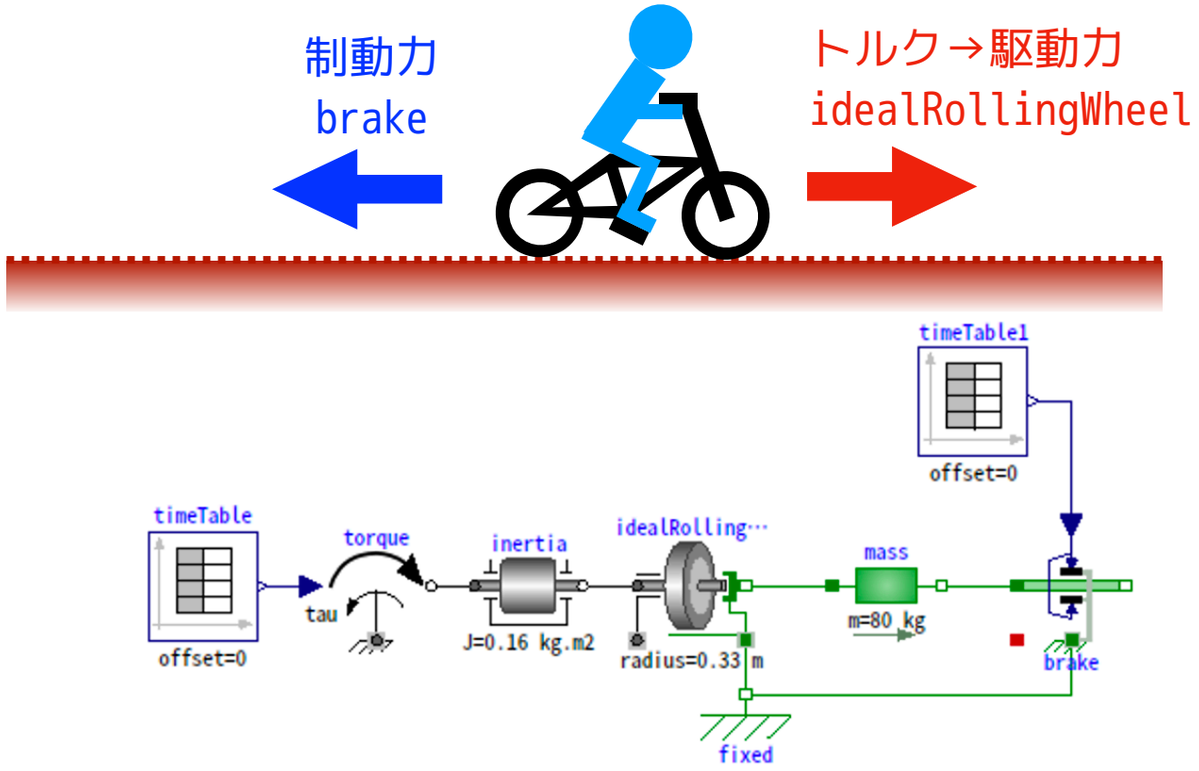
OpenModelicaによる
一次元並進機構モデル入門

Modelica.Mechanics.Translational.Components の
全コンポーネントのオリジナル例題と構成



2021/10/18

2021/11/10 改訂　　有限会社アマネ流研(https://www.amane.to) 田中 周 (finback)

はじめに

Modelica Standard Library の中でも一次元機構系ライブラリである Mechanics.Translational は、Spring（ばね）やMass（質量のある物体）など馴染み深いコンポーネントが含まれていて、比較的容易に使用できます。しかしながら、次のステップに進むのが難しいのが現状だと思います。
そこで、Modelica.Mechanics.Translational.Components にある全コンポーネント１８個について簡単なオリジナル例題を２０個作成しました。また、コンポーネントの継承関係や内部に含まれる方程式もできるだけ示しました。これらを紹介します。

本文書ではモデルどうしの依存関係をUMLのクラス図的に表しています。
Modelica のクラスの概念に関しては、

1. Modelica のクラスの概要 https://www.amane.to/wp-content/uploads/2018/12/8ec4f21241c98ee8413280240090c942.pdf

が参考になると思います。

- The purpose of this document is introducing the Modelica.Mechanics.Translational package which is included in the Modelica Standard Library (MSL). This document uses libraries, software, figures, and documents included in MSL and those modifications. Licenses and copyrights of those are written in next page.

Copyright (c) 2021, Amane Tanaka
Released under the MIT License
https://opensource.org/licenses/mit-license.php

Modelica Standard Library License
https://github.com/modelica/ModelicaStandardLibrary/blob/master/LICENSE
BSD 3-Clause License

Copyright (c) 1998-2020, Modelica Association and contributors
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

*** Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.**

*** Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.**

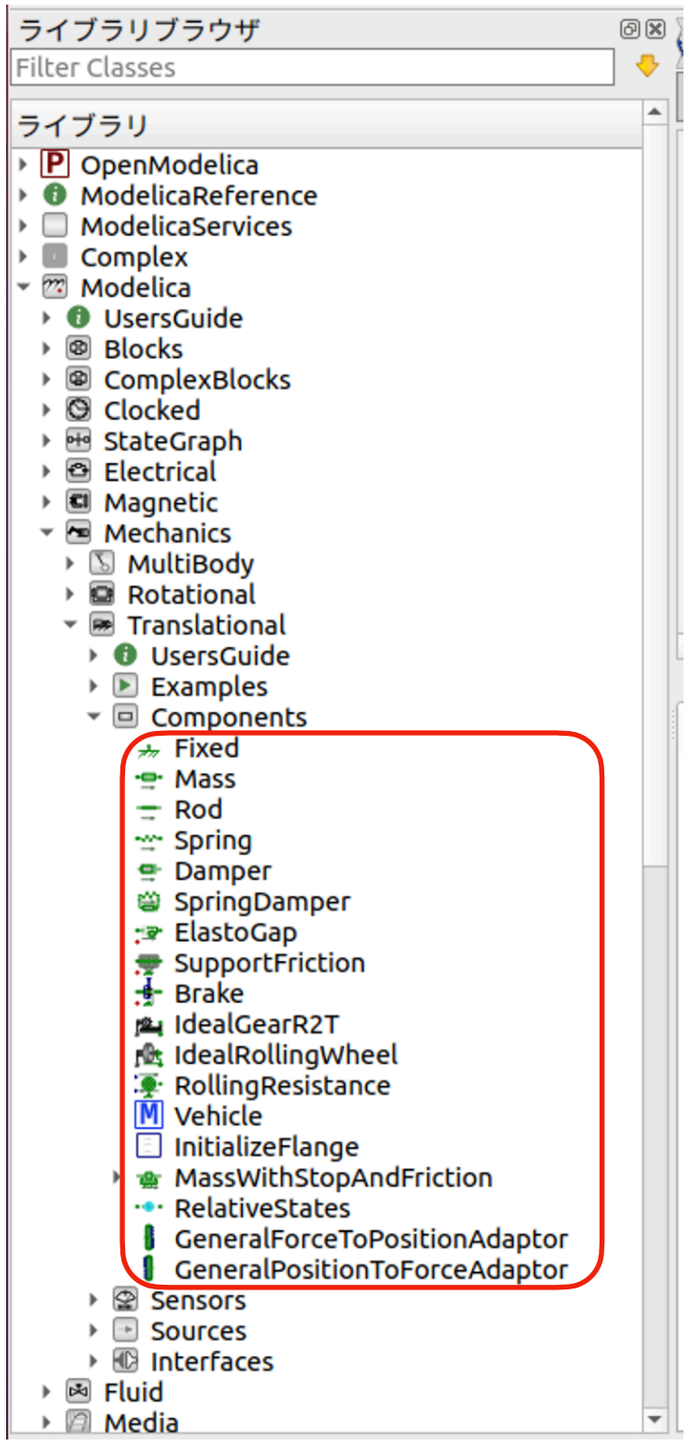
*** Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.**

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contents of Examples

Modelica.Mechanics.Translational.Components

OpenModelicaのライブラリブラウザ



基本的なコンポーネント

Fixed 空間的に固定された点のモデル
Mass 慣性質量のある物体のモデル
Spring ばねのモデル
Rod 質量が無視できる棒のモデル

摩擦力のコンポーネント

Damper 速度に比例した抵抗力のモデル
SpringDamper ばねとダンパを並列したモデル
ElastoGap 弾性体に接触したときの弾力と抵抗力のモデル
SupportFriction 速度と運動状態に依存した摩擦力のモデル
Brake 垂直効力が制御できる摩擦力のモデル
MassWithStopAndFriction 摩擦力と可動範囲のある物体

車両関係のコンポーネント

IdealGearR2T, IdealRollingWheel 回転と並進の変換
RollingResistance 転がり抵抗力のモデル
Vehicle トルクで駆動する車両の簡易モデル

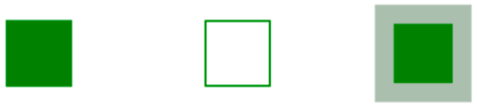
その他のコンポーネント

RelativeStates 相対位置、相対速度を状態変数にするモデル
InitializeFlange 入力信号でFlangeを初期化するモデル
GeneralForceToPositionAdaptor Flangeを入出力信号に変換する(1)
GeneralPositionToForceAdaptor Flangeを入出力信号に変換する(2)

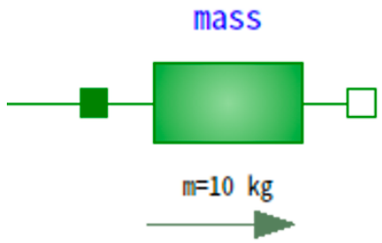
基本コンポーネント

基本的なコンポーネント
Flange 一次元並進機構系のコネクタ
Fixed 空間的に固定された点のモデル
Force 力のモデル
Mass 慣性質量のある物体のモデル
Spring ばねのモデル
Rod 質量が無視できる棒のモデル

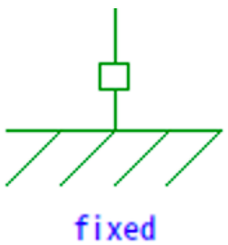
Flange



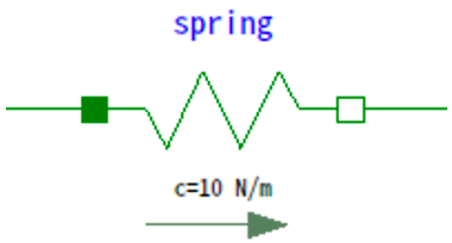
Mass



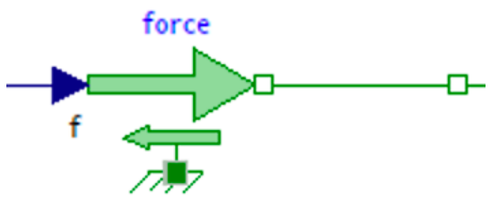
Fixed



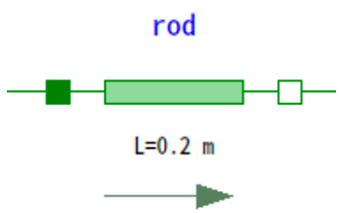
Spring



Force



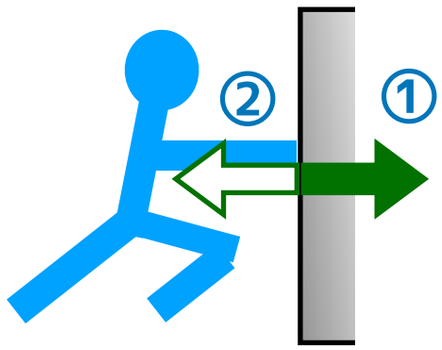
Rod



Example1 壁を押す

概要

力を加える人 固定壁面
force　fixed



壁の位置 0 [m]

人が10[N]の力で壁を押します。

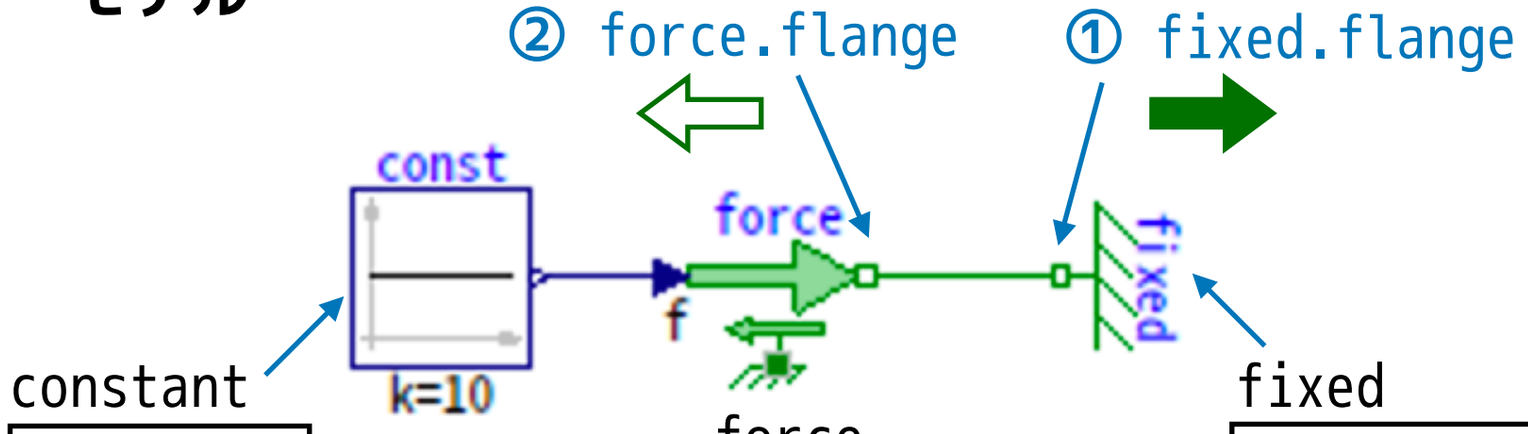
① → 10 [N]　作用

壁面が人から加えられる力
(人が壁面に加える力)

② ← -10 [N]　反作用

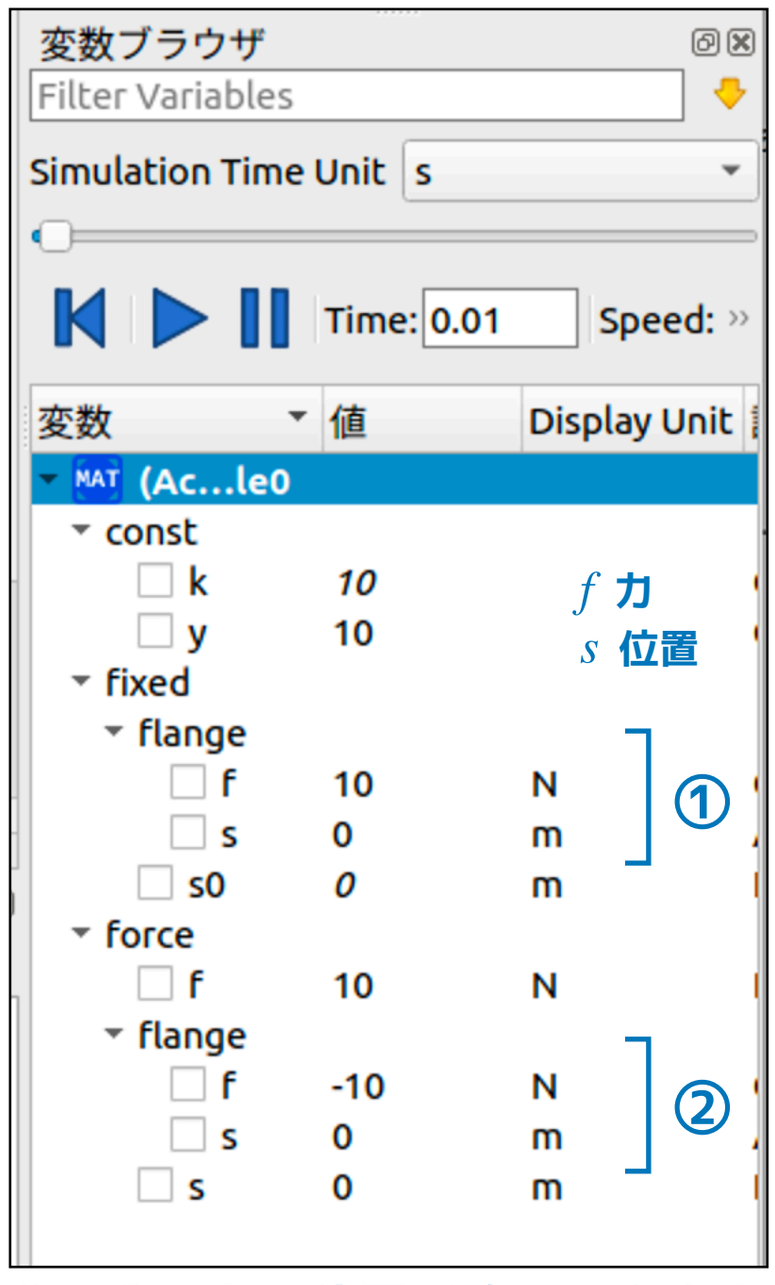
人が壁面から加えられる力

モデル



Modelica.Blocks.Sources.Constant
パラメータで設定した実数定数の値を出力します。

シミュレーション結果



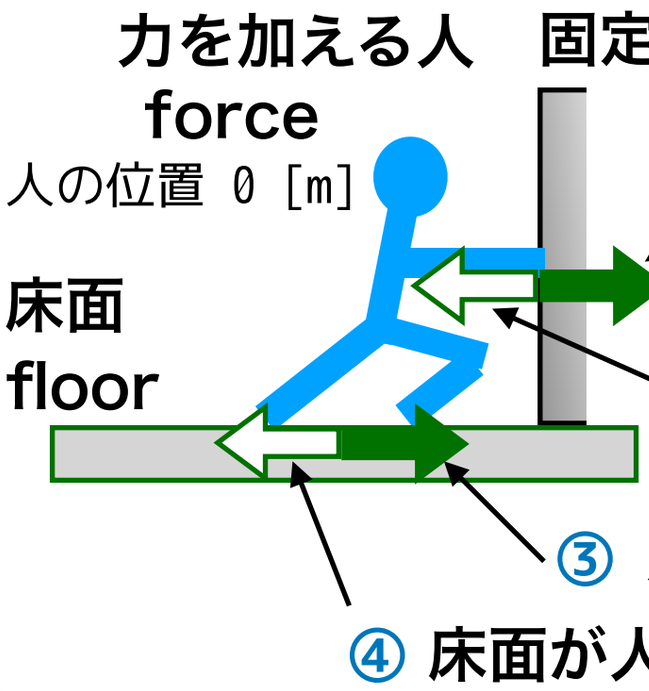
①と②の力と位置が確認できます。

コンポーネントの詳細は、後述します。　□で囲まれた部分がコンポーネントに設定するパラメータです。

Example2 壁を押す（床面がある場合）

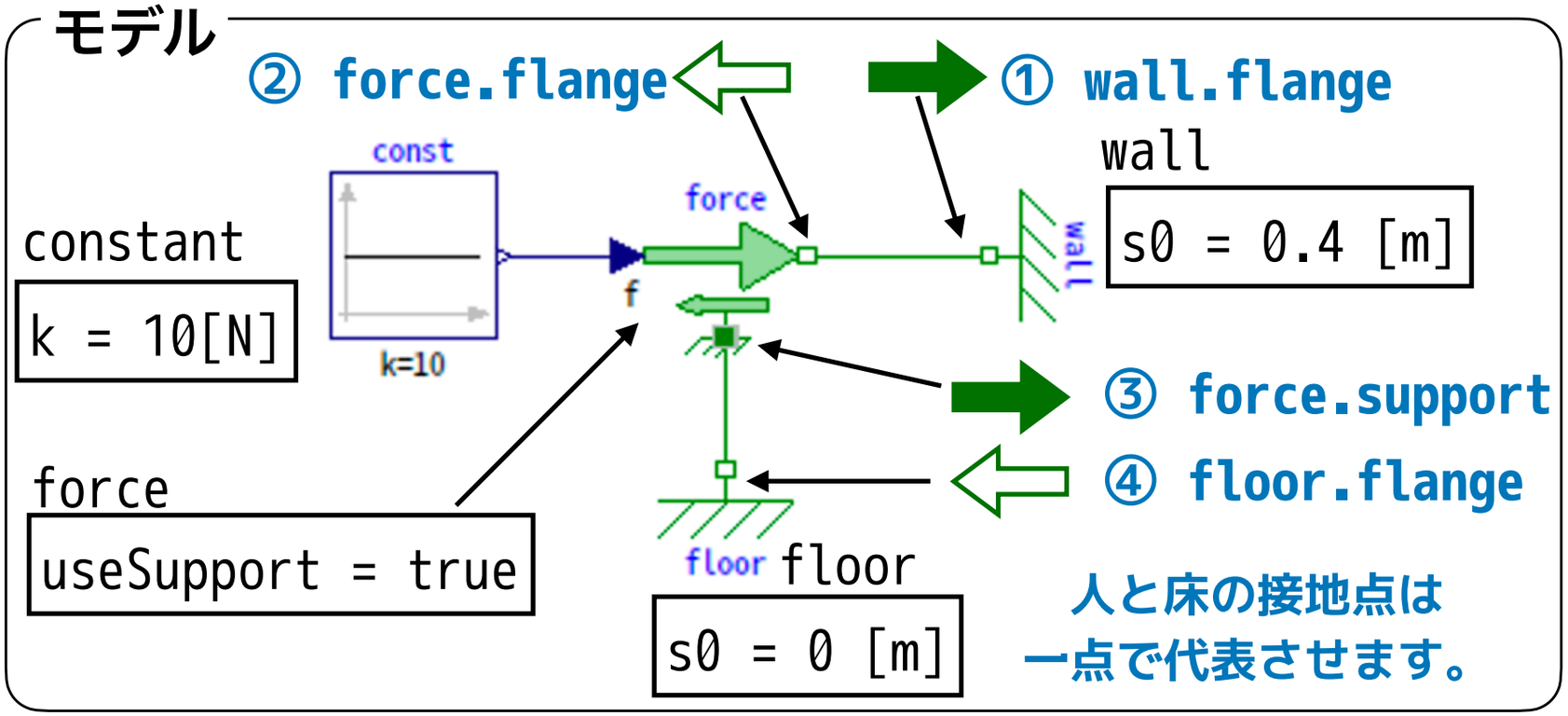
概要

床面に働く力も考えます。



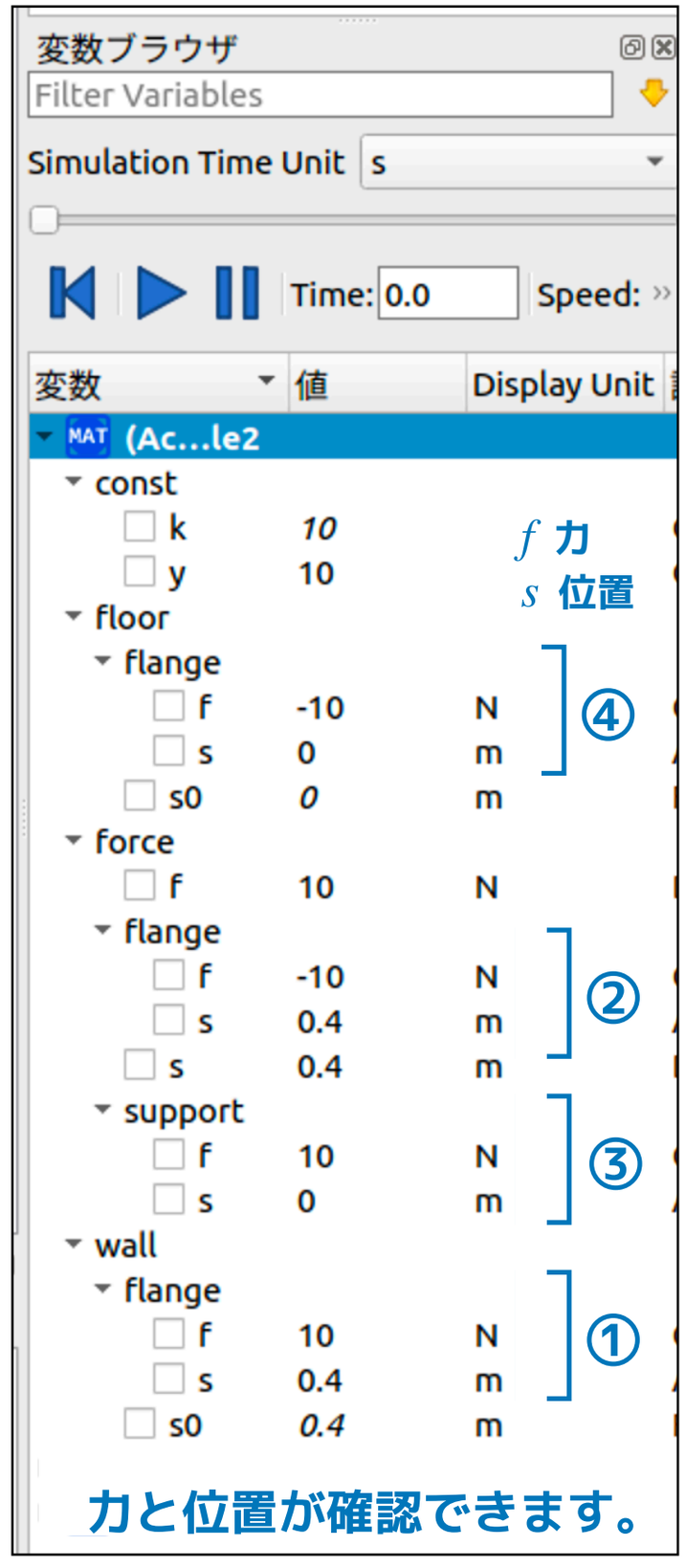
① 壁面が人から加えられる力
10 [N]
② 人が壁面から加えられる力
③ 人が床面から加えられる力
④ 床面が人から加えられる力

