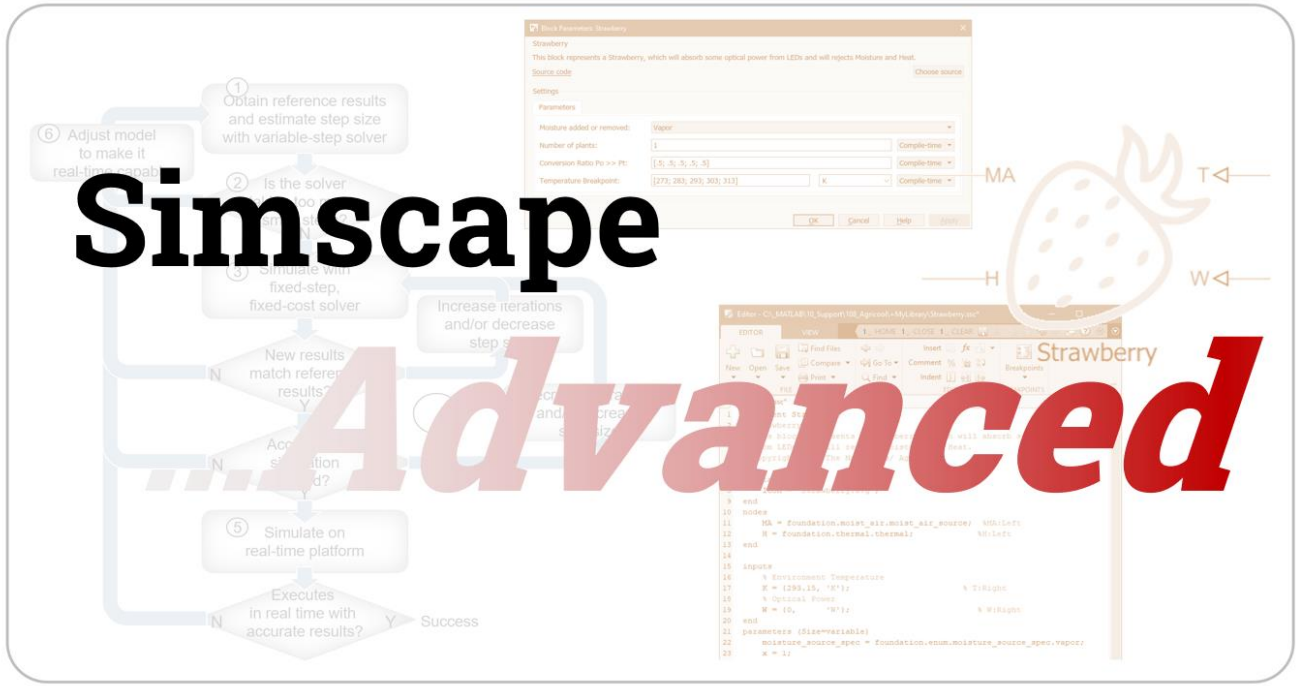
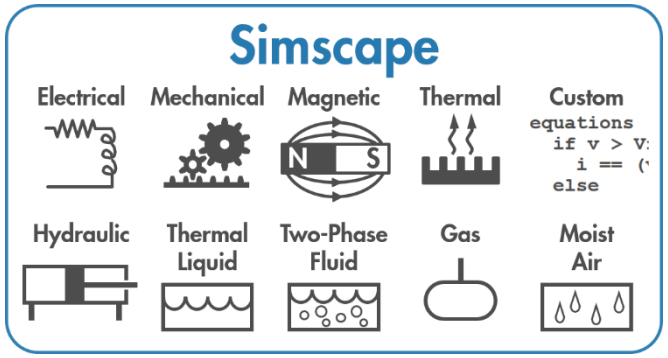
Simscape Advanced

10/01/2024



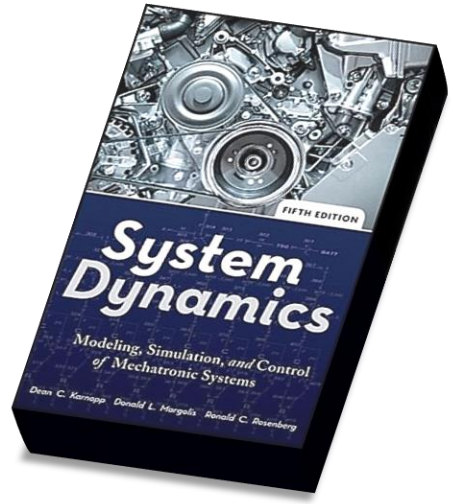
Kévin Roblet – Application Engineer

Simscape Principles



- Different type of physical domain
- For each domain, 2 types of variables: '**Through**' and '**Across**'
- **Through** x **Across** = Power (W)

