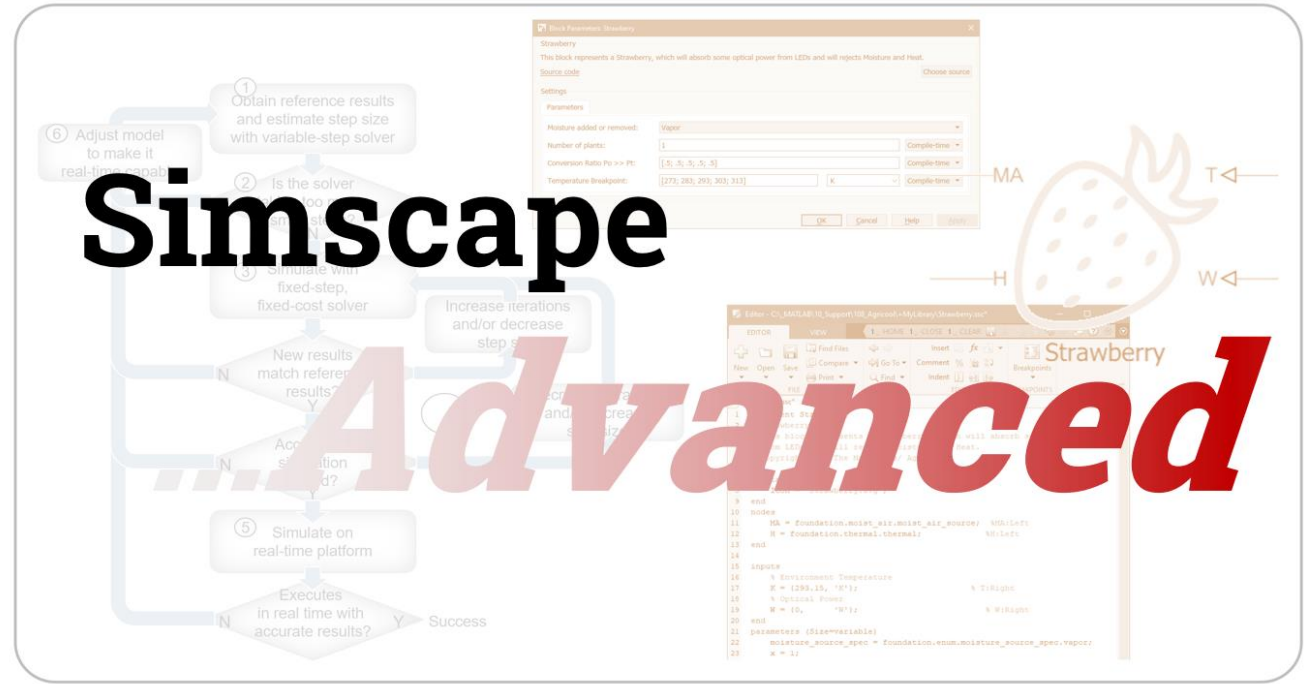


Simscape Advanced

10/01/2024

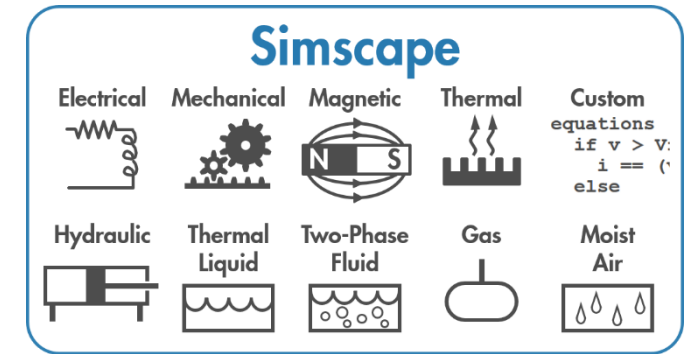
Kévin Roblet

- Application Engineer

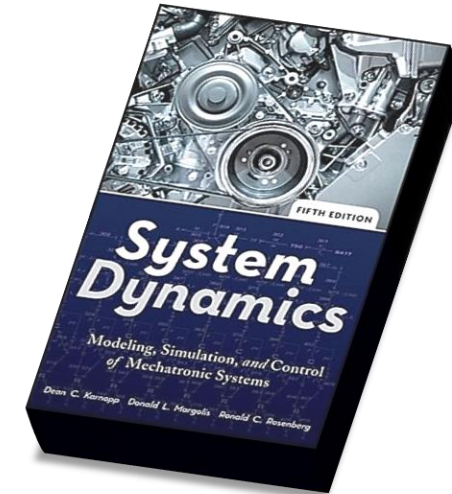


Simscape Principles

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



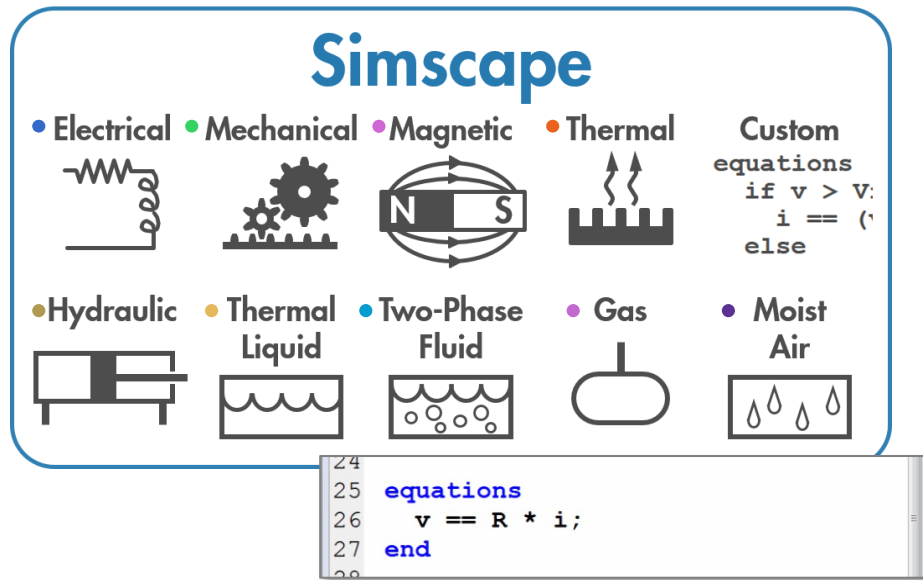
Domain	Through variable	Across variable