モデル



人と床の接地点は
一点で代表させます。

シミュレーション結果

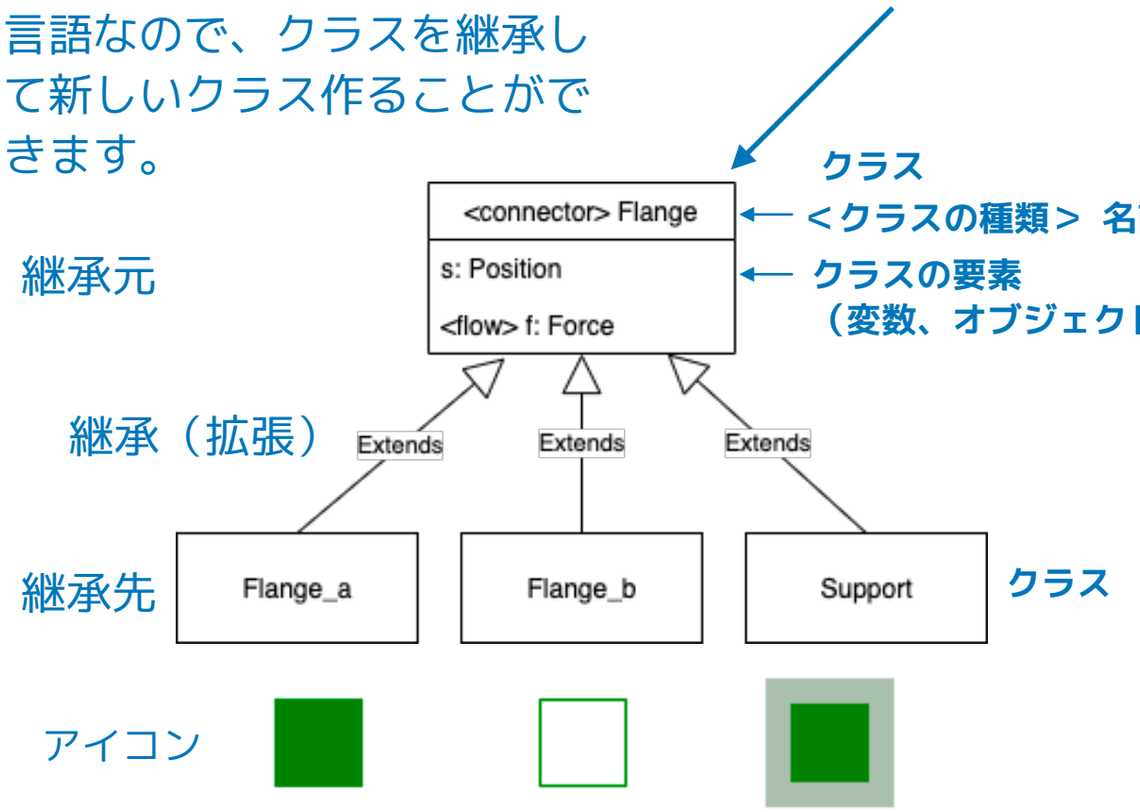


力と位置が確認できます。

Flange — 一次元並進機構系のコネクタ

Modelica.Mechanics.Translational のコンポーネントどうしを接続するコネクタです。

Modelicaはオブジェクト指向言語なので、クラスを継承して新しいクラス作ることができます。



- Flange_a, Flange_b, Support は、Flangeを継承（拡張、extends）して作られています。
- これらは、アイコンが異なるだけで同じものです。

Flangeは力点や作用点をモデル化したものと考えるとわかりやすいと思います。

Modelica.Mechanics.Translational.Interfaces.Flangeのソースコード

```
within Modelica.Mechanics.Translational.Interfaces;
connector Flange "One-dimensional translational flange"

  SI.Position s "Absolute position of flange";
  flow SI.Force f "Cut force directed into flange";
  annotation ( …);
    Documentation(info="<html>
end Flange;
```

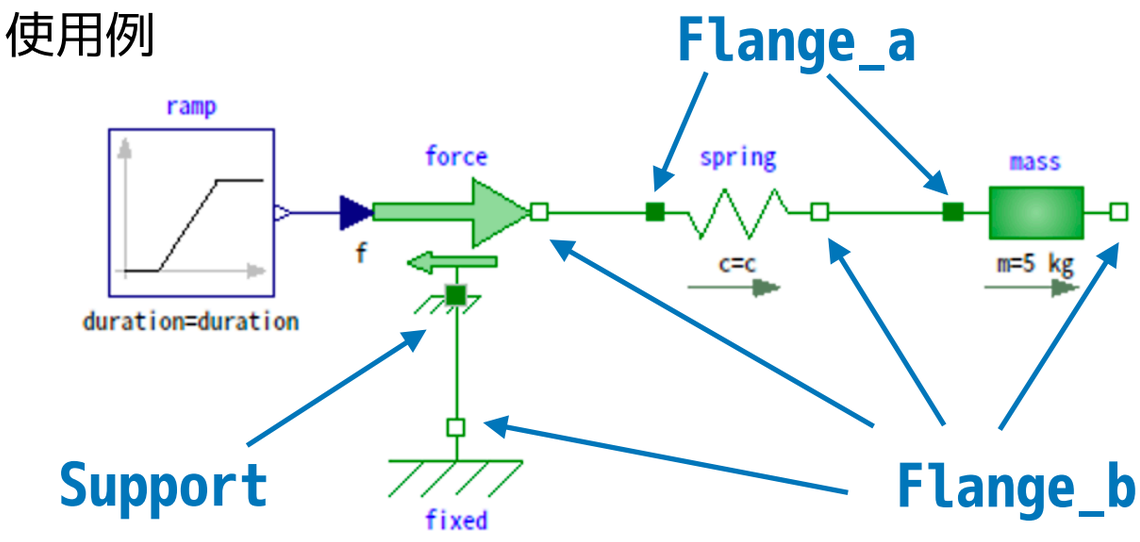
potential変数：接続されたコネクタ間で値が同じ

s：作用点(面)の位置 [m]

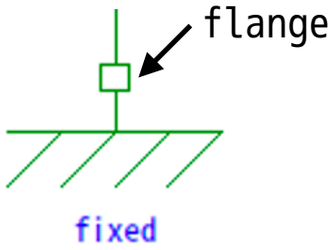
flow変数：接続されたコネクタで和がゼロ

f：コンポーネントに加えられる力 [N]（部品）

使用例



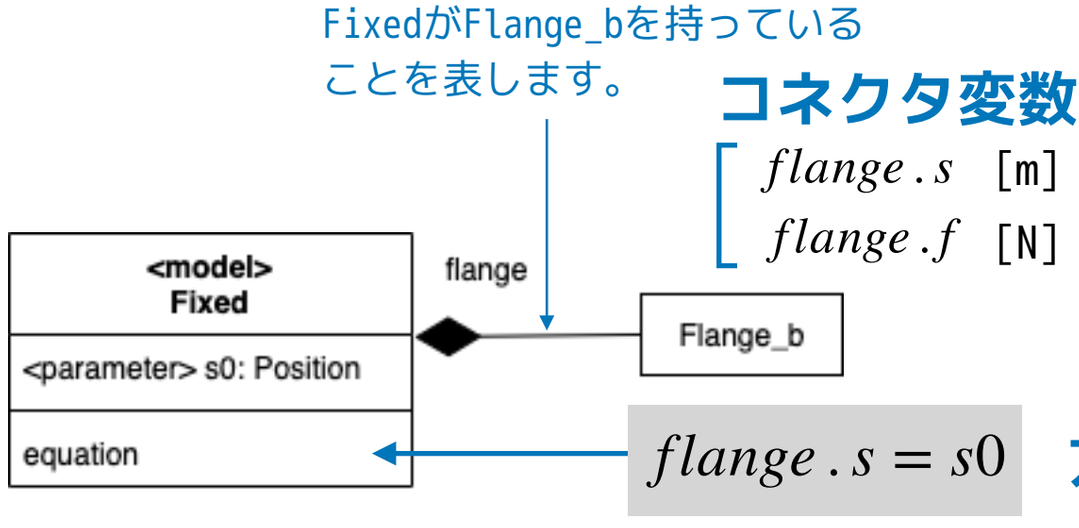
Fixed — 空間的に固定された点のモデル



空間的に固定された点を表すモデル。
パラメータ $s0$ で位置 flange.s を設定します。

構成と方程式

$s0$ ： 固定位置［m］



modelなどのクラスのふるまいとして方程式やアルゴリズムを記述できることがModelica言語の特徴の一つです。

本文書では
方程式は背景色灰色で示します。

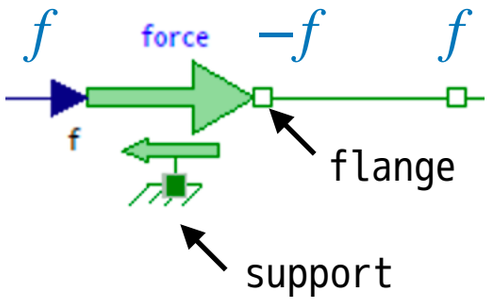
MOdelica.Mechanics.Translational.Components.Fixedのソースコード

```
within Modelica.Mechanics.Translational.Components;
model Fixed "Fixed flange"
  parameter SI.Position s0=0 "Fixed offset position of housing";

  Interfaces.Flange_b flange annotation ( …);
equation
  flange.s = s0;
  annotation ( …);
end Fixed;
```

equation セクション

力点や作用点の位置をパラメータ s_0 で設定します。

Force — 力のモデル



実数信号入力（RealInput）f で大きさを設定する力のモデル
アイコンの矢印は正の実数信号を与えた場合の力の向き(加速される向き)を表します。
外部に力 f を加えるため、反作用としてflangeに $-f$ の力が加えられます。

構成と方程式

パラメータ useSupport=trueのときsupportフランジが使えるようになります。

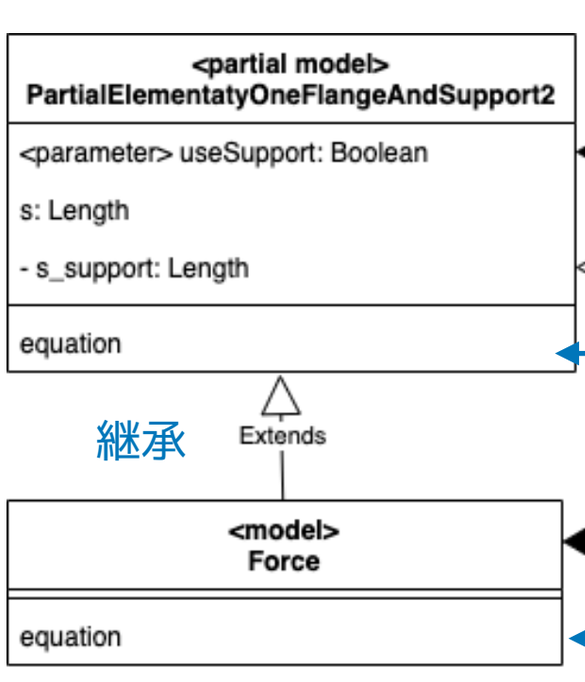
ソースコードの一部

```
Support support(s=s_support, f=-flange.f) if useSupport
```

1個のFlangeと１個のSupportをもつモデル

s ： 相対位置（距離）[m]

$s\_support$ ： 基準位置 [m]



f ： 力（実数入力信号）[N]

実数信号入力（RealInput）で大きさを設定する力のモデル

$$\begin{bmatrix} flange.s\ [\mathrm{m}] \\ flange.f\ [\mathrm{N}] \end{bmatrix}$$

if useSupport

$$support.s = s\_support$$
$$supoort.f = -flange.f$$

$$s = flange.s - s\_support$$

if not useSupport

$$s\_support = 0$$

end if

$$flange.f = -f$$

力点や作用点の力を、実数入力信号 f で設定します。

Example1で作成した連立方程式

コンポーネントを接続すると、次のような連立方程式ができます。
OpenModelica などのシミュレーションツールは、このような連立方程式を解きます。

変数

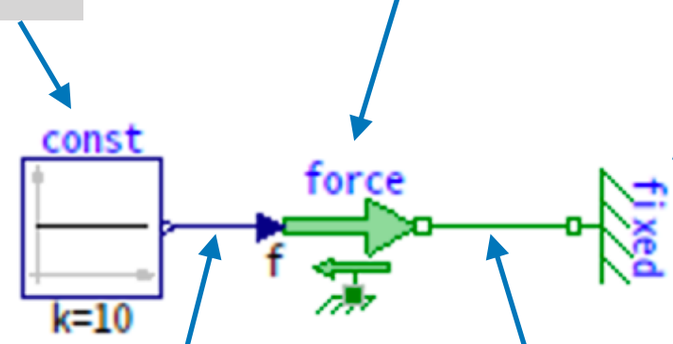
$const.y$
$force.f$
$force.s$
$force.s\_support$
$force.flange.f$
$force.flange.s$
$fixed.flange.f$
$fixed.flange.s$

連立方程式

$force.flange.f = -force.f$
$fors.s\_support = 0$
$force.s = force.flange.s - force.s\_support$

$const.y = 10$

$fixed.flange.s = 0$



$const.y = force.f$

$force.flange.s = fixed.flange.s$
$force.flance.f + fixed.flange.f = 0.0$

Flangeどうしを接続することによって、
力を及ぼす方が力点、受ける方が作用点になります。

Example2 で作成した連立方程式

OpenModelica などのシミュレーションツールは、このような連立方程式を解きます。

変数

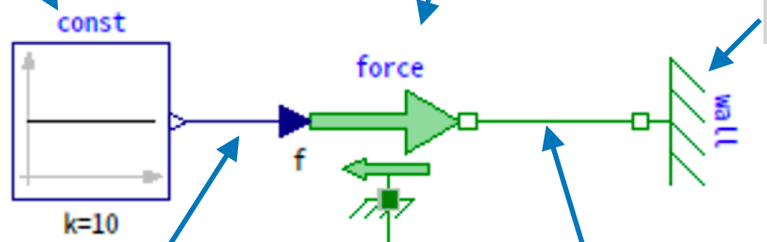
$const.y$
$force.f$
$force.s$
$force.s\_support$
$force.flange.f$
$force.flange.s$
$force.support.f$
$force.support.s$
$wall.flange.f$
$wall.flange.s$
$floor.flange.f$
$floor.flange.s$

連立方程式

$$force.flange.f = -force.f$$
$$force.s = force.flange.s - force.s\_support$$
$$force.support.s = force.s\_support$$
$$force.support.f = -force.flange.f$$

$const.y = 10$

$wall.flange.s = 0.4$



$const.y = force.f$

$floor.flange.f + force.support.f = 0.0$
$floor.flange.s = force.support.s$

$force.flange.s = wall.flange.s$
$wall.flance.f + force.flange.f = 0.0$

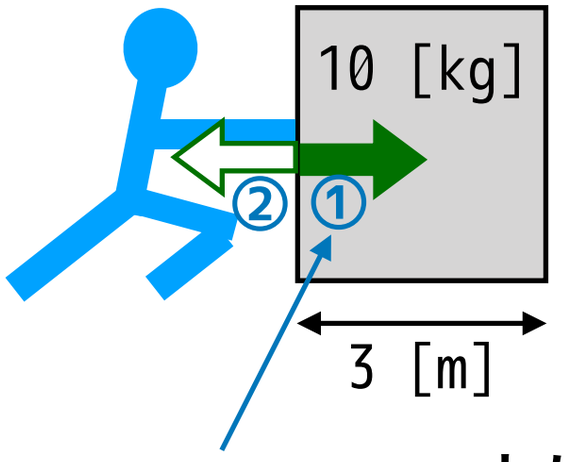
$floor.flange.s = 0$

Example3 質量のある物体に力を加える

概要

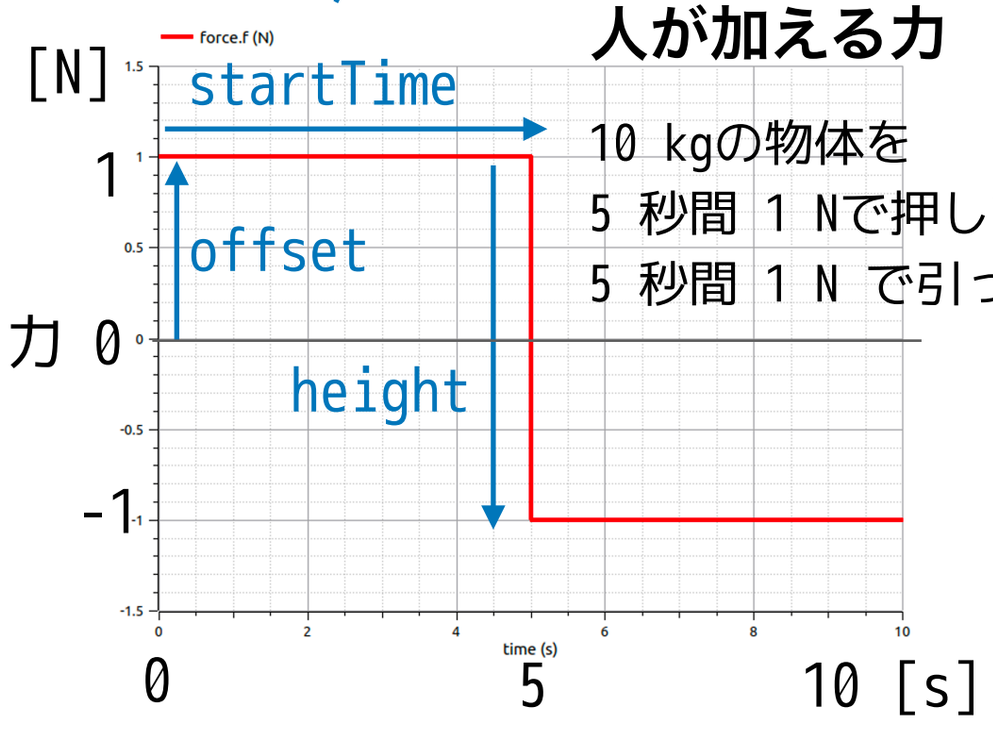
力を加える人　　質量のある物体

force　　mass



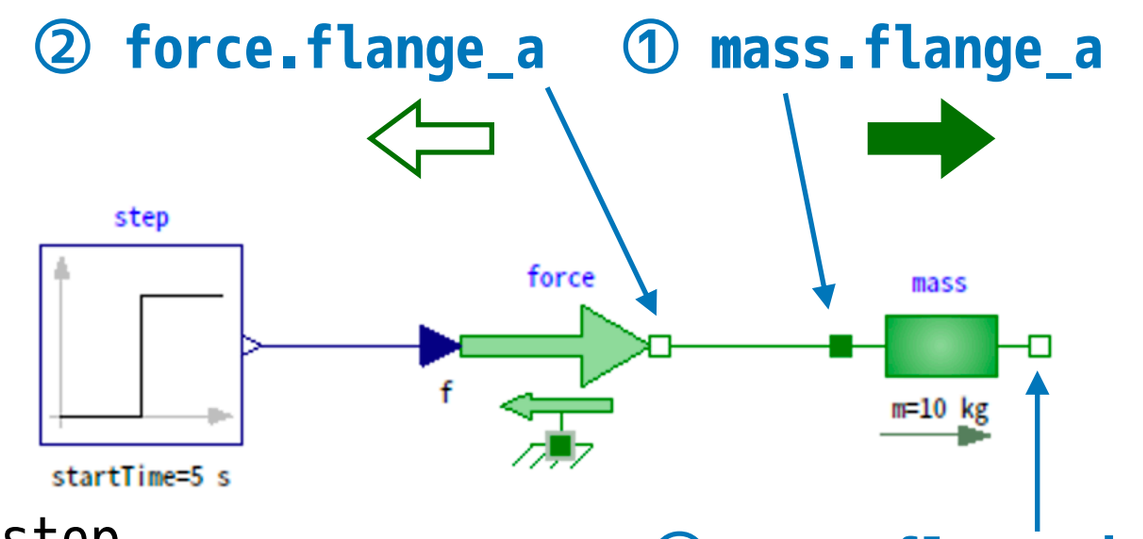
初期位置 0 [m]
初期速度 0 [m/s]

