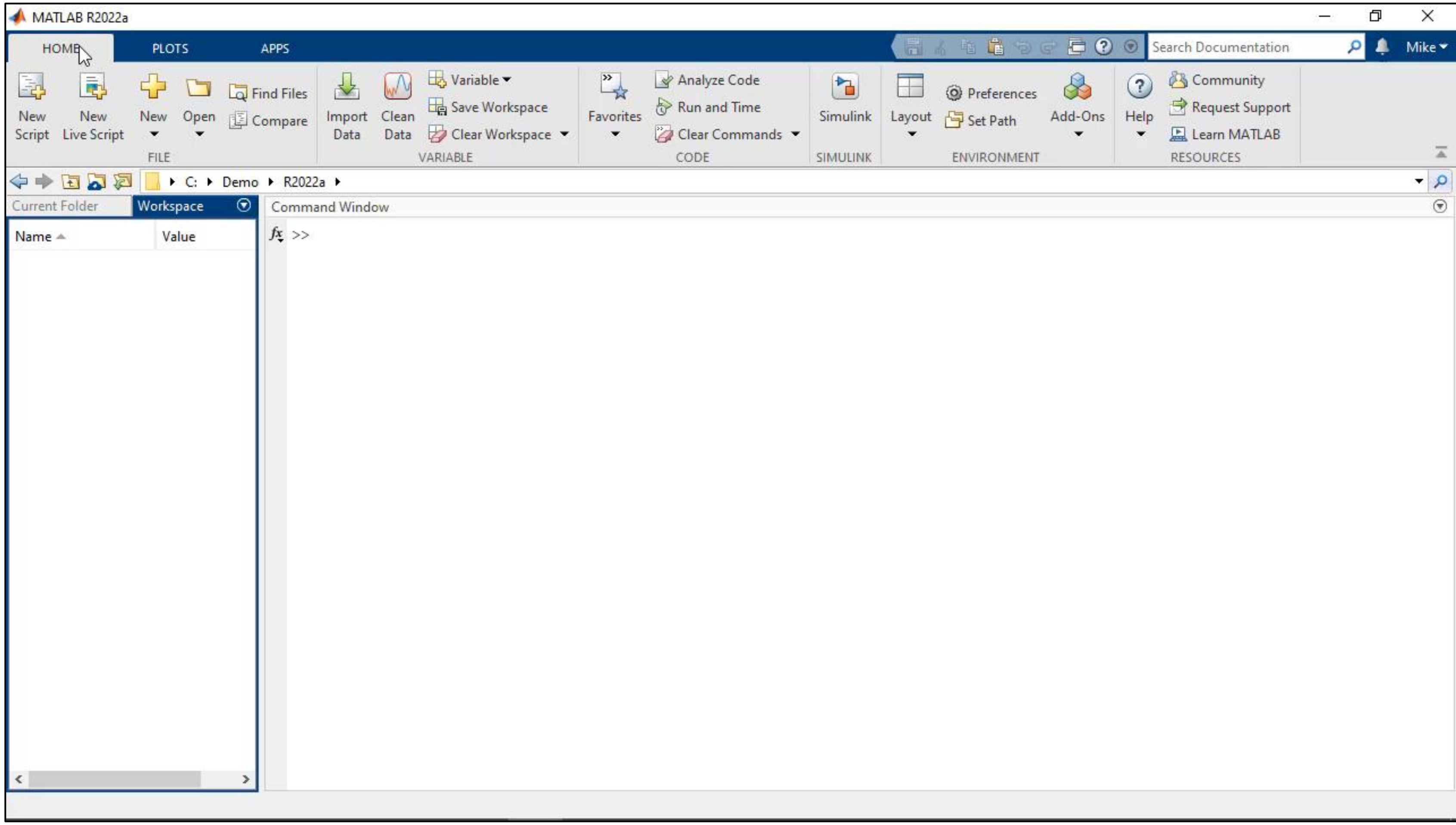
Pre-built reference applications



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
- Model can be further customized