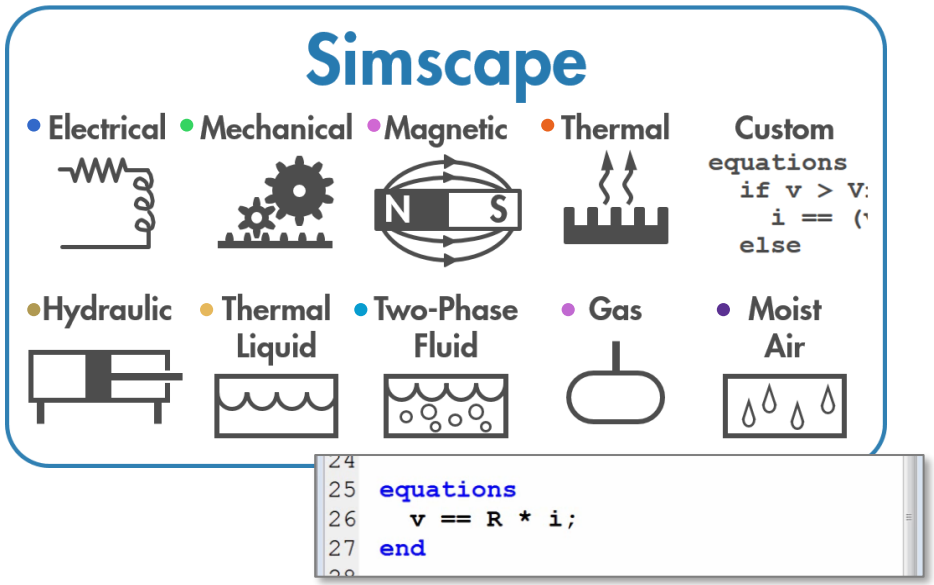
Domain	Through variable	Across variable
Electrical	Current (A)	Voltage (V)
Mechanical translation	Force (N)	Speed (m/s)
Mechanical rotation	Torque (N.m)	Speed (rad/s)
Hydraulics	Mass flow (m^3/s)	Pressure (Pa)
Thermal	Heat flow (J)	Temperature (°K)
…	…	…



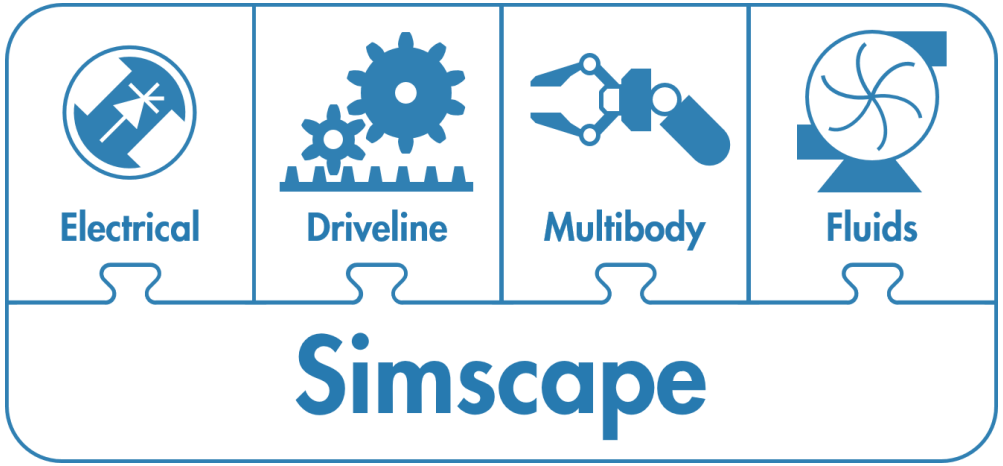
System Dynamics
Modeling, Simulation, and Control of Mechatronic Systems