Electrical	Current (A)	Voltage (V)
Mechanical translation	Force (N)	Speed (m/s)
Mechanical rotation	Torque (N.m)	Speed (rad/s)
Hydraulics	Mass flow (m ³ /s)	Pressure (Pa)
Thermal	Heat flow (J)	Temperature (°K)
...



System Dynamics
 Modeling, Simulation, and Control of
 Mechatronic Systems

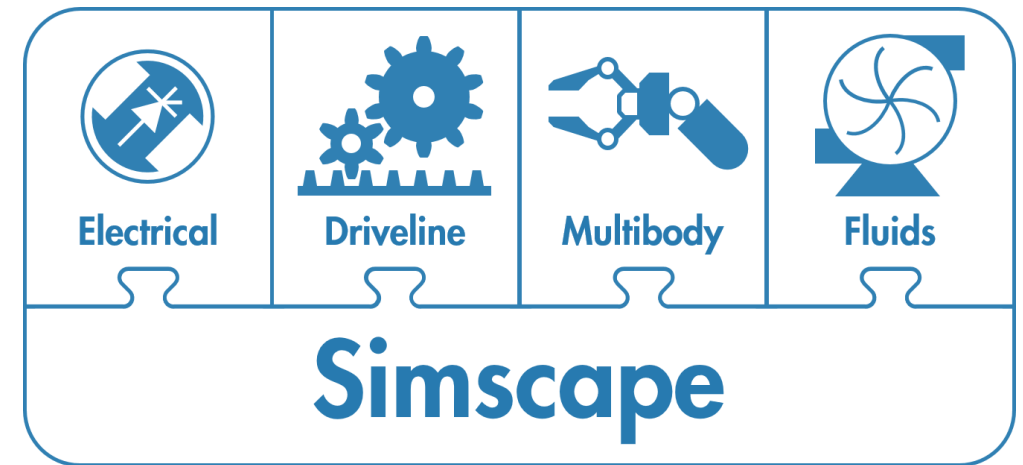
Dean C. Karnopp Donald L. Margolis Ronald C. Rosenberg