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

$$E_m = f(SOC)$$

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

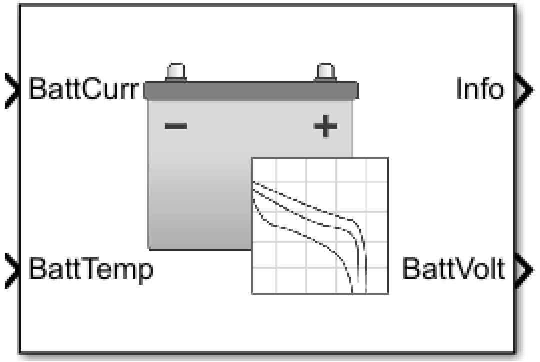
$$V_T = E_m + I_{batt} R_{int}$$

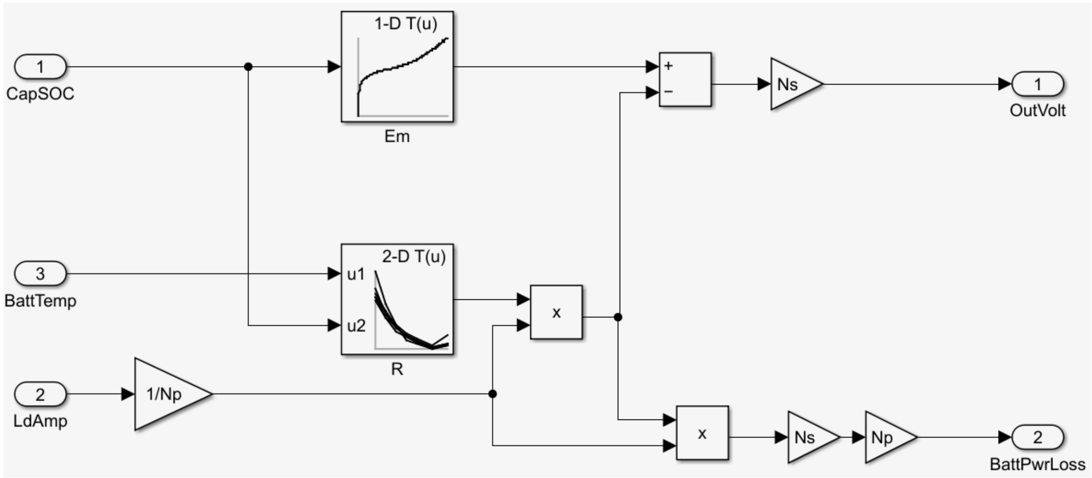
$$I_{batt} = \frac{I_{in}}{N_p}$$

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

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

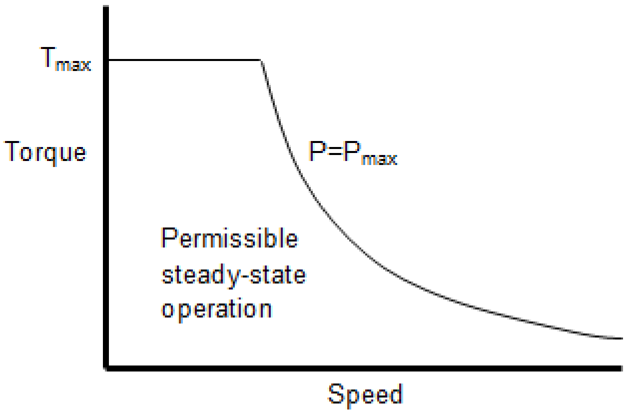
$$Ld_{AmpHr} = \int_0^t I_{batt} dt$$

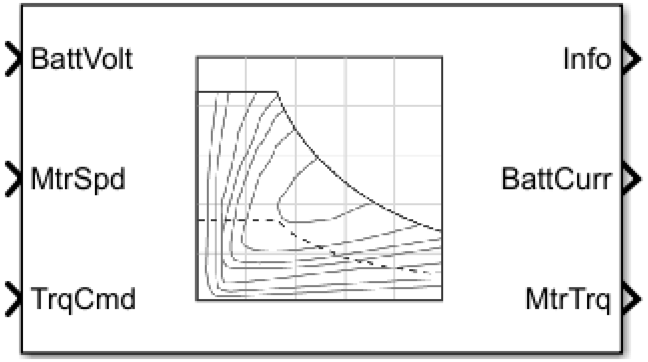


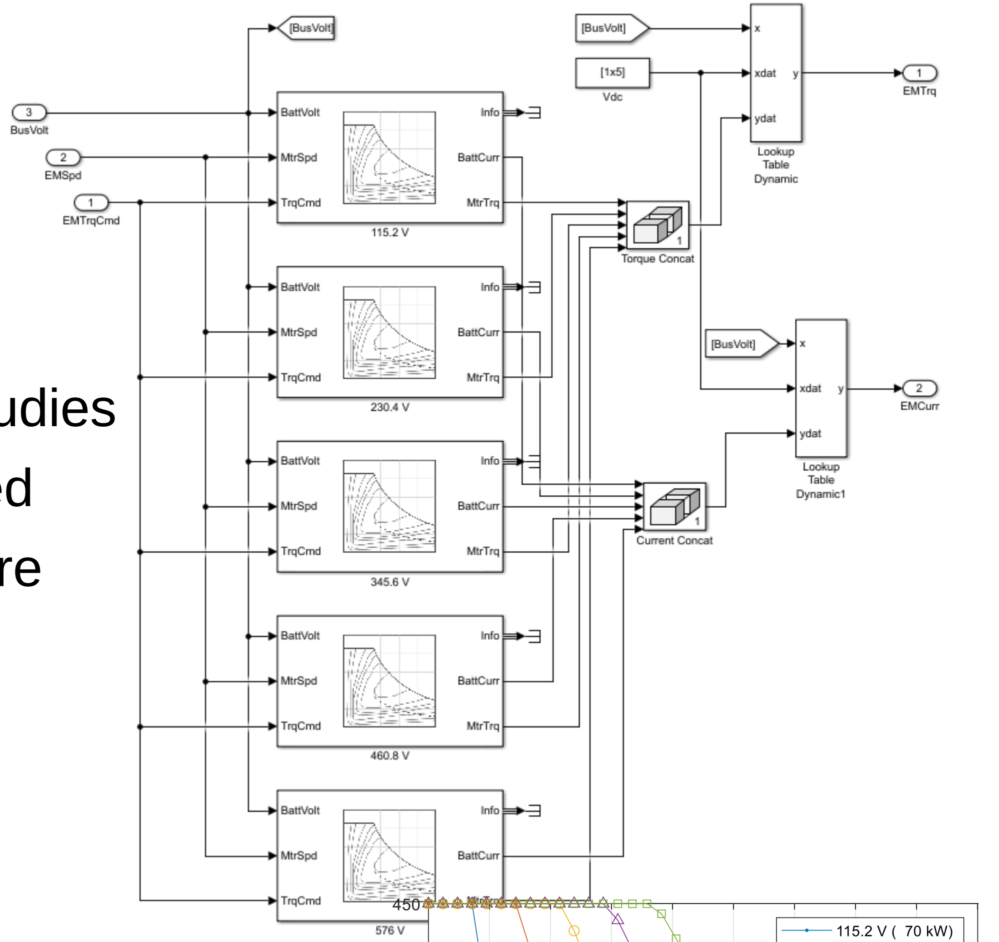


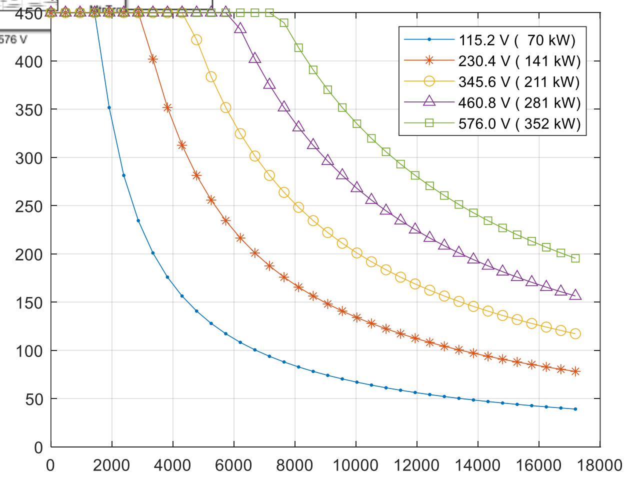
Motor model

- 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









Agenda

- Problem description
- Modeling of a Battery Electric Vehicle with the Virtual Vehicle Composer
- **Parameter sweep**
- Numerical optimization of the battery size and additional parameters
- Summary and outlook