Dean C. Karnopp Donald L. Margolis Ronald C. Rosenberg

Simscape Foundation



Add-on libraries

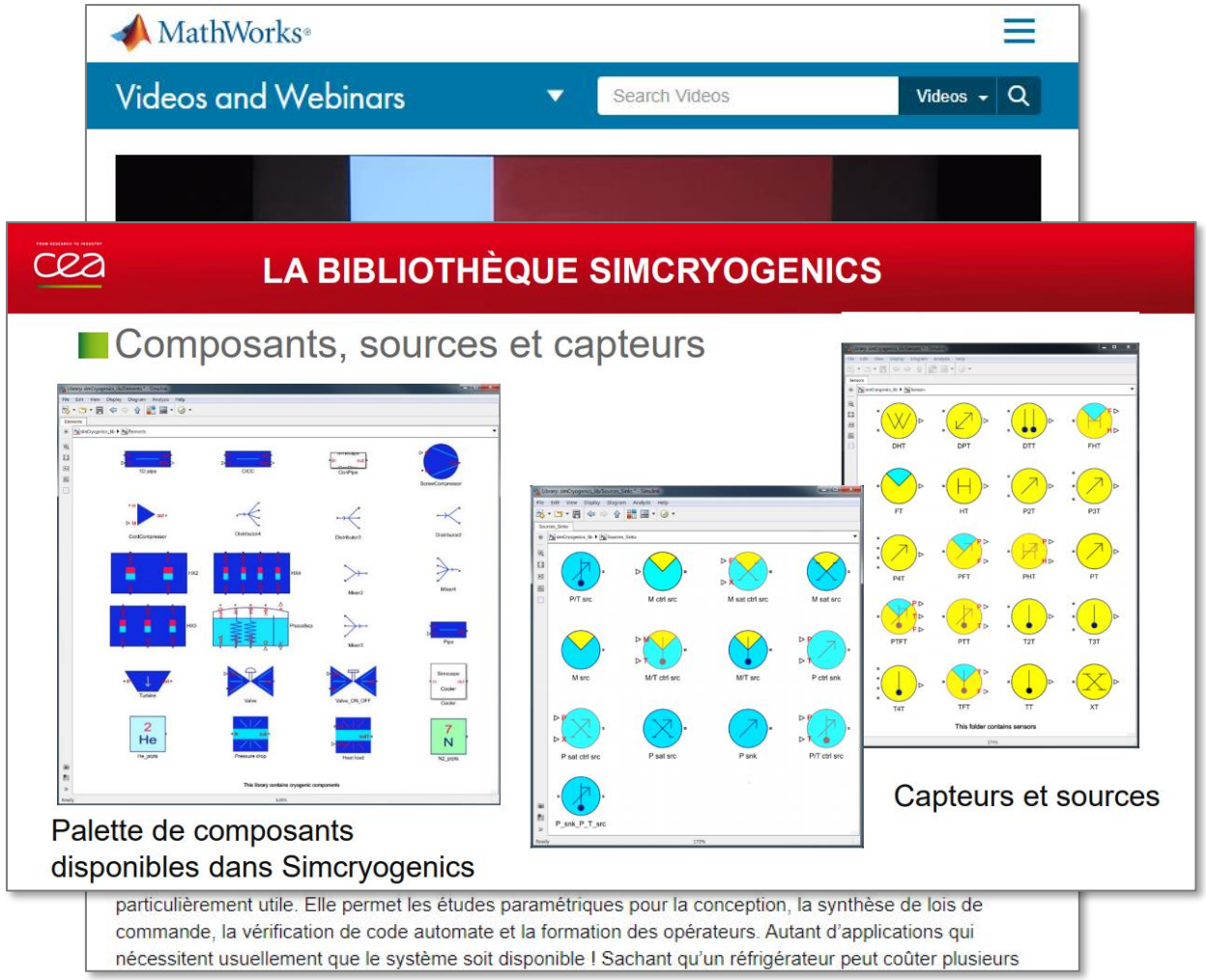


Option 1: Custom Components

Option 2: Ready-to-use Component

What is the Simscape Language?

- MATLAB-based, object-oriented physical modeling language for use in the Simulink Environment
- It enables engineers to:
  - Build physical modeling custom component in the provided physical domains
  - Define new domains and create custom components for the same
  - Build custom block libraries ➔



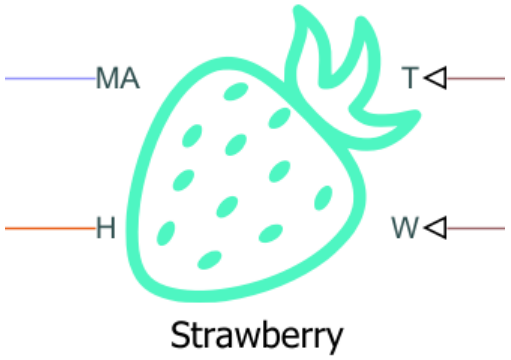
More info: https://www.mathworks.com/videos/cryogenic-systems