Simscape Foundation



Option 1: Custom Components

Add-on libraries



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 ➔

MathWorks®

Videos and Webinars

Search Videos

Videos

CEA

LA BIBLIOTHÈQUE SIMCRYOGENICS

Composants, sources et capteurs

Capteurs et sources

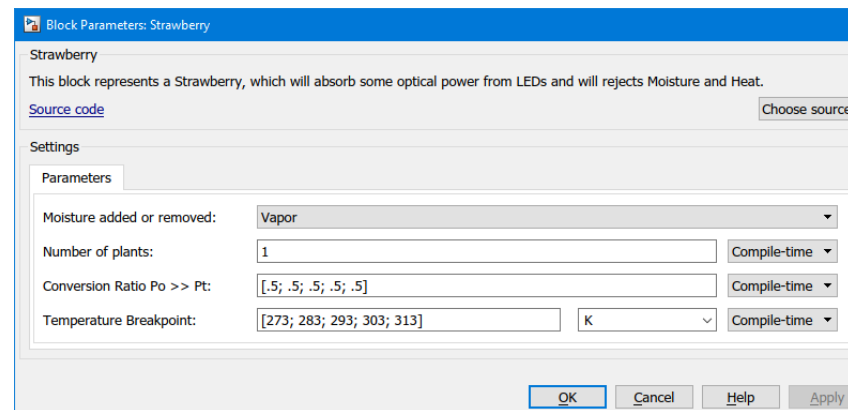
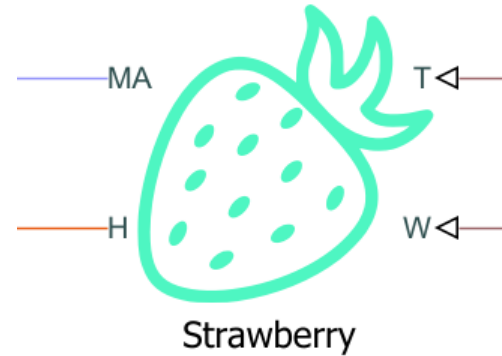
particulièrement utile. Elle permet les études paramétriques pour la conception, la synthèse de lois de commande, la vérification de code automate et la formation des opérateurs. Autant d'applications qui nécessitent usuellement que le système soit disponible ! Sachant qu'un réfrigérateur peut coûter plusieurs