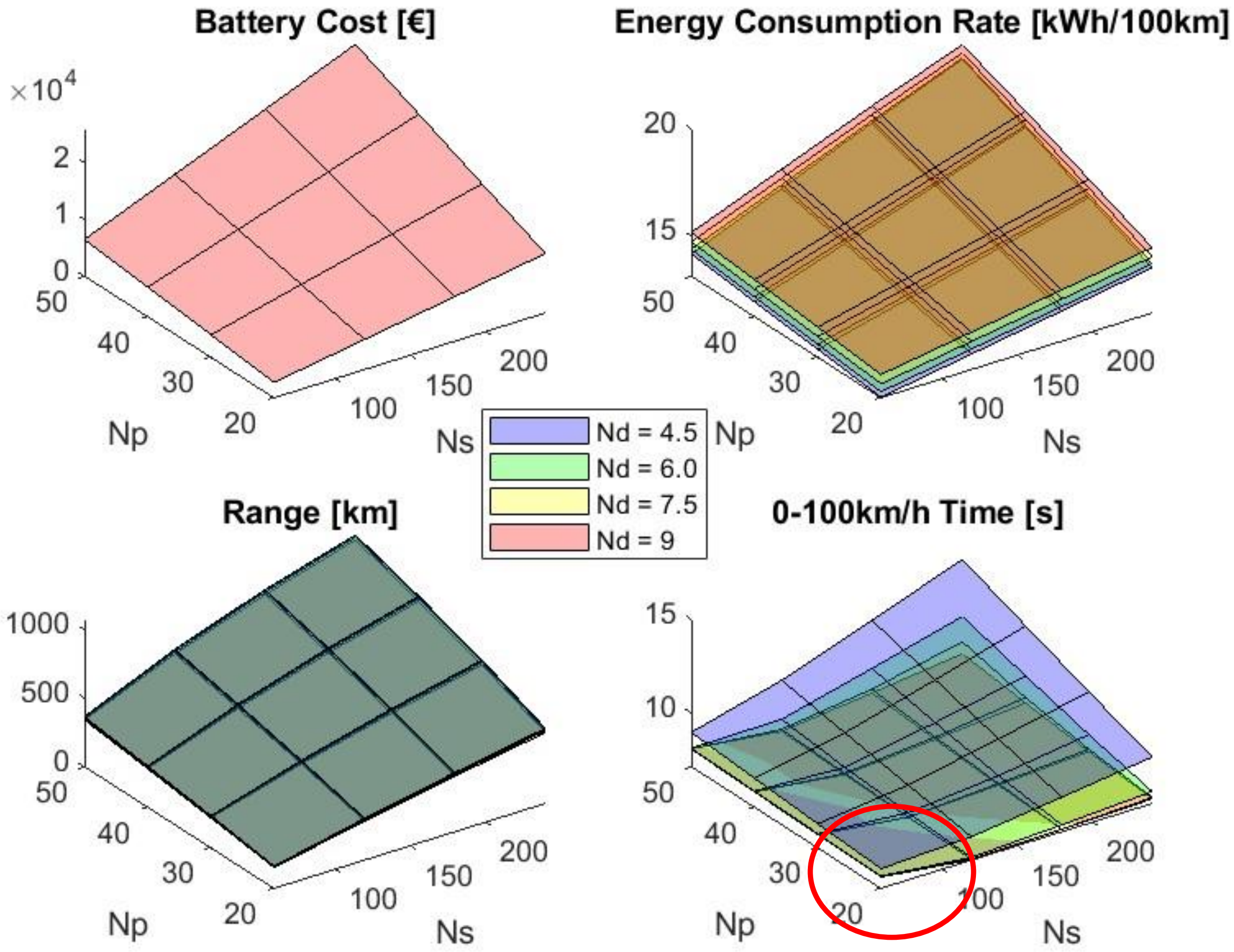
Base vehicle properties (midsize limousine)

Parameters (fixed)	Value
Vehicle mass (without battery) [kg]	1291
Wheelbase; Width; Height [m]	2.875; 1.85; 1.44
C_D (Aerodynamic drag)	0.23
Front area [m^2]	2.4
Cell voltage [V]	3.6
Cell capacity [Ah]	4.8
Battery energy density [Wh/kg]	145
Battery costs [€/kWh]	125

Metric	Baseline
Cell configuration (Ns,Np)	96s31p
Gearbox ratio (Nd)	9.0
Range [km]	340
Energy consumption [kWh/100km]	15.1
Battery cost [€]	6428
Acceleration time $t_{0\text{-}100}$ [s]	7.14