人が加える力



10 kgの物体を
5 秒間 1 Nで押してから
5 秒間 1 N で引っ張る。

モデル



step

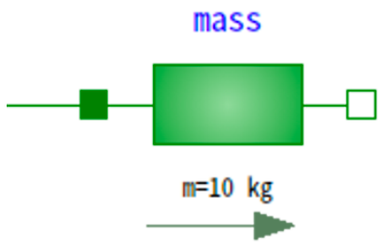
height = -2
offset = 1
startTime =5

Modelica.Blocks.Sources.Step
ステップ関数状の信号を生成します。

mass

m = 10 [kg]	質量
L = 3 [m]	長さ
v.start = 0 [m/s]	速度の初期値
v.fixed = true	速度の初期値を設定する
s.start = 0 [m]	中心位置の初期値
s.fixed = true	中心位置の初期値を設定する

Mass — 慣性質量のある物体のモデル



アイコンの矢印は位置 s が増大する向きを表します。

- 物体は flange_a.f と flange_b.f の合力よにって加速されます。
- 合力が正の場合、アイコンの矢印の向き(sが増大する向き)に加速されます。

構成と方程式

2個のFlangeをもつモデル

$$\begin{bmatrix} flange\_a.s & [\mathrm{m}] \\ flange\_a.f & [\mathrm{N}] \end{bmatrix}$$

$$\begin{bmatrix} flange\_b.s & [\mathrm{m}] \\ flange\_b.f & [\mathrm{N}] \end{bmatrix}$$

大きさのある硬いものを表すモデル

s:物体の中心位置 [m]

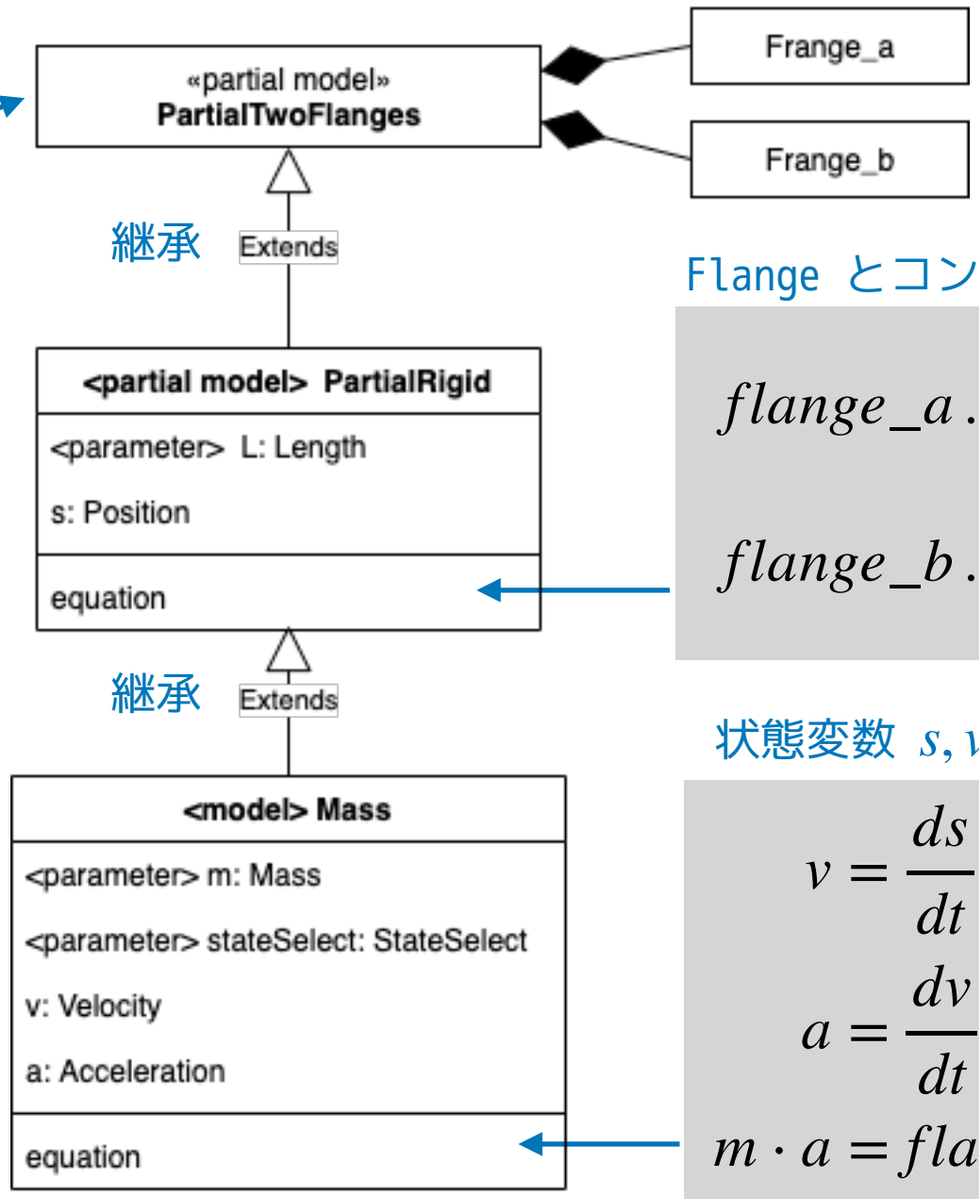
L: 物体の長さ [m]

慣性質量のある物体のモデル

v: 速度 [m/s]

a: 加速度 [m/s2]

m: 質量 [kg]



Flange とコンポーネントの位置関係

$$flange\_a.s = s - \frac{L}{2}$$

$$flange\_b.s = s + \frac{L}{2}$$

状態変数 s, v に関する微分方程式

$$v = \frac{ds}{dt}$$ 速度

$$a = \frac{dv}{dt}$$ 加速度

物体の運動方程式

$$m \cdot a = flange\_a.f + flange\_b.f$$

Example3 で作成した連立方程式

変数

$force.s$
$force.flange.s$
$force.flange.f$
$force.s\_support$
$force.f$
$mass.s$
$mass.flange\_a.s$
$mass.flange\_a.f$
$mass.flange\_b.s$
$mass.flange\_b.f$
$mass.v$
$mass.a$
$step.y$

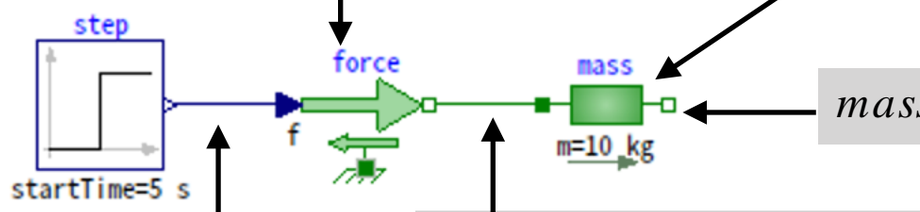
$mass.s$ と $mass.v$ は **状態変数(state variable)** と呼ばれます。

連立方程式

$$step.y = step.offset + (if\ time < step.startTime\ then\ 0.0\ else\ step.height)$$

$$mass.m \cdot mass.a = mass.flange\_a.f + mass.flange\_b.f$$
$$mass.v = \frac{d(mass.s)}{dt}$$
$$mass.a = \frac{d(mass.v)}{dt}$$
$$mass.flange\_a.s = mass.s - \frac{mass.L}{2}$$
$$mass.flange\_b.s = mass.s + \frac{mass.L}{2}$$

$$force.flange.f = -force.f$$
$$force.s\_support = 0.0$$
$$force.s = force.flange.s - force.s\_support$$



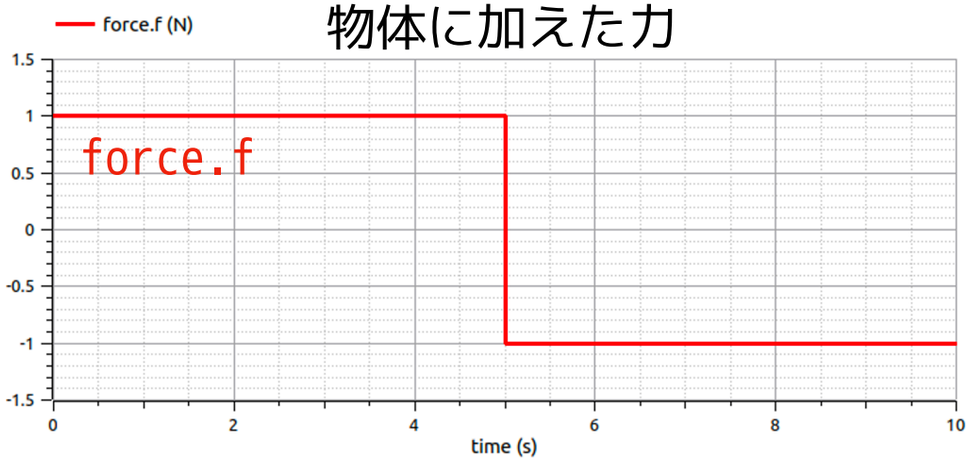
$$mass.flange\_b.f = 0$$

$$step.y = force.f$$

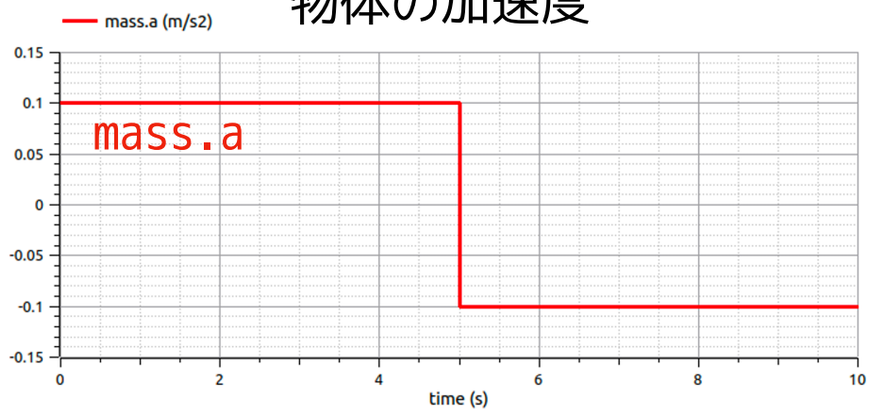
$$force.flange.s = mass.flange\_a.s$$
$$mass.flange\_a.f + force.flange.f = 0.0$$

シミュレーション結果

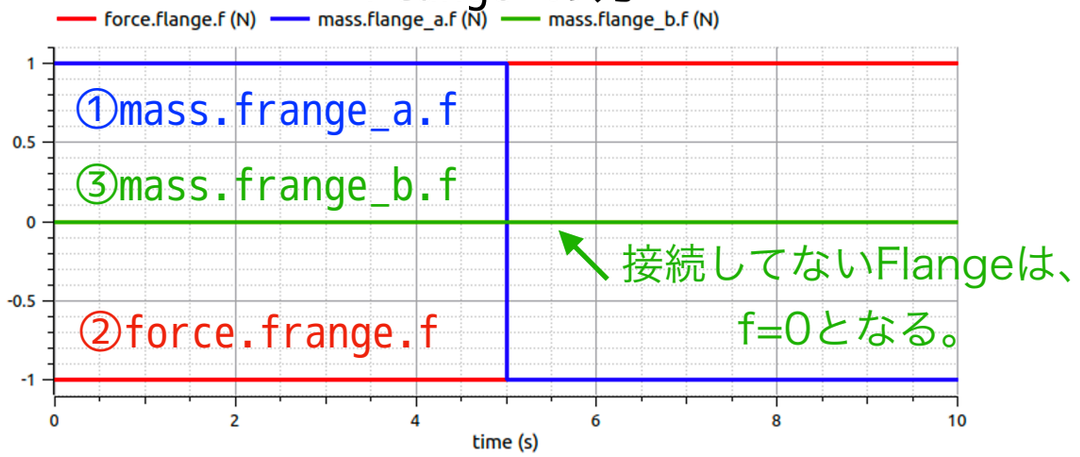
物体に加えた力



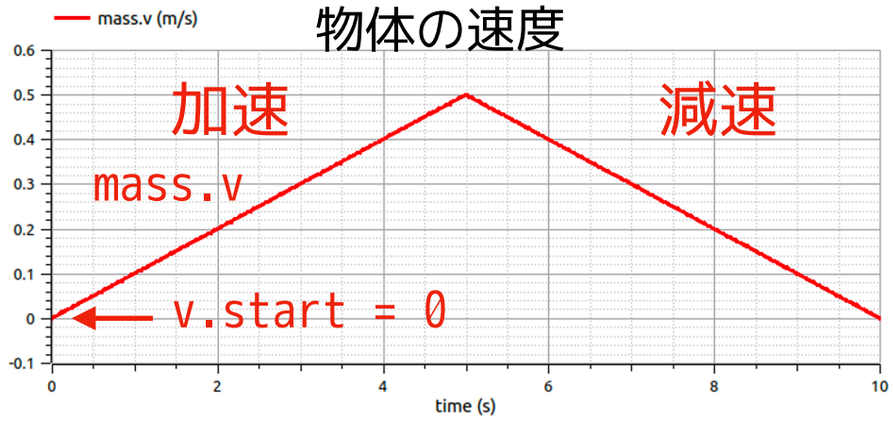
物体の加速度



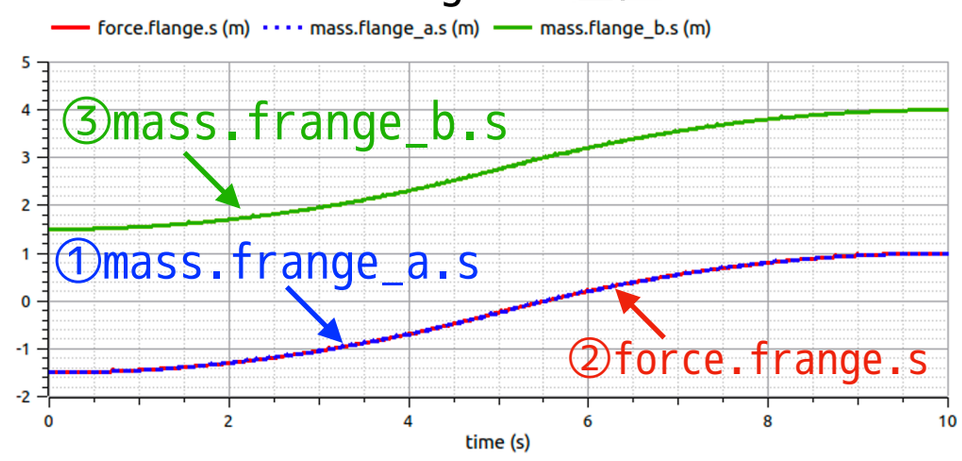
Flange の力



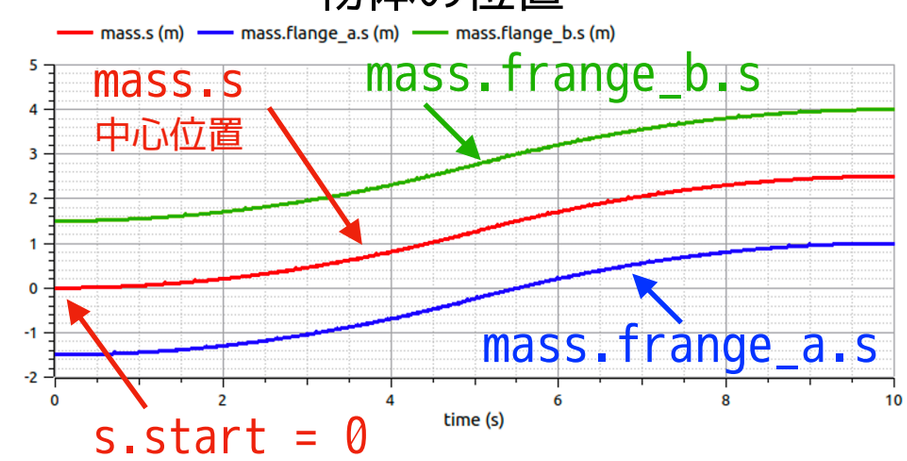
物体の速度



Flange の位置



物体の位置



Modelicaの方程式の種類

noEvent と reinit を除く、OpenModelicaなどのツールで変換されたModelicaの方程式の種類は以下のように分類でます。

$$0 = f_x(v, c)$$

微分代数方程式(DAE)、状態変数の時間的発展が数値積分によって解かれます。

$$c := f_c(relation(v))$$

条件式、数値積分の間に評価されモニタされます。

$$0 = \begin{cases} f_z(v, c), & \textbf{at event} \\ z - pre(x), & \textbf{otherwise} \end{cases}$$

離散的実数変数の方程式、条件式によって発生したイベントにより計算処理が駆動されます。

$$m := f_m(v, c)$$

値が離散的な変数の方程式、条件式によって発生したイベントにより計算処理が駆動されます。

$$v := [p; t; \dot{x}; x; y; z; m; pre(x); pre(m)]$$

- p パラメータまたは定数、すなわち時間依存しない変数。
- t 時間、(実数型)独立変数。
- $x(t)$ 明示的に微分で表される実数変数。
- $y(t)$ 連続的変数。明示的に微分で表されない実数変数（＝代数変数）。
- $z(t)$ 離散的実数変数。イベントが発生した時間 t_e に更新される変数。直前のイベント時の値$pre(z)$があります。
- $m(t)$ 値が離散的な変数(Boolean, Integerなど)。イベントが発生した時間 t_e に更新される変数。直前のイベント時の値 $pre(m)$ があります。
- $c(t_e)$ when節を変換したもの含む全てのif文で表される条件。
- $relation(v)$ 変数 v_i を含む関係式

Modelica Language Specification version 3.5, p288. https://modelica.org/documents/MLS.pdf より

State Selection (状態変数選択)

微分代数方程式の最も一般的な形
陰的微分方程式

$$\mathbf{F}(t, \mathbf{y}, \mathbf{y}') = 0$$

物理学的、工学的現象の数学的モデル

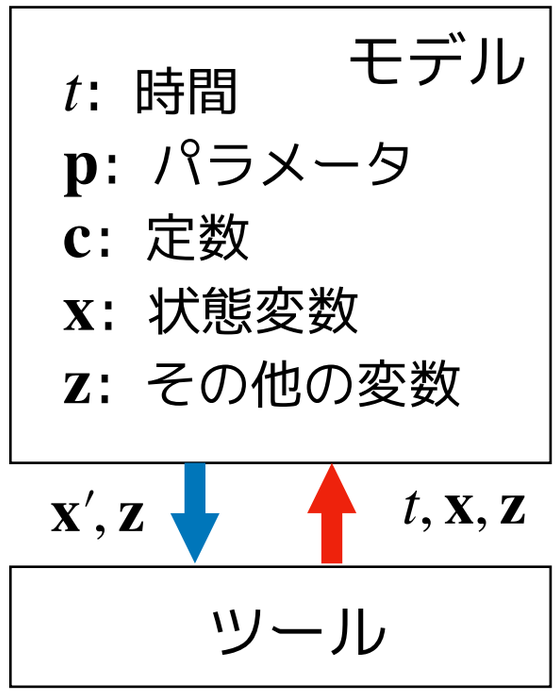
ツールによる translation

半陽的微分代数方程式
制約付きの常微分方程式

数値解析に適したモデル

$$\begin{cases} \mathbf{x}' &= \mathbf{f}(t, \mathbf{x}, \mathbf{z}) \\ \mathbf{0} &= \mathbf{g}(t, \mathbf{x}, \mathbf{z}) \end{cases}$$

微分方程式
代数方程式（制約条件）



ツールによる simulation

OpenModelicaなどのツールが
どの変数を状態変数(state variables)として微分方程式を構成するかを選択します。

例

$$x = y + c \quad \Leftrightarrow \text{同値} \quad y = x - c$$

$$\frac{dy}{dt} = f(x, y) \quad \Leftrightarrow \quad \frac{dx}{dt} = f(x, y)$$

x, y のどちらに対して微分方程式を構成するかツールが選択します。

実数変数の StateSelect パラメータ

ツール（ソルバー）が実数変数を状態変数として扱うプライオリティを制御するパラメータです。

Modelica Lanuage Specification 3.5, 4.8.7.1, p.57　https://modelica.org/documents/MLS.pdf

低 ↑ プライオリティ ↓ 高

```
type StateSelect = enumeration(
  never "Do not use as state at all.” ,
  avoid "Use as state, if it cannot be avoided (but only if variable appears
         differentiated and no other potential state with attribute default,
         prefer, or always can be selected).” ,
  default "Use as state if appropriate, but only if variable appears differentiated.” ,
  prefer "Prefer it as state over those having the default value
         (also variables can be selected, which do not appear differentiated). ",
  always "Do use it as a state.”
);
```