Component File Overview

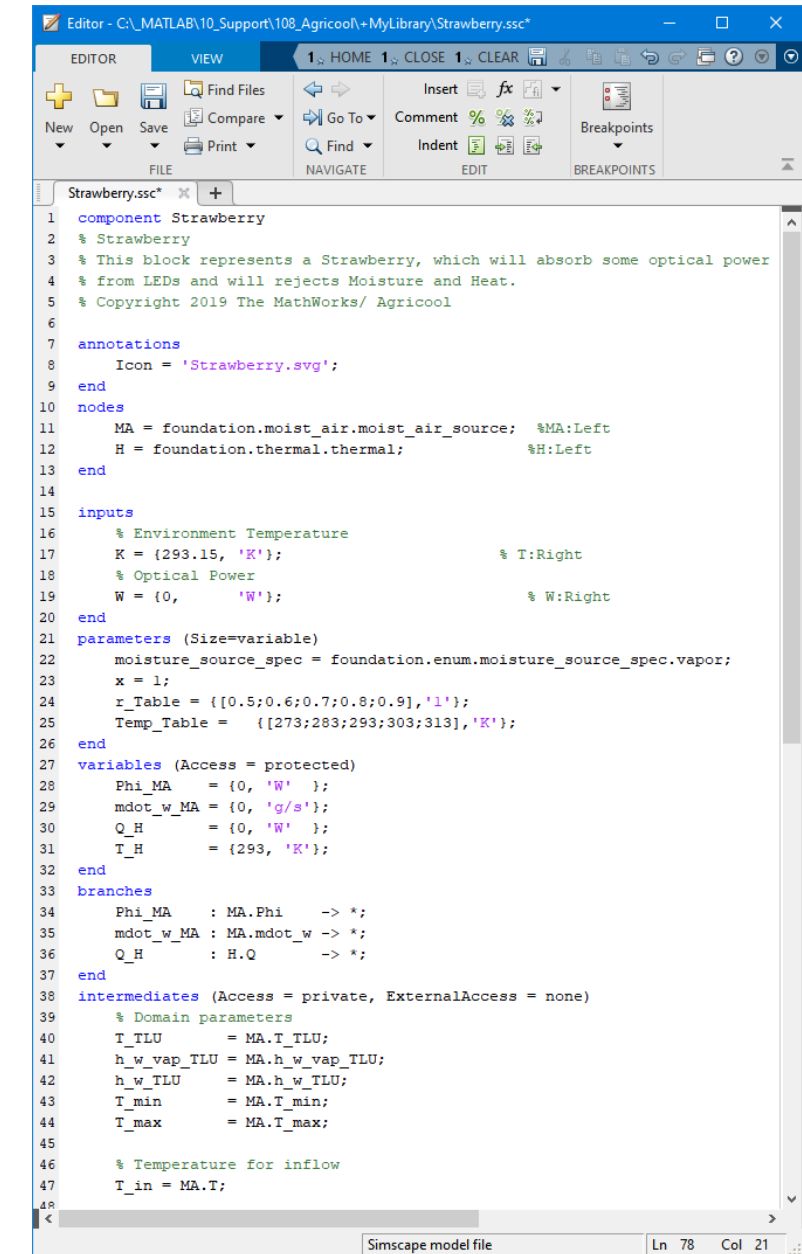
Language Syntax

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

```

1 component Strawberry
2 % Strawberry
3 % This block represents a Strawberry, which will absorb some optical power
4 % from LEDs and will reject Moisture and Heat.
5 % Copyright 2019 The MathWorks/ Agricoool
6
7 annotations
8     Icon = 'Strawberry.svg';
9 end
10 nodes
11     MA = foundation.moist_air.moist_air_source; %MA:Left
12     H = foundation.thermal.thermal; %H:Left
13 end
14
15 inputs
16     % Environment Temperature
17     K = {293.15, 'K'}; % T:Right
18     % Optical Power
19     W = {0, 'W'}; % W:Right
20 end
21 parameters (Size=variable)
22     moisture_source_spec = foundation.enum.moisture_source_spec.vapor;
23     x = 1;
24     r_Table = {[0.5;0.6;0.7;0.8;0.9], '1'};
25     Temp_Table = {[273;283;293;303;313], 'K'};
26 end
27 variables (Access = protected)
28     Phi_MA = {0, 'W'};
29     mdot_w_MA = {0, 'g/s'};
30     Q_H = {0, 'W'};
31     T_H = {293, 'K'};
32 end
33 branches
34     Phi_MA : MA.Phi -> *;
35     mdot_w_MA : MA.mdot_w -> *;
36     Q_H : H.Q -> *;
37 end
38 intermediates (Access = private, ExternalAccess = none)
39     % Domain parameters
40     T_TLU = MA.T_TLU;
41     h_w_vap_TLU = MA.h_w_vap_TLU;
42     h_w_TLU = MA.h_w_TLU;
43     T_min = MA.T_min;
44     T_max = MA.T_max;
45
46     % Temperature for inflow
47     T_in = MA.T;
48
  
