

The slide features a dark blue background with a large, light blue triangular shape on the right side. Inside this triangle, there is a 3D wireframe plot with a color gradient from yellow at the top to blue at the bottom. In the background, there are faint white lines representing a signal waveform and a circuit diagram with various components and arrows.

MATLAB EXPO 2017

How Simscape™ Supports Innovation for Cyber-Physical Systems

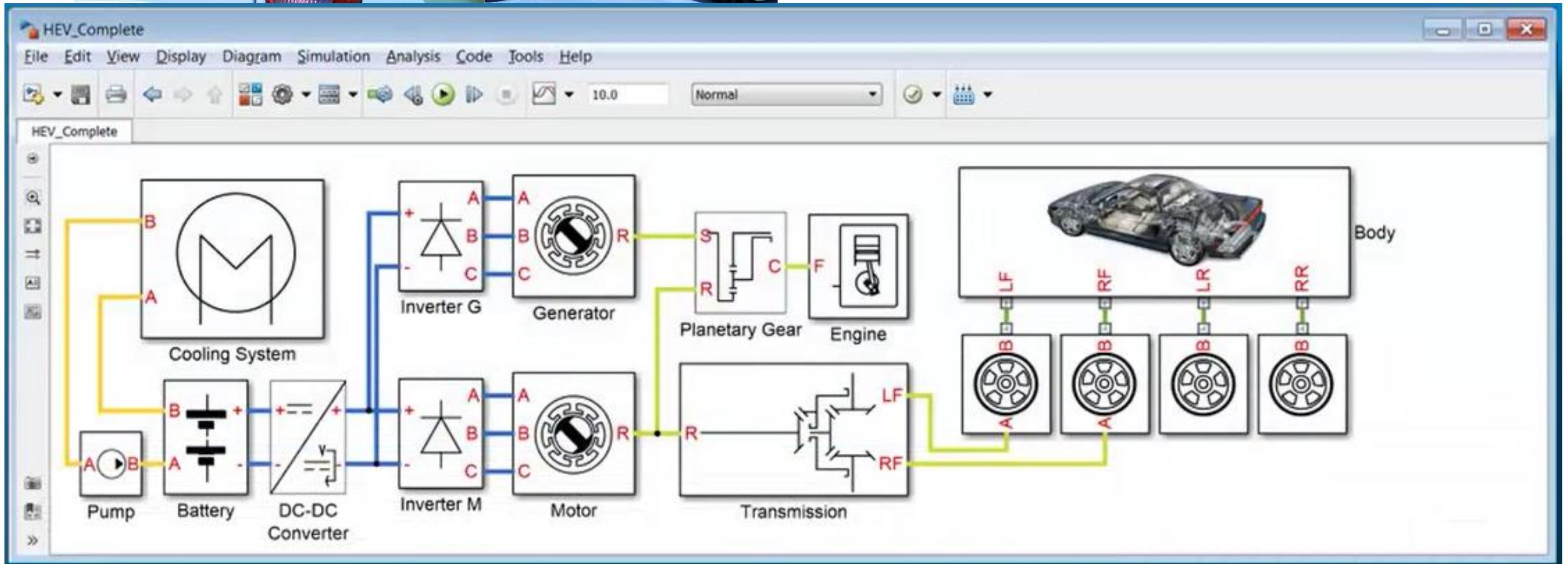
Rick Hyde

How can we use system-level modelling to support *innovative product design*?

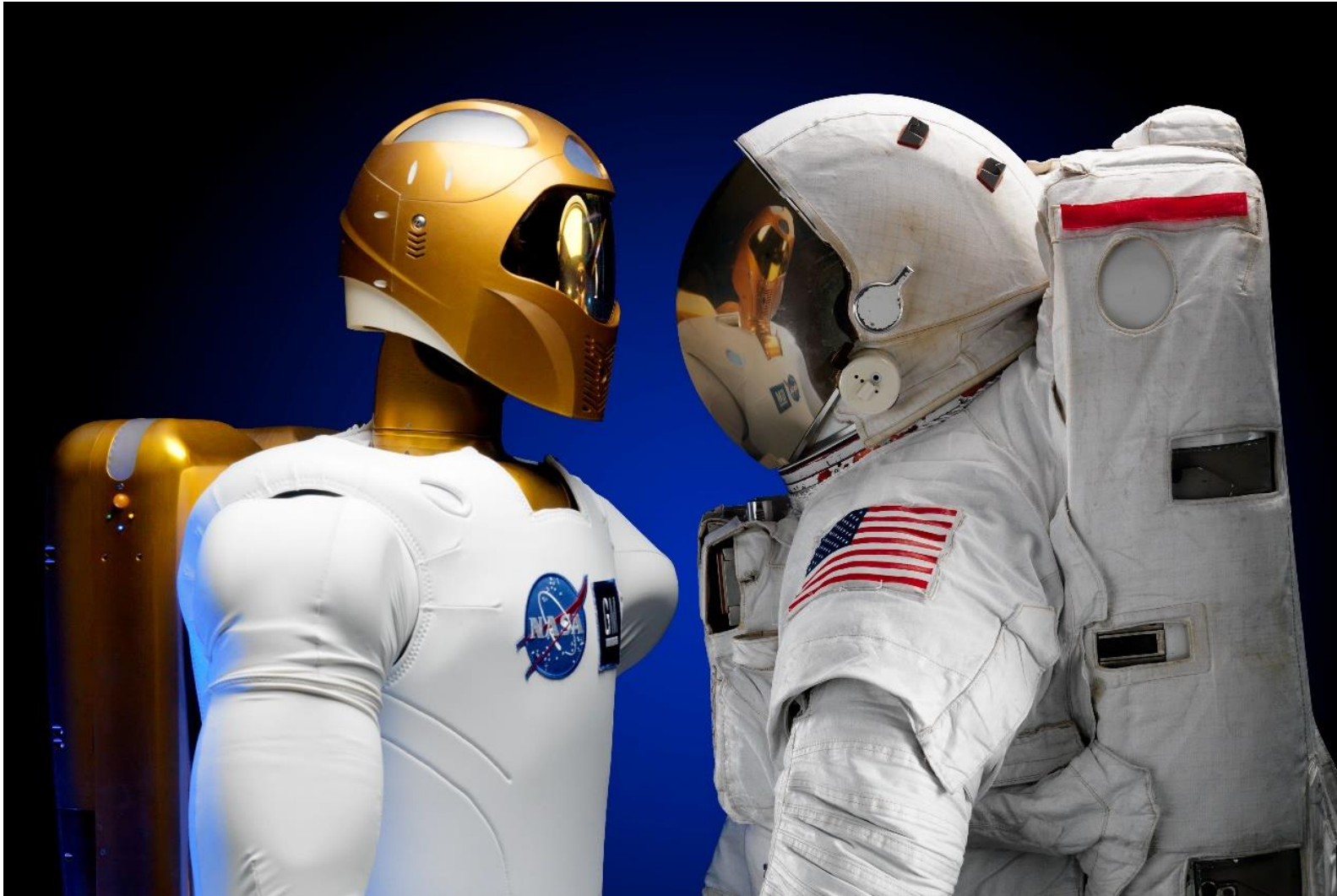
Innovation in electric and hybrid vehicles



- **Electrical**
- **Mechanical**
- **Fluid**



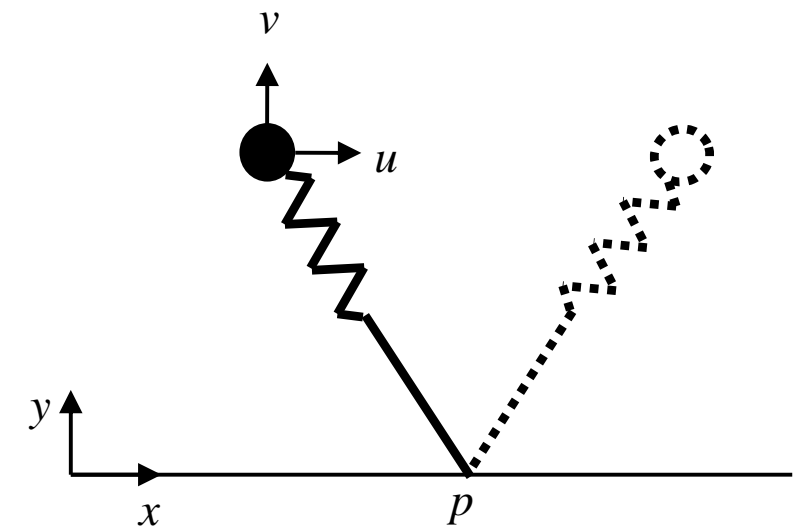
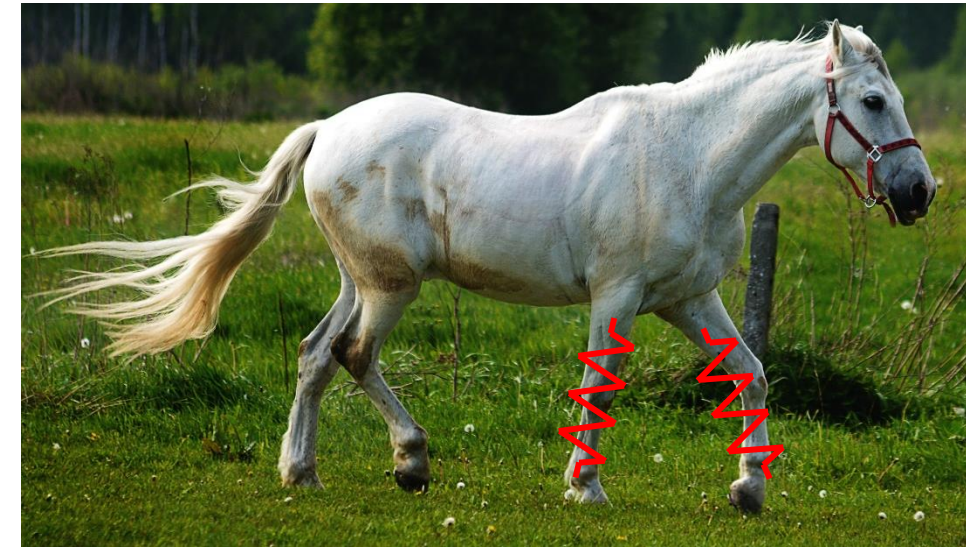
Innovation in robotics



Example: Quadruped running robot

Biologically-inspired design (Biomimetics)

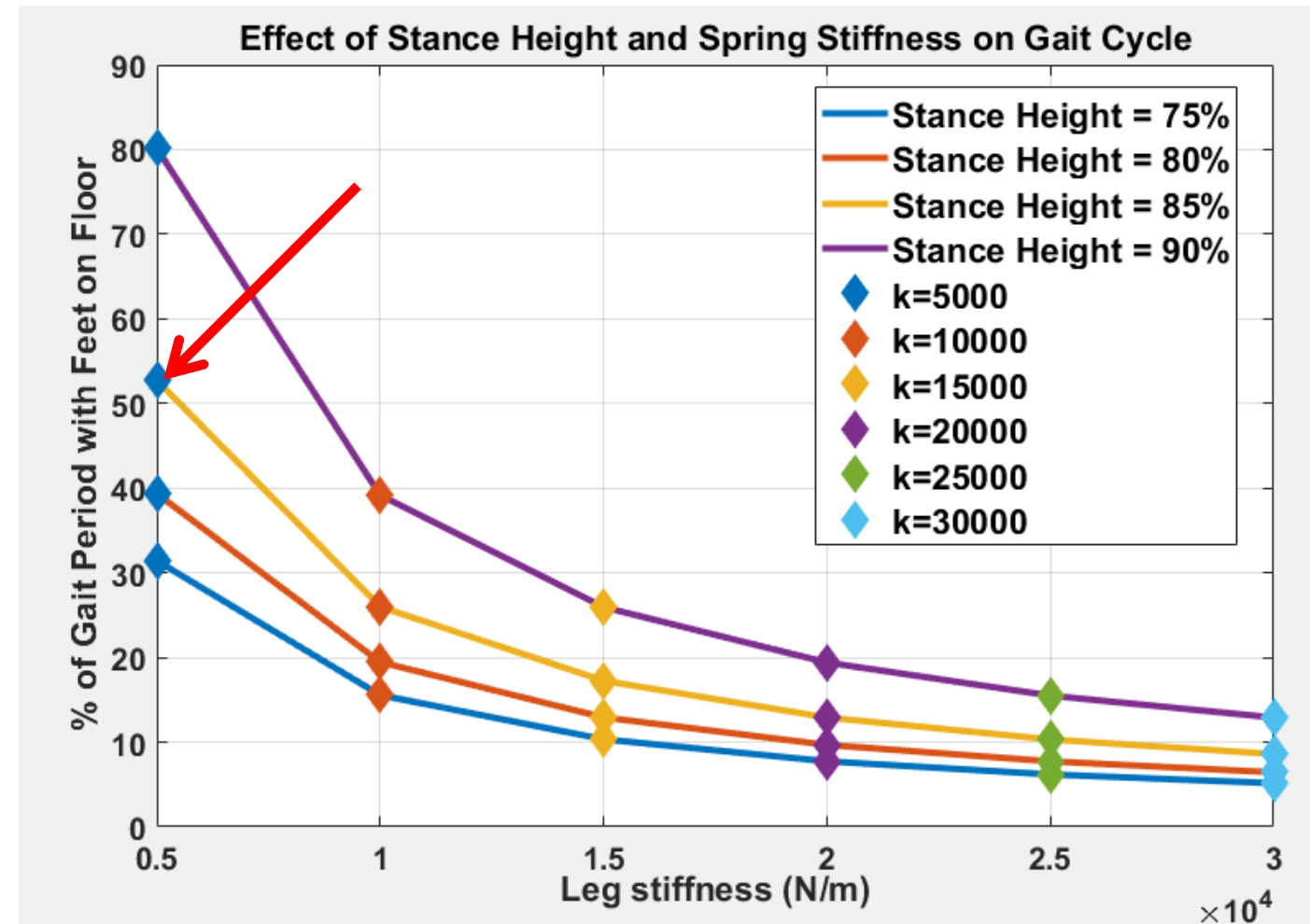
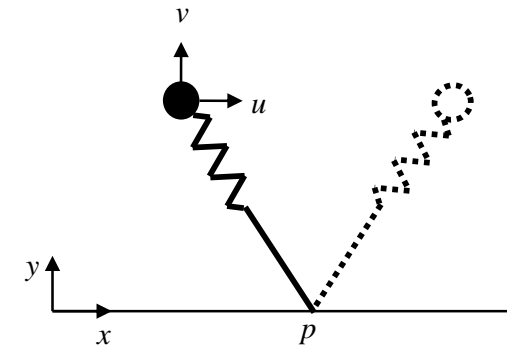
- Animal terrestrial motion
 - Muscles are inefficient (30%)
 - Muscles are also the energy store
 - Running gait uses kinetic energy recovery
 - The leg is well modelled by a linear spring
- Innovation: Use equivalent inverted pendulum model as basis for robot