実数変数が微分の形で表されている場合、状態変数として扱う

Modelica.Mechanics.Translational.Components.Mass のソースコードを確認。

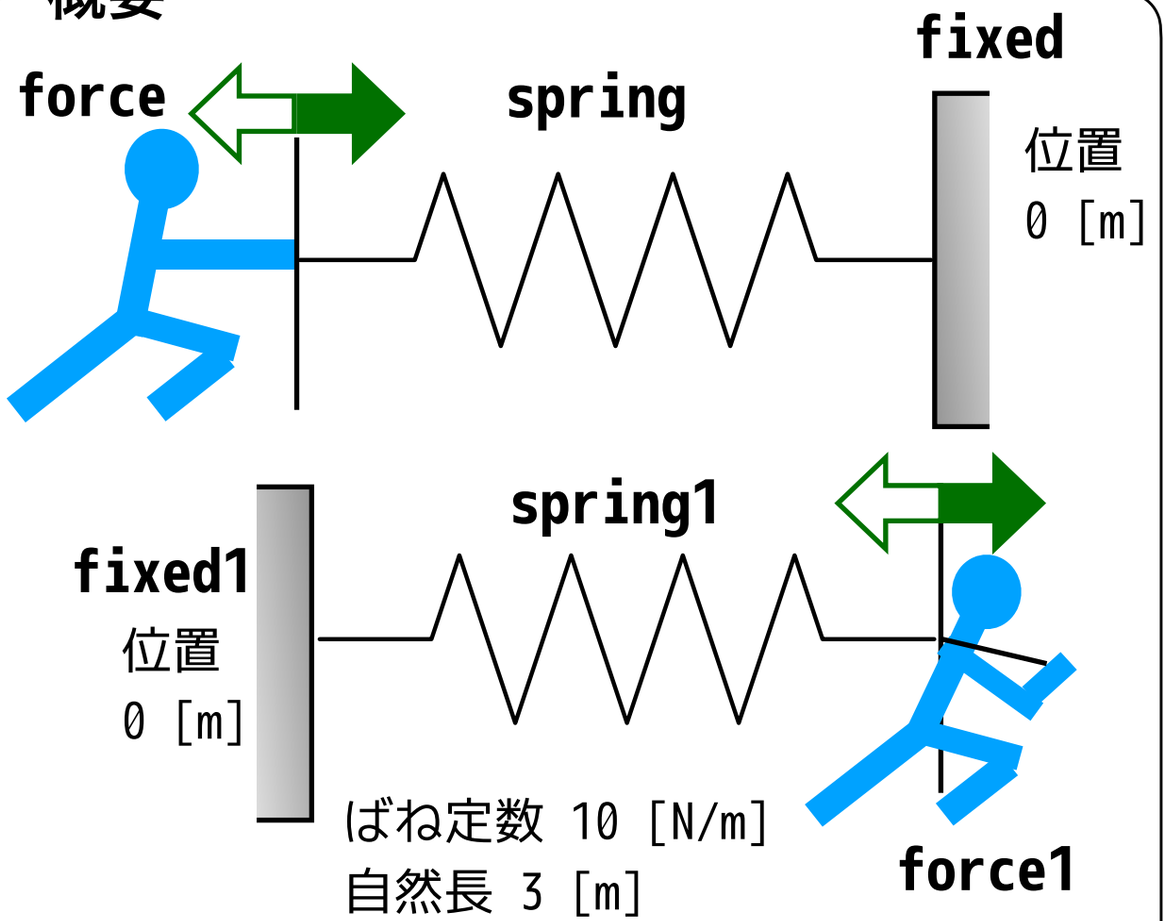
```
model Mass "Sliding mass with inertia"
  parameter SI.Mass m(min=0, start=1) "Mass of the sliding mass";
  parameter StateSelect stateSelect=StateSelect.default
    "Priority to use s and v as states" annotation (Dialog(tab="Advanced"));
  extends Translational.Interfaces.PartialRigid(L=0,s(start=0, stateSelect=
          stateSelect));
  SI.Velocity v(start=0, stateSelect=stateSelect)
    "Absolute velocity of component";
  SI.Acceleration a(start=0) "Absolute acceleration of component";
equation
  v = der(s);
  a = der(v);
  m*a = flange_a.f + flange_b.f;
  annotation ( …);
end Mass;
```

変数 s, v は、以下により状態変数として扱われます。

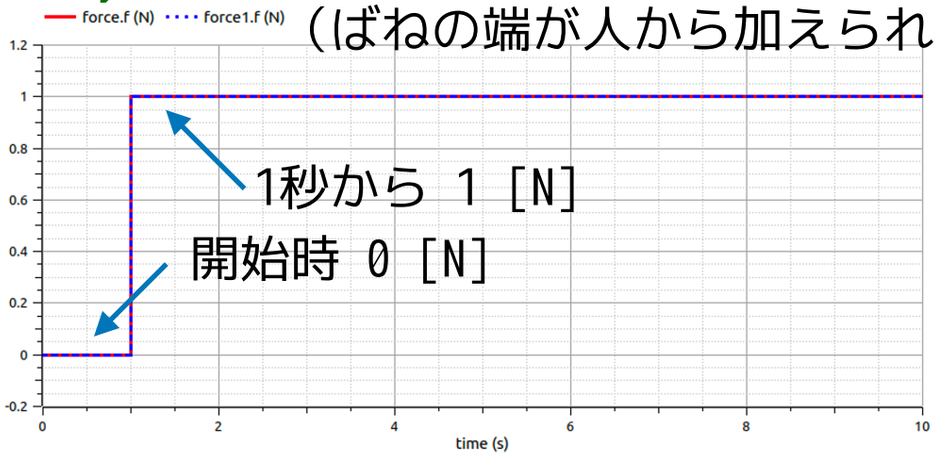
- **stateSelect = defalut**
- **微分の形で表されている**

Example4 ばねを押す。ばねを引っ張る。

概要



➡ 人がばねの端に加える力
（ばねの端が人から加えられる力）

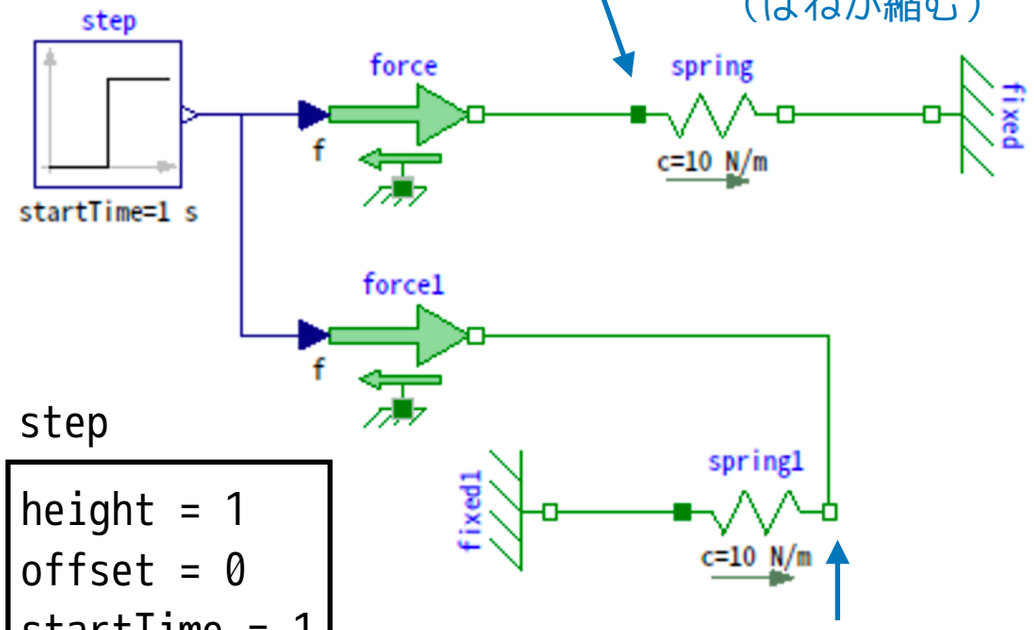


モデル

矢印の向きを合わせます！

spring.flange_a

➡ 正の力を加えると
flange_aの座標が増大
（ばねが縮む）



spring1.flange_b.f

➡ 正の力を加えると
flange_bの座標が増大
（ばねが伸びる）

step

```
height = 1
offset = 0
startTime = 1
```

fixed, fixed1

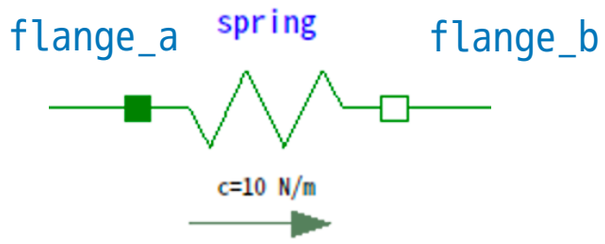
```
s0 = 0
```

spring, spring1

```
c = 10 [N/m]          ばね定数
s_rel0 = 3            自然長
s_rel.fixed = false   長さの初期値を設定しない
```

長さの初期値は、力の初期値 0 [N] につりあう自然長になる。

Spring — ばねのモデル



アイコンの矢印は Flange の位置が増大する向きを表します。

- flange_a に正の力を加えるとflange_aの位置が増大します。（ばねが縮む）
- flange_b に正の力を加えるとflange_bの位置が増大します。（ばねが伸びる）
- flange_a と flange_b の合力はゼロになります。(独立に入力できません)

構成と方程式

2個のFlangeをもつモデル

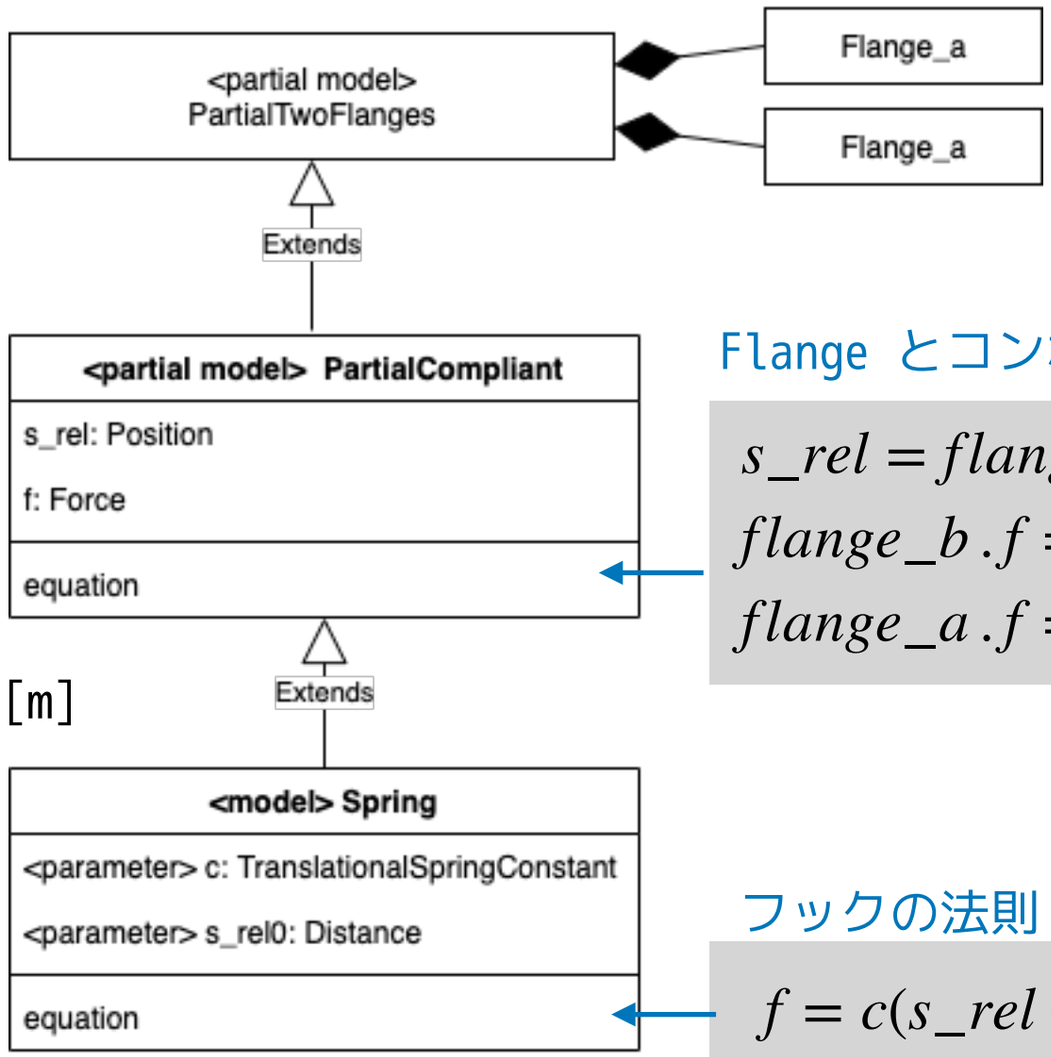
$$flange\_a.s$$
$$flange\_a.f$$
$$flange\_b.s$$
$$flange\_b.f$$

変形するものを表すモデル
相対位置(長さ)と力

s_rel ： flange_aに対する flange_bの相対位置 [m]
f ： 力 [N]

ばねのモデル
ばね定数、自然長をもつ

c ： ばね定数 [N/m]
s_rel0 ： ばねの自然長 [m]