```

Object-Oriented Modeling

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

The screenshot shows the MATLAB Editor with two files open. The left file, `constant_area_orifice.ssc`, is the subclass. It starts with the line `component constant_area_orifice < foundation.hydraulic.branch`, indicating inheritance. The right file, `branch.ssc`, is the base class. It defines the `branch` component with nodes, variables, branches, and equations. The subclass file contains specific parameters and assertions for the orifice model.

```

1 component constant_area_orifice < foundation.hydraulic.branch
2 % Constant Area Hydraulic Orifice : 1.5
3 % The block models a sharp-edged constant-area orifice, flow rate through
4 % which is proportional to the pressure differential across the orifice.
5 %
6 % Connections A and B are conserving hydraulic ports
7 % associated with the orifice inlet and outlet, respectively.
8 % The block positive direction is from port A to port B. This
9 % means that the flow rate is positive if fluid flows from A to B,
10 % and the pressure differential is determined as p = p_A - p_B.
11 %
12 % Copyright 2005-2016 The MathWorks, Inc.
13
14 parameters
15     area      = {1e-4, 'm^2'}; % Orifice area
16     C_d       = {0.7, '1'}; % Flow discharge coefficient
17     lam_spec  = {1, '1'}; % Laminar transition specification
18     % 1 - Pressure ratio
19     % 2 - Reynolds number
20     B_lam     = {0.999, '1'}; % Laminar flow pressure ratio
21     Re_cr     = {12, '1'}; % Critical Reynolds number
22 end
23
24 parameters (Access=private)
25     p_atm = {101325, 'Pa'}; % Atmospheric pressure
26 end
27
28 equations
29     % Assertion
30     assert(area > 0)
31     assert(C_d > 0)
32     if lam_spec == 1

```

Subclass

```

1 component (Hidden=true) branch
2 % Hydraulic Branch
3 % Defines a hydraulic branch with external hydraulic-conserving ports A
4 % and B. Also defines associated through and across variables q and p.
5 %
6 % Copyright 2005-2013 The MathWorks, Inc.
7
8 nodes
9     A = foundation.hydraulic.hydraulic; % A:left
10    B = foundation.hydraulic.hydraulic; % B:right
11 end
12
13 variables
14     q = { 1e-3, 'm^3/s' }; % Flow rate
15     p = { 0, 'Pa' }; % Pressure differential
16 end
17
18 branches
19     q : A.q -> B.q;
20 end
21
22 equations
23     p == A.p - B.p;
24 end
25
26 end
27

```

Base Class

Name	Type
lib	MATLAB Code
hydraulic	Simscape Model
<input checked="" type="checkbox"/> branch	Simscape Model
gui	File folder
+utilities	File folder
+sources	File folder
+sensors	File folder
+elements	File folder

➔ 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](#)

Describe

Library: Simscape / Driveline / Couplings & Drives



The Trans-
line model
resistance.