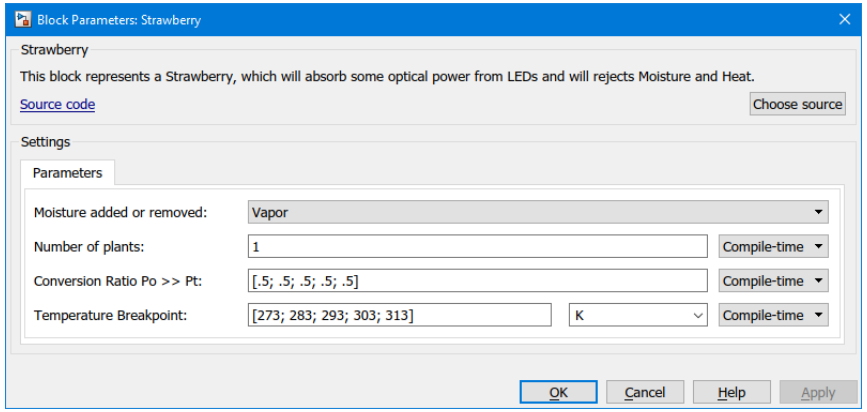
Component File Overview

Language Syntax

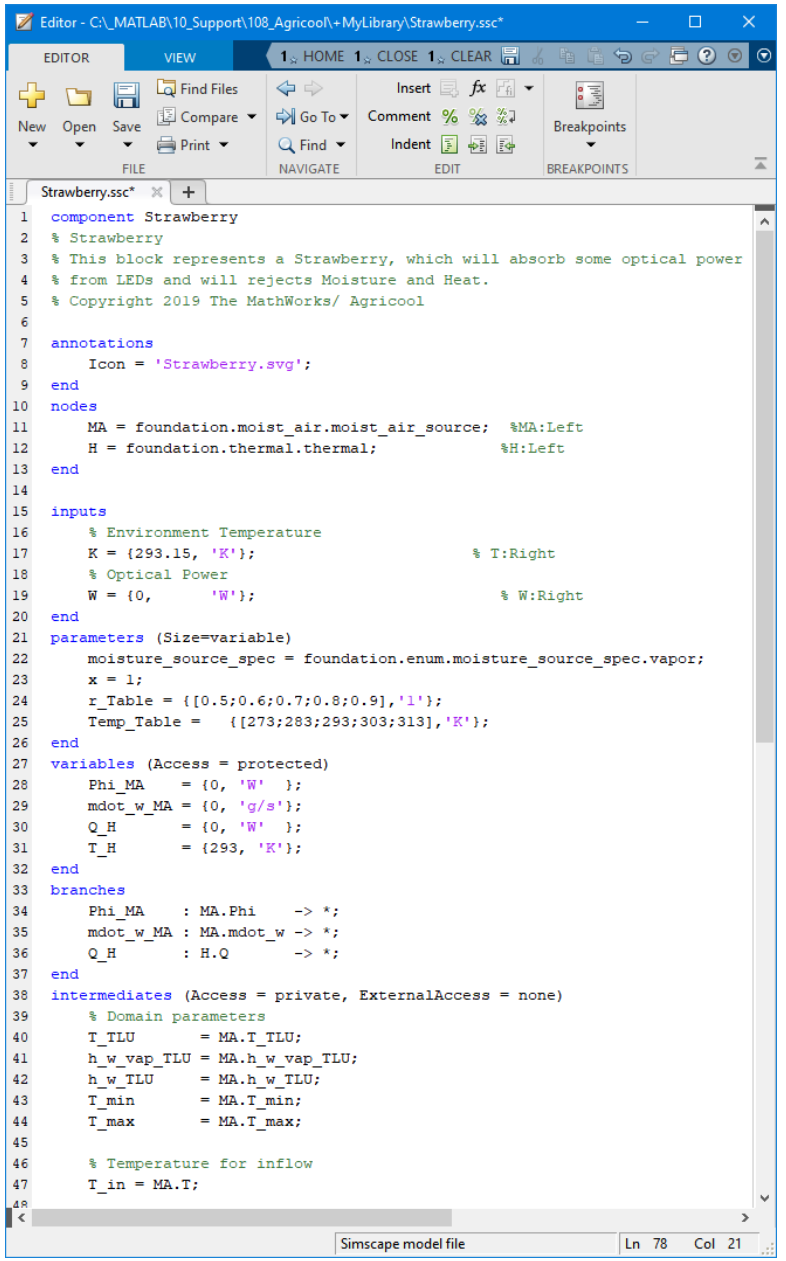
- annotations
- inputs
- outputs
- nodes
- parameters
- variables
- components
- setup
- branches
- connections
- equations
- intermediates

- assert
- function
- events
- edge





Automatic UI generation



Component File Overview

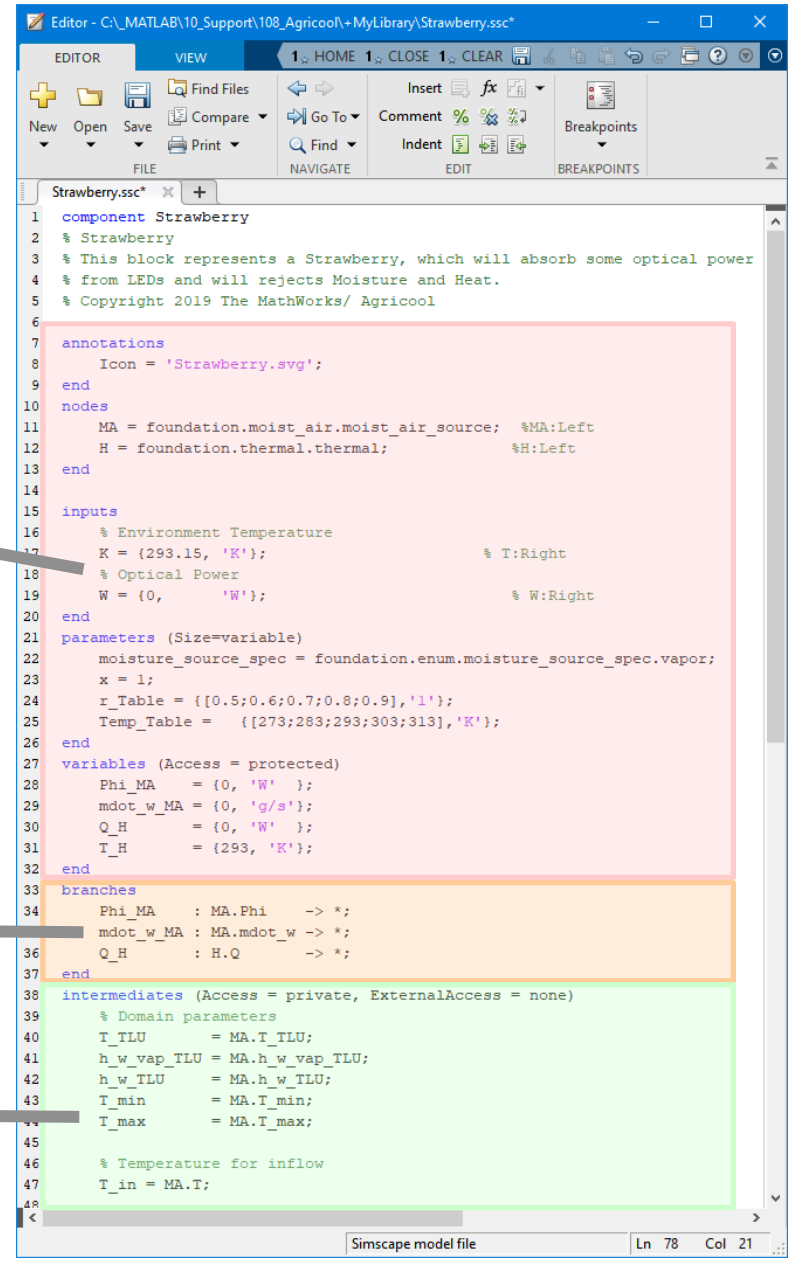
Component Model

Declaration
- Nodes
- Inputs, Outputs
- Variables
  - (through, across, and internal)
- Parameters