Flange とコンポーネントの変数の関係

$$s\_rel = flange\_b.s - flange\_a.s$$
$$flange\_b.f = f$$
$$flange\_a.f = -f$$

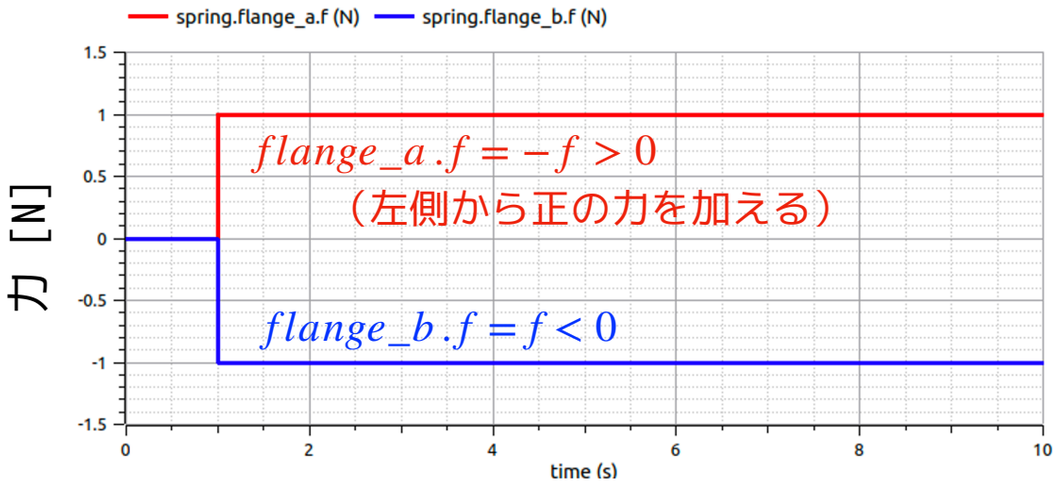
フックの法則

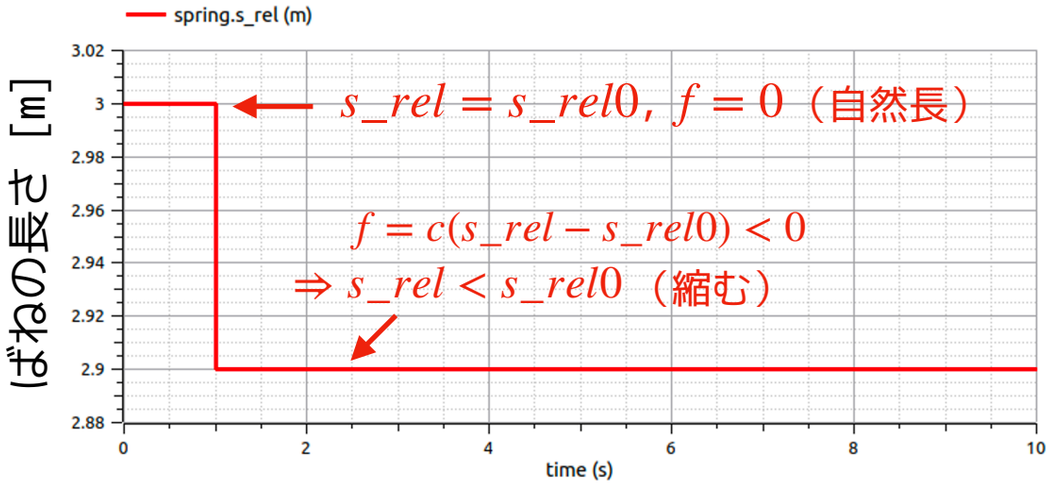
$$f = c(s\_rel - s\_rel0)$$

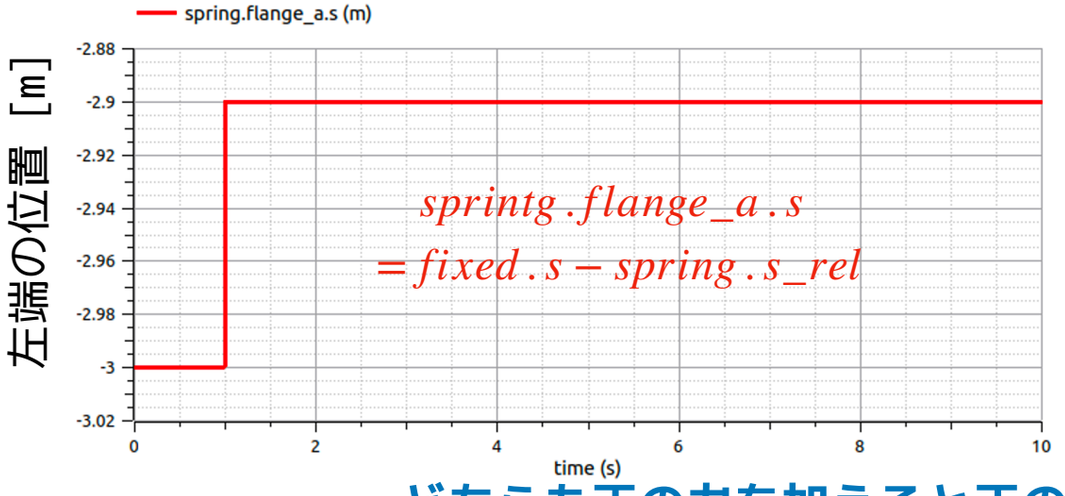
シミュレーション結果

慣性がないので力を変化させると
瞬時にばねの長さや位置が変化します

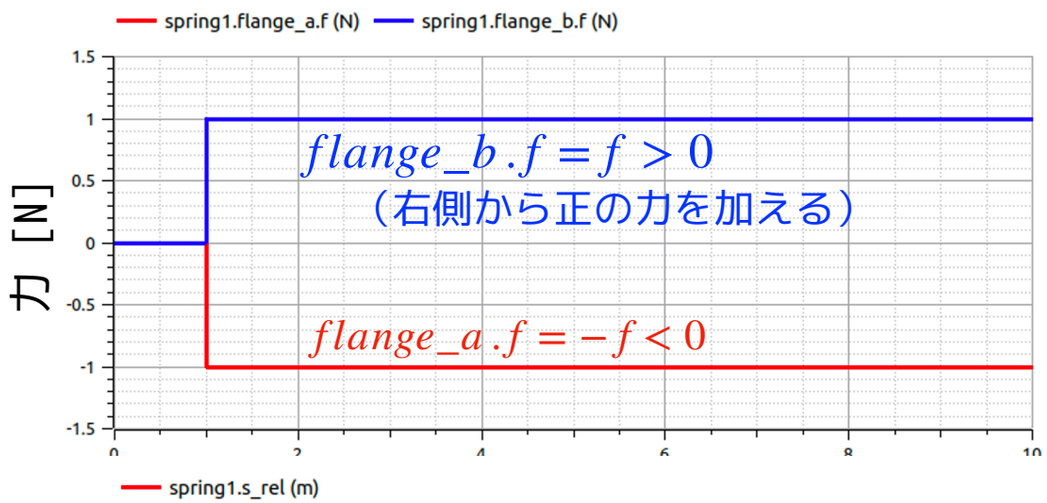
spring

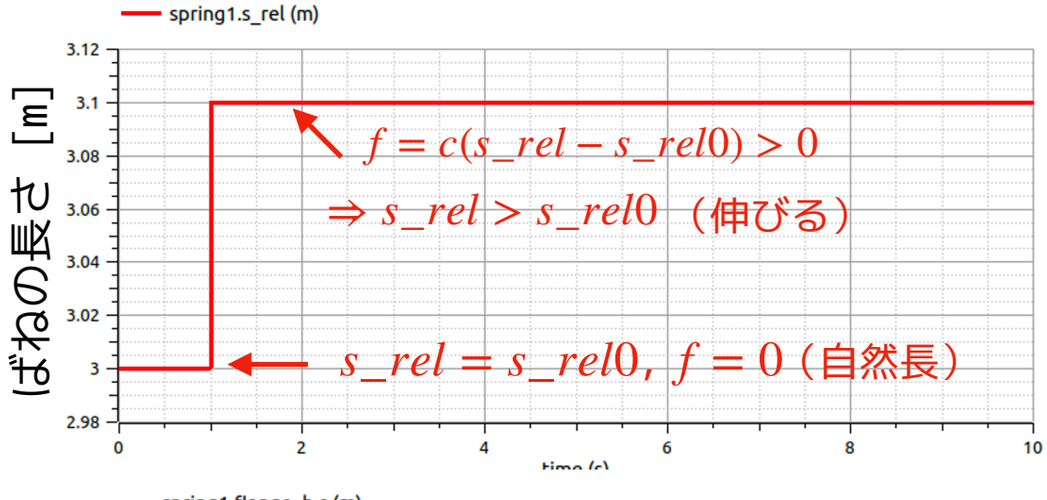


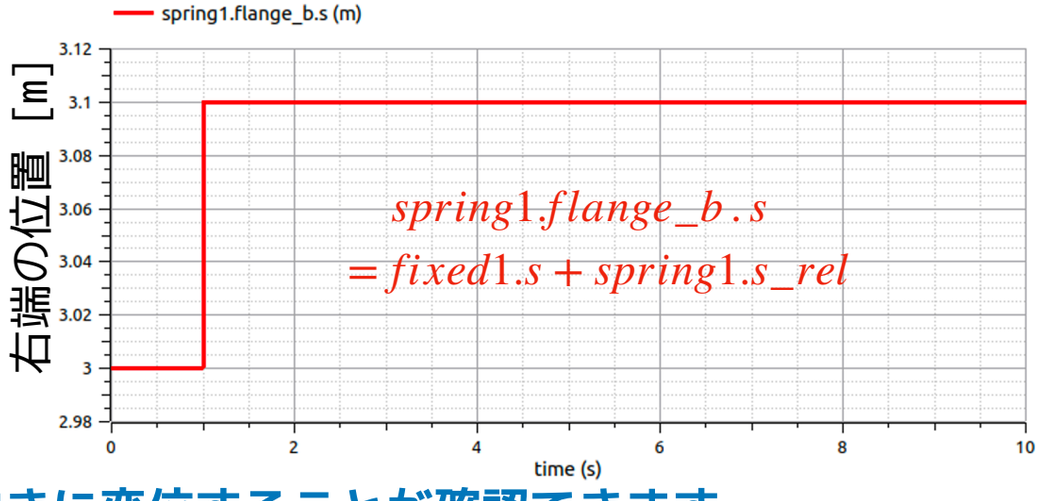




spring1



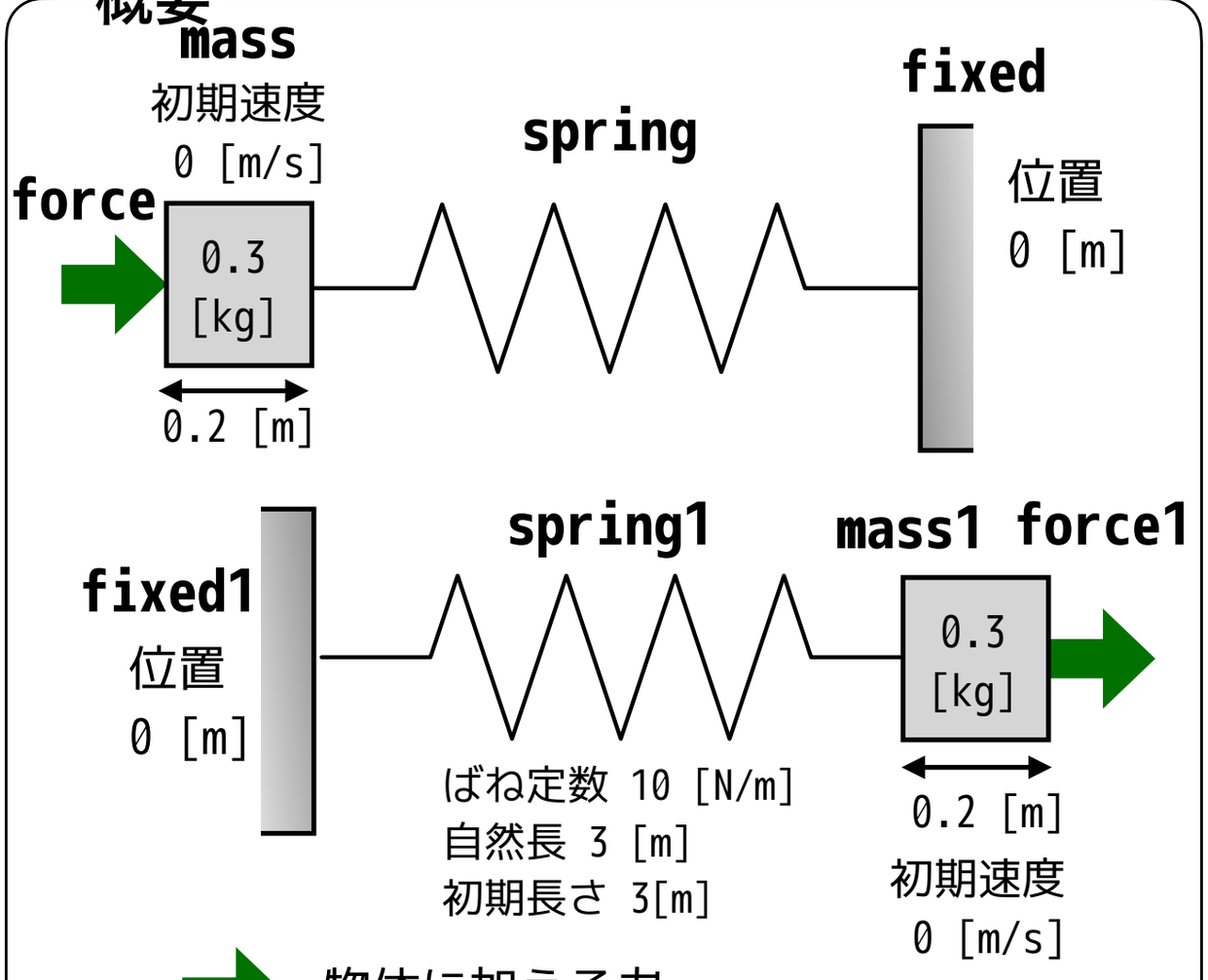




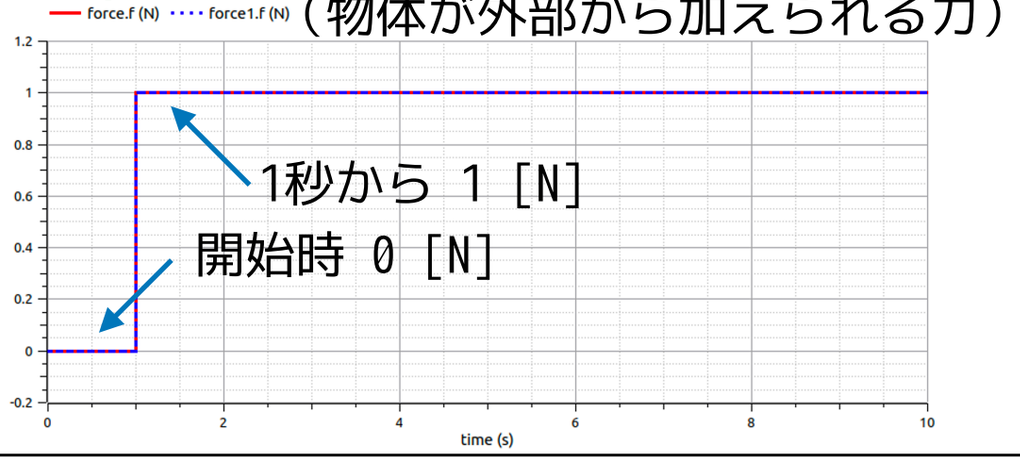
どちらも正の力を加えると正の向きに変位することが確認できます。

Example5 ばねがついた物体を押す。引っ張る。

概要

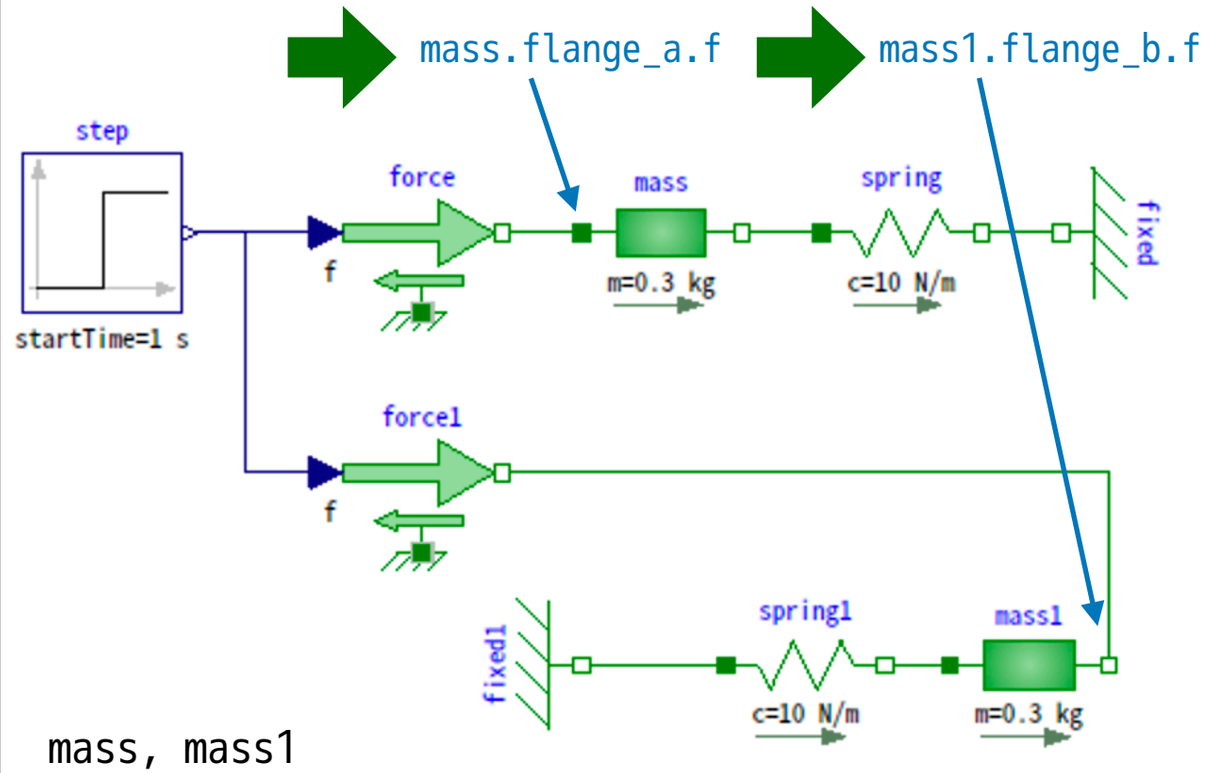


物体に加える力（物体が外部から加えられる力）



モデル

Example4 に mass, mass1 を追加します。



mass, mass1

```
m = 0.3 [kg] 質量
L = 0.2 [m]  長さ
v.start = 0 [m/s] 初期速度
v.fixed = true    初期速度を設定する。
```

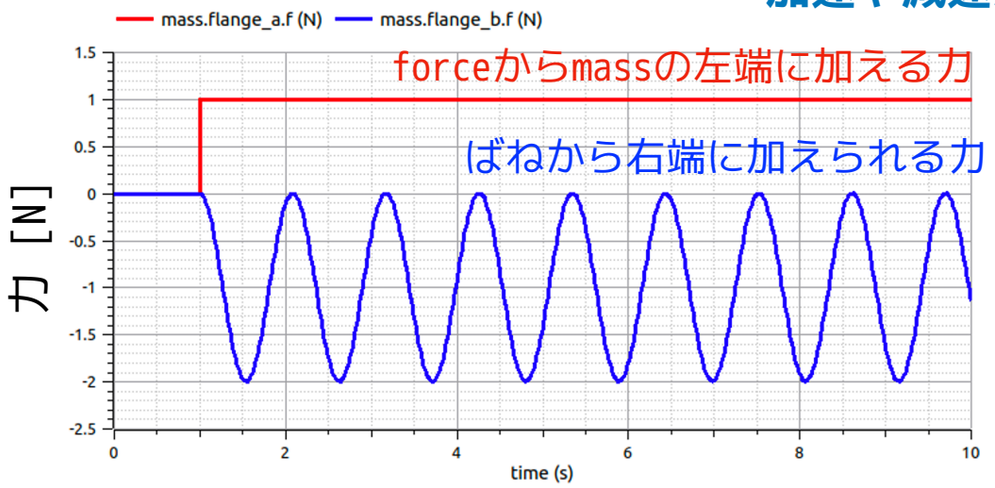
spring, spring1

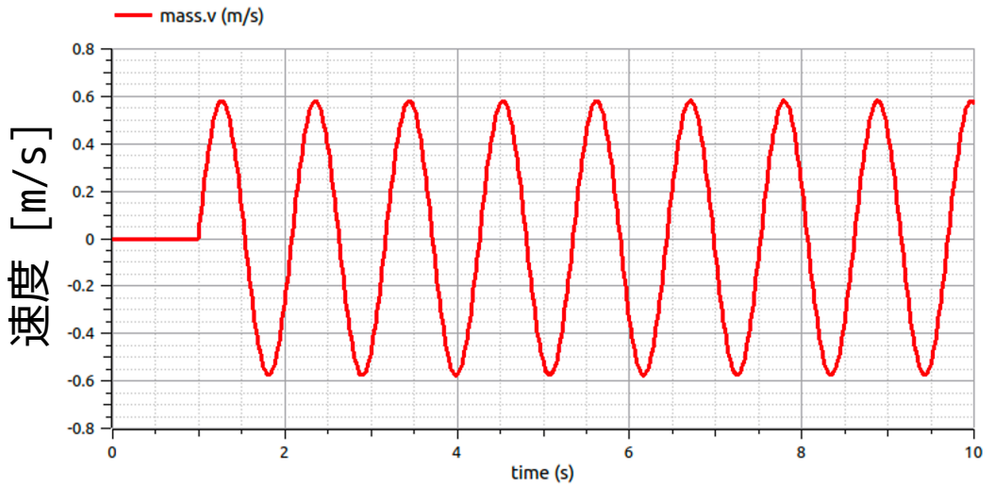
```
c = 10 [N/m]          ばね定数
s_rel0 = 3 [m]        自然長
s_rel.start = 3 [m] 初期長さ
s_rel.fixed = true  初期長さを設定する
```