Torsion Mk

Segmented Pipe LP

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

The figure

For the torsic
lumped mass
a final inertia

Library

Low-Pressure Blocks

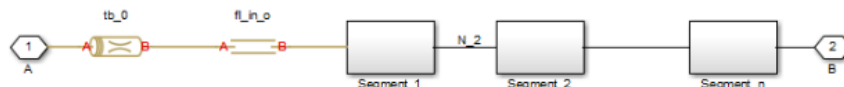


Description

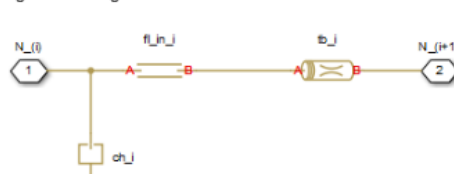
The Segmented Pipe LP block models hydraulic pipelines with circular cross sections. Hydraulic pipelines, which are inherently distributed parameter elements, are represented with sets of identical, connected in series, lumped parameter segments. It is assumed that the larger the number of segments, 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

Torsic
Eqi
Physica
Flexibl



Segment Configuration



To increas
N-segmen

To improvi
values for

Ports

- One spring
- One dam
- Two inert
- consolida

Conserv

> ~1 — Three-pl
electrical

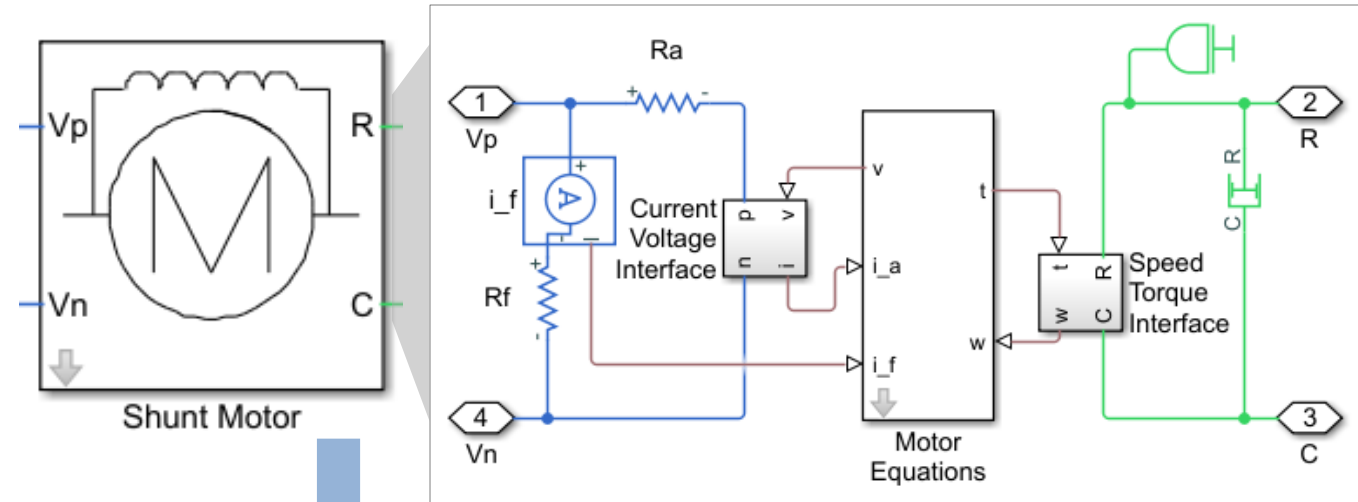
The model contains as many Constant Volume Hydraulic Chamber blocks as there are segments. The chamber lumps fluid volume equal to

Define components using resizable arrays of elements

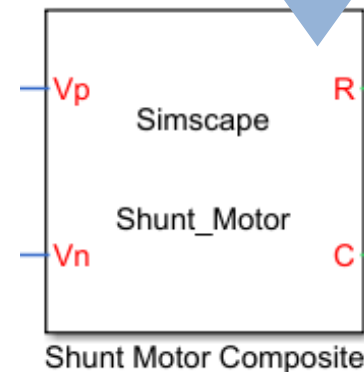
```
import foundation.isothermal_liquid.*;
component SegmentedPipeline
parameters
    N = 10; % Number of segments
    segm_length = { 5, 'm' }; % Length of each segment
end
% Ports at the two ends of the pipeline
nodes
    A = isothermal_liquid; % A:left
    B = isothermal_liquid; % B:right
end
% Declare 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>)



Shunt_Motor.ssc