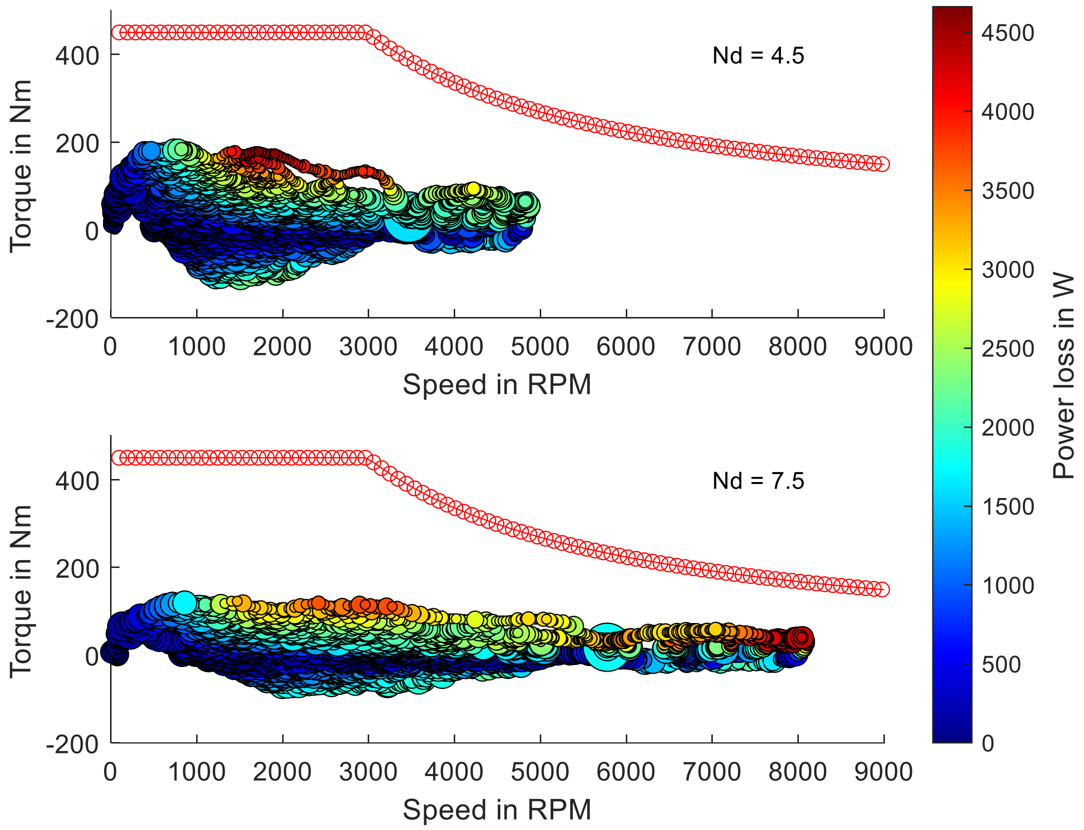
Initial assessment

- Performed initial parametric study
  - Sweep of Np, Ns and Nd
  - Study problem statement before launching long optimization study
- Lessons learned
  - High gearbox ratio (Nd) → better acceleration time but worse energy consumption
  - Higher number of cells → higher range but worse energy consumption



Initial assessment

- Performed initial parametric study
  - Sweep of Np, Ns and Nd
  - Study problem statement before launching long optimization study
- Lessons learned
  - High gearbox ratio (Nd) → better acceleration time but worse energy consumption
  - Higher number of cells → higher range but worse energy consumption
  - WLTP never pushed motor to max torque / power limits



Design trade-offs

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

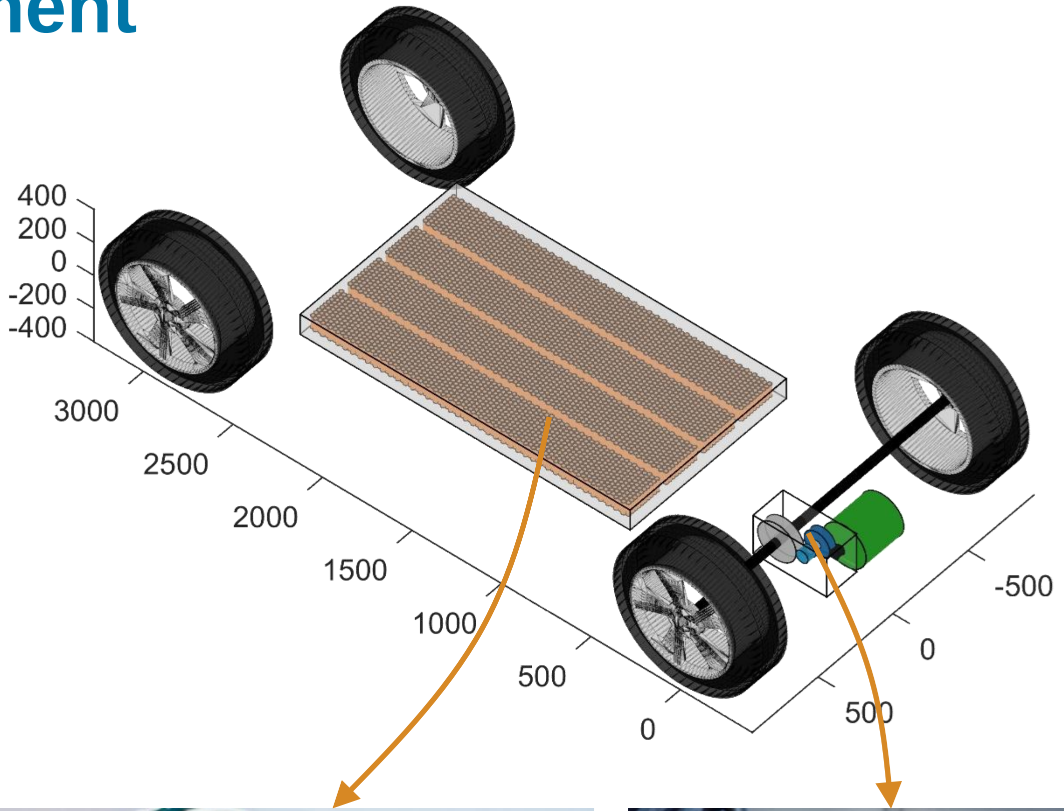
Numerical optimization provides a rigorous method to balance competing objectives

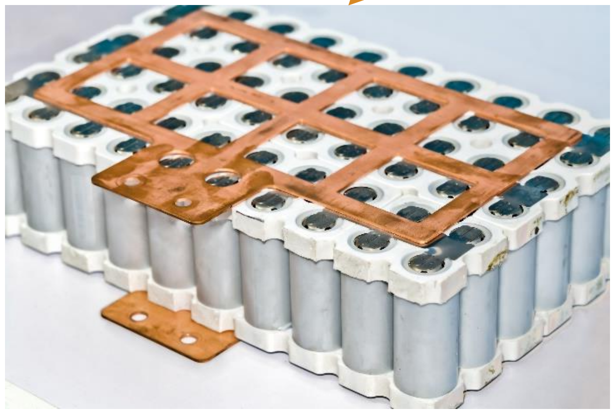
Agenda

- Problem description
- Modeling of a Battery Electric Vehicle with the Virtual Vehicle Composer
- Parameter sweep
- **Numerical optimization of the battery size and additional parameters**
- Summary and outlook

Component sizing problem statement

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





Component sizing problem statement

- Goals:

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

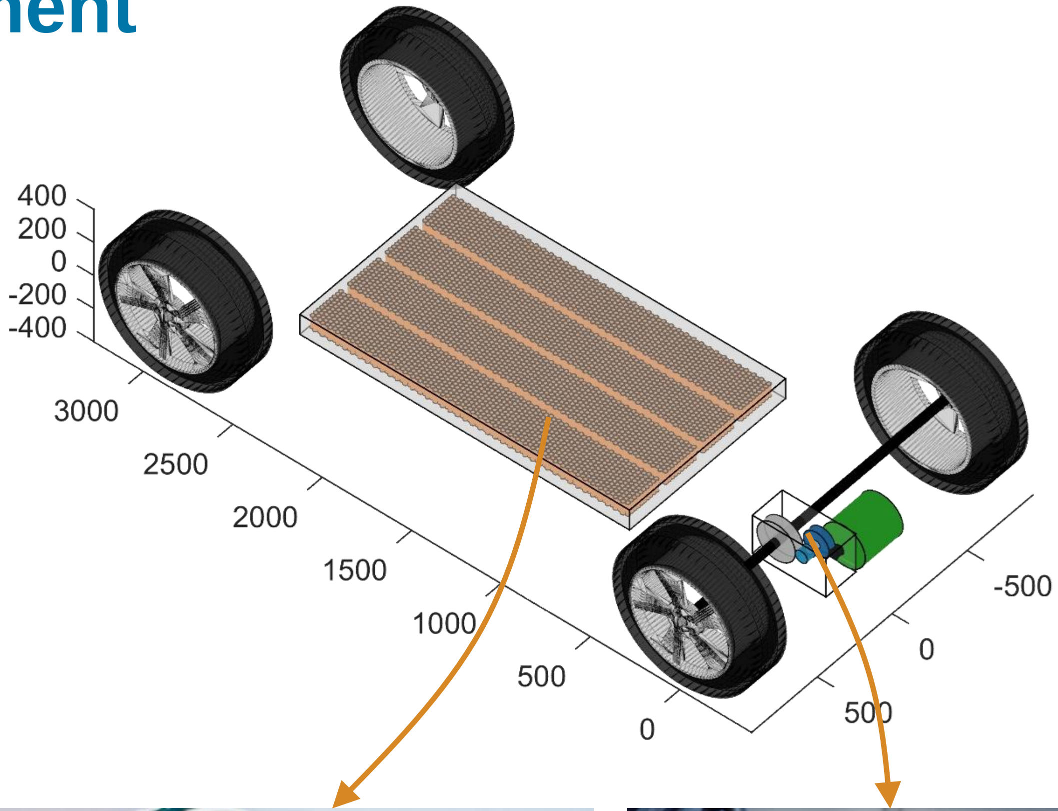
ECR = Energy Consumption Rate [Wh/km]

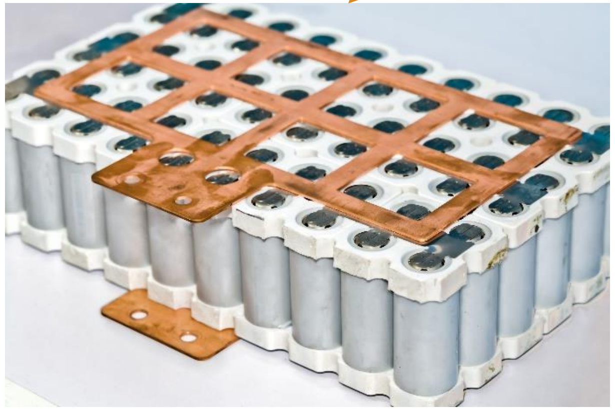
- Constraints:

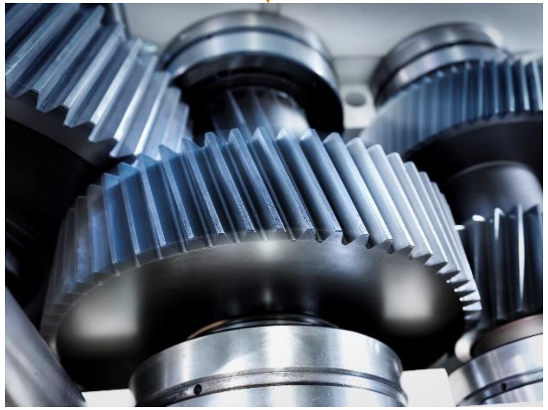
g_1: DriveCycleFault ≤ 0
g_2: Range ≥ 400 km
g_3: $t_{0\text{-}100} \leq 8$ sec

- Design Variables:

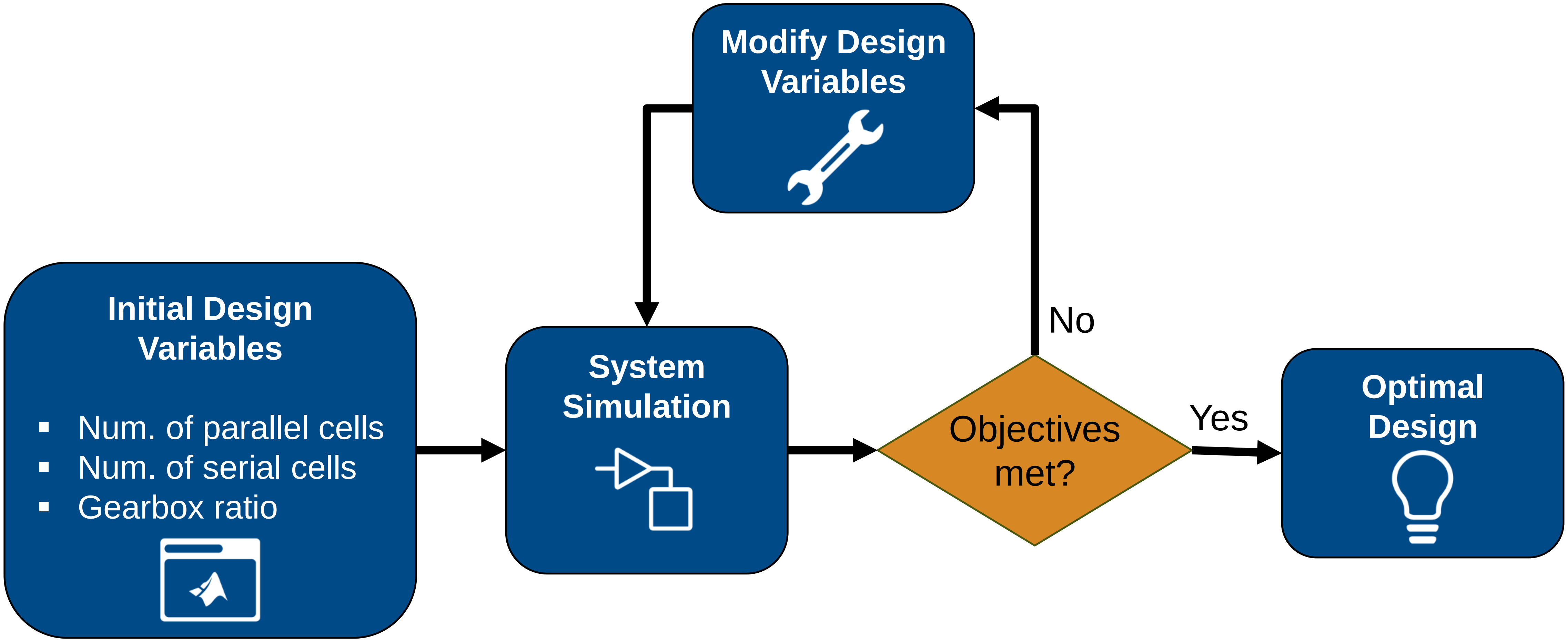
x_1: $20 \leq Np \leq 50$ *(Integer)*
x_2: $320V/3.6V \leq Ns \leq 600V/3.6V$ *(Integer)*
x_3: $2 \leq Nd \leq 10$ *(Continuous)*