Setup
- Parameter checking
- Derived parameters

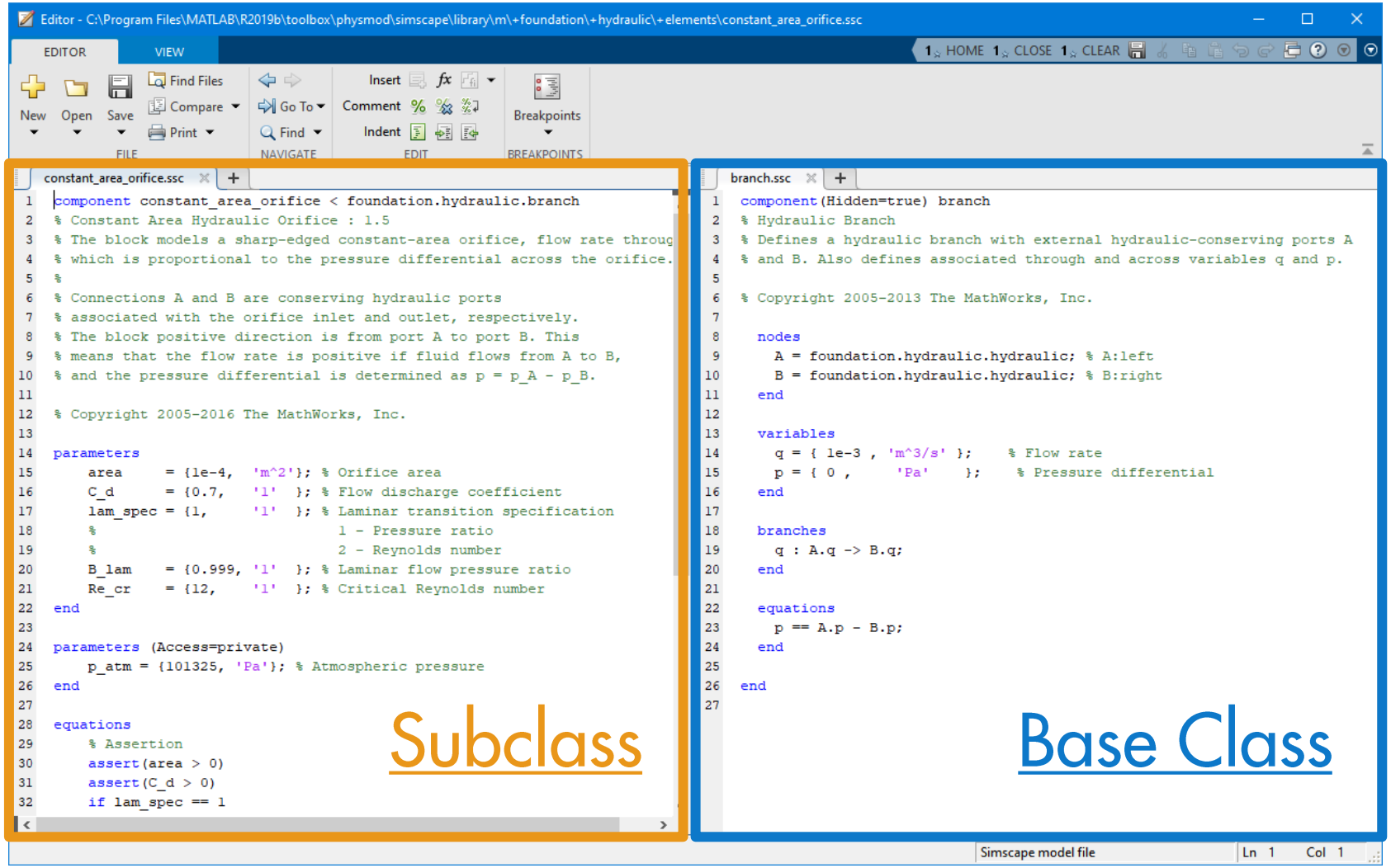
Branches
- Defining relationship between through variables and nodes

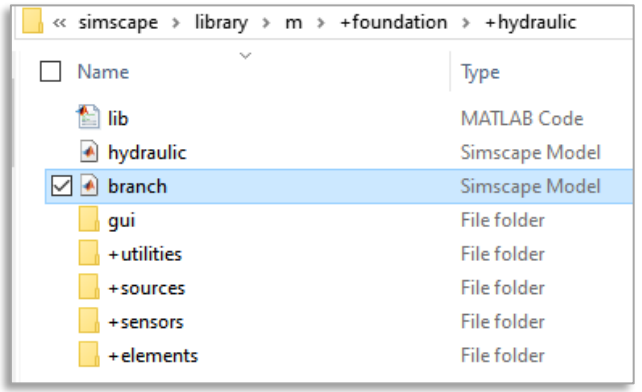
Equations
- Algebraic, discontinuous, differential