Running robot design example

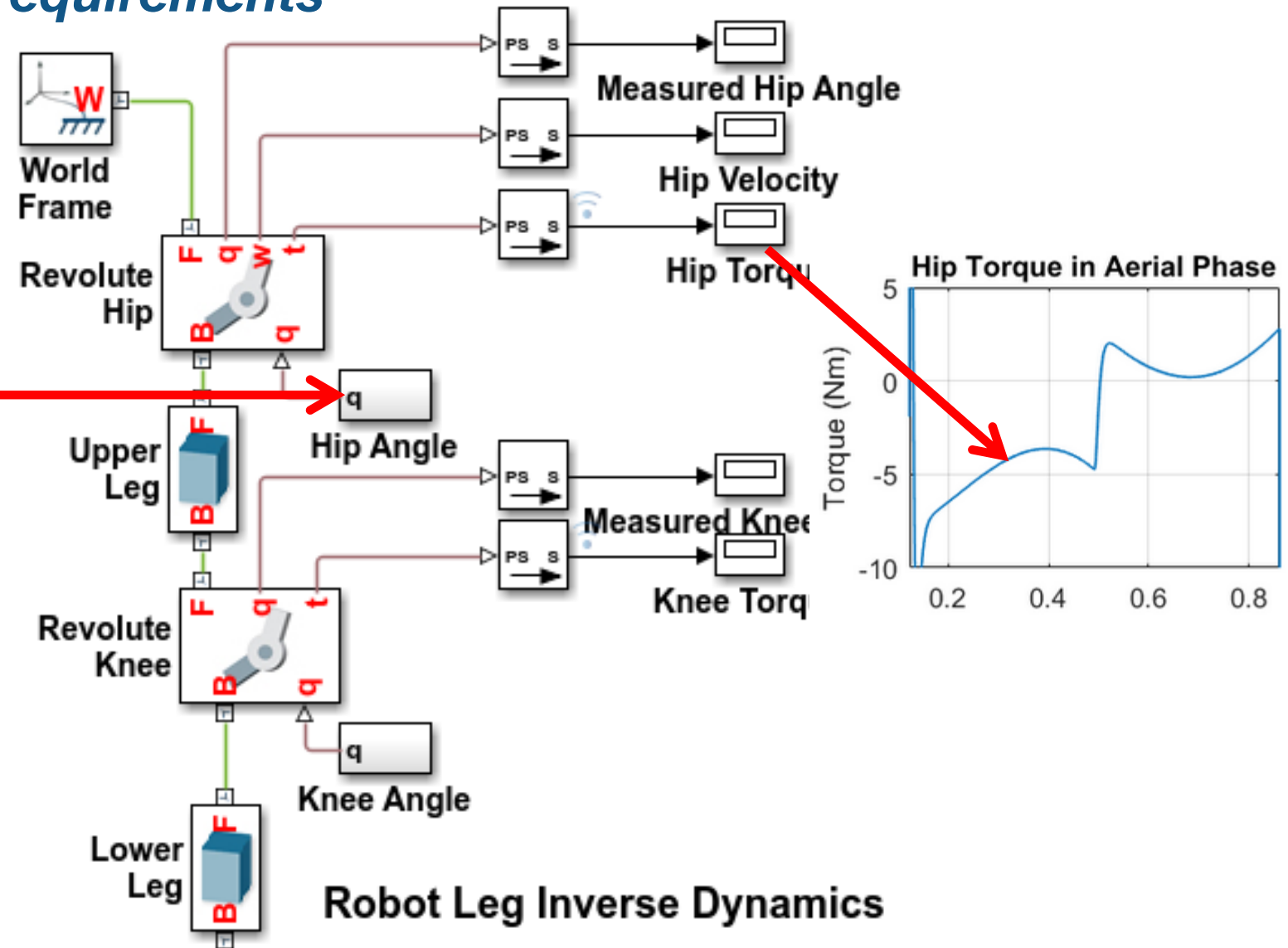
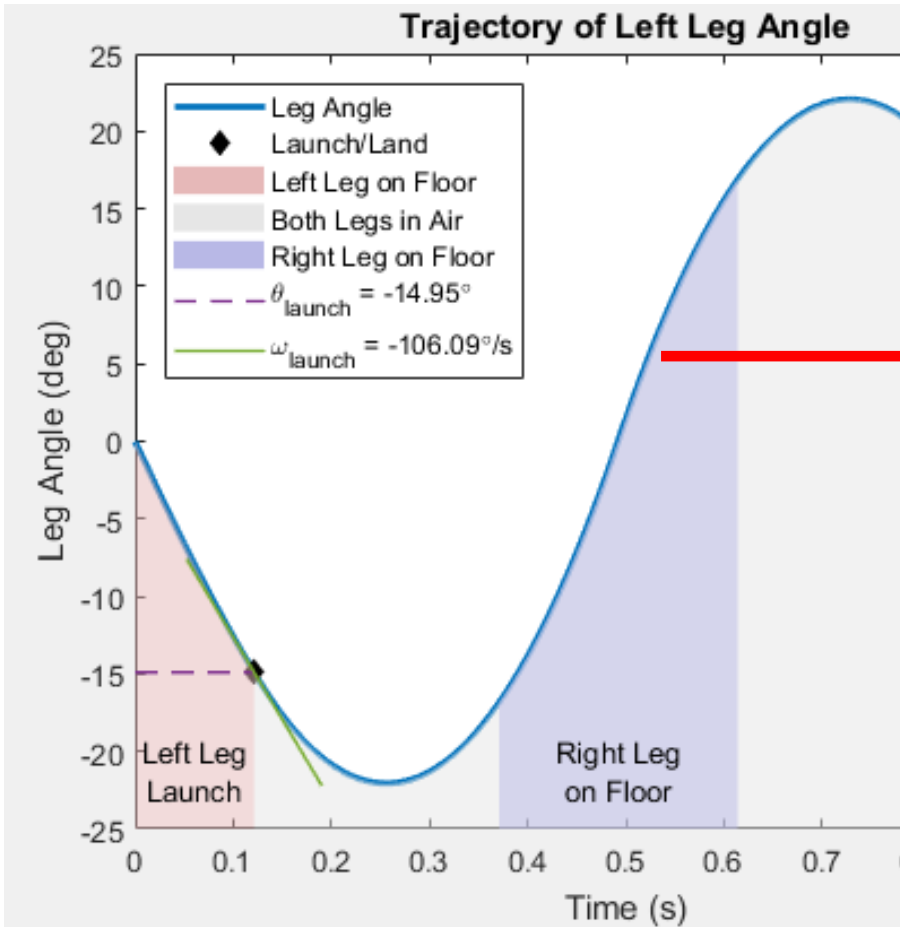
Design step #1 – gait selection

- Fixed parameters
 - Leg length
 - Running speed
 - Mass
- Design parameters
 - Leg (spring) stiffness
 - Stance height
- Simple point-mass model
 - MATLAB script for trade-off



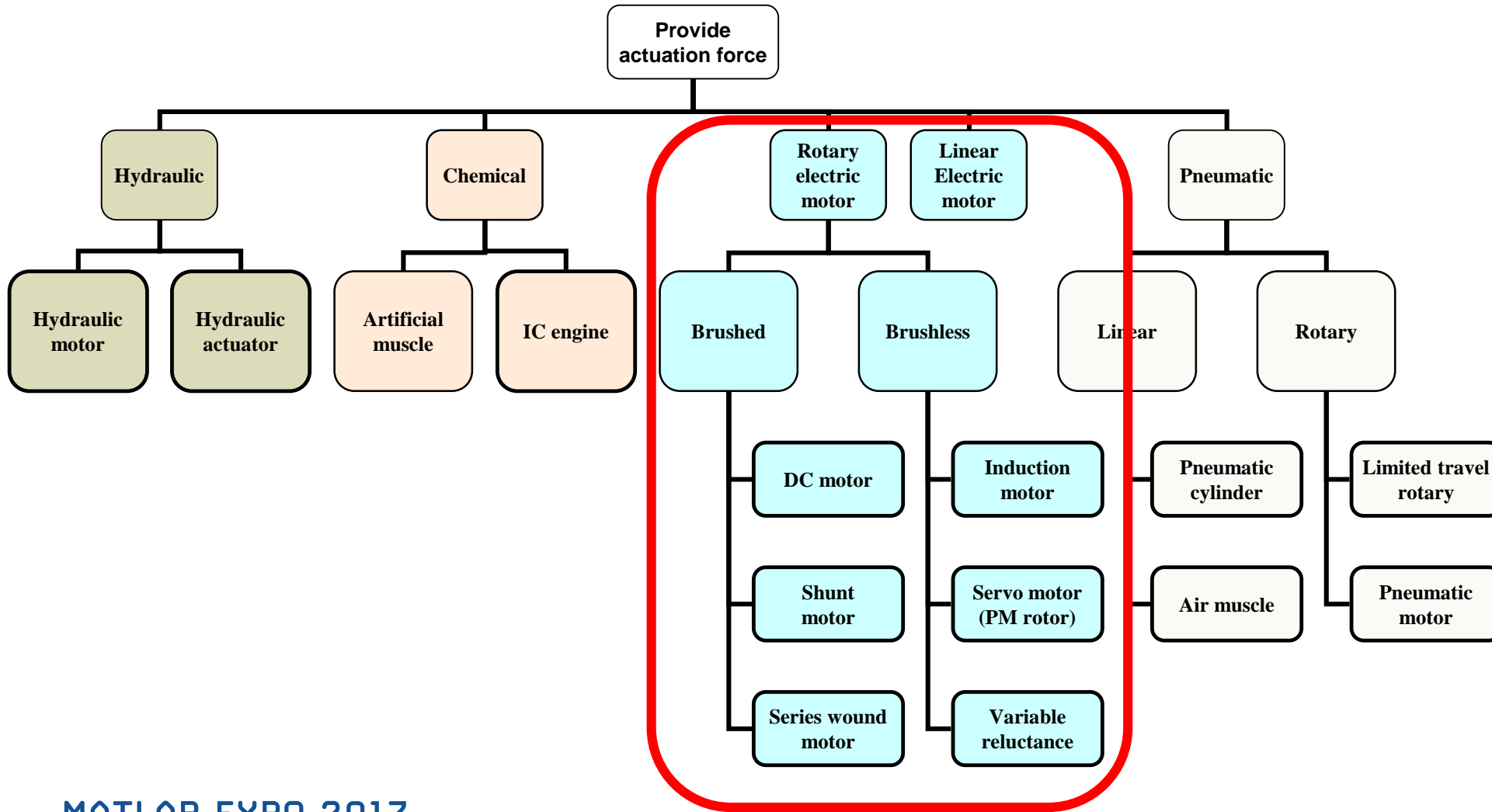
Running robot design example

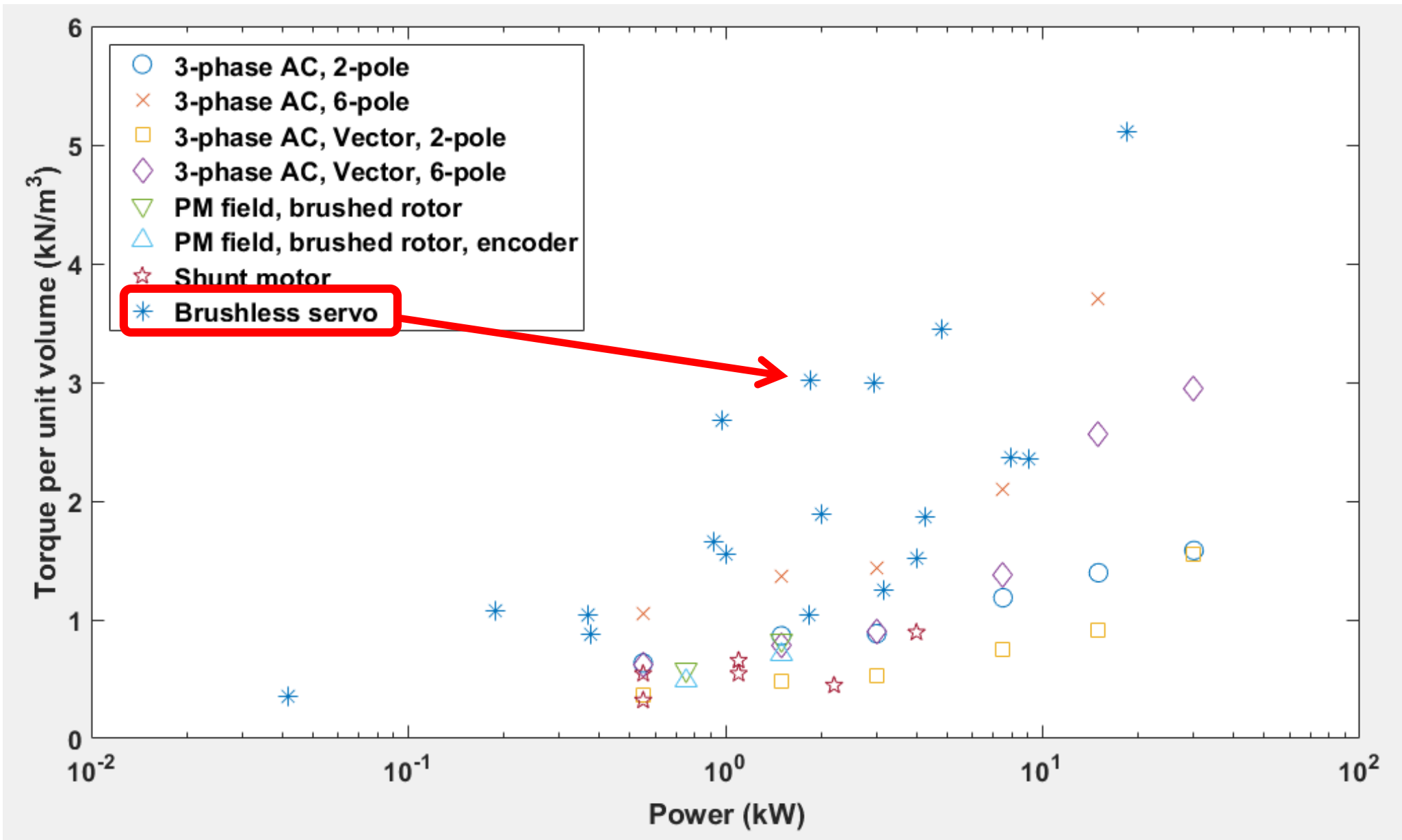
Design step #2 – actuator requirements from inverse dynamics