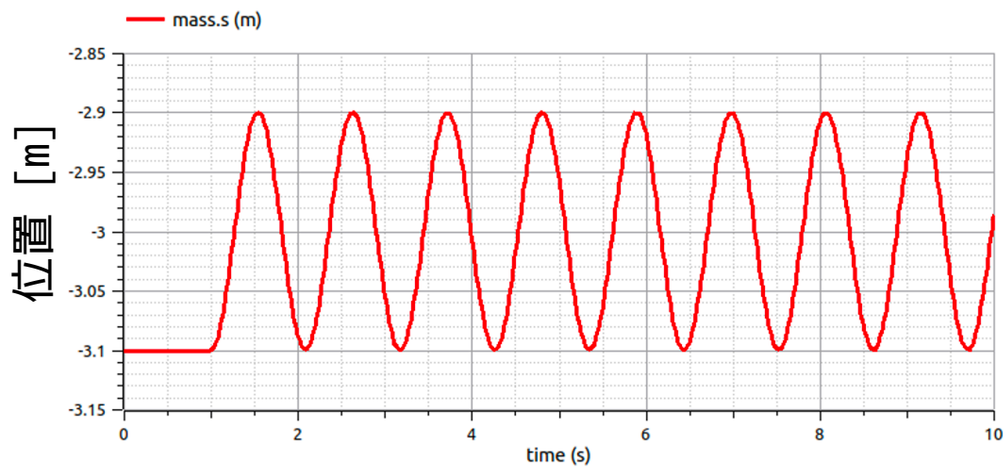
シミュレーション結果

Massの慣性とSpringの力によって
加速や減速が生じ物体は振動します。

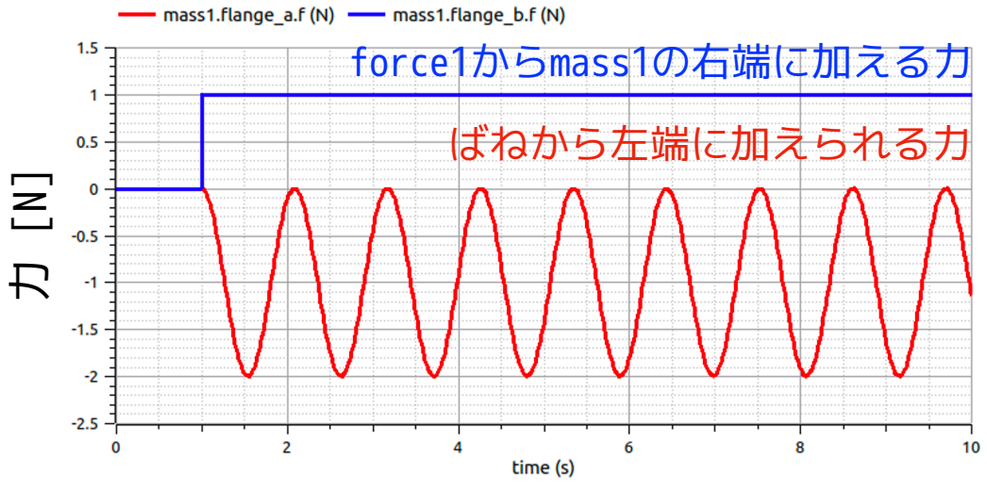
mass

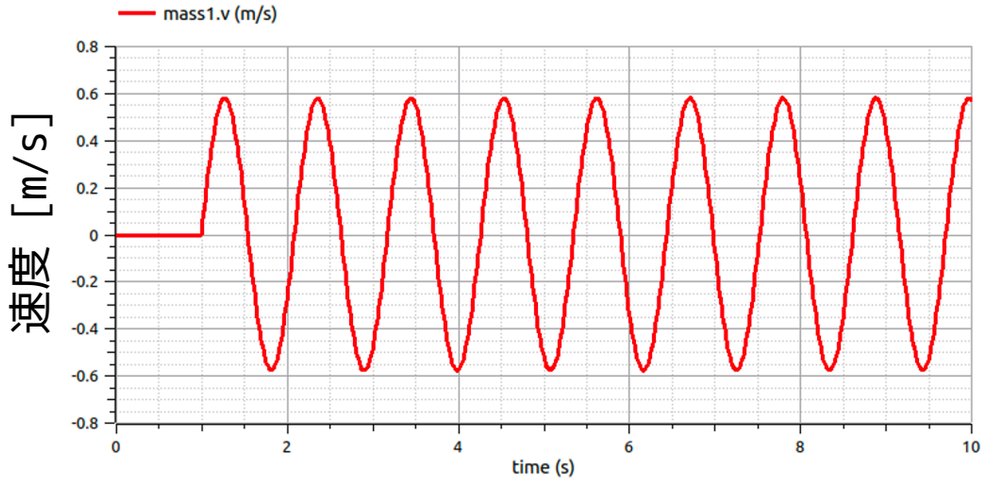


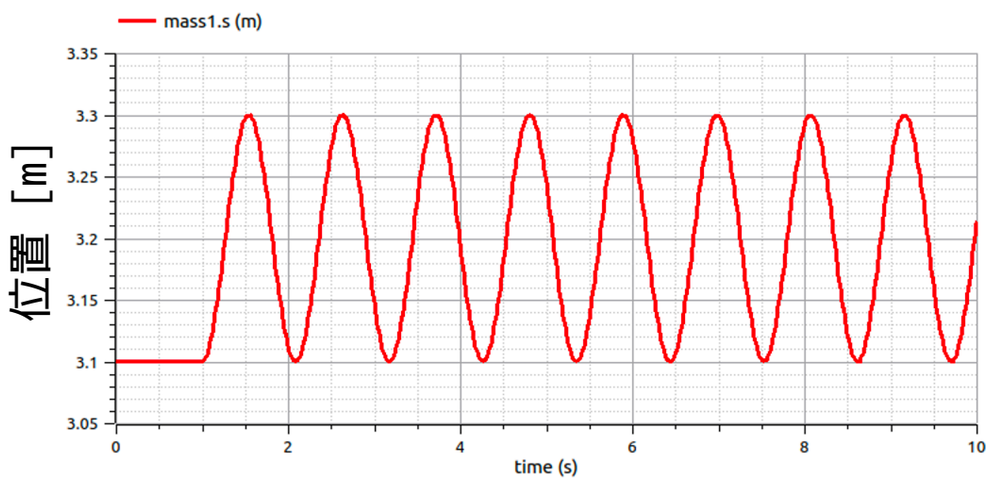




mass1

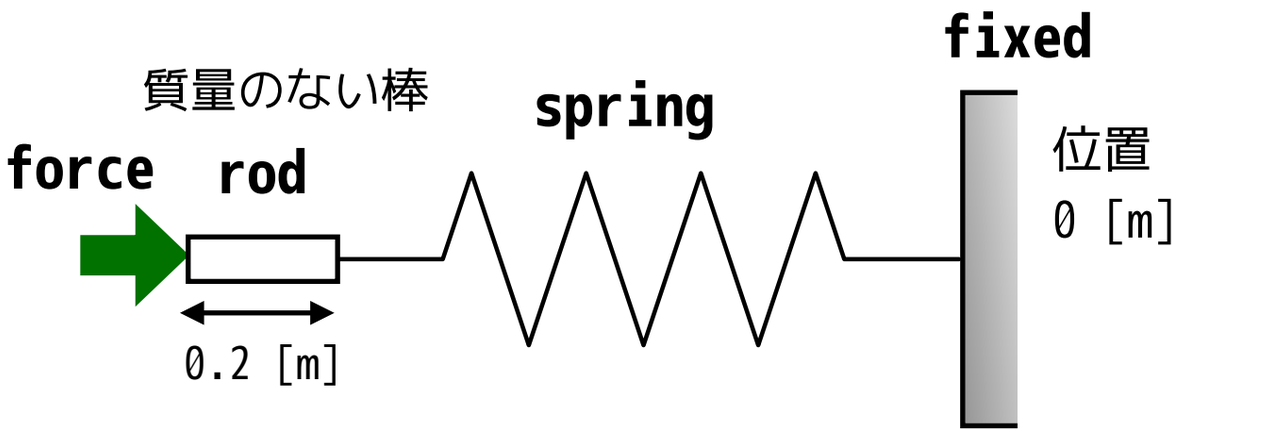


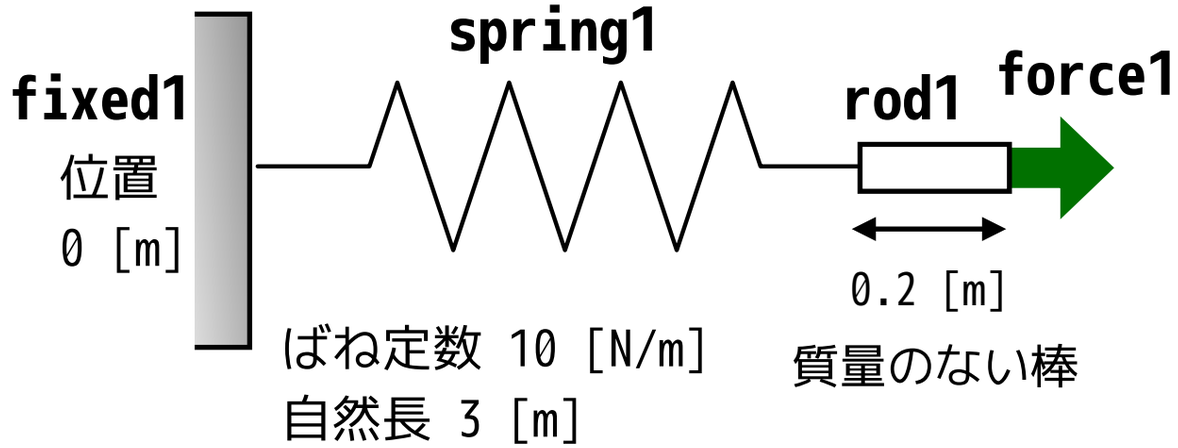




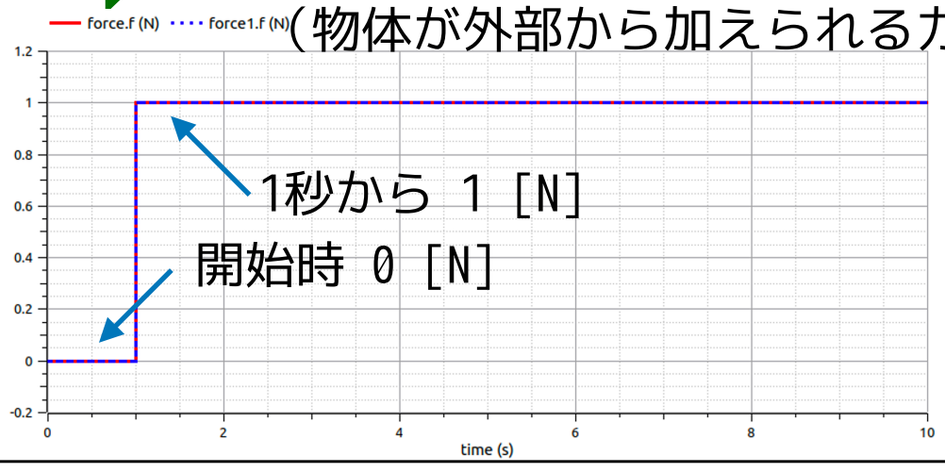
Example6 質量の無い棒でばねを押す。引っ張る。

概要





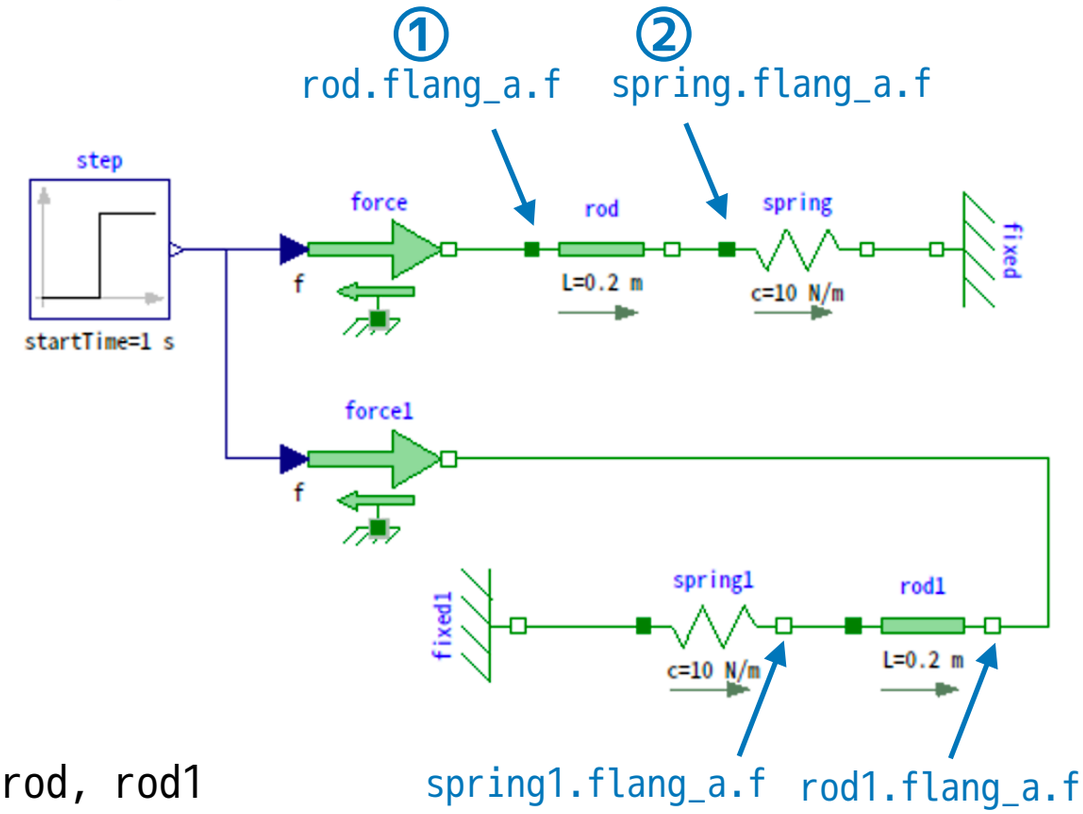
物体に加える力



（物体が外部から加えられる力）

モデル

Example4に rod, rod1 を追加します。



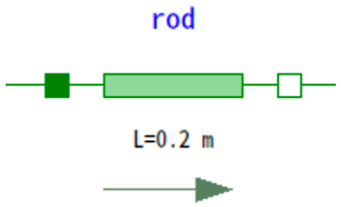
rod, rod1

L = 0.2 [m]　長さ

spring, spring1

c = 10 [N/m]	ばね定数
s_rel0 = 3 [m]	自然長
s_rel.fixed = false	初期長さを設定しない

Rod — 質量が無視できる棒のモデル



パラメータ L で指定した距離だけ作用点の位置をずらすモデル
複数の作用点を持つ物体などで、作用点間の相対位置を設定するときに使用します。

構成と方程式

2個のFlangeをもつモデル

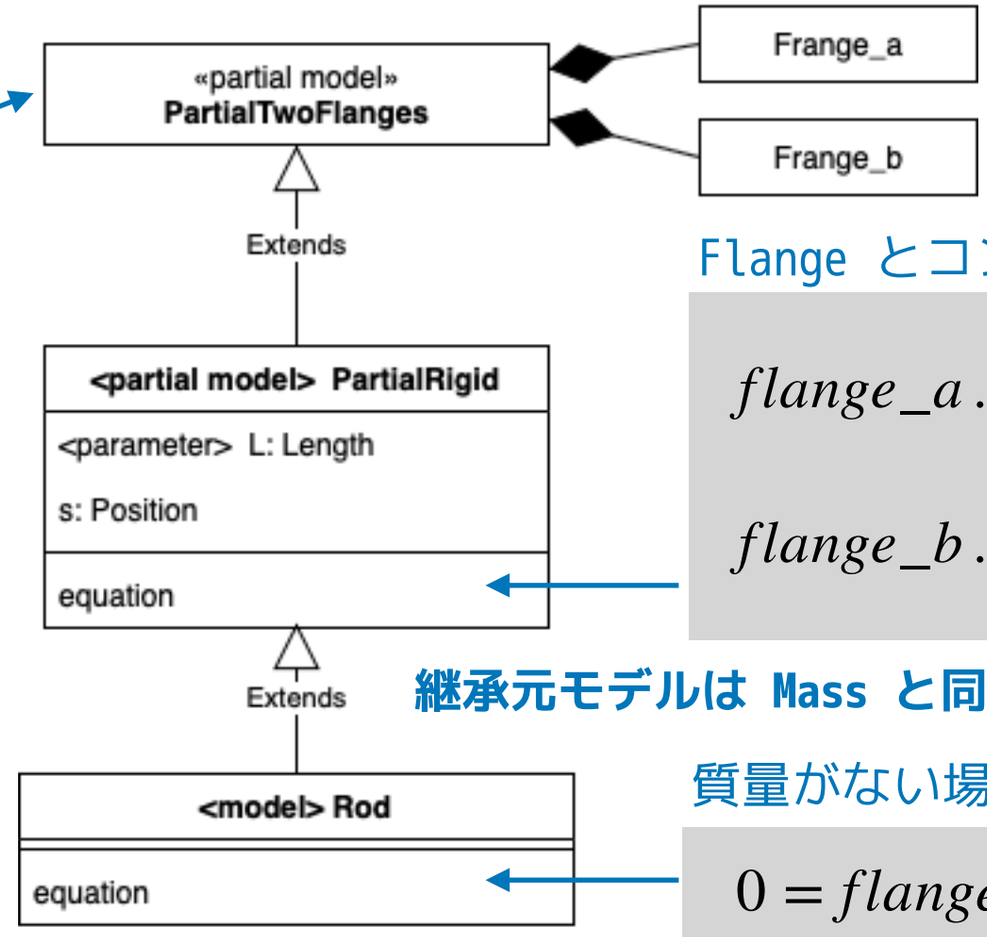
$$\begin{bmatrix} flange\_a.s & [\mathrm{m}] \\ flange\_a.f & [\mathrm{N}] \end{bmatrix}$$

$$\begin{bmatrix} flange\_b.s & [\mathrm{m}] \\ flange\_b.f & [\mathrm{N}] \end{bmatrix}$$

大きさのある硬いものを表すモデル

s:棒の中心位置 [m]

L: 棒の長さ [m]



Flange とコンポーネントの位置関係

$$flange\_a.s = s - \frac{L}{2}$$

$$flange\_b.s = s + \frac{L}{2}$$

継承元モデルは Mass と同じです。

質量がない場合の力のつり合い

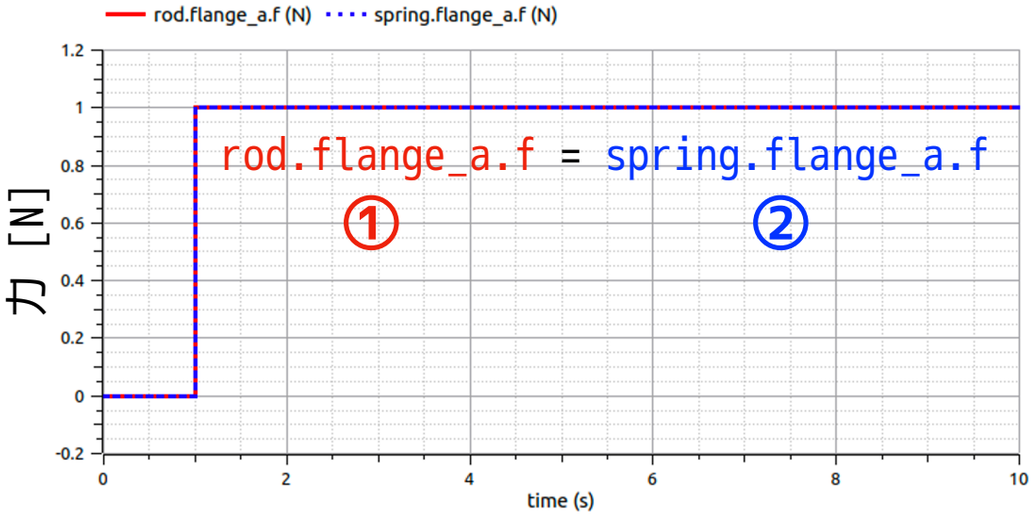
$$0 = flange\_a.f + flange\_b.f$$

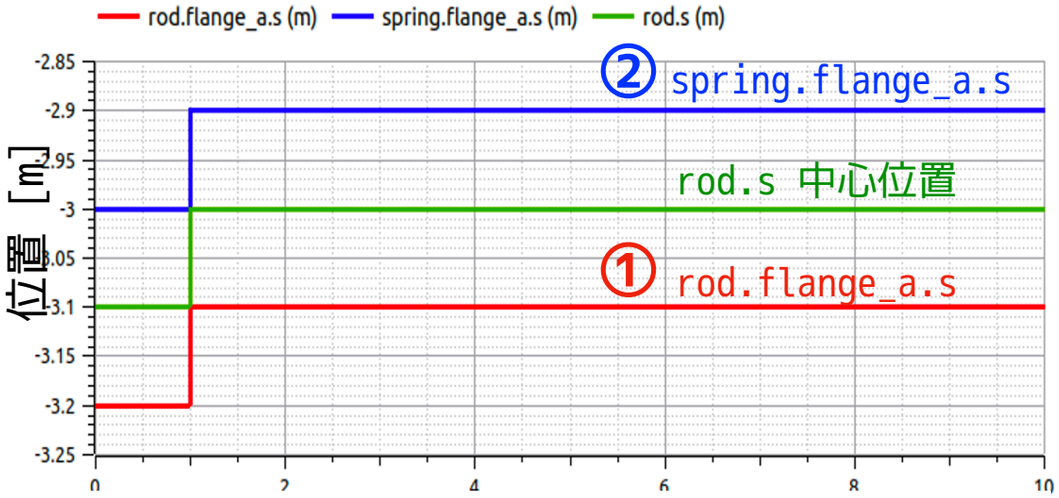
このように継承元のモデルを再利用することによって、
新しいモデルの開発が容易になります。