Object-Oriented Modeling

```
component constant_area_orifice < foundation.hydraulic.branch
```





➔ Subclass inherits all of the members from base class

➔ Equations of both are included in the overall system

Lumped Parameter Model

Transmission Line (Three-Phase)

Three-phase transmission line using lumped-parameter pi-section line model

expand all in page

Library: Simscape / Electrical / Passive / Lines

Flexible Shaft

Shaft with torsional and bending compliance

expand all in page

Library: Simscape / Driveline / Couplings & Drives

Segmented Pipe LP

Hydraulic pipeline with resistive, fluid inertia, fluid compressibility, and elevation properties

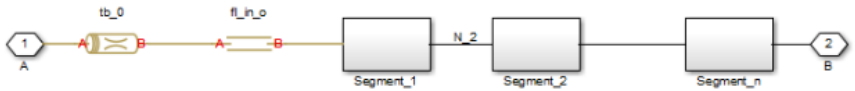
Library

Low-Pressure Blocks

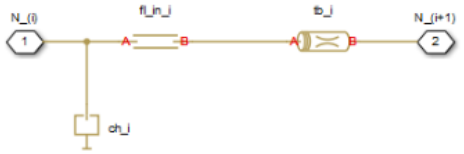
Description

The Segmented Pipe LP block models hydraulic pipelines with circular cross sections. Hydraulic pipelines, which are inherently distributed parameter elements, are represented with sets of identical, connected in series, lumped parameter segments. It is assumed that the larger the number of segments, the closer the lumped parameter model becomes to its distributed parameter counterpart. The equivalent circuit of a pipeline adopted in the block is shown below, along with the segment configuration.

Pipeline Equivalent Circuit



Segment Configuration



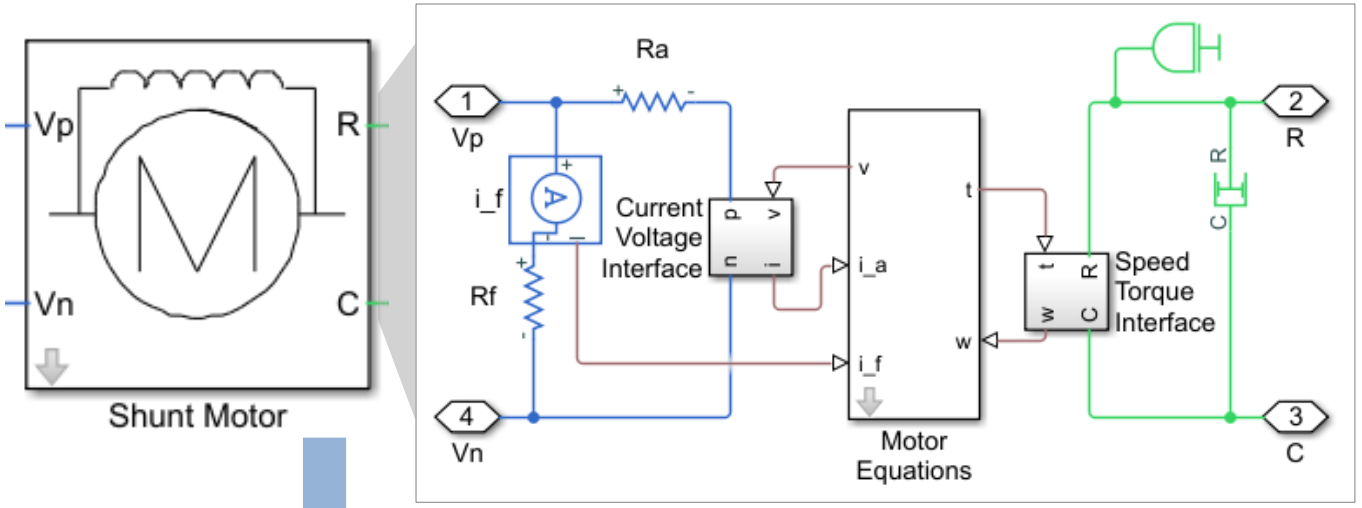
The model contains as many Constant Volume Hydraulic Chamber blocks as there are segments. The chamber lumps fluid volume equal to

Define components using resizable arrays of elements

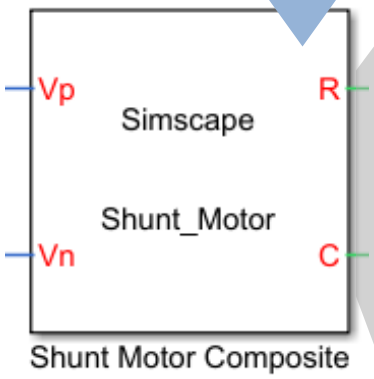
```
import foundation.isothermal_liquid.*;
component SegmentedPipeline
    parameters
        N = 10;                 % Number of segments
        segm_length = { 5, 'm' };  % Length of each segment
    end
    % Ports at the two ends of the pipeline
    nodes
        A = isothermal_liquid; % A:left
        B = isothermal_liquid; % B:right
    end
    % Declare array of N components
    for i=1:N
        components (ExternalAccess=none)
            pipe(i) = elements.pipe(length = segm_length);
        end
    end
    % Connect all segments in series
    for i=1:(N-1)
        connections
            connect(pipe(i).B, pipe(i+1).A);
        end
    end
    % Connect two ends of pipeline to first and last segment
    connections
        connect(A, pipe(1).A);
        connect(B, pipe(N).B);
    end
end
```