Running robot design example

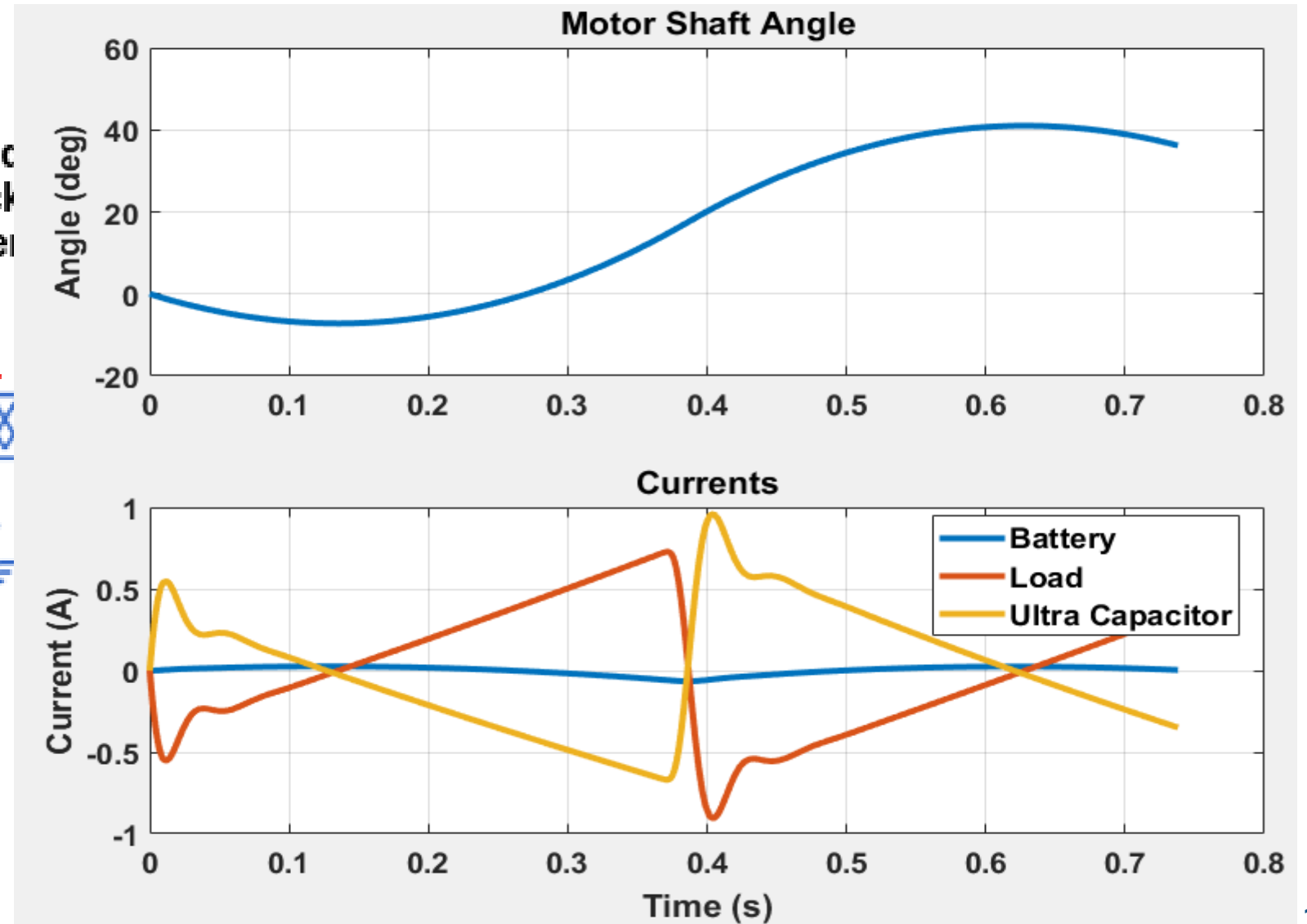
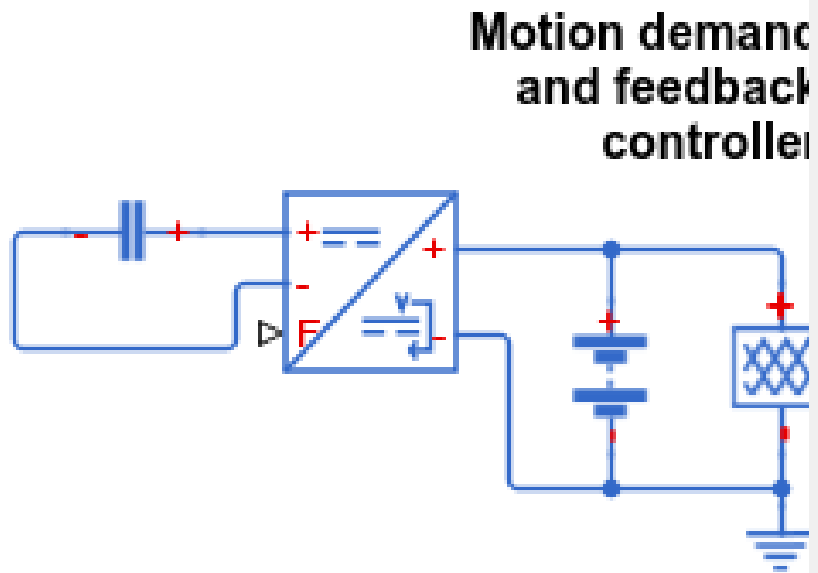
Design step #3 – actuator selection





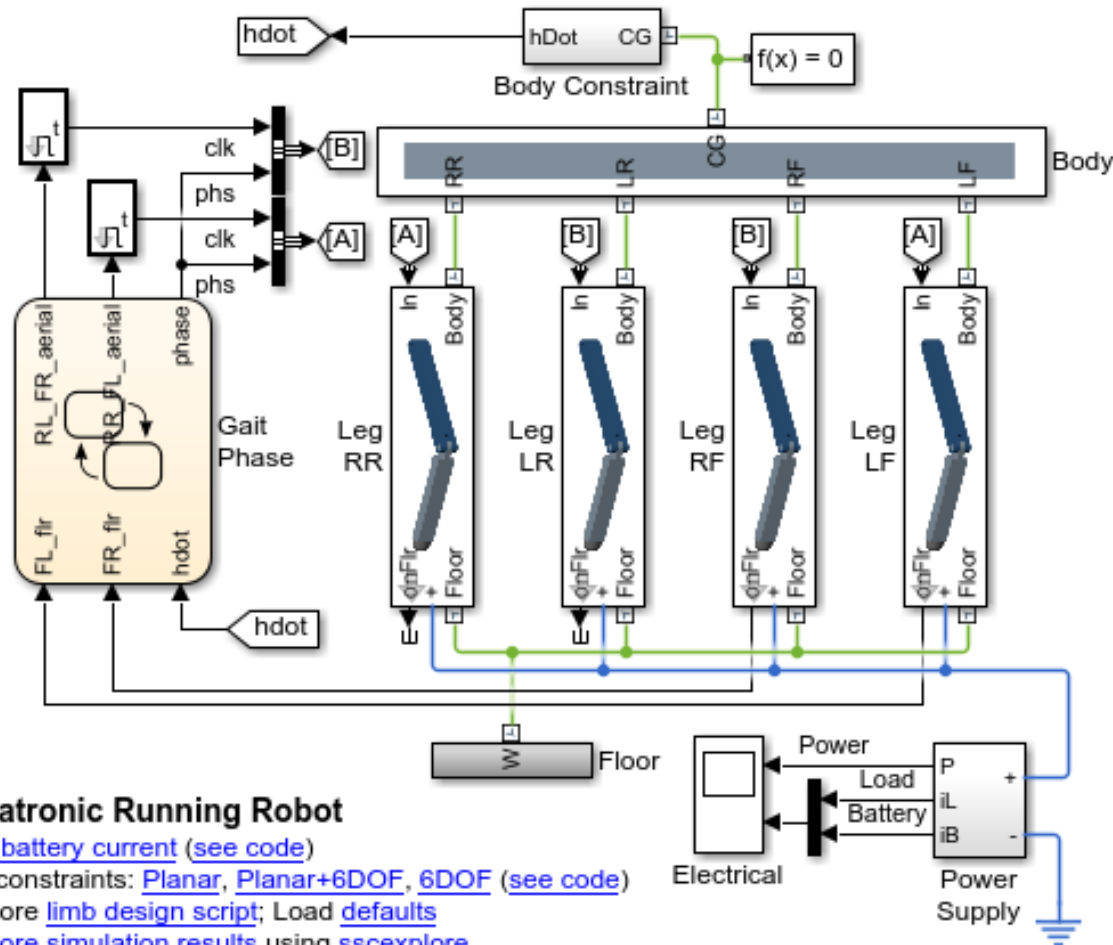
Running robot design example

Design step #4 – actuation validation



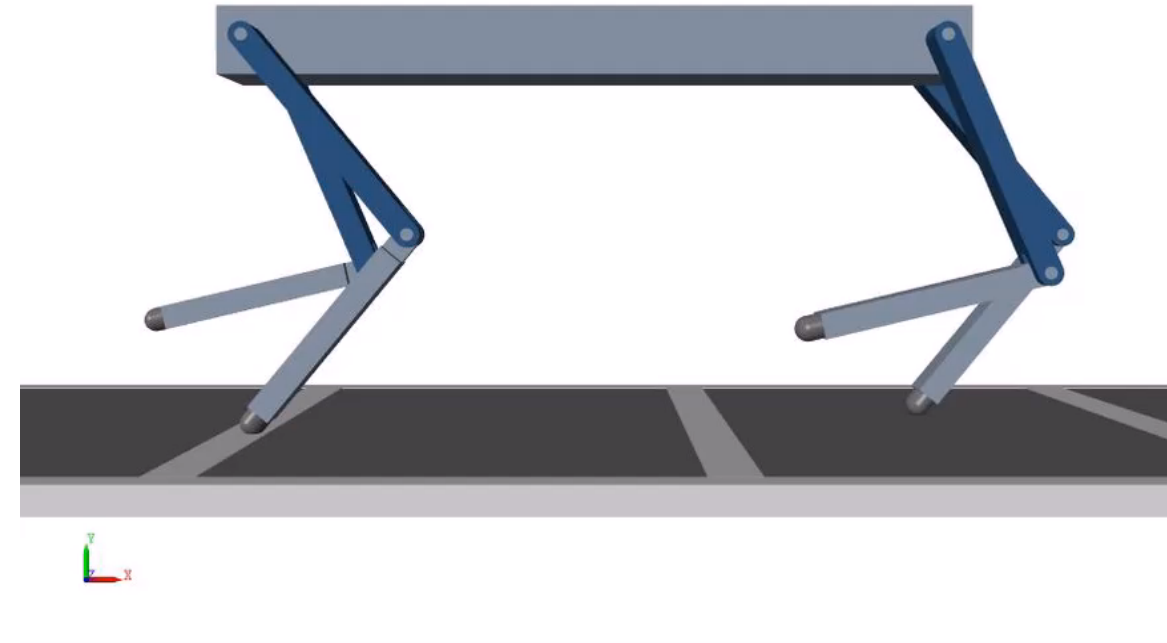
Running robot design example

Design step #5 – evaluation



Mechatronic Running Robot

1. [Plot battery current](#) (see code)
2. Set constraints: [Planar](#), [Planar+6DOF](#), [6DOF](#) (see code)
3. Explore [limb design script](#); Load defaults
4. [Explore simulation results](#) using [sscexplore](#)
5. [Learn more](#) about this example



Motor efficiency at rated load = 95%
 Motor efficiency for trotting gait = 84%

Running robot design example

Design step automation using MATLAB scripting

```
%% Generate nominal gait, leg length and payload mass

% Biomechanical parameters
L = 1.0;    % Leg length (m)
m = 25;    % Mass (Kg)
k = 5315;  % Leg stiffness (N/m)

% Initial conditions for normalized positions and speeds
x0 = 0.0;   % Horizontal position of mass in middle of stance phase ()
y0 = 0.85*L; % Height of mass in middle of stance phase ()
u0 = 2.0;   % Horizontal speed in middle of stance phase (/s)
```

- Automation permits greater understanding of design trade-offs
 - e.g. see effect of gearbox ratio on efficiency

Gear ratio	80	100	120
Efficiency	84%	81%	78%

Running robot design example

Key points

1. Multiple models
2. Each model matched to a design task
3. Design data passed between models
4. Automation to support analysis & optimisation
5. Code generation for HIL testing

Enables product design innovation in a way that starting with the CAD tool could never do

Building the right model for the task at hand can be challenging

Requirements
not understood
by project
management

Identification of
required
modelling detail

Identify required modelling detail for *PMSM* drives

1. System-level simulation

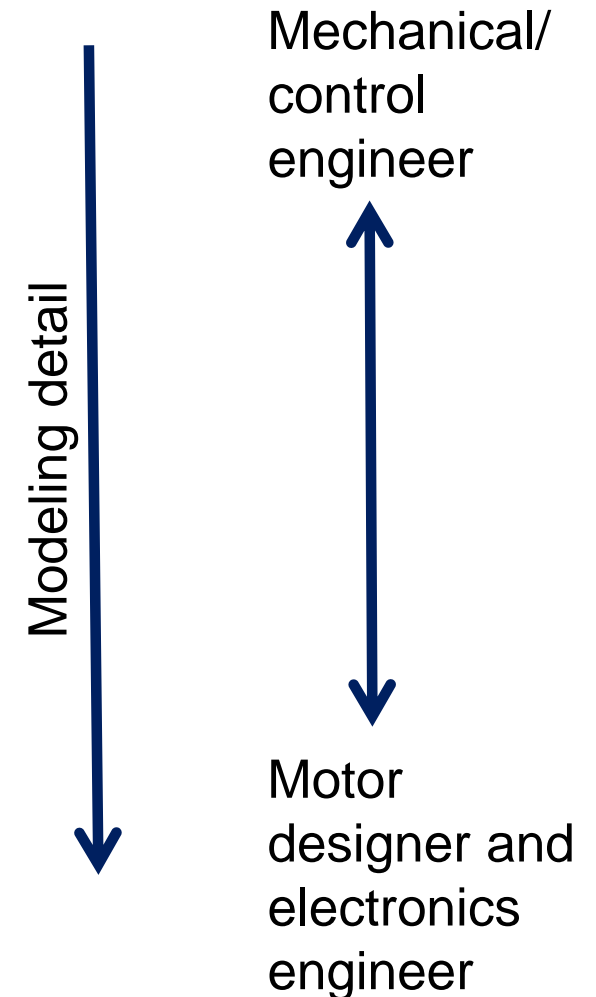
- Torque-speed behaviour
- Model motor losses as part of overall efficiency calculation
- Thermal & fault modelling

2. Component validation

- Ensure motor stays within manufacturer operating limits
- Detailed analysis of impact on other components e.g. power harmonics

3. Component design

- Motor and/or drive circuitry
- Determine overall actuation losses
- Understand/predict fault behaviour



Building the right model for the task at hand can be challenging

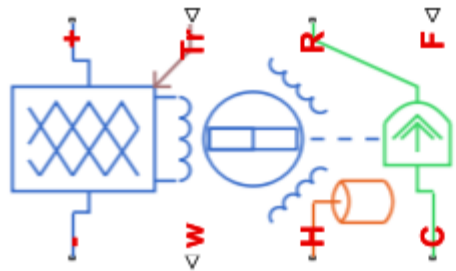
Requirements
not understood
by project
management

Identification of
required
modelling detail

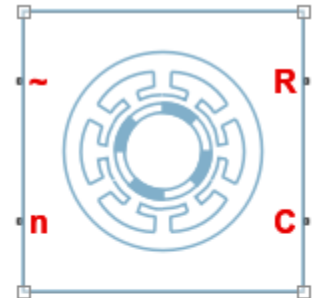
Limited time
and nothing to
build on –
starting from
scratch

Lacking
domain
knowledge

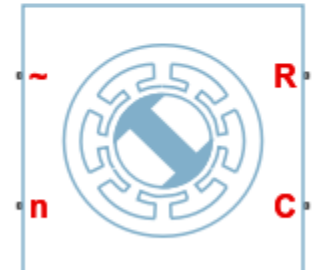
Simscape library components provide a useful starting point and encapsulate some domain knowledge