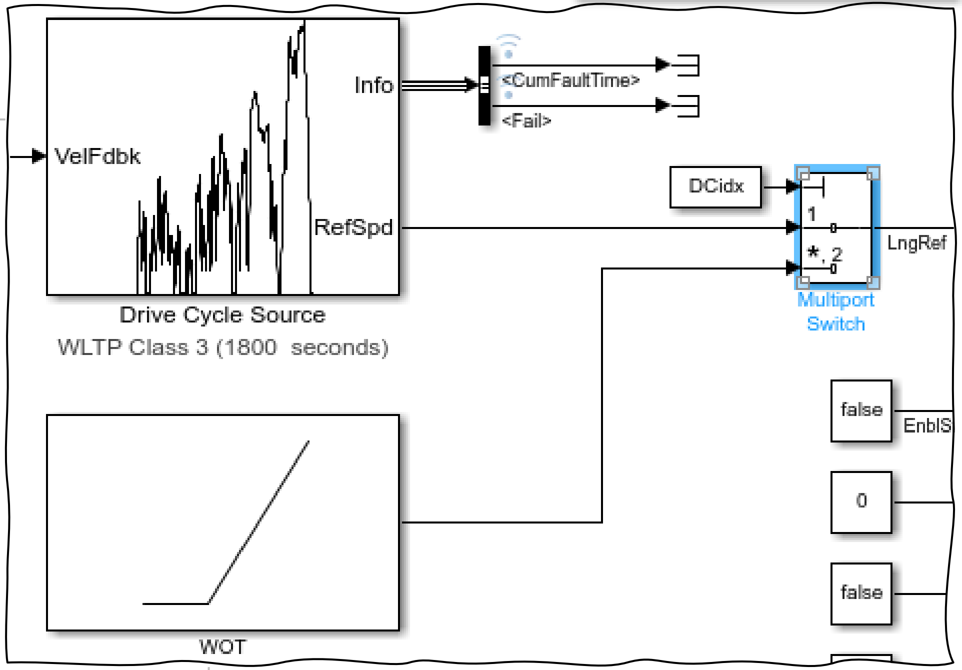
Optimization workflow



Running simulations as a function call

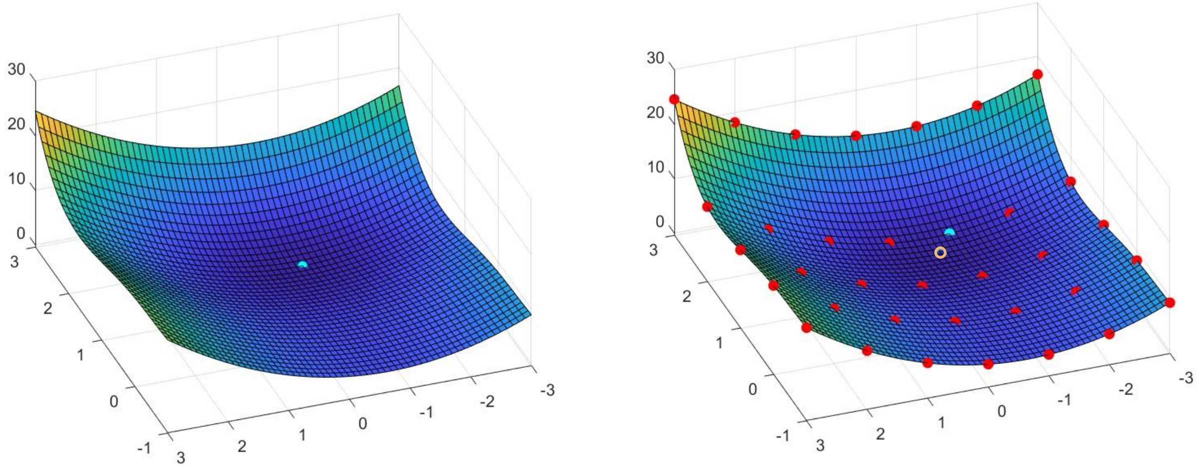
```matlab
function [f, g, ECR, cost] = RunEV(Np, Ns, Ndiff, model)
% RunEV runs a series of EV sims for a given set of parameters

% 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 overrides
in = Simulink.SimulationInput(model
in = in.setVariable('PlntVehMass',
in = in.setVariable('PlntBattNumCel
in = in.setVariable('PlntBattNumCel
in = in.setVariable('PlntDiffrntlRa

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

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