```
connections
connect (Speed_Torque_Interface.w, Motor_Equations.t);
connect (Motor_Equations.t, Speed_Torque_Interface);
connect (i_f.I, Motor_Equations.i_f);
connect (Motor_Equations.v, Current_Voltage_Interface.i);
connect (Current_Voltage_Interface.i, Motor_Equations.v);
connect (Rf0.p, i_f.n);
connect (Vp, Ra0.p);
```

ssc_protect()

Shunt_Motor.sscp

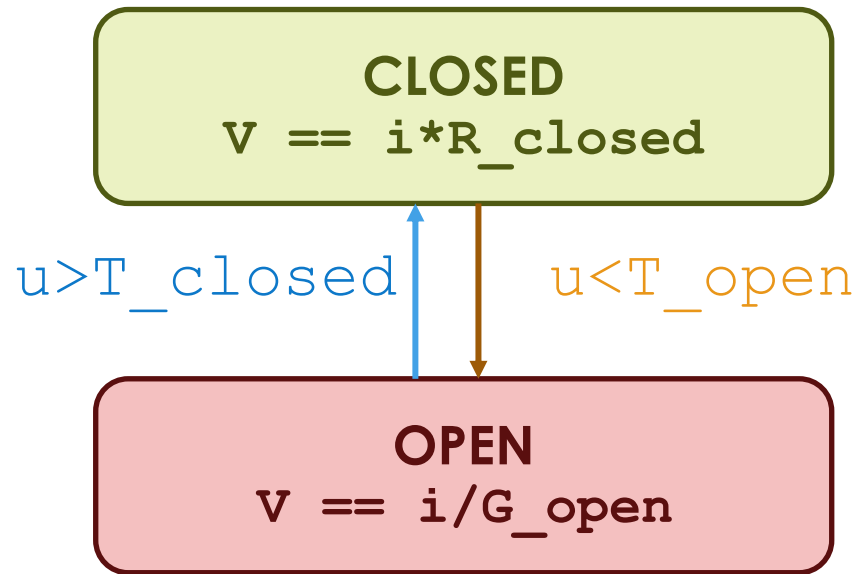