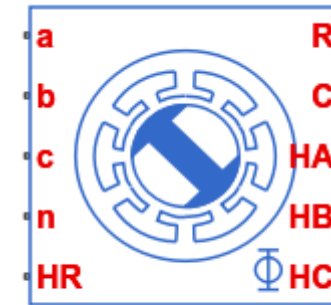
Servomotor



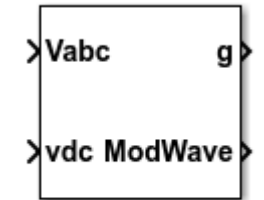
Brushless DC Motor



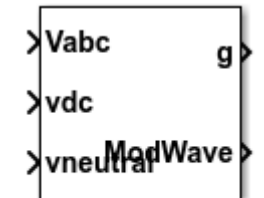
Permanent Magnet Synchronous Motor



FEM-Parameterized PMSM



PWM Generator (3-phase, 2-level)



PWM Generator (3-phase, 3-level)



Encoder



Resolver

Modeling detail

Building the right model for the task at hand can be challenging

Requirements
not understood
by project
management

Identification of
required
modelling detail

Limited time
and nothing to
build on –
starting from
scratch

No data

Lacking
domain
knowledge

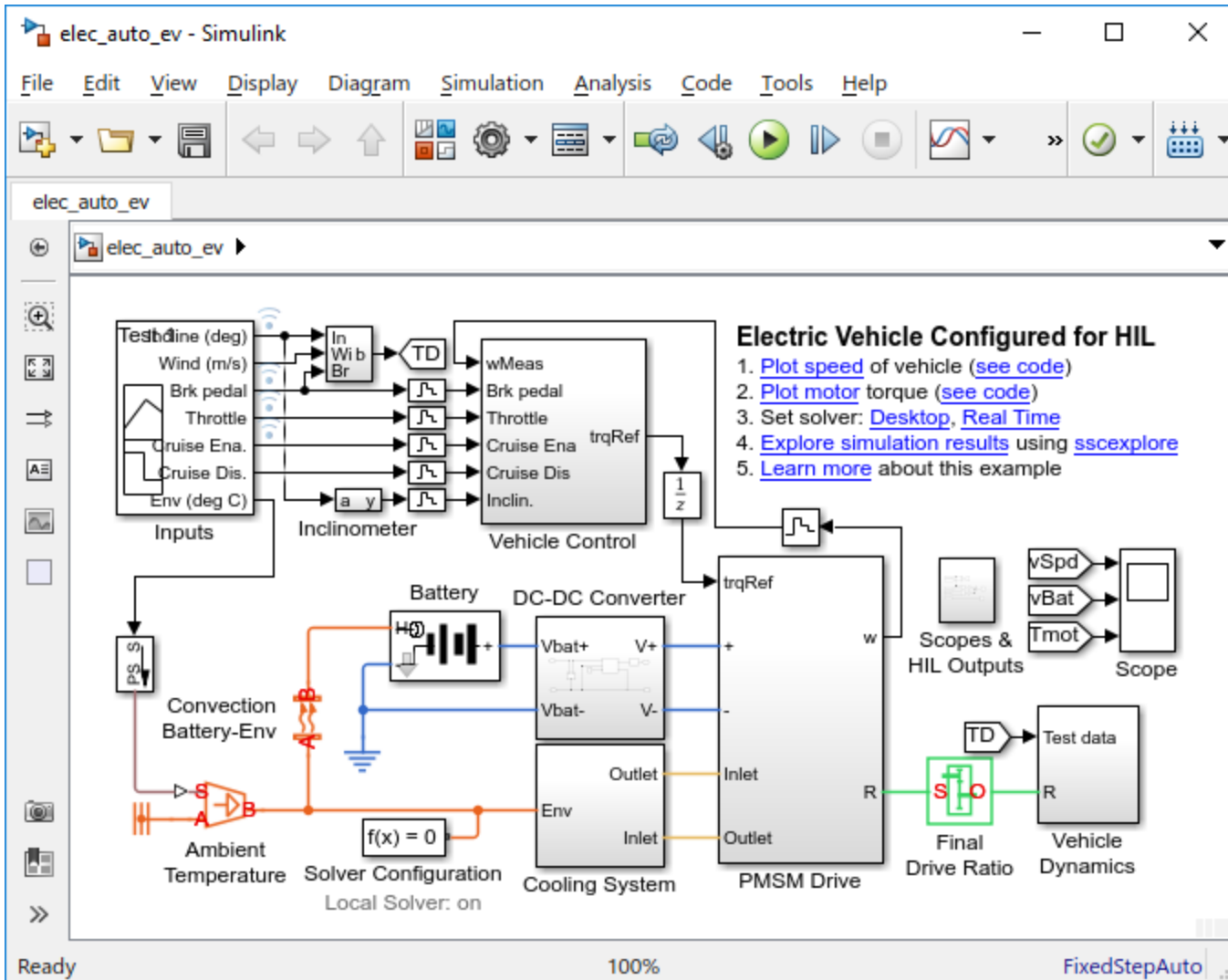
Modelling a PMSM with limited supplier data

Tune to measurement data

See PMSM parameter identification example in Track 2 at 16:15pm



Using abstraction to deal with limited data



R2017a/R2017b:
elec_auto_ev.slx

R2016b/R2016a/R2015b:
elec_electric_vehicle.slx

elec_auto_ev - Simulink

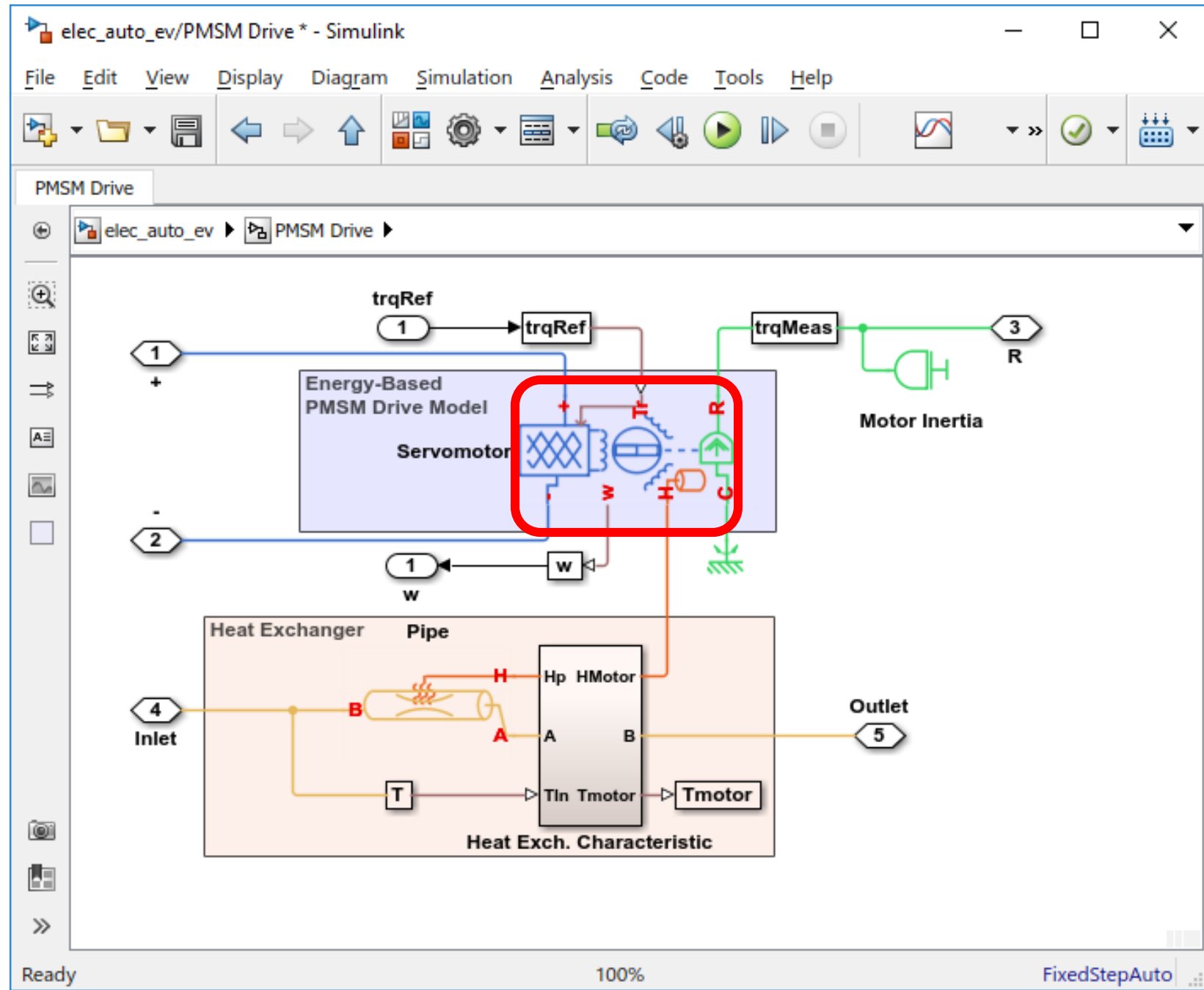
File Edit View Display Diagram Simulation Analysis Code Tools Help

elec_auto_ev

Electric Vehicle Configured for HiL

1. [Plot speed](#) of vehicle ([see code](#))
2. [Plot motor torque](#) ([see code](#))
3. Set solver: [Desktop](#), [Real Time](#)
4. [Explore simulation results](#) using [sscexplore](#)
5. [Learn more](#) about this example

Ready 100% FixedStepAuto



Block Parameters: Servomotor

Servomotor

This block represents a servomotor and drive electronics operating in torque-control mode, or equivalently current-control mode. The motor's permissible range of torques and speeds is defined by a torque-speed envelope, and the output torque is assumed to track the torque reference demand T_r with time constant T_c .

The servomotor block should be connected to a DC supply. Electrical losses are assumed to be the sum of a constant term plus two additional terms that are proportional to the square of the torque and the square of the speed respectively. In addition, a resistor in series with the supply can be included to model transmission losses between power supply and servomotor driver.

The block produces a positive torque acting from the mechanical C to R ports.

Settings

Electrical Torque | **Electrical Losses** | Mechanical | Temperature Dependence | Thermal Port

Parameterize losses by: Tabulated loss data

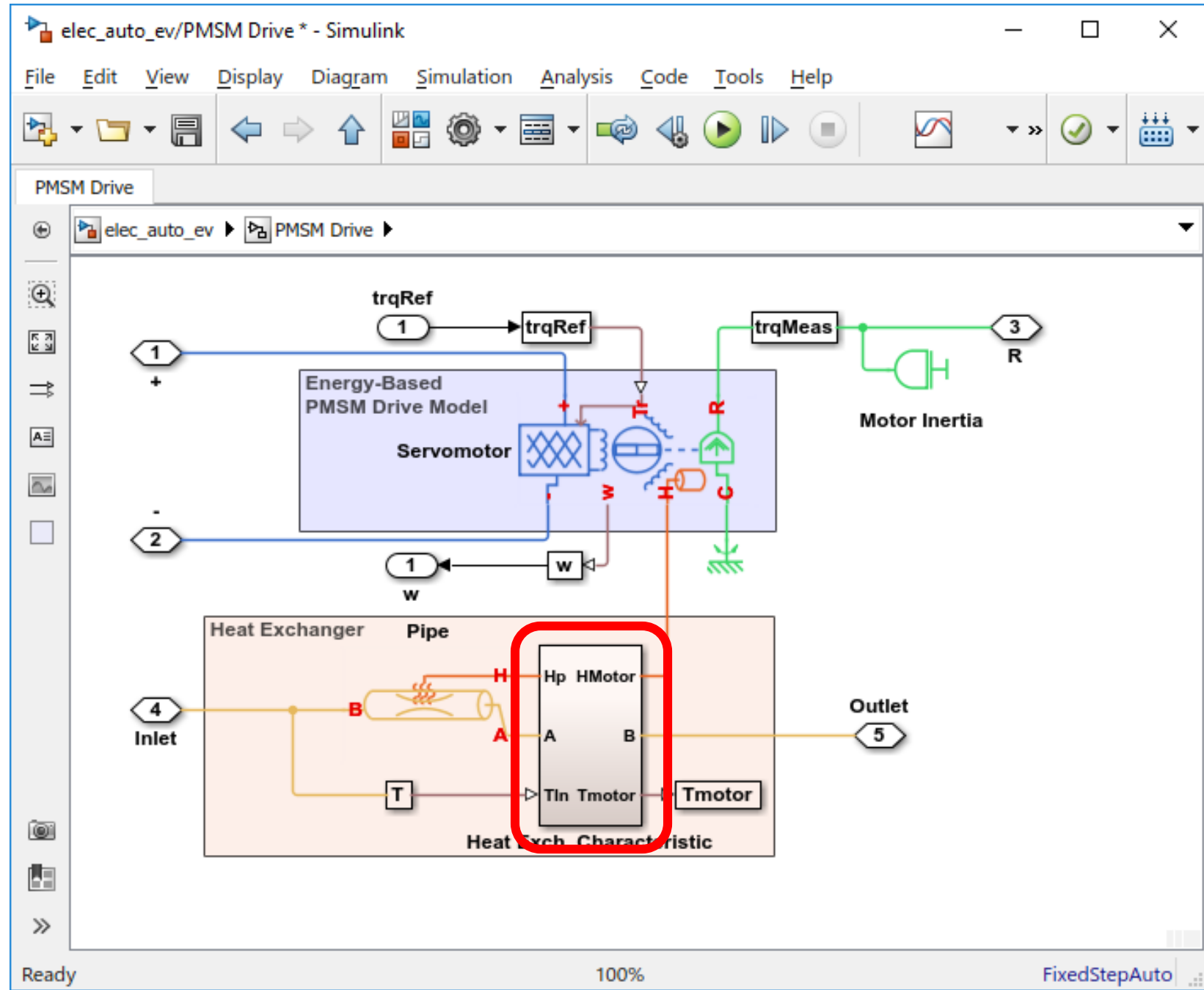
Vector of speeds (w) for tabulated losses: rpmVec rpm