```matlab
% 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
```

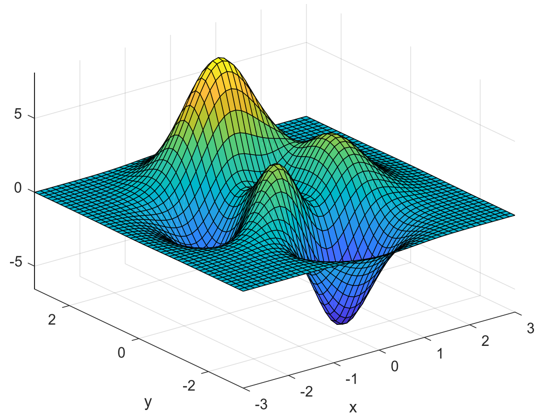


Selecting the appropriate optimization algorithm

- Design Variable Space
  - Continuous
  - Integer (discrete)
  - Mixed Integer
- Local / global optimization task
  - Optimization Toolbox (local)
    - Functions for finding parameters that **minimize or maximize objectives** while **satisfying constraints**
  - Global Optimization Toolbox
    - Functions that **search for global solutions** to problems that contain **multiple maxima or minima** on **smooth or nonsmooth** problems
- Problems
  - Linear / quadratic / cone programming
  - Least-squares and nonlinear equations
  - Multiobjective optimization
  - Nonlinear optimization
    - Surrogate, Genetic Algorithm, …



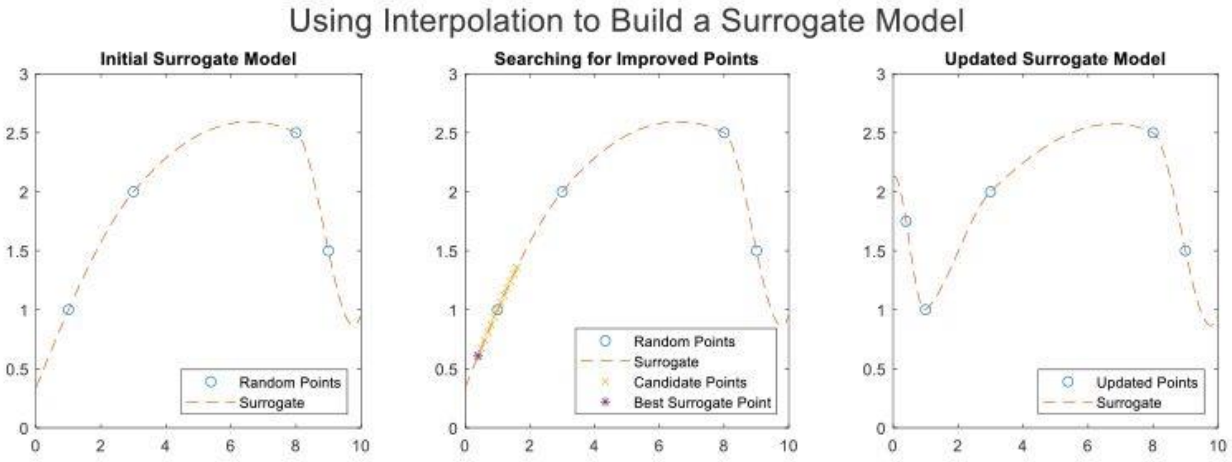
Objective with single minimum



Objective with multiple minima

Solve expensive nonlinear problems with `surrogateopt`

- Concept
  - Create a surrogate model of the objective / constraints
  - Find the best point on the surrogate model, then sample new points
    - near the best point found so far (refine solution)
    - far from any sample (improve model accuracy)
- Benefits
  - Automatically builds a cheap-to-evaluate surrogate model
  - Searches for global solution
  - Works in continuous or **integer design variables**
  - Accepts nonlinear, linear, and integer constraints



Description

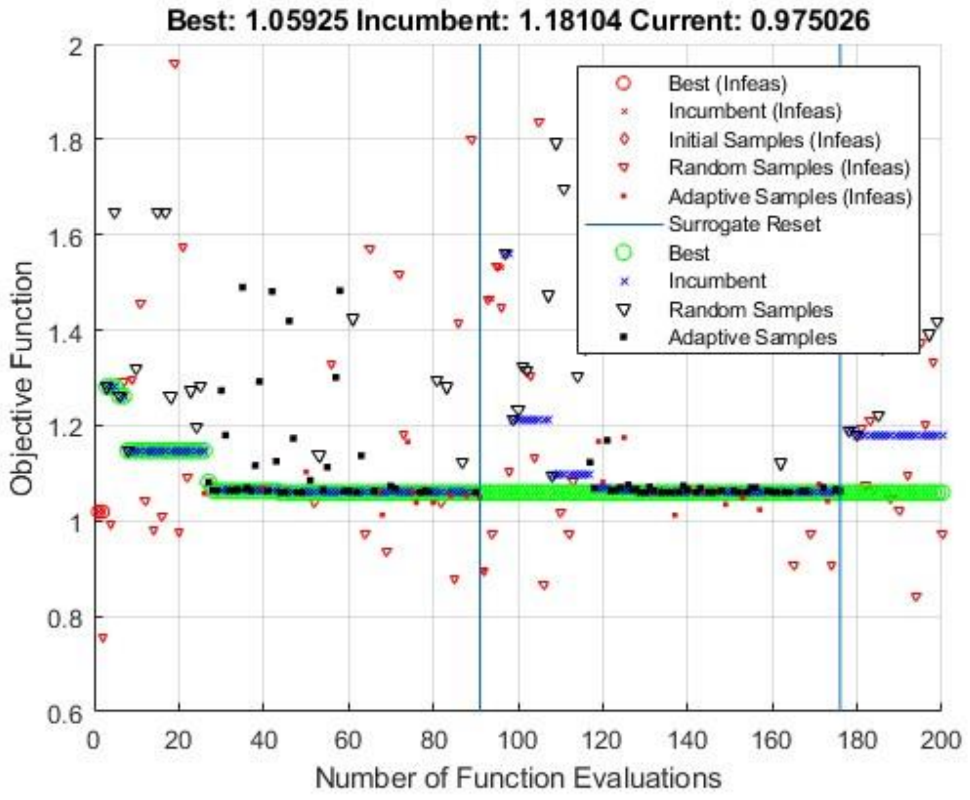
`surrogateopt` is a global solver for time-consuming objective functions.

`surrogateopt` attempts to solve problems of the form

$$\min_x f(x) \text{ such that } \begin{cases} \mathrm{lb} \le x \le \mathrm{ub} \\ A \cdot x \le b \\ \mathrm{Aeq} \cdot x = \mathrm{beq} \\ c(x) \le 0 \\ x_i \text{ integer, } i \in \mathrm{intcon}. \end{cases}$$

Optimization Results

Metric	Baseline	Target	Optimized (% improvement)
Energy consumption [kWh/100km]	15.1	< 15	14.4 (-4.6%)
Cost [€]	6428	< 7500	7232 *(+12.5%)*
Range [km]	340	> 400	401 (+17.6%)