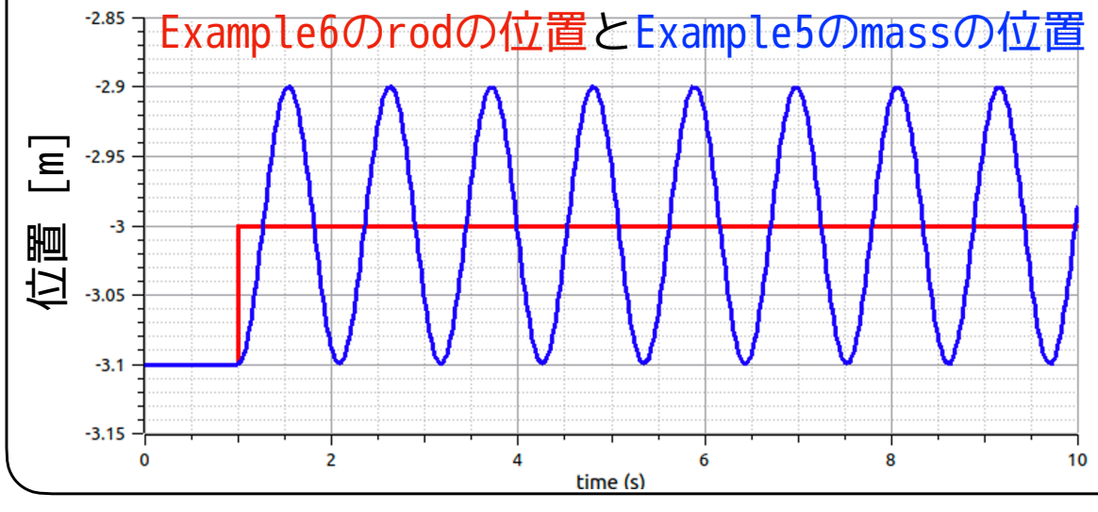
シミュレーション結果

質量による慣性が無いので振動しません！

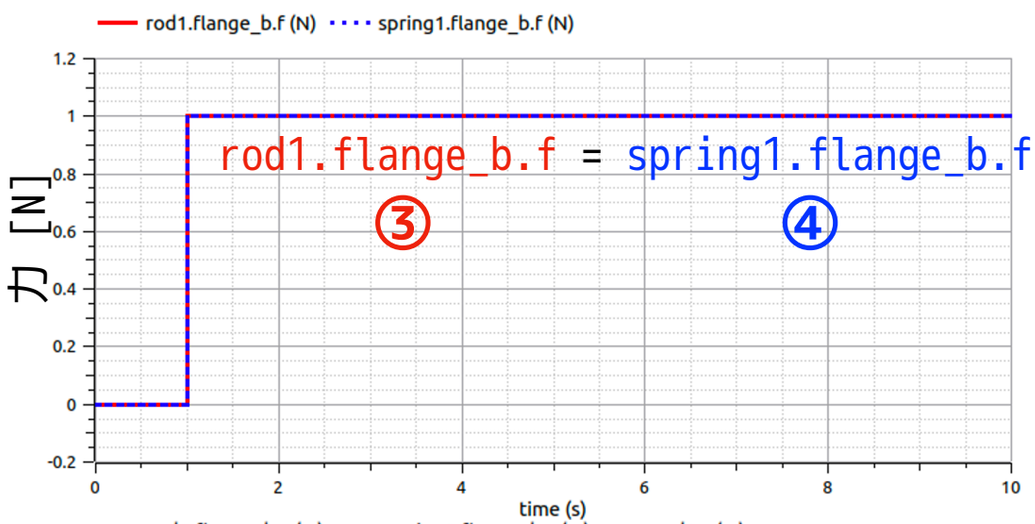
rod

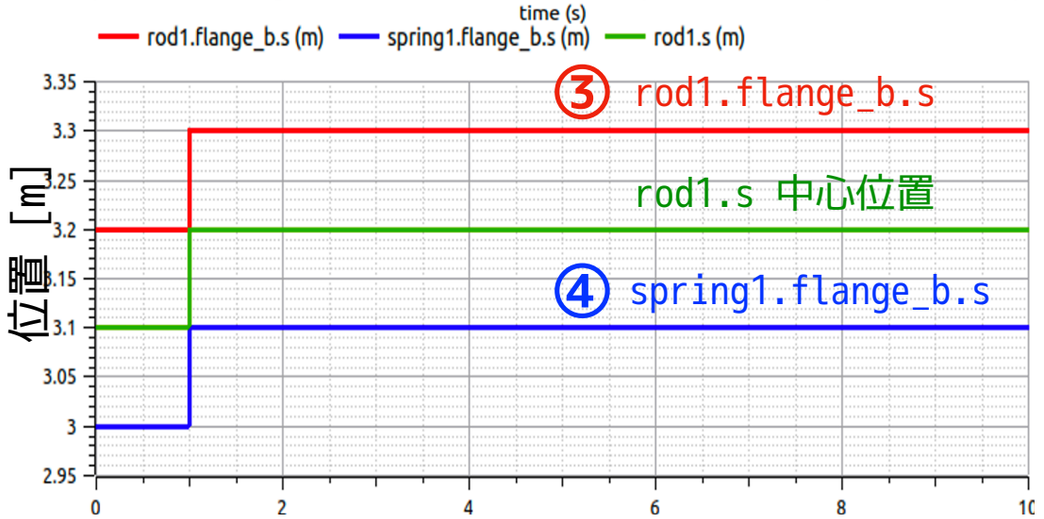


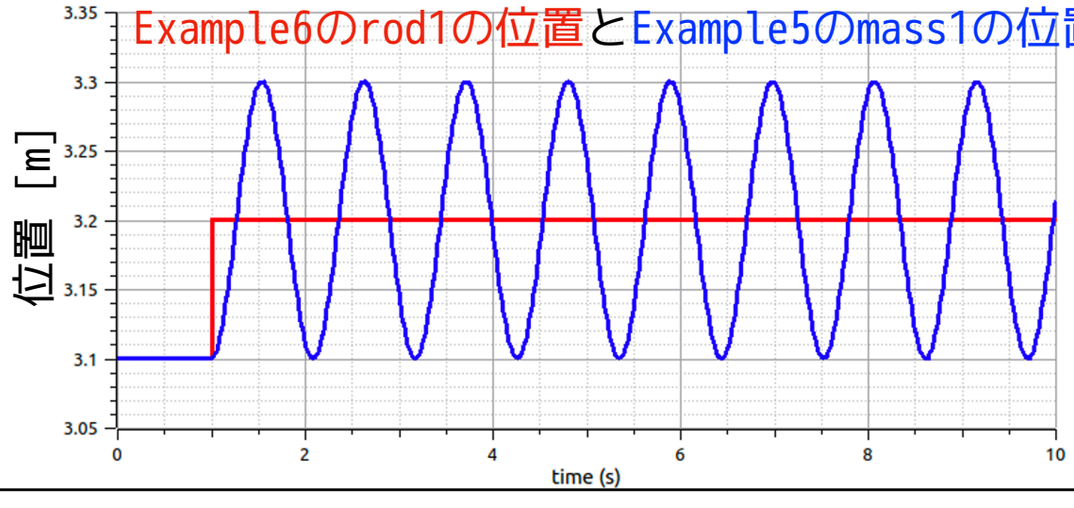




rod1

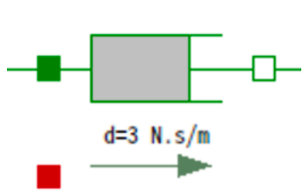




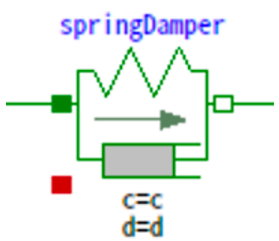


摩擦力のコンポーネント

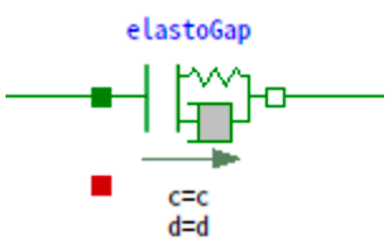
Damper



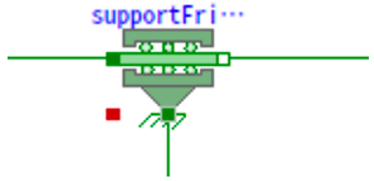
SpringDamper



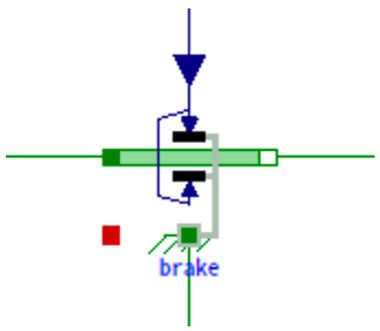
ElastoGap



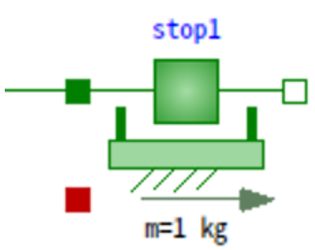
SupportFriction



Brake

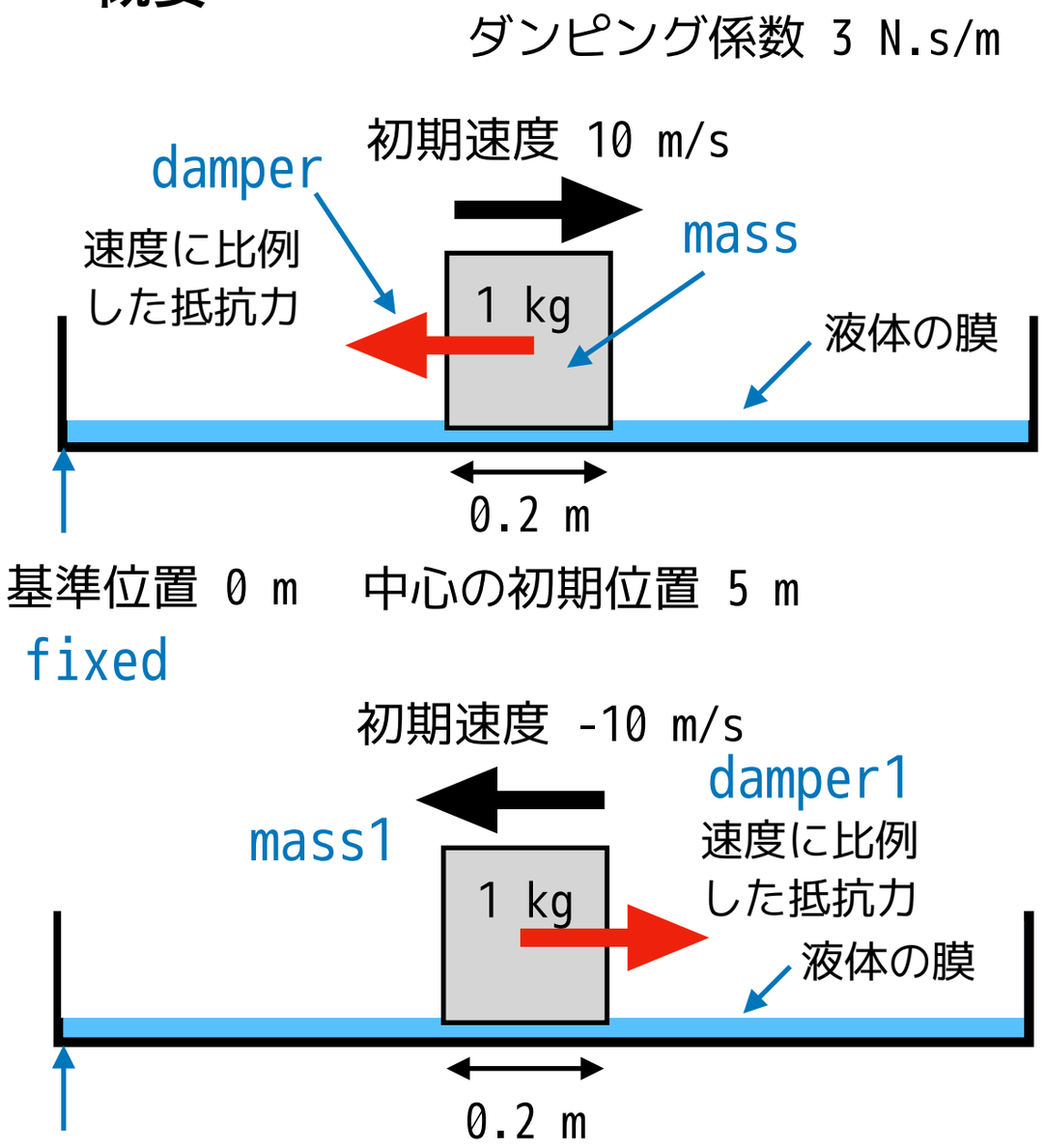


MassWithStopAndFriction



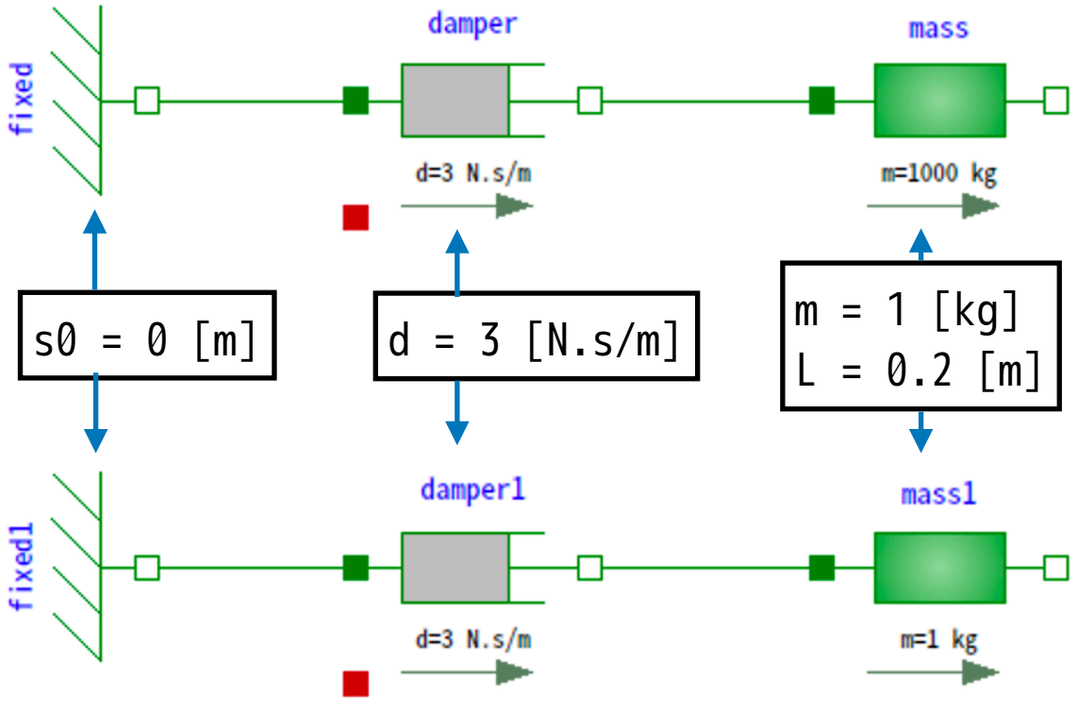
Example7 運動している物体に粘性抵抗力を加えて止める。

概要



液体の膜の上で物体を滑らせます。

モデル



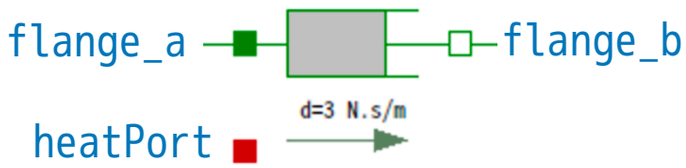
mass

```
v.fixed = true      初期速度を設定する。
v.start = 10 [m/s]  初期速度
s.fixed = true      初期位置を設定する。
s.start = 5 [m]     初期位置
```

mass1

```
v.fixed = true      初期速度を設定する。
v.start = -10 [m/s] 初期速度
s.fixed = true      初期位置を設定する。
s.start = 5 [m]     初期位置
```

Damper — 速度に比例した抵抗力のモデル

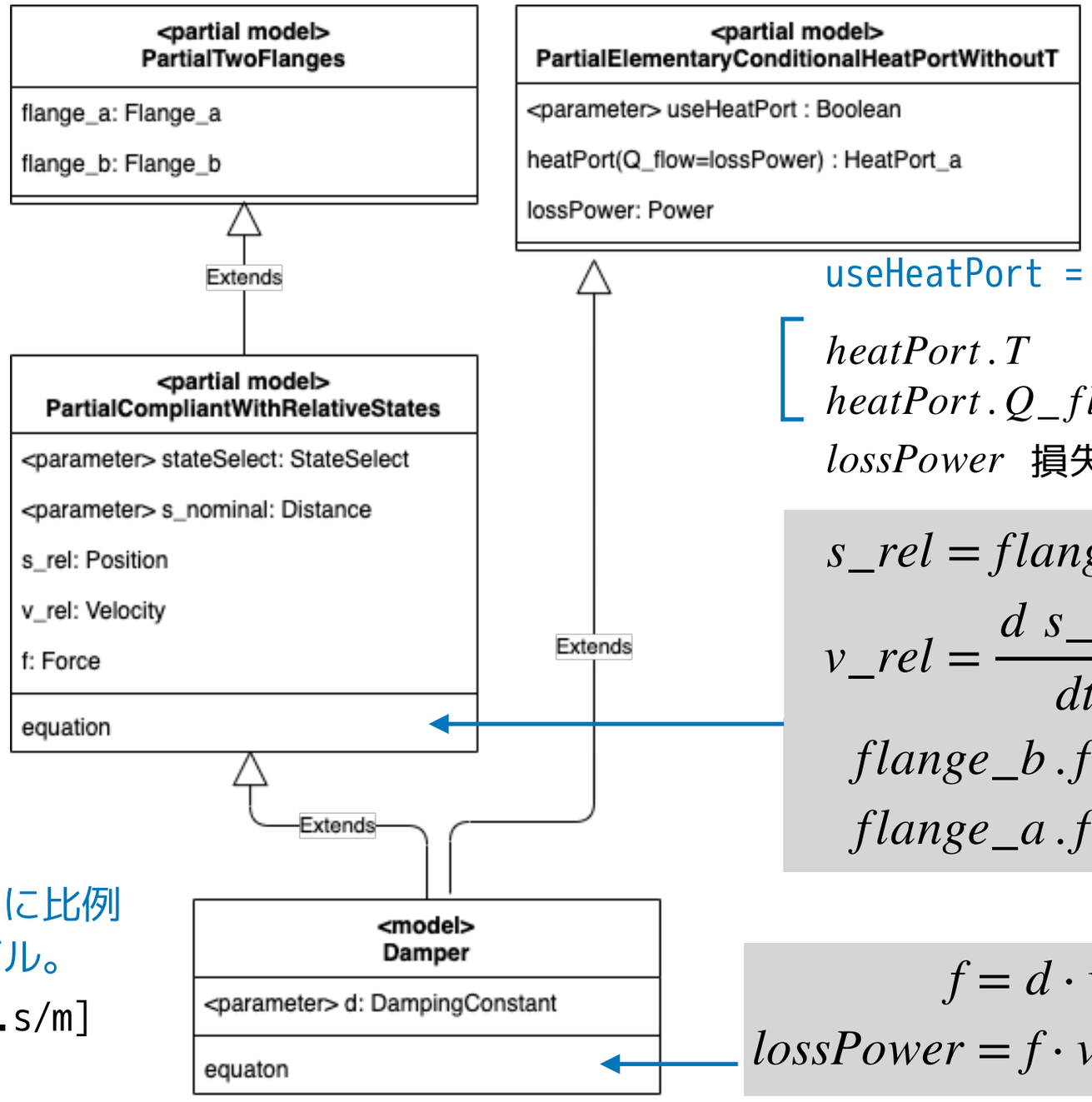


Frange間の相対速度が正のとき（伸びるとき）
$v\_rel > 0,\ flange\_a.f < 0,\ flange\_b.f > 0$
→flange_a は外部に正の力を加え、flange_b は外部に負の力を加えます。

構成と方程式

2個のFlangeをもつモデル

$flange\_a.s$ [m]
$flange\_a.f$ [N]
$flange\_b.s$ [m]
$flange\_b.f$ [N]



損失される仕事率を heatPortの熱流量として出力できるモデル

useHeatPort = true のとき 有効になる

$heatPort.T$　温度
$heatPort.Q\_flow = lossPower$　熱流量
$lossPower$　損失される仕事率

Flangeの相対位置を状態変数とするモデル

s_rel 相対位置
v_rel 相対速度
f 抵抗力

$$s\_rel = flange\_b.s - flange\_a.s$$
$$v\_rel = \frac{d\ s\_rel}{dt}$$

相対位置
相対速度

$$flange\_b.f = f$$
$$flange\_a.f = -f$$

Flangeと抵抗力の関係

ダンパーのモデル
粘性抵抗など相対速度に比例した抵抗力を示すモデル。
d ダンピング係数 [N.s/m]

$$f = d \cdot v\_rel$$ 抵抗力
$$lossPower = f \cdot v\_rel$$ 損失される仕事率

ダンピング係数の見積もり方法

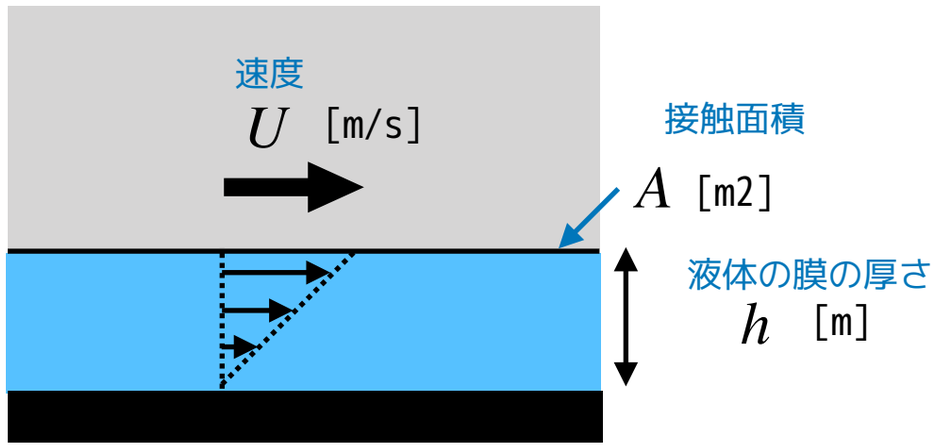
粘性率[Pa.s]　ずり速度[1/s]

ずり応力[Pa]

$$\tau = \mu \frac{\partial U}{\partial y} \sim \mu \frac{U}{h}$$

$$f = A\tau = \frac{\mu A}{h} U$$

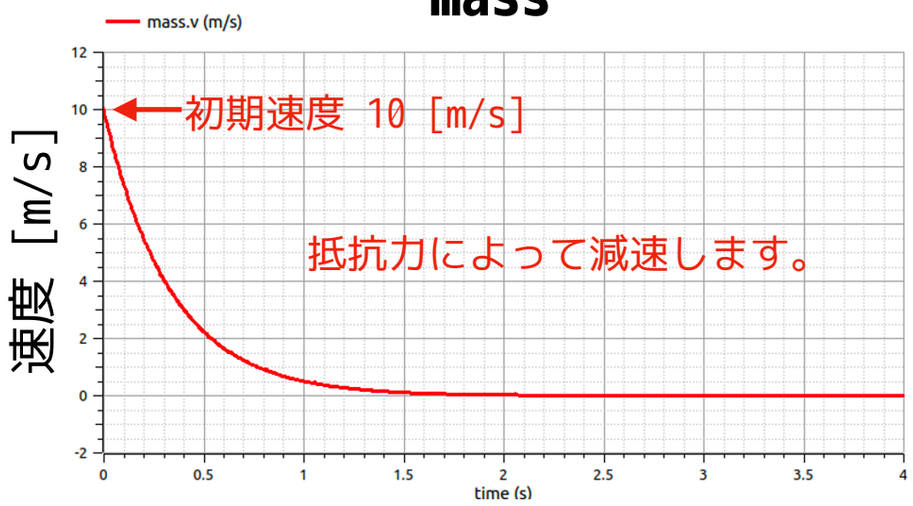
ダンピング係数　$d = \frac{\mu A}{h}$ [N.s/m]