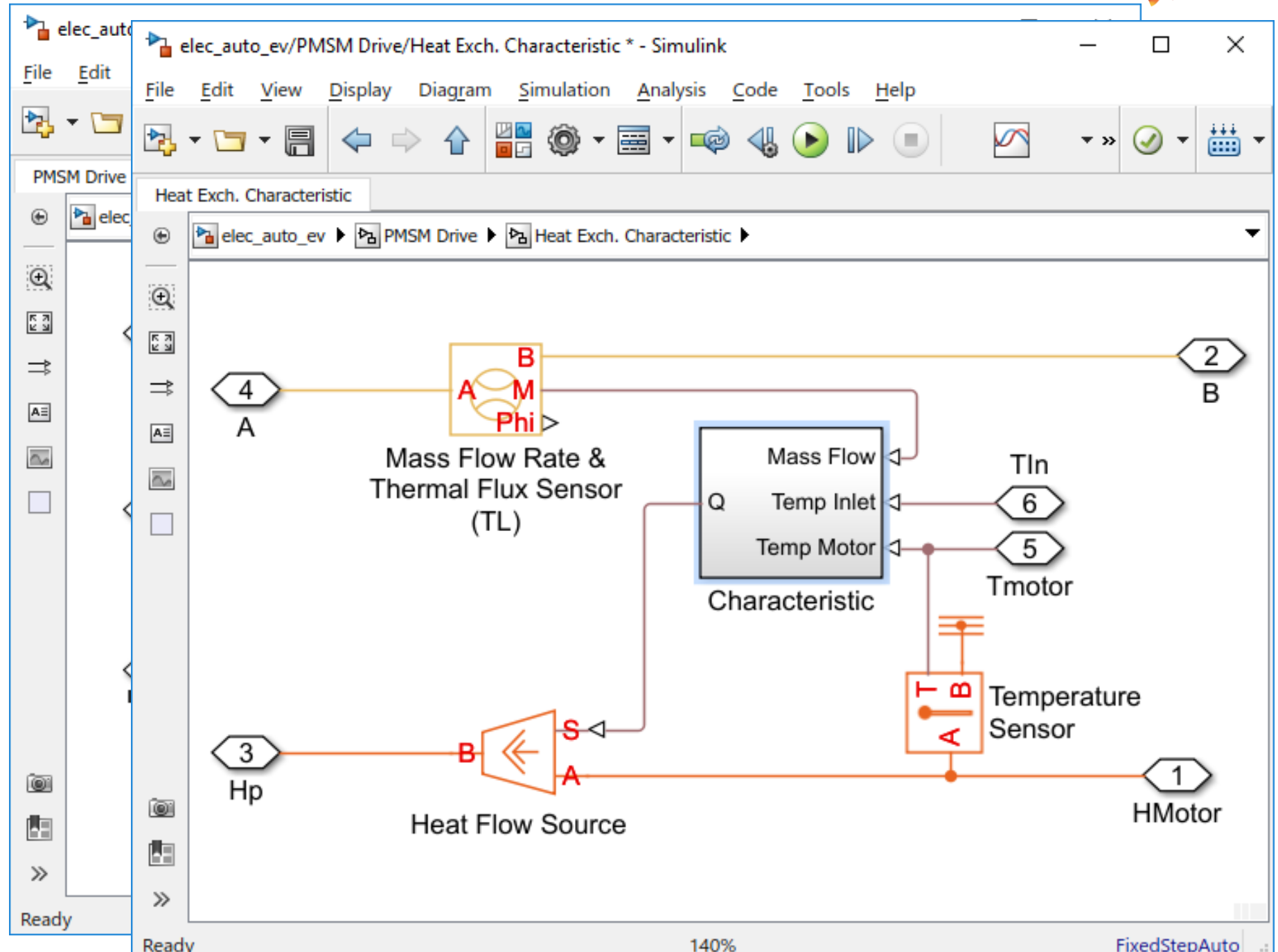
Vector of torques (T) for tabulated losses: trqVec N*m

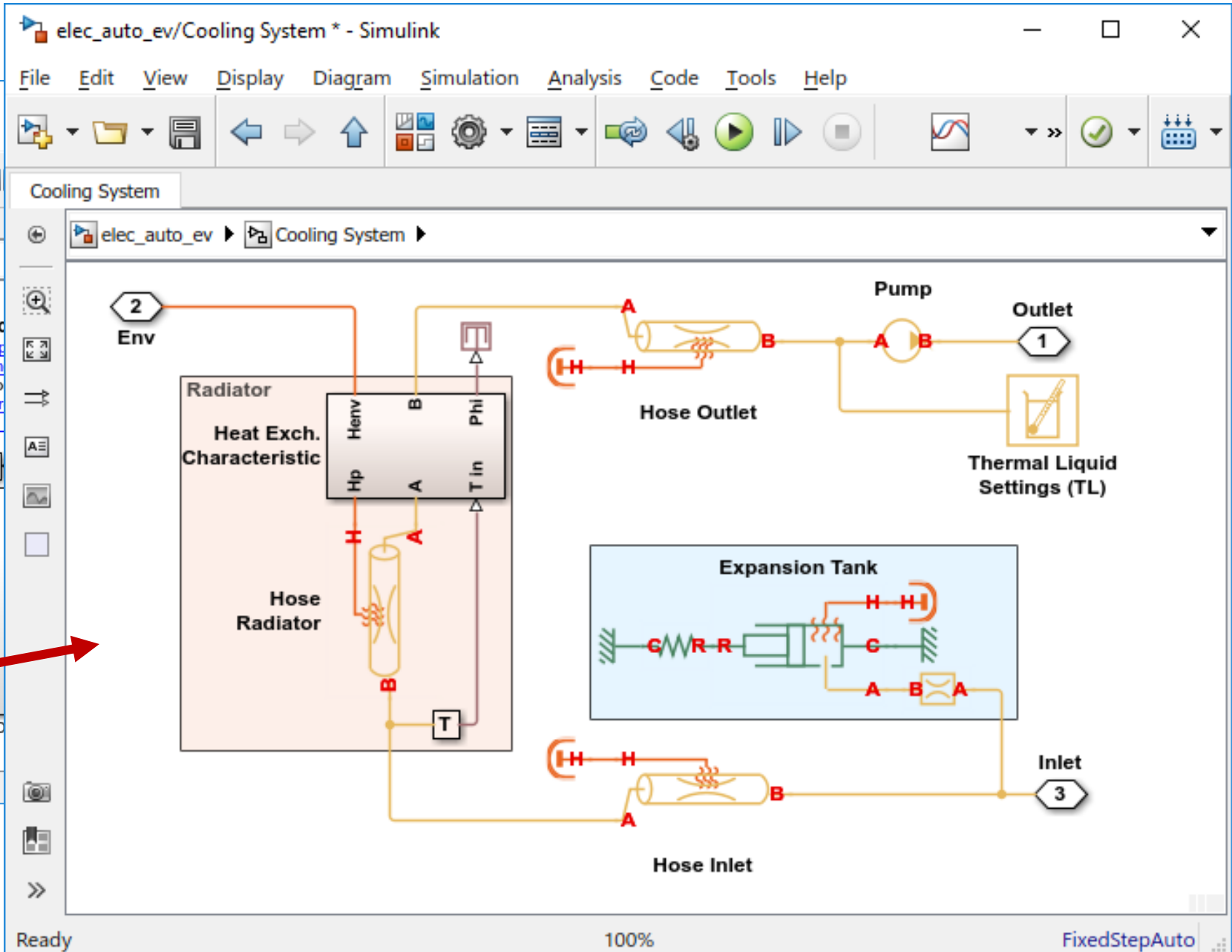
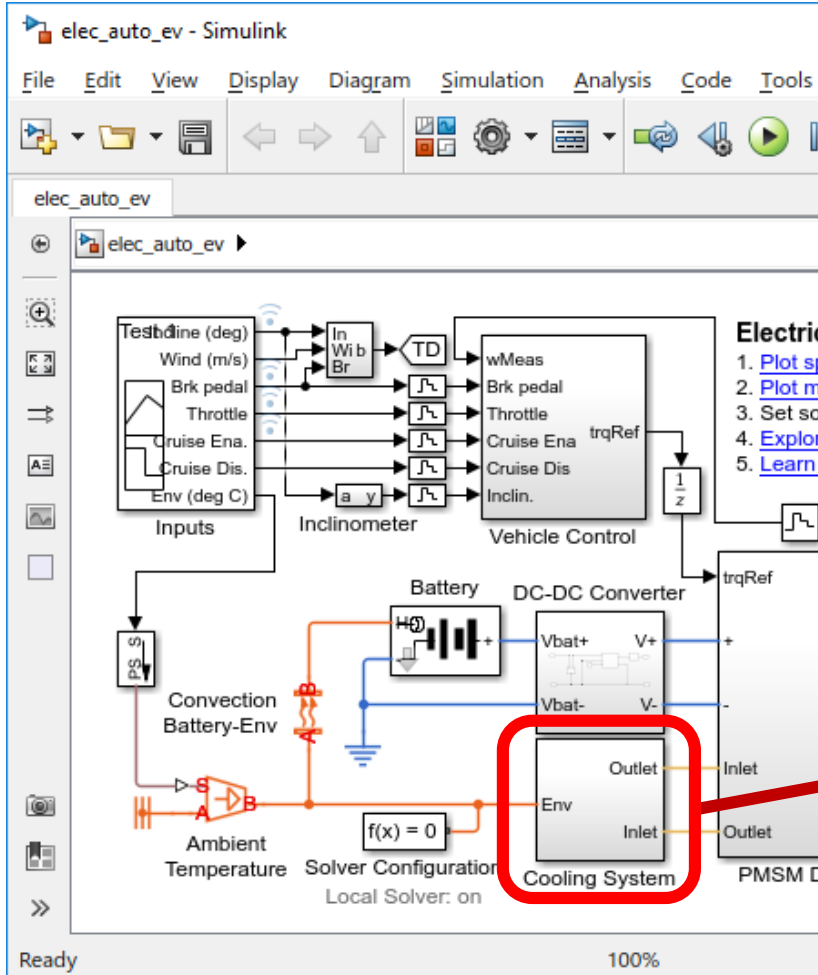
Corresponding losses, $P(w,T)$: LossesMat W