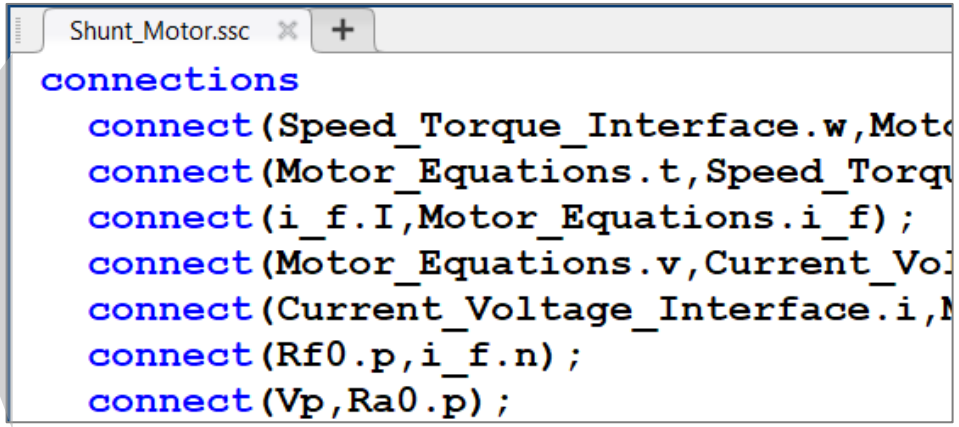
`subsystem2ssc` Function

- Convert subsystem assembled from Simscape blocks to a Simscape Language file
  - Parameters promoted automatically
  - Can be hierarchical
  - Resulting file can be converted to a binary using **`ssc_protect`**