Inertia:	.00022	kg*m^2
Back EMF:	5.11	
Armature Resistance:	110	Ohm
Field Resistance:	2460	Ohm

Discrete Changes Behavior

- Use Event or Mode Chart Modeling

Switch Behavior

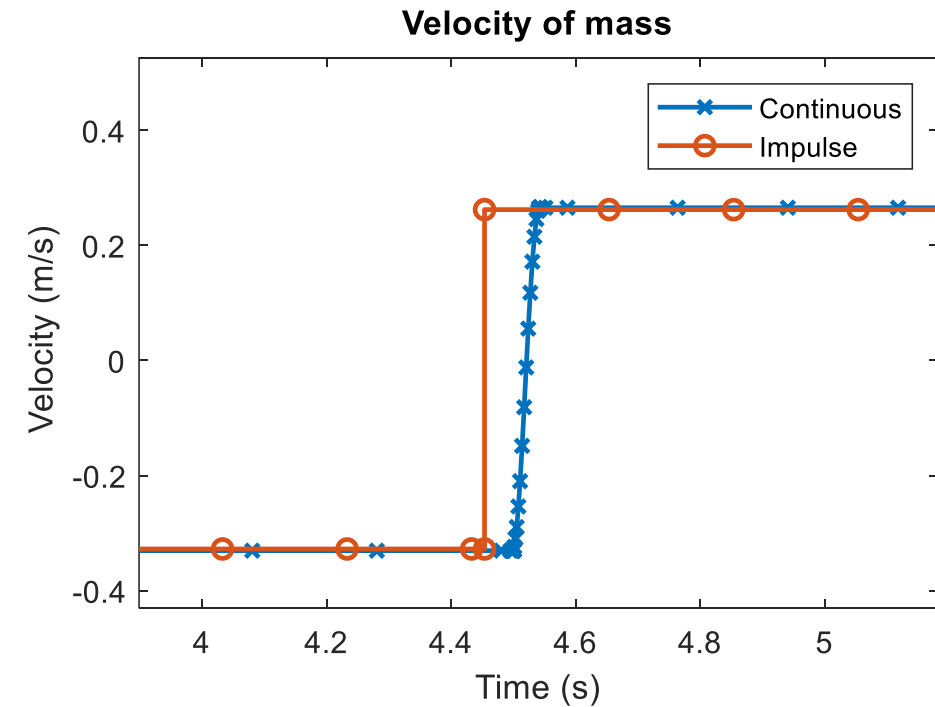
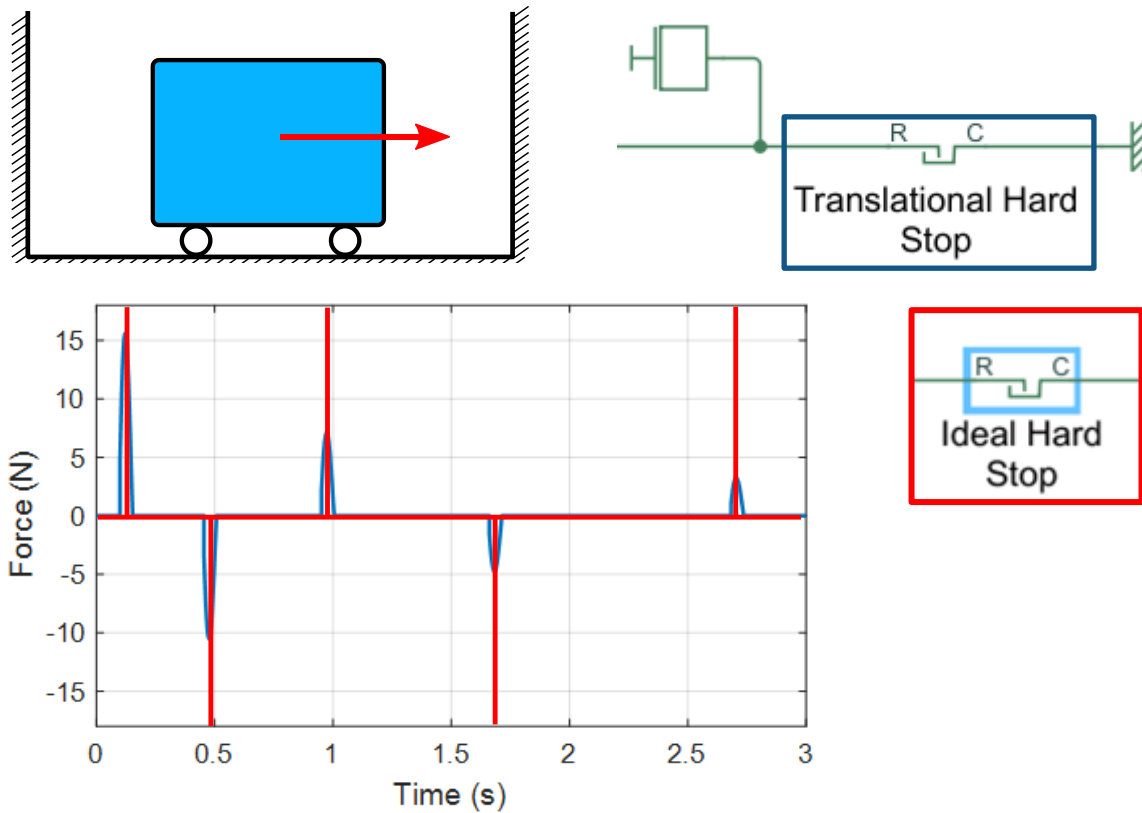


```

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
  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