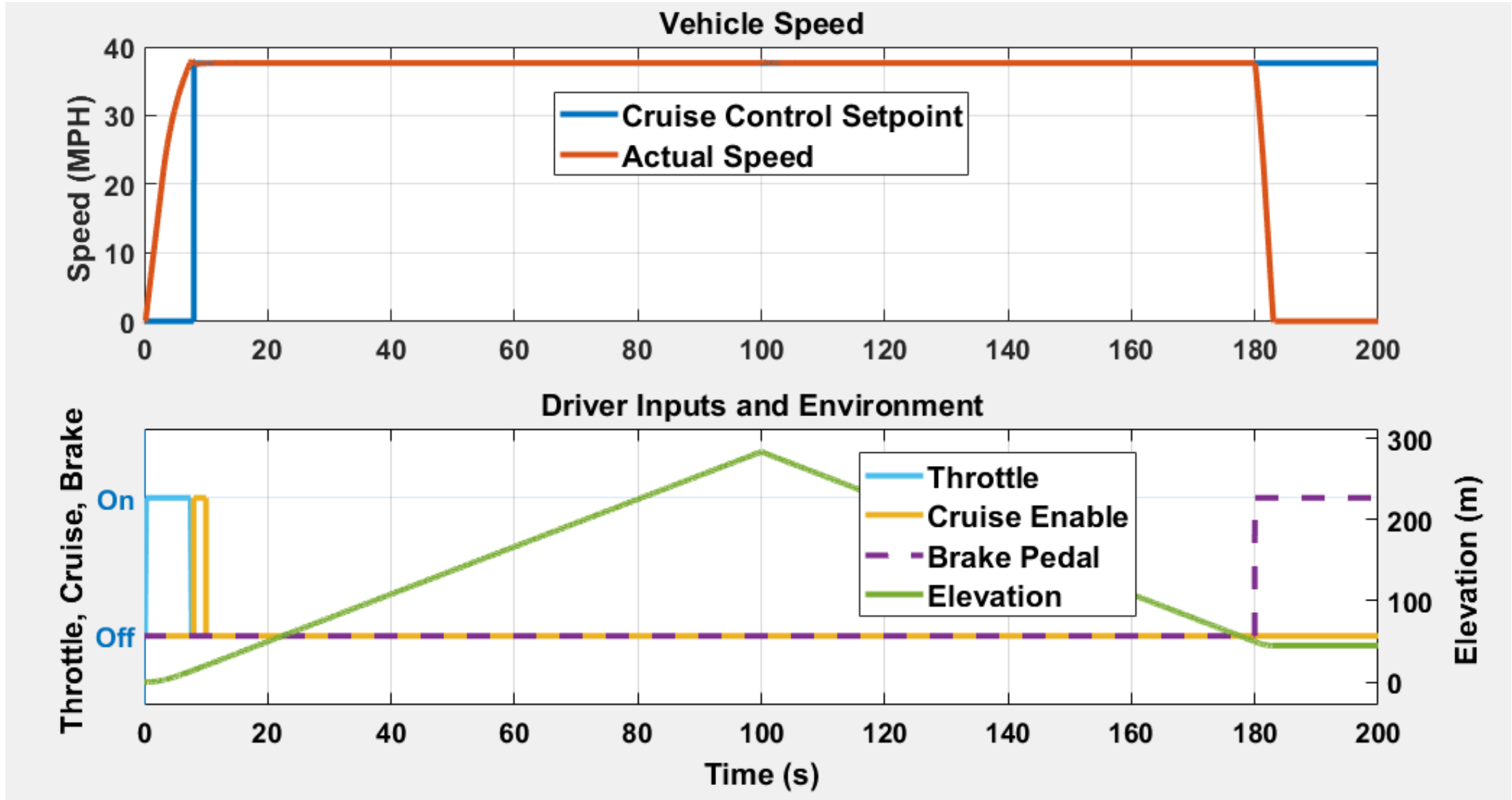
Supply series resistance: 0 Ohm

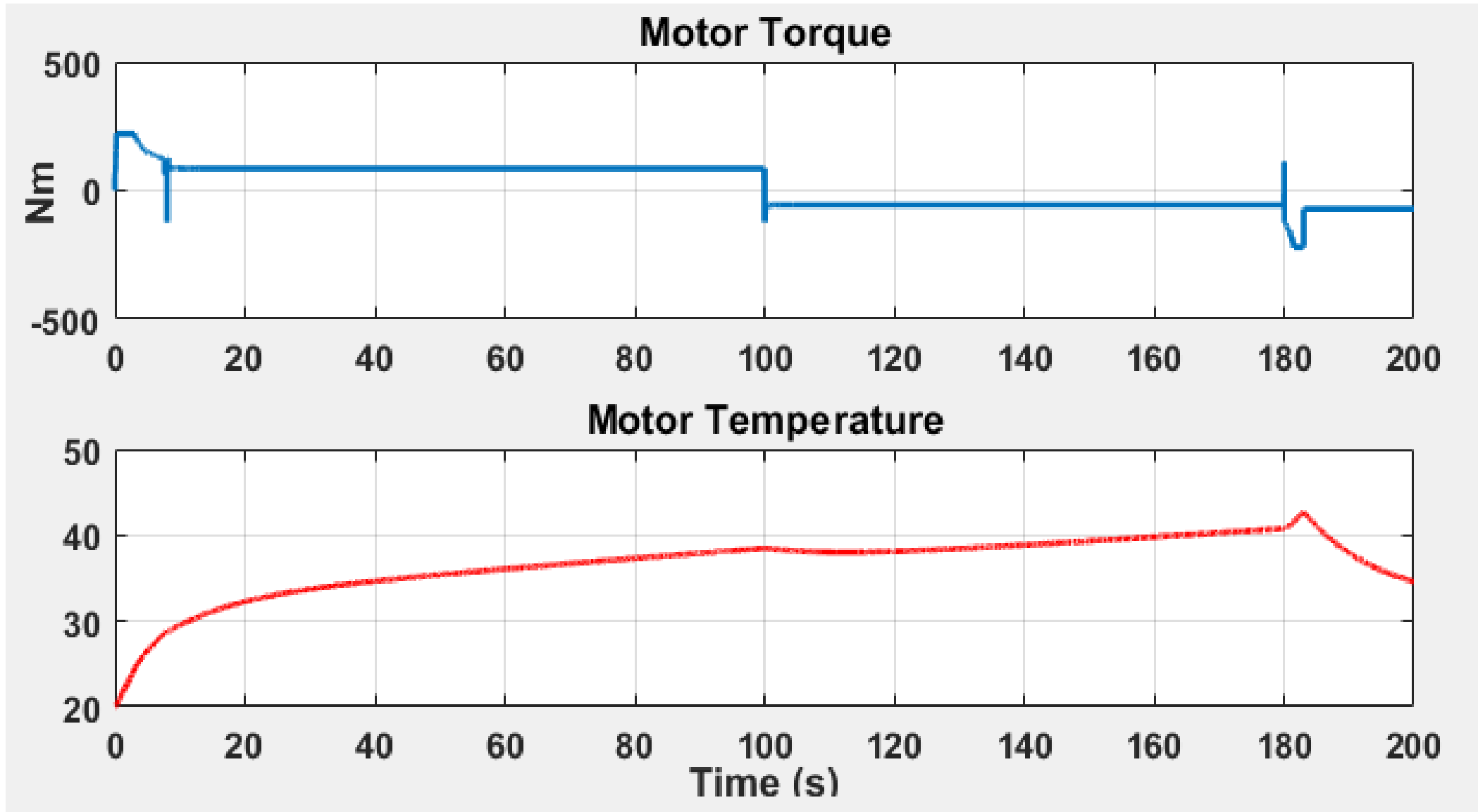
OK Cancel Help Apply











Building the right model for the task at hand can be challenging

Requirements
not understood
by project
management

Identification of
required
modelling detail

Limited time
and nothing
to build on –
starting from
scratch

No data

Lacking
domain
knowledge

Building the right model for the task at hand can be challenging

Requirements
not understood
by project
management

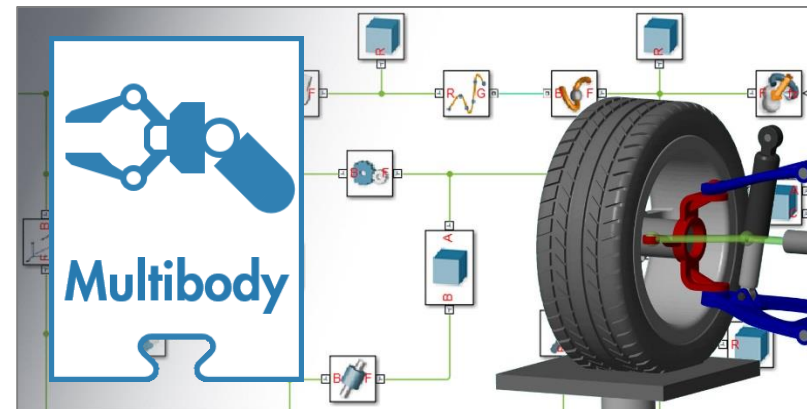
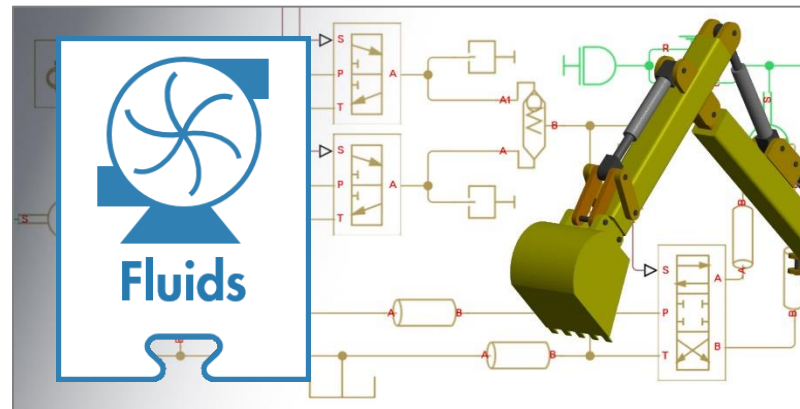
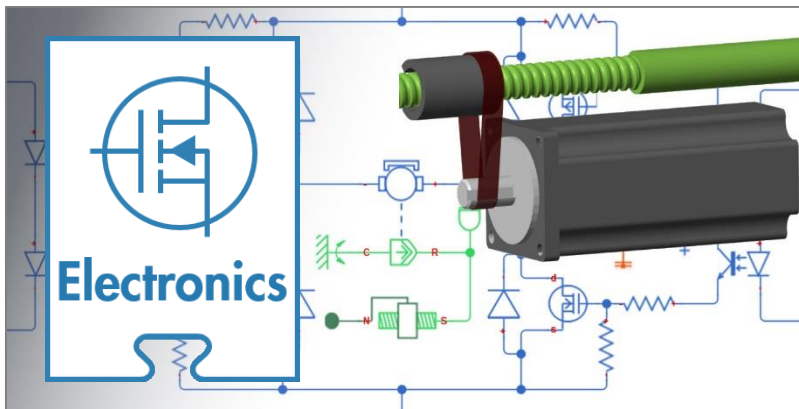
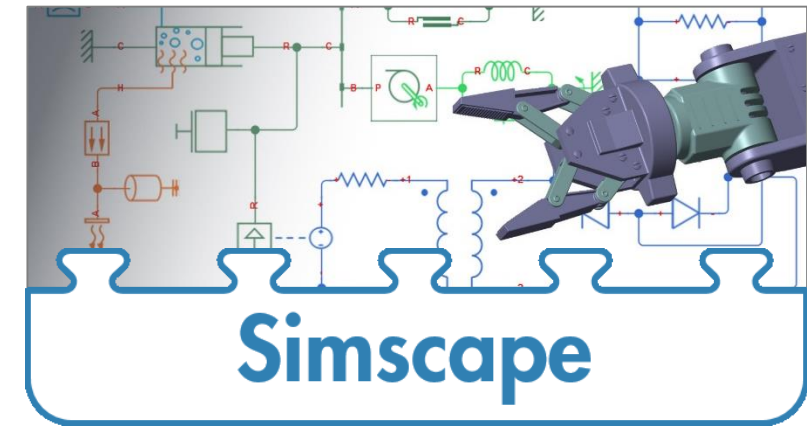
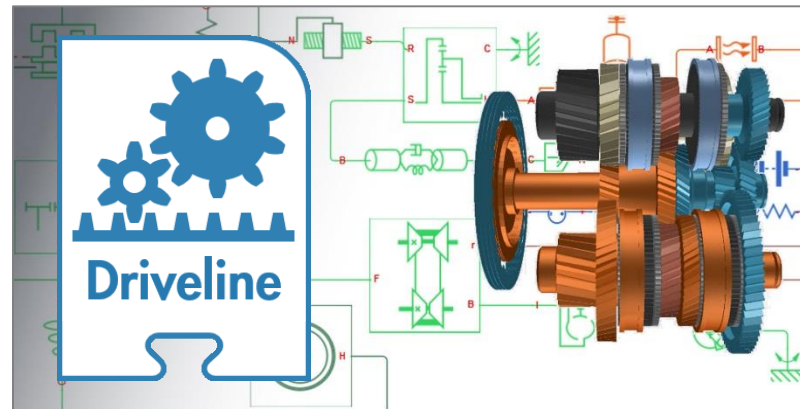
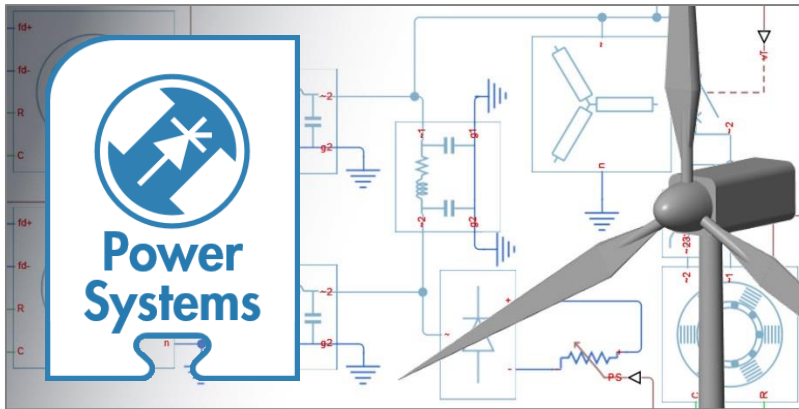
Identification of
required
modelling detail

No data

**Limited time
and nothing
to build on –
starting from
scratch**

**Lacking
domain
knowledge**

Simscape libraries enable you to build representative models fast

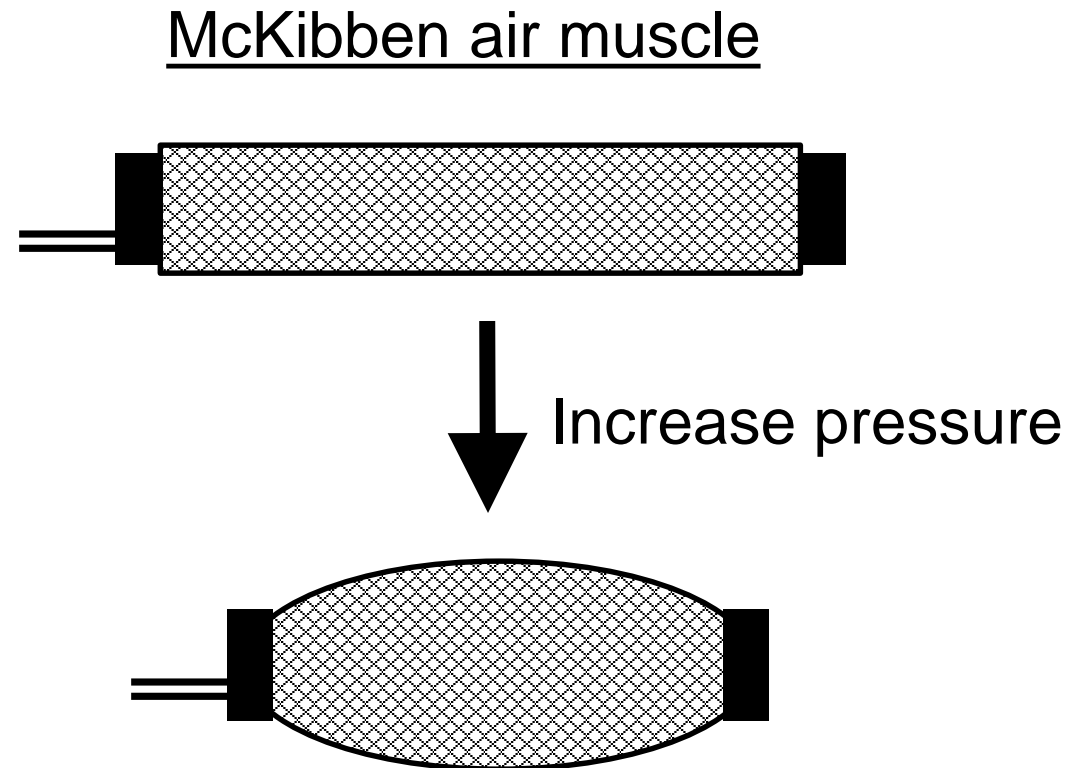


Creating custom Simscape components

Example: McKibben air muscle

Steps:

1. Write out defining equations
2. Find starting point in Simscape foundation library
3. Incrementally add functionality, testing as you go



Creating custom Simscape components

Step 1: Write out equations

L_u = Un-stretched length

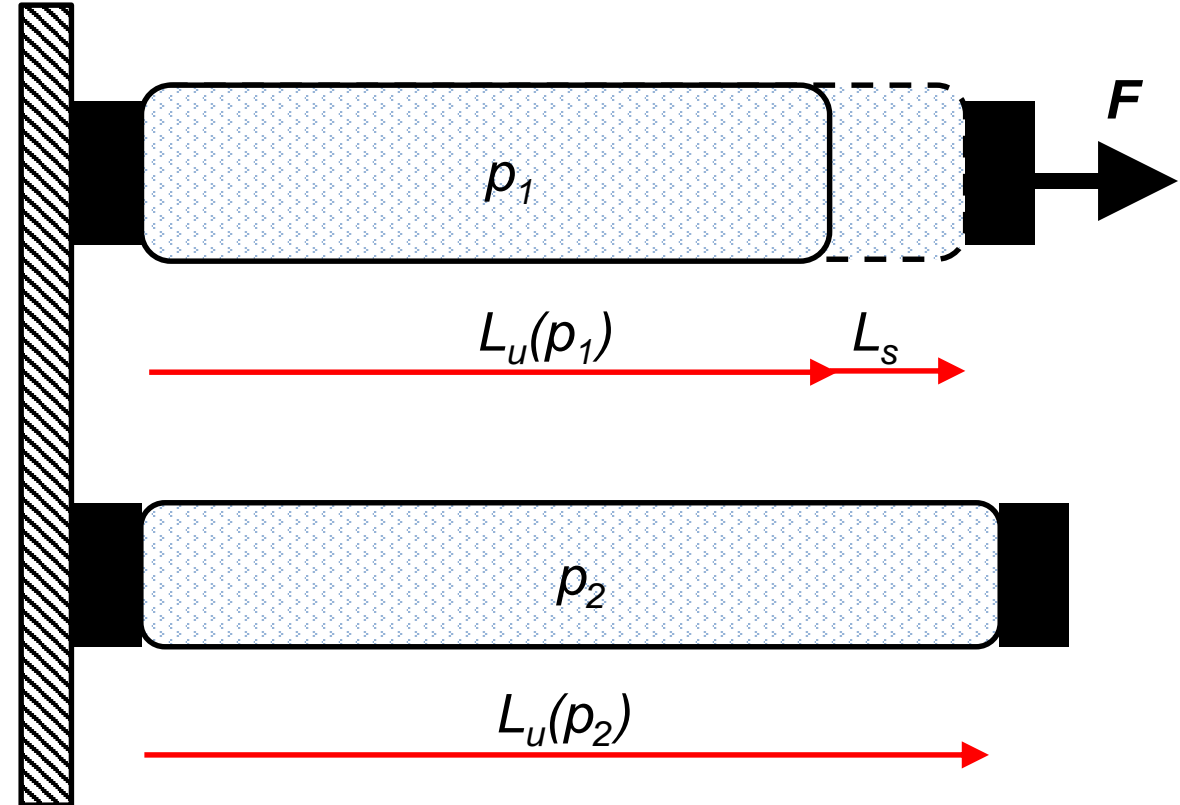
L_s = Additional stretch due to force, F

Assumptions:

- Volume is approximately constant
- Stretch force is proportional to L_s

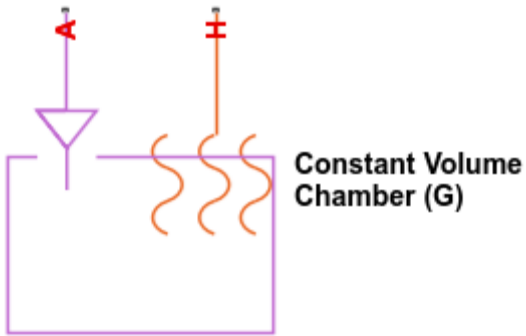
Equations:

- $L = L_u(p) + L_s$
- $F = k \times L_s$
- $pV = nRT$



Creating custom Simscape components

Step 2: Find starting point from foundation library



- Has equation of state
- Need to add mechanical ports & equations

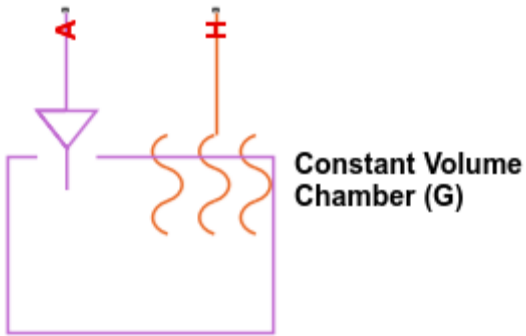
```

C:\Program Files\MATLAB\R2017a2\toolbox\physmod\simscaps\library\m\+foundation\+gas\+elements\constant_volume_chamber.ssc
EDITOR VIEW
New Open Save Find Files Compare Go To Comment % % % Breakpoints
Print Find Indent Breakpoints
FILE NAVIGATE EDIT BREAKPOINTS
1 component constant_volume_chamber < foundation.gas.one_port_vertical
2 % Constant Volume Chamber (G)
3 % This block models mass and energy storage in a gas network. The chamber
4 % contains a constant volume of gas. The pressure and temperature evolve
5 % based on the compressibility and thermal capacity of this gas volume.
6 %
7 % Port A is the gas conserving port associated with the chamber inlet. Port
8 % H is the thermal conserving port associated with the temperature of the
9 % gas inside the chamber.
10
11 % Copyright 2016 The MathWorks, Inc.
12
13 nodes
14     H = foundation.thermal.thermal; % H:top
15 end
16
17 parameters
18     volume = {0.001, 'm^3'}; % Chamber volume
19     area_A = {0.01, 'm^2'}; % Cross-sectional area at port A
20 end
21
Simscape model file Ln 1 Col 1

```

Creating custom Simscape components

Step 3: Incrementally add functionality



Add:

- Two mechanical ports

```

1  component air_muscle < foundation.gas.one_port_vertical
2  % Air Muscle (G)
3  % This block models a McKibben air muscle.
4
5  % Copyright 2016-2017 The MathWorks, Inc.
6
7  nodes
8  |   H = foundation.thermal.thermal; % H:top
9  |   R = foundation.mechanical.translational.translational; % R:bottom
10 |   C = foundation.mechanical.translational.translational; % C:top
11 end

```

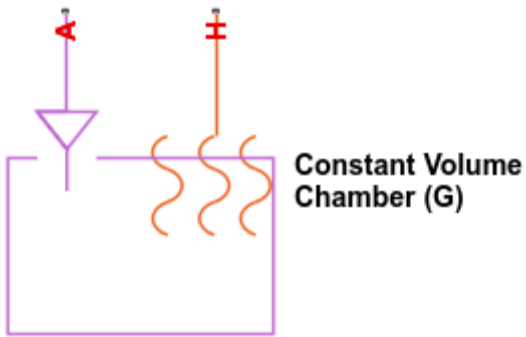
- Two additional new equations

$$L = L_u(p) + L_s \longrightarrow 152 \quad \mathbf{L} \quad == \quad \mathbf{Ls} + \mathbf{Lu};$$

$$F = k \times L_s \longrightarrow 153 \quad \mathbf{force} == \mathbf{K} * \mathbf{Ls};$$

Creating custom Simscape components

Step 3: Incrementally add functionality



Add definitions for:

- Variables
- Parameters

```

33 variables
34     % Mechanical variables
35     force = {0, 'N'}; % Force
36     Ls = {0, 'm'}; % Stretch
37 end

21 parameters
22     K = {140, 'N/cm'}; % Stiffness
23     pVec = {[0 1 2 3 4 5 6], 'bar'}; %
24     LuVec = {[30 27.3 25.1 23.5 22.3
25 end
  
```

Creating custom Simscape components

Step 4: Build library and run test model

Block Parameters: McKibben Air Muscle

Air Muscle (G)

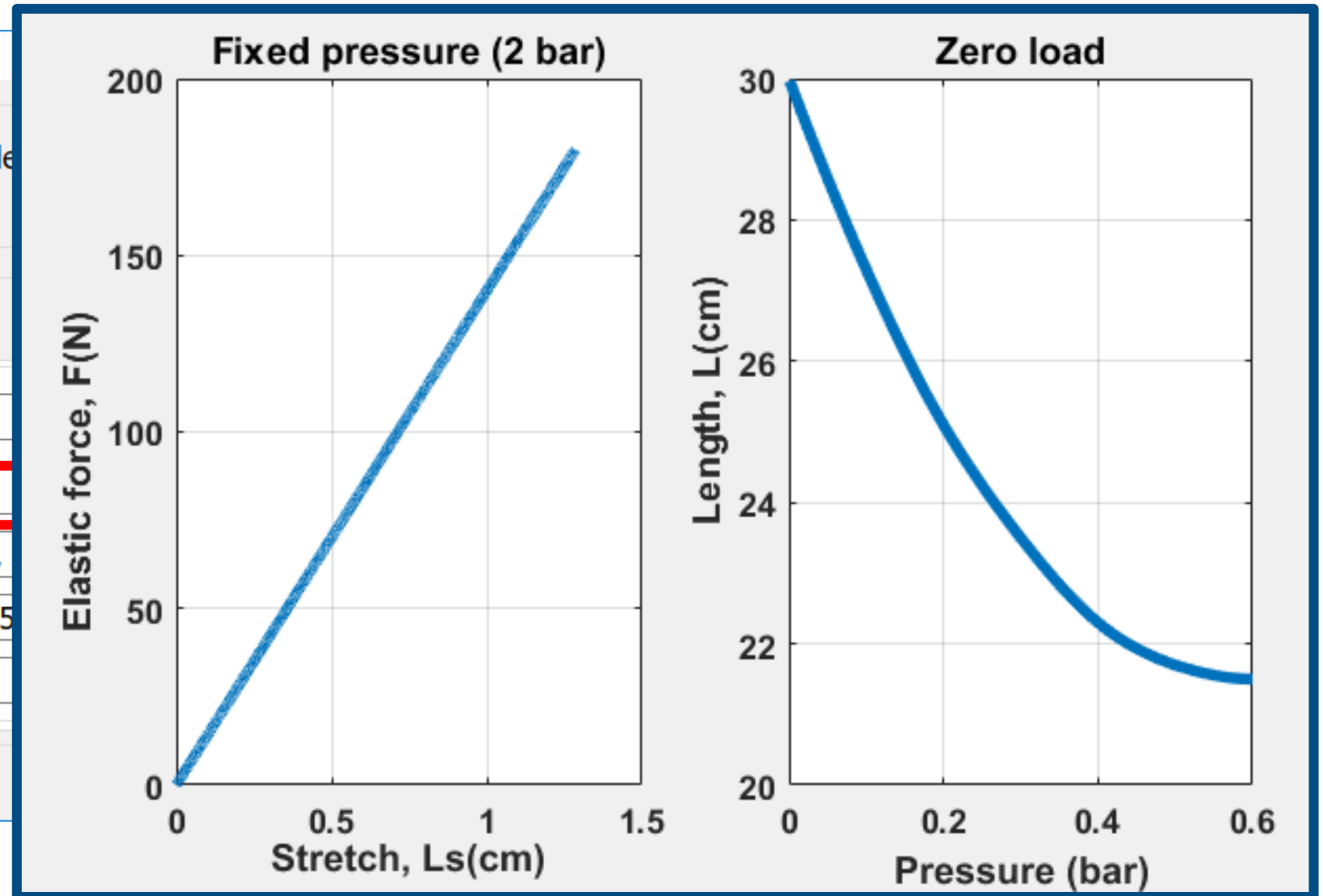
This block models a McKibben air muscle

[Source code](#)

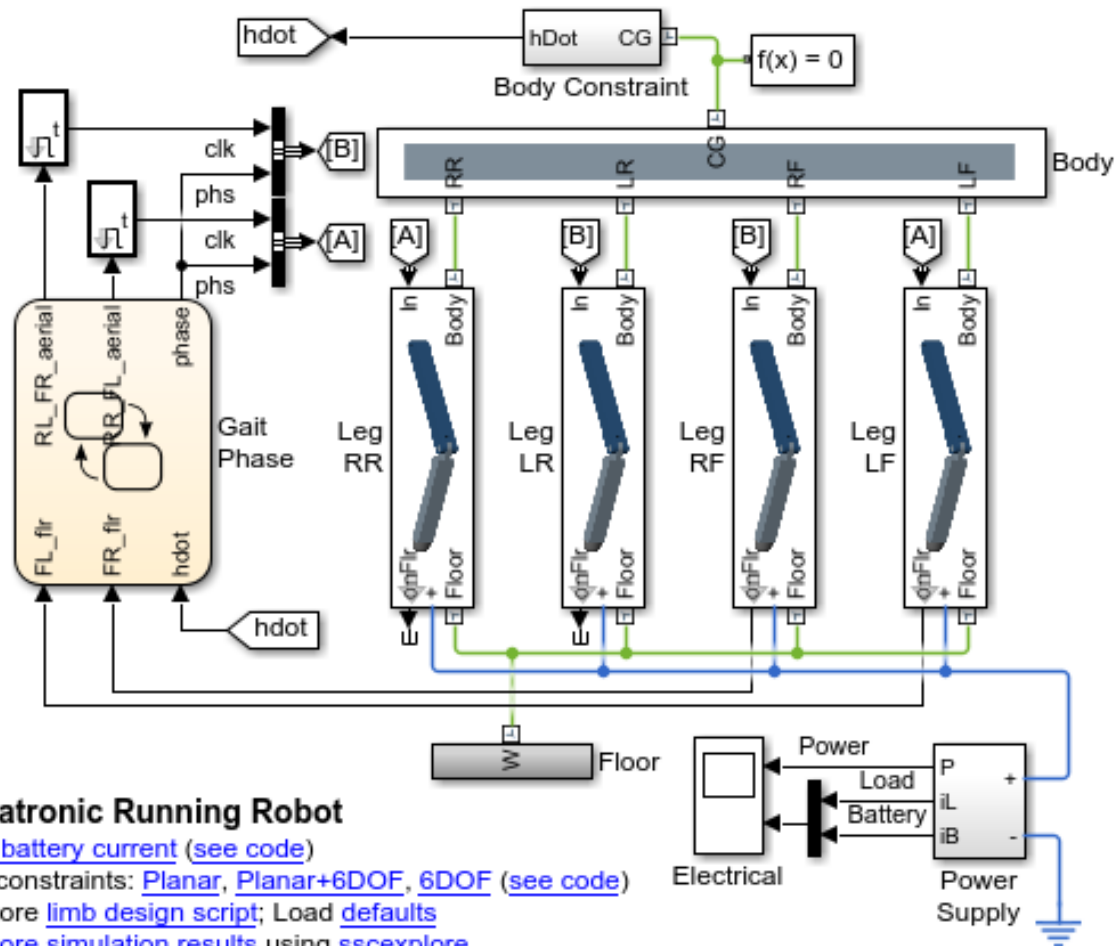
Settings

Parameters Variables

Cross-sectional area at port A:	0.01
Stiffness:	140
Pressures:	[0.0, 1.0, 2.0,
Unstretched lengths:	[7.3, 25.1, 23.5
Volume:	85

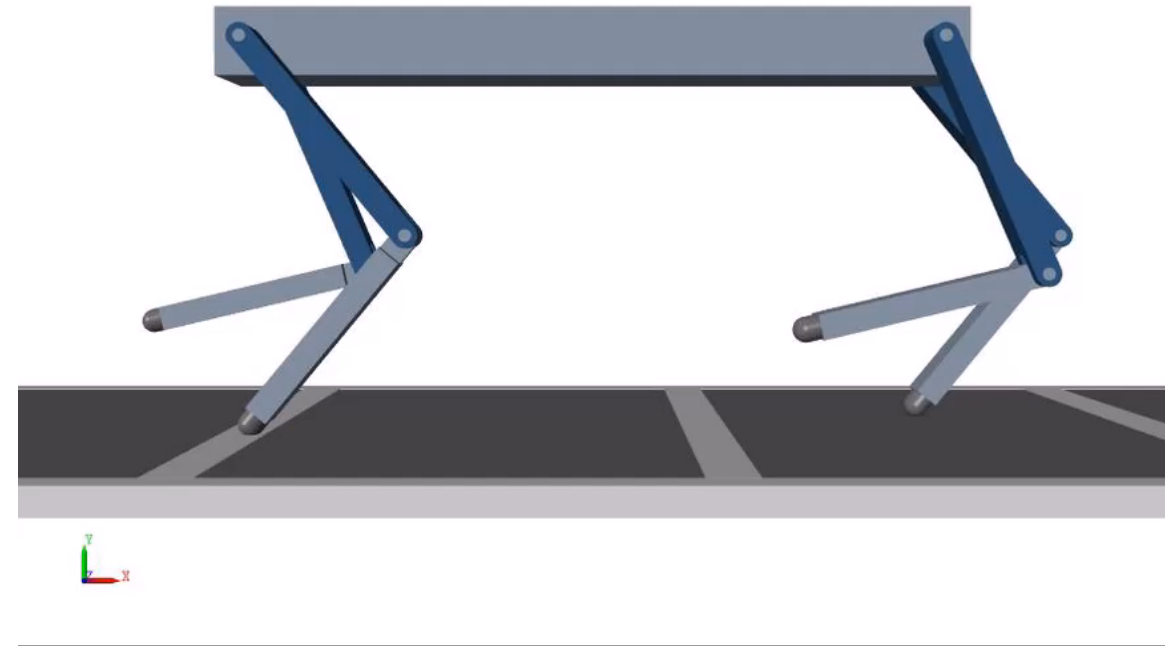


Why use Simscape?