Acceleration time $t_{0\text{-}100}$ [s]	7.14	< 8	8.0 (+12.0%)
Gearbox ratio Nd	9		5.05
Cell configuration	96s31p		108s31p
Bus voltage [V]	345.6		388.8
Capacity [kWh]	51.4		57.9



Performed 200 function calls (~2,5 hours)

Agenda

- Problem description
- Modeling of a Battery Electric Vehicle with the Virtual Vehicle Composer
- Parameter sweep
- Numerical optimization of the battery size and additional parameters
- **Summary and outlook**

Summary

- Key take-away:
  - **Simulink Virtual Vehicle Composer** with **Global Optimization Toolbox** are rigorous means to identify optimal design parameter configurations

- Topics discussed:
  - BEV model with **Virtual Vehicle Composer**
  - Apply BEV model to estimate:
    - Acceleration time in s
    - Energy consumption in Wh/km
    - Range in km
    - Battery costs in €
  - Parameter sweep to understand interdependencies and trade-offs
  - Optimize using *Surrogate Optimization* from the **Global Optimization Toolbox**

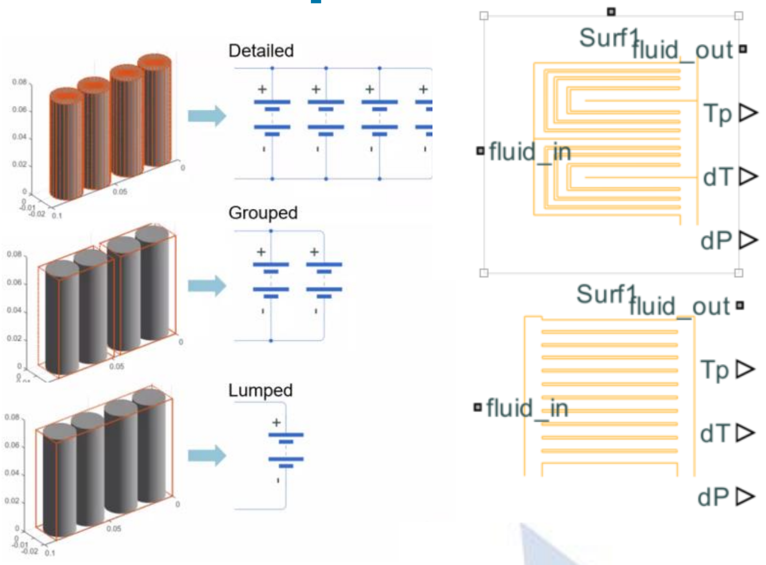
Outlook

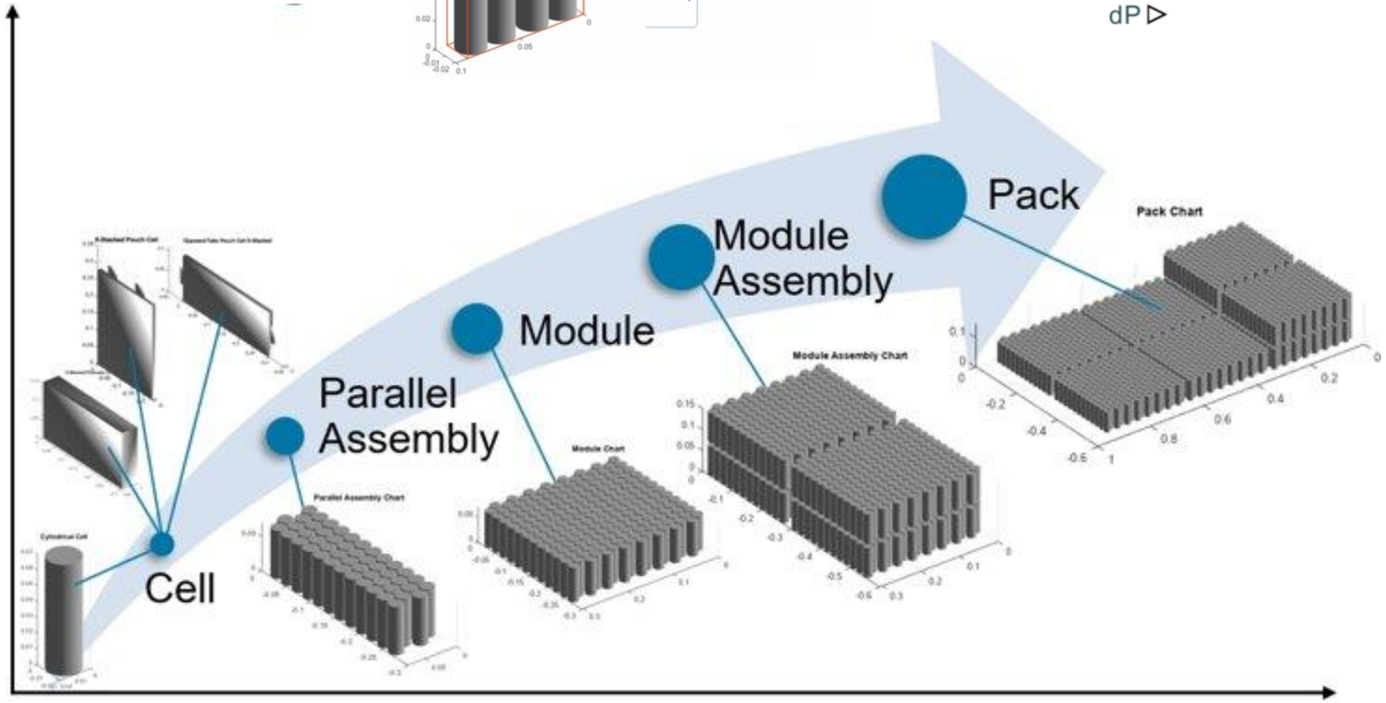
- The **Virtual Vehicle Composer** offers different powertrain architectures
  - Battery Electric Vehicles
  - Internal Combustion Vehicles
  - Hybrid Electric Vehicles
- The virtual vehicle model can be easily modified/extended
- Optimization at component- and system-level
- Wide range of alternatives for optimization algorithms
- For a deeper focus on the battery, we advise **Simscape Battery**

Next steps: Modeling battery packs with Simscape Battery

Key Features

- Battery Pack Builder (MATLAB API)
  - Automatically assemble cell models into battery pack
  - Define electrical and thermal connections (series, parallel)
  - Adjust tradeoff of simulation speed and model fidelity
- Cooling plate models
  - Parallel channel
  - U-shaped channel
- Battery management algorithms
  - Charge/discharge cycles
  - SOC, SOH estimators
  - Cell balancing, thermal management
- Support for C-code generation
- Application-specific examples
  - EV charging, Microgrid with BESS





Q&A

MathWorks®