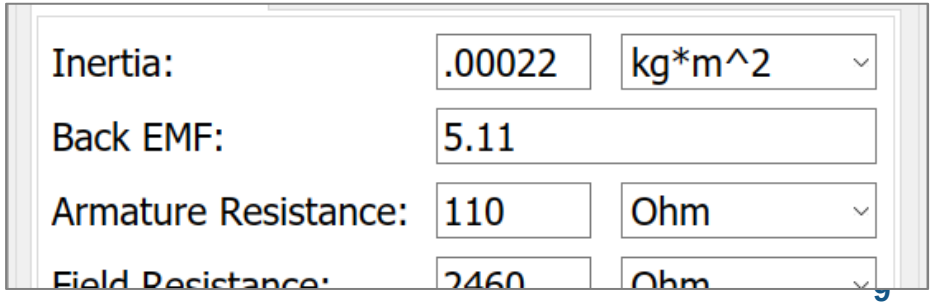
`subsystem2ssc(<subsystem>)`





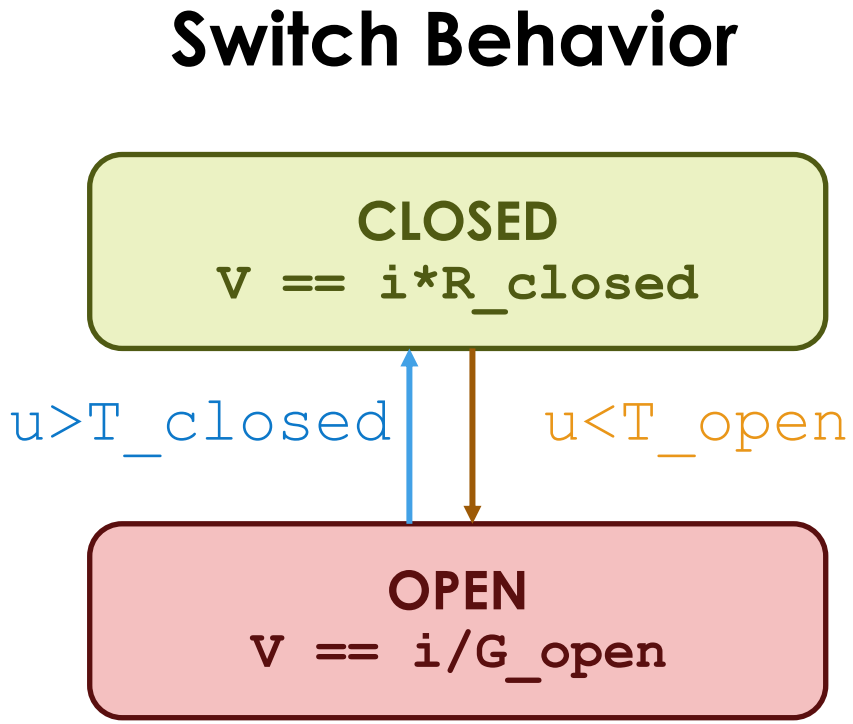
`ssc_protect()`

`Shunt_Motor.sscp`



Discrete Changes Behavior

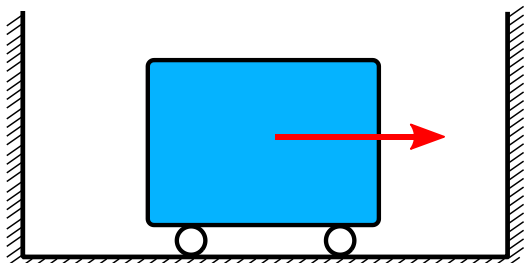
- Use Event or Mode Chart Modeling

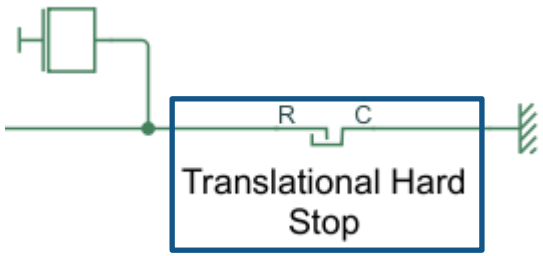


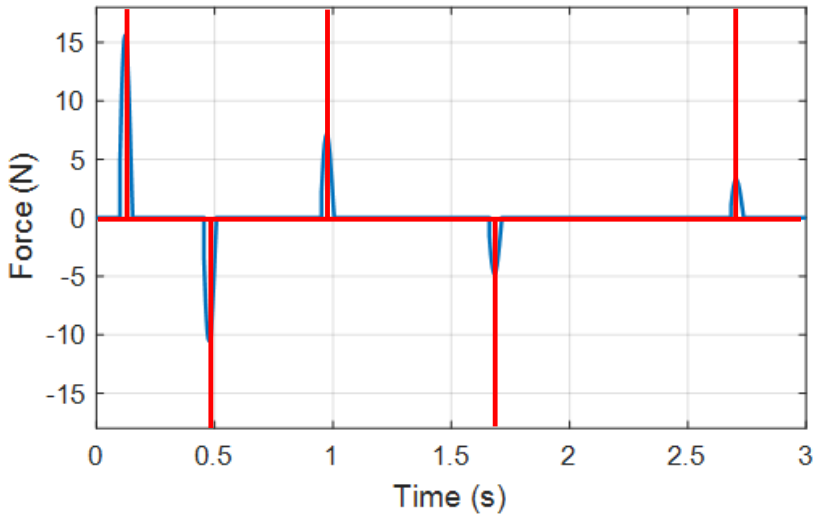

```
modecharts(ExternalAccess = observe)
  m1 = modechart
    modes
      mode CLOSED
        equations
          v == i*R_closed;
        end
      end
      mode OPEN
        equations
          v == i/G_open;
        end
      end
    end
    transitions
        CLOSED -> OPEN : u < T_open;
        OPEN -> CLOSED : u > T_closed;
    end
    initial
        OPEN : InitMode <= 0;
    end
end
```

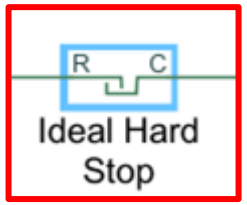
State reset at events

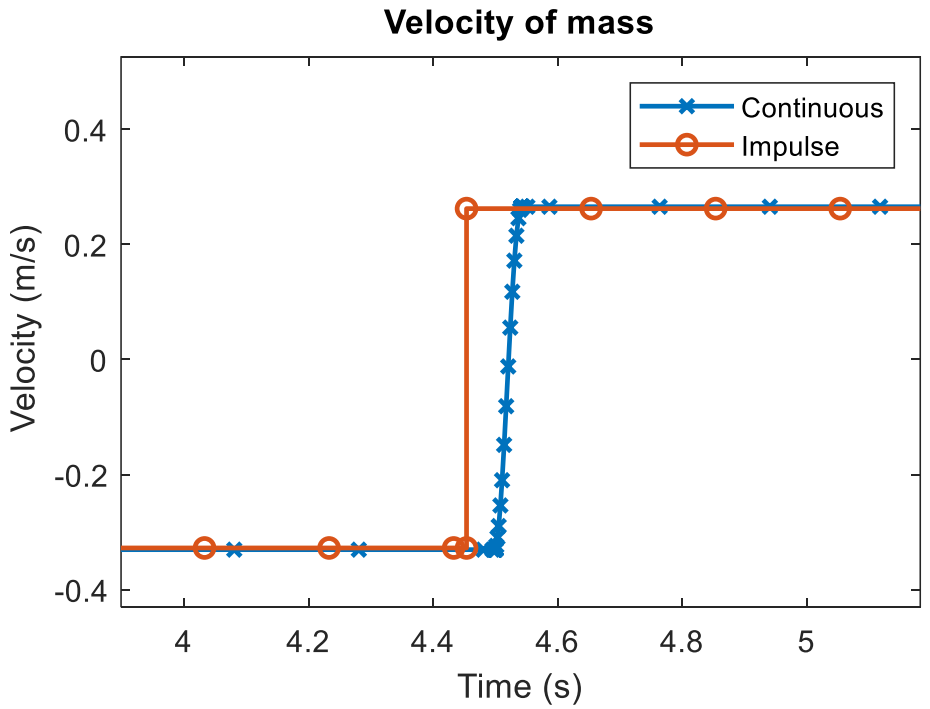
- Reinitialize state variables at events
- Model physical phenomena such as collisions or clutches engaging











Continuous – **407** time steps
Impulse-based – **63** time steps