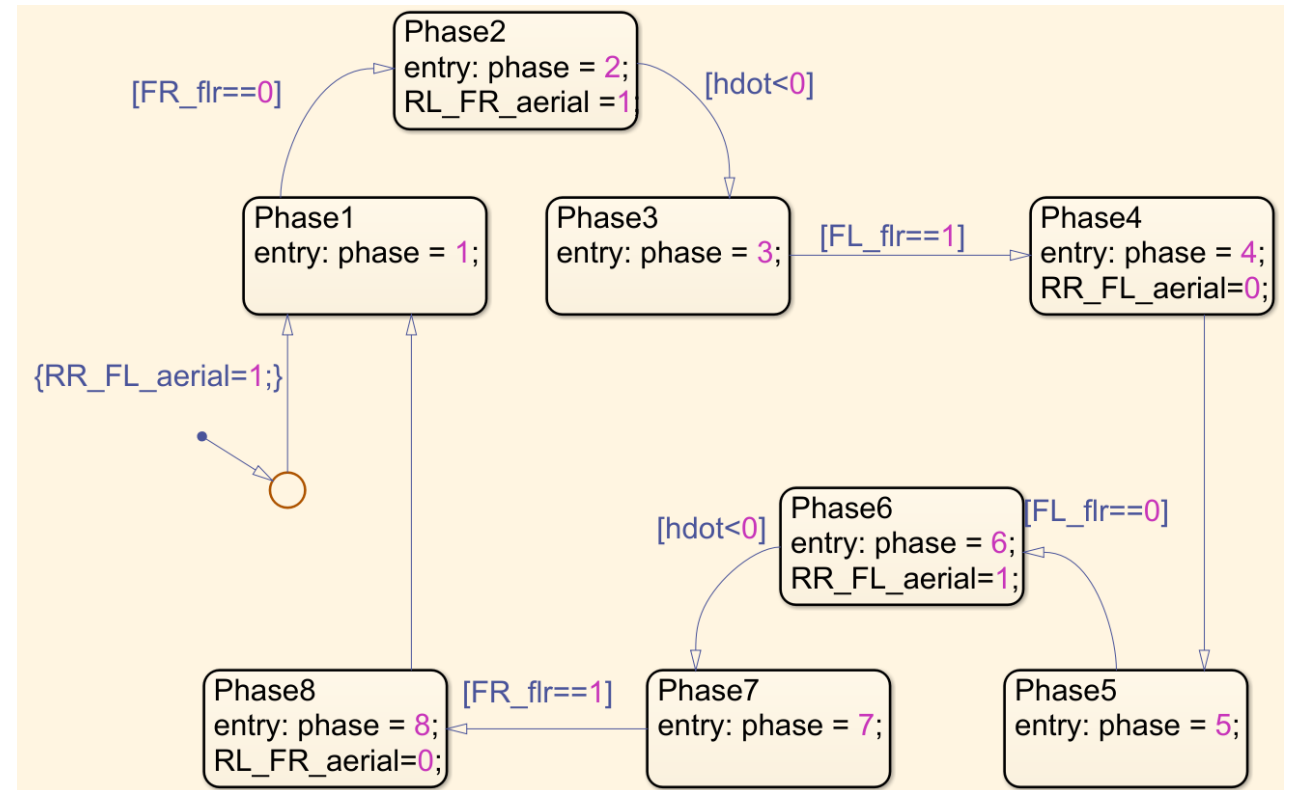
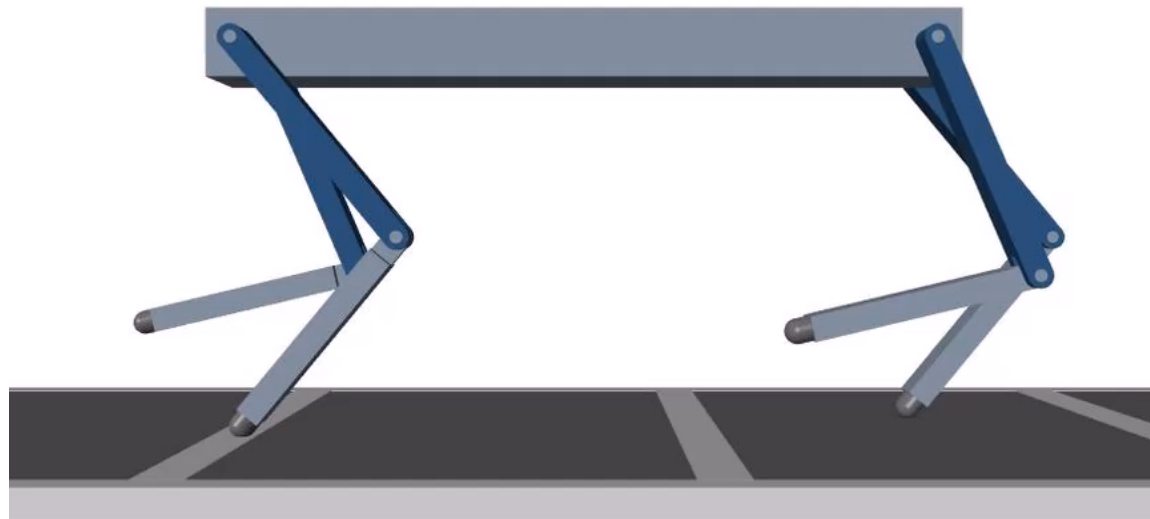
Mechatronic Running Robot

1. [Plot battery current](#) (see code)
2. Set constraints: [Planar](#), [Planar+6DOF](#), [6DOF](#) (see code)
3. Explore [limb design script](#); Load defaults
4. [Explore simulation results](#) using [sscxplore](#)
5. [Learn more](#) about this example



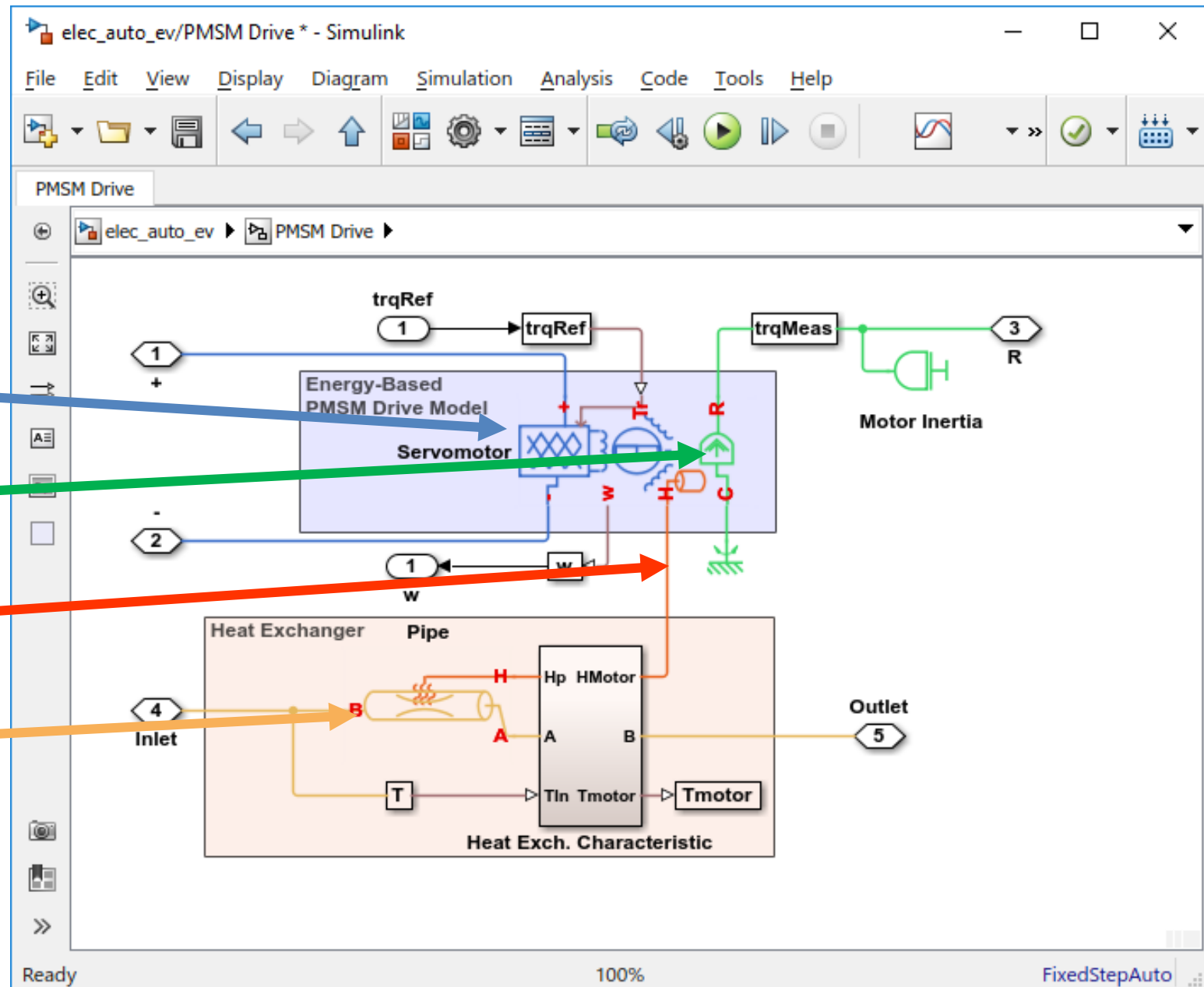
Why use Simscape?

- Plant and control



Why use Simscape?

- Plant and control
- **Multidomain**
 - Electrical
 - Mechanical
 - Thermal
 - Fluid

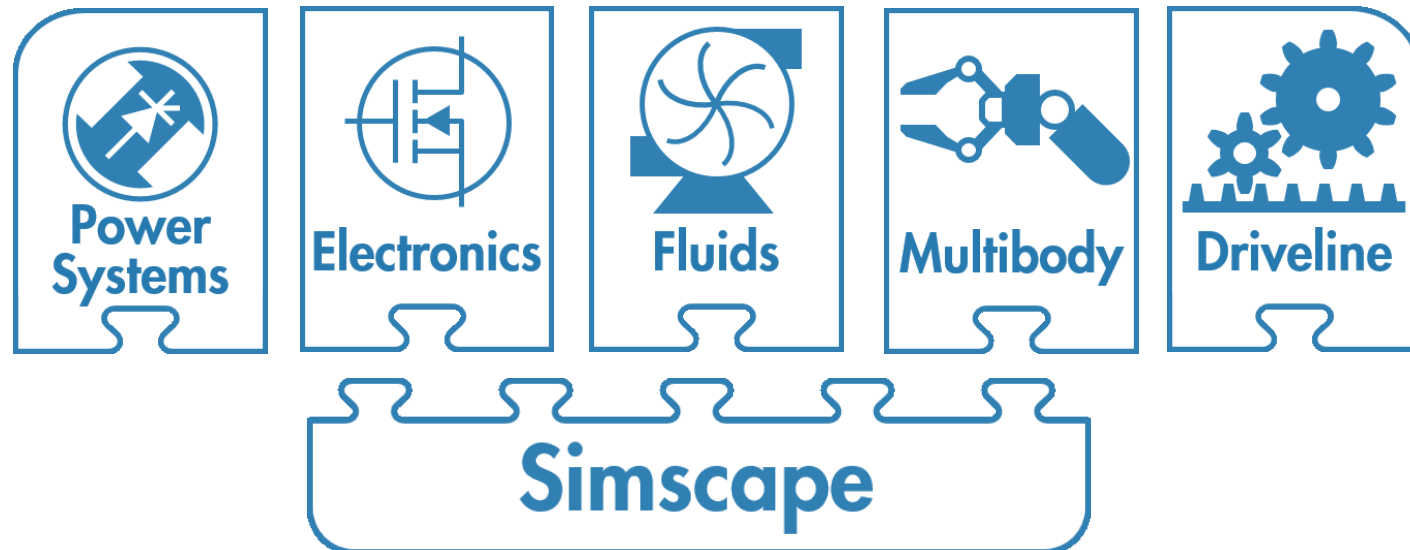


Why use Simscape?

- Plant and control
- Multidomain
- **Code generation and V&V tools**
 - Test controller on HIL plant
 - Deploy to simulator
 - Use plant model in real-time controller

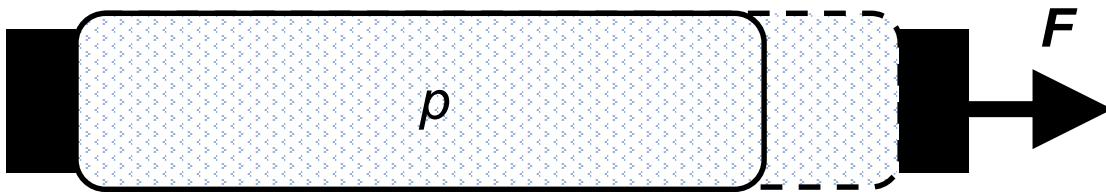
Why use Simscape?

- Plant and control
- Multidomain
- Code generation and V&V tools
- **Libraries, examples, documentation & webinars**



Why use Simscape?

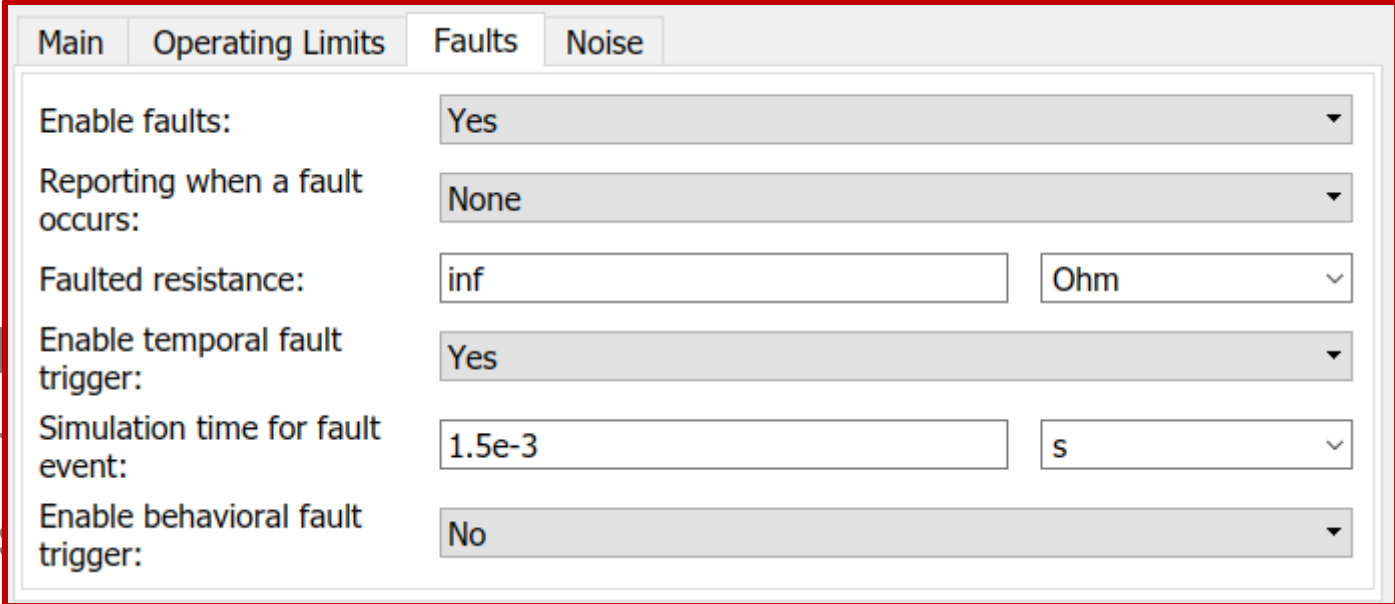
- Plant and control
- Multidomain
- Code generation and V&V tools
- Libraries, examples, documentation & webinars
- **Simscape language – build custom components**



```
21 parameters
22     K = {140, 'N/cm'}; % Stiffness
23     pVec = {[0 1 2 3 4 5 6] , 'bar'}; % Pressures
24     LuVec = {[30 27.3 25.1 23.5 22.3 21.7 21.5], 'cm'};
25 end
```

Why use Simscape?

- Plant and control
- Multidomain
- Code generation and V&V tool
- Libraries, examples, documents
- Simscape language – build custom models
- **Workflow**
 - **Tight integration with MathWorks control and optimization tools**
 - **MATLAB for scripting and automation**
 - **Fault-capable components (R, L, C, Servomotor, ...)**



The image shows a screenshot of the Simscape configuration panel, specifically the 'Faults' tab. The panel is divided into four sections: 'Main', 'Operating Limits', 'Faults', and 'Noise'. The 'Faults' section is currently selected and contains the following settings:

Parameter	Value
Enable faults:	Yes
Reporting when a fault occurs:	None
Faulted resistance:	inf Ohm
Enable temporal fault trigger:	Yes
Simulation time for fault event:	1.5e-3 s
Enable behavioral fault trigger:	No

Why use Simscape?

- Plant and control
- Multidomain
- Code generation and V&V tools
- Libraries, examples, documentation & webinars
- Simscape language – build custom components
- Workflow
 - Tight integration with MathWorks control and optimization tools
 - MATLAB for scripting and automation
 - Fault-capable components (R, L, C, Servomotor, ...)
- **Support, training, consulting**
- **MATLAB Central**

How to find out more

- MathWorks physical modelling page:
 - <https://www.mathworks.com/solutions/physical-modeling.html>
- Steve Miller's introduction video
 - <https://www.mathworks.com/videos/physical-modeling-introduction-75883.html>
- MATLAB Central File Exchange
 - <https://www.mathworks.com/matlabcentral/fileexchange/>
- Contact us direct