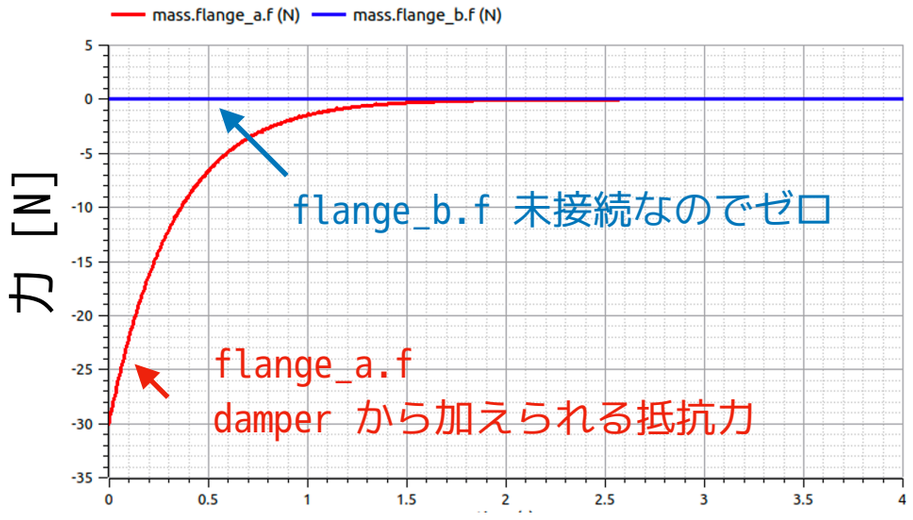


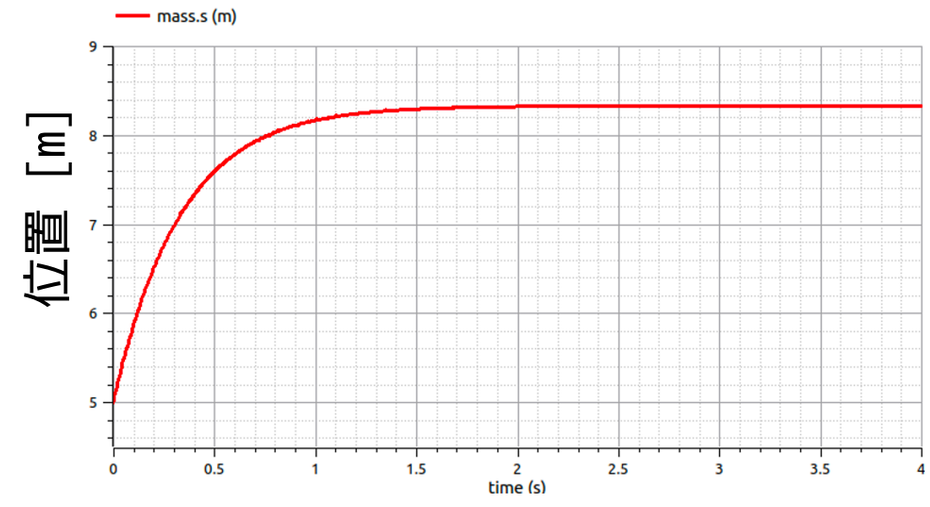
層流(クウェット流)
を仮定する。

シミュレーション結果

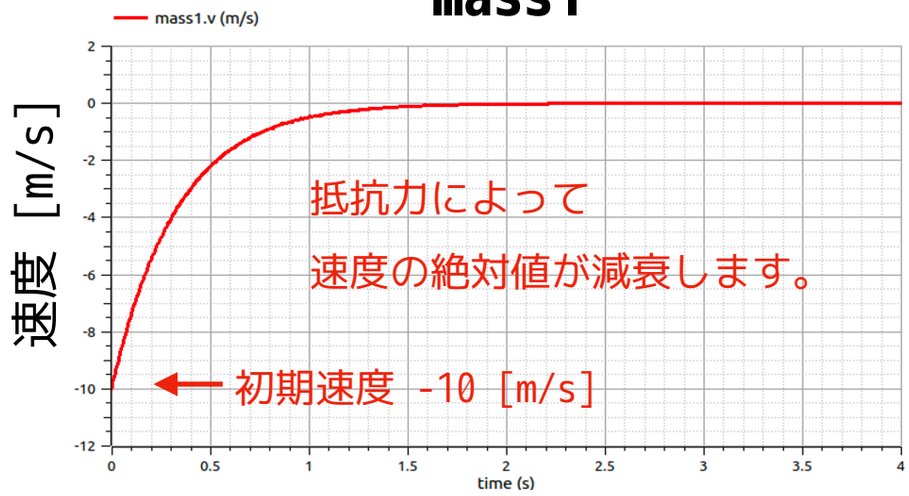
mass

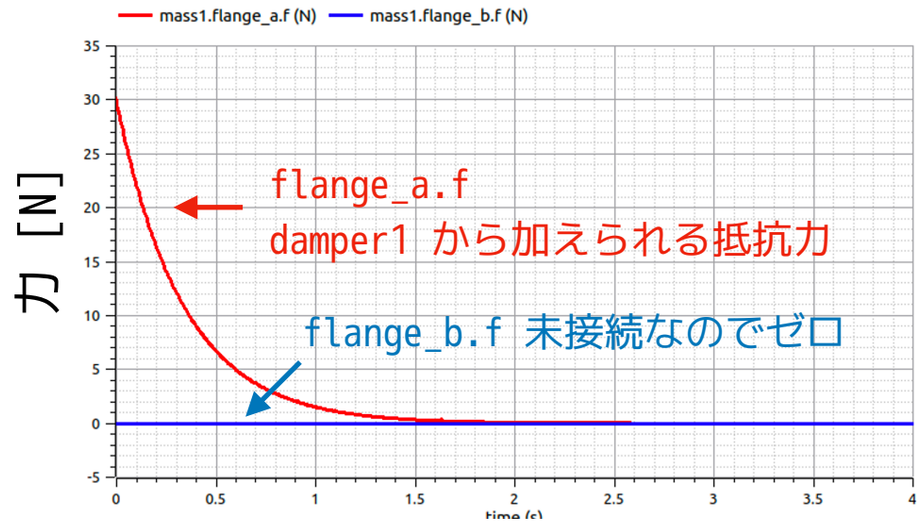


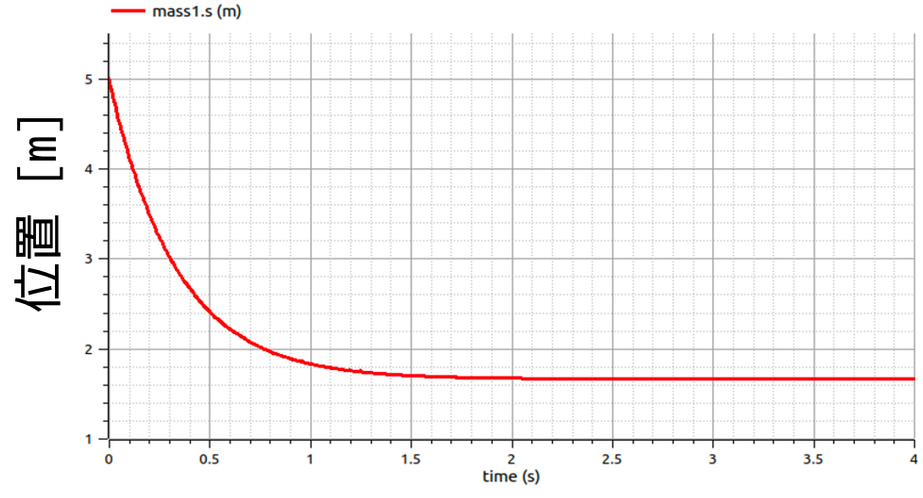




mass1

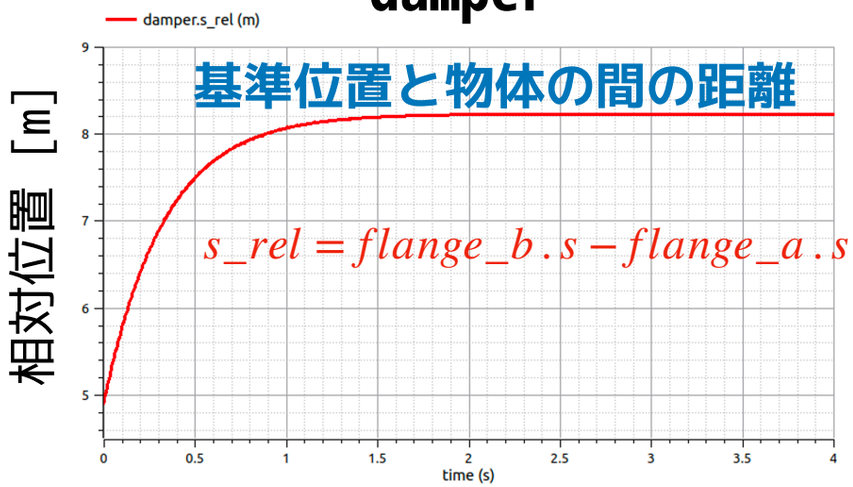


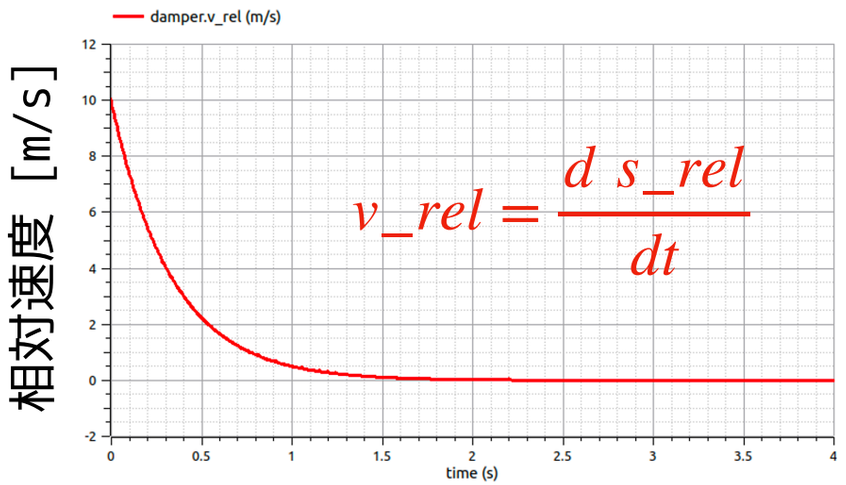


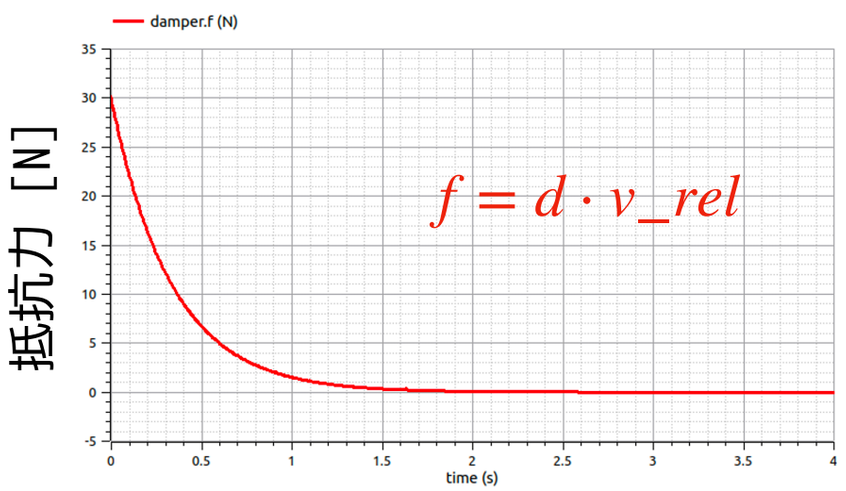


シミュレーション結果

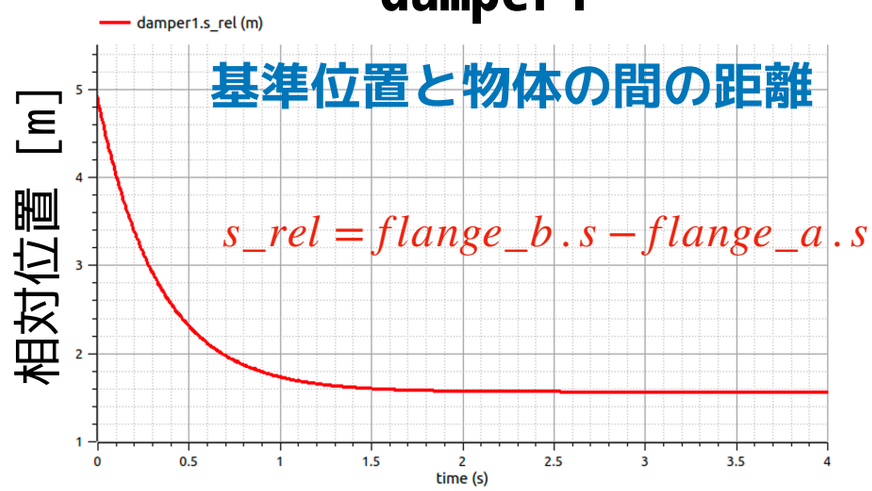
damper

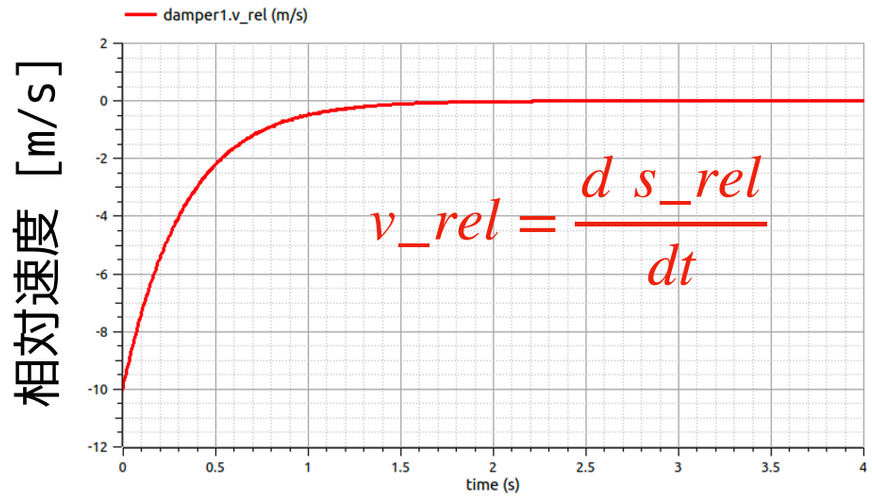


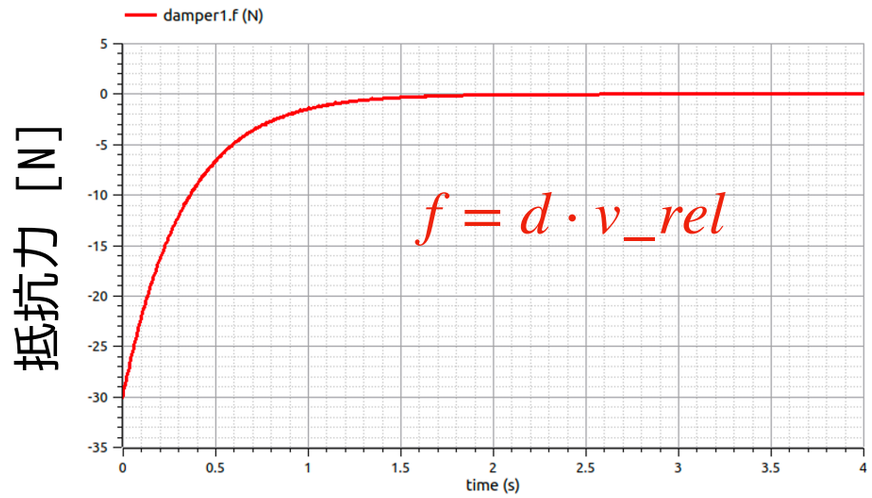




damper1

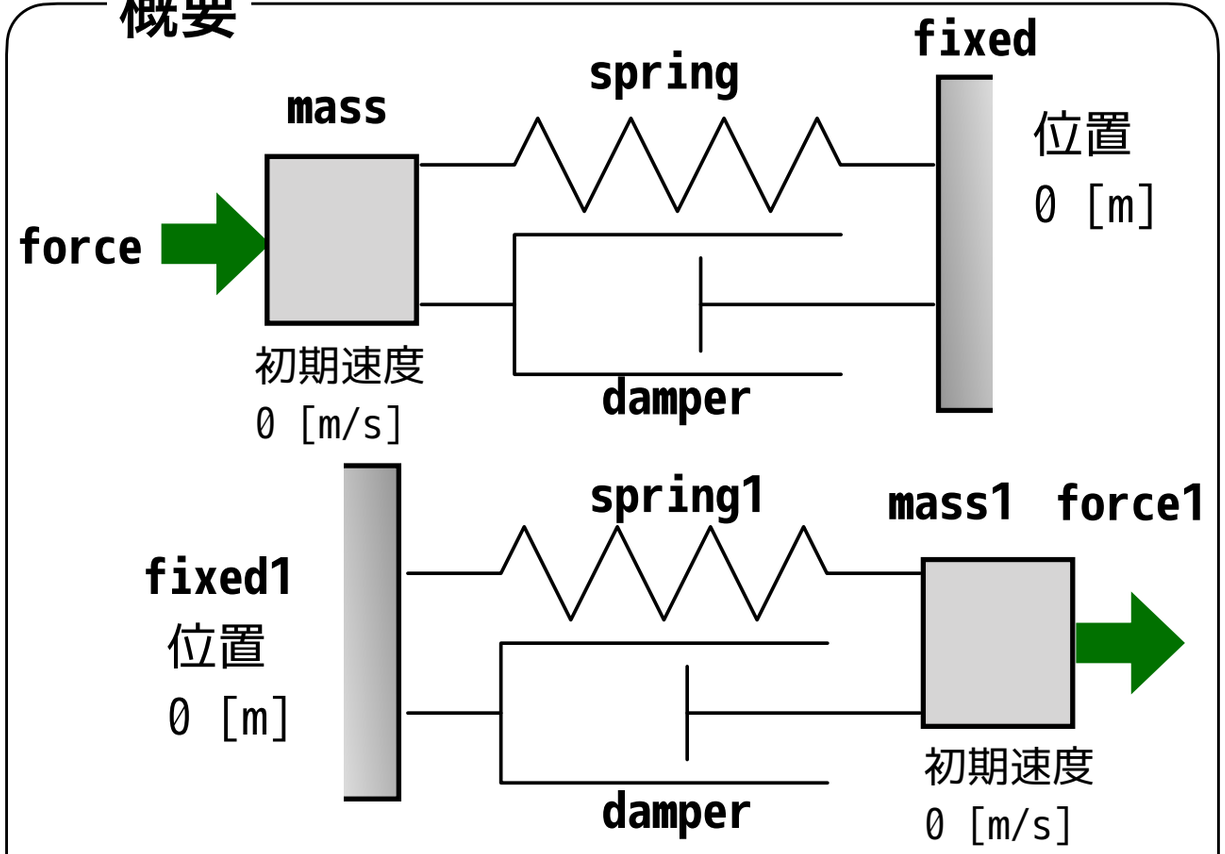






Example8 ばねにダンパをつける

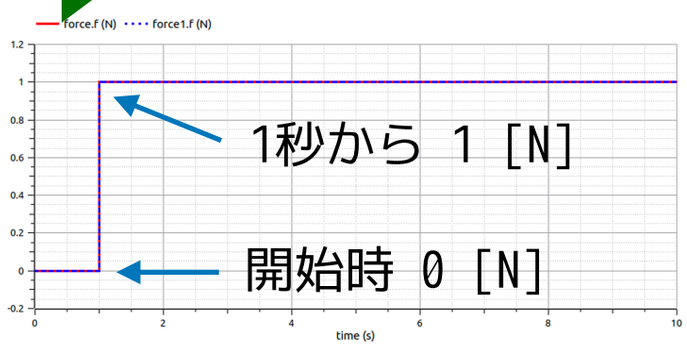
概要



ダンピング係数 0.2 [N.s/m]

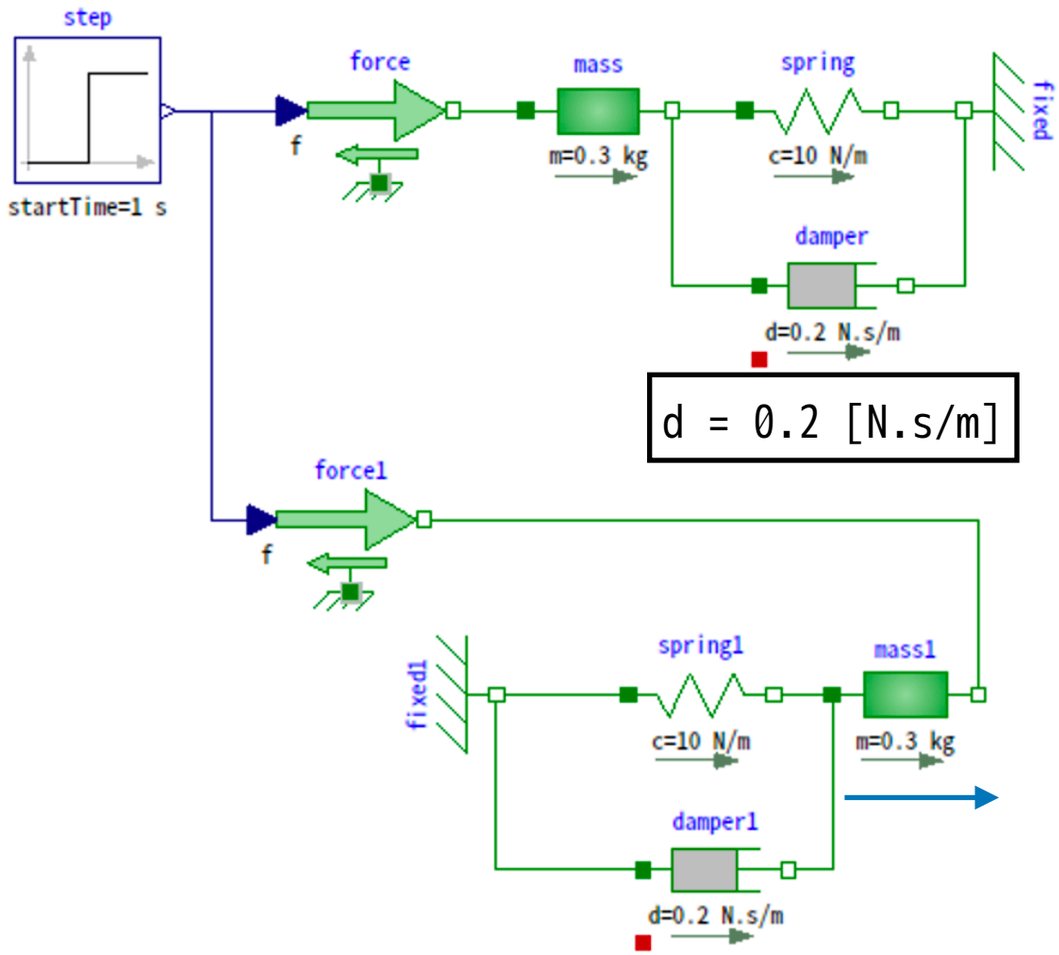
ばね定数 10 [N/m]
自然長 3 [m]
初期長さ 3 [m]

物体に加える力



モデル

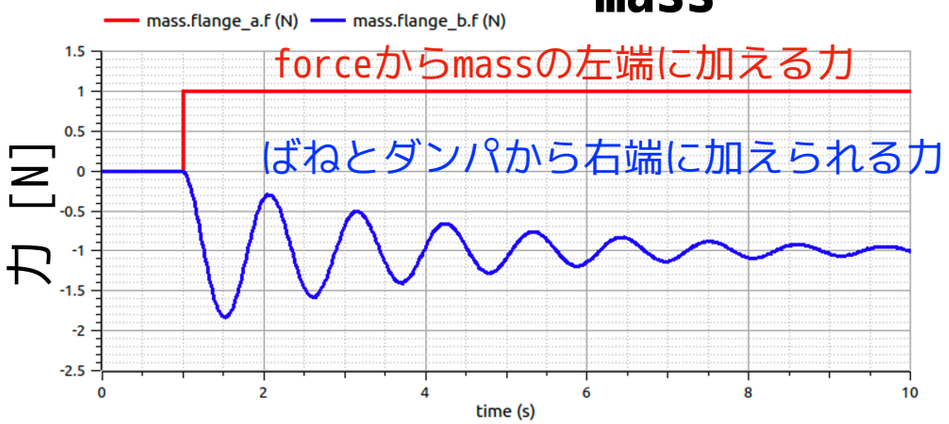
Example5 の spring, spring1 に
それぞれ damper, damper1 を並列につなぎます。

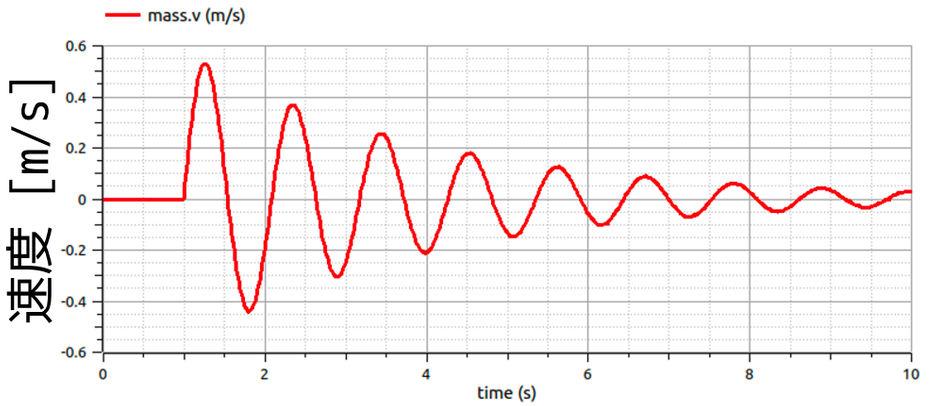


シミュレーション結果

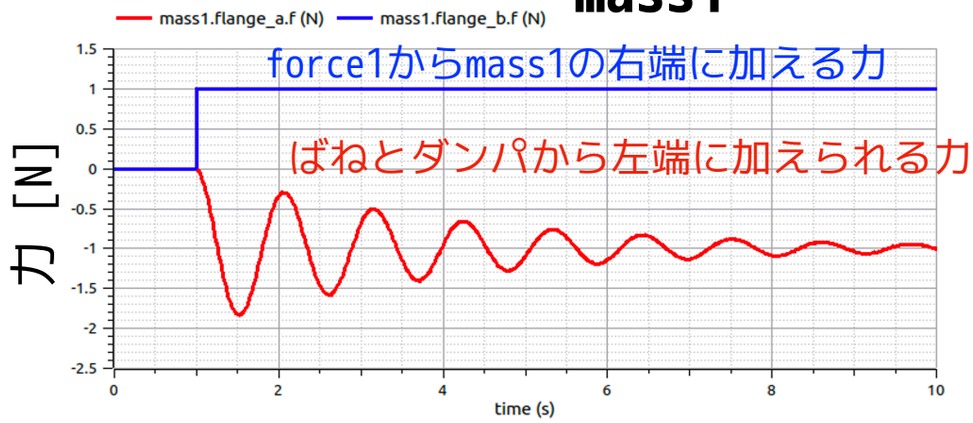
Damperの効果により振動が減衰します。

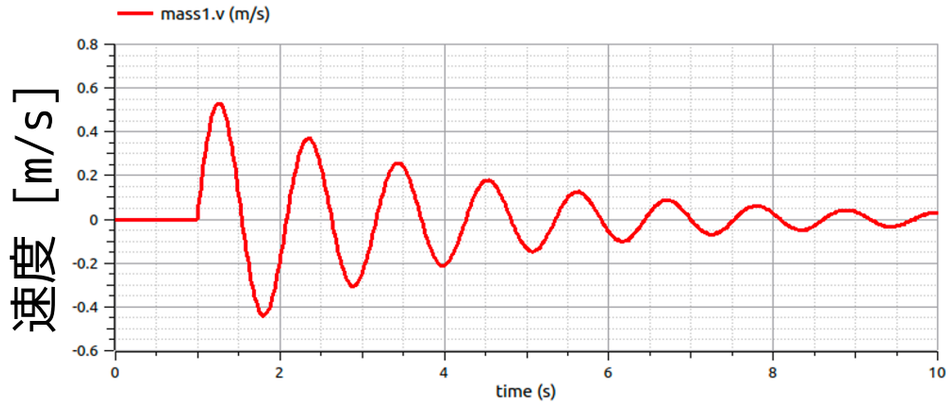
mass



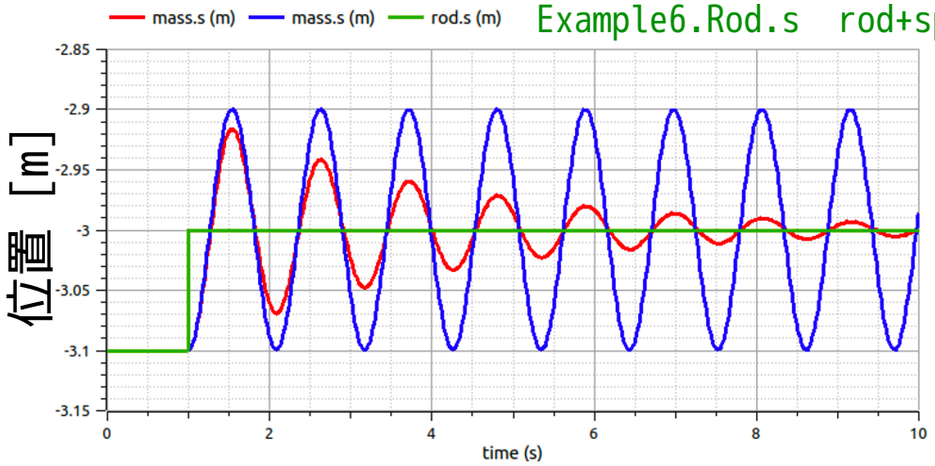


mass1

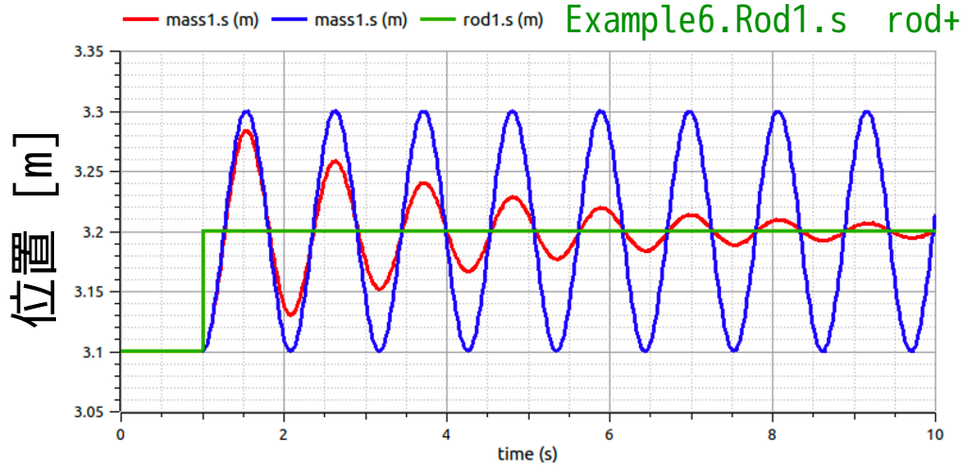




Example8.Mass.s mass+spring+damper
Example5.Mass.s mass+spring
Example6.Rod.s rod+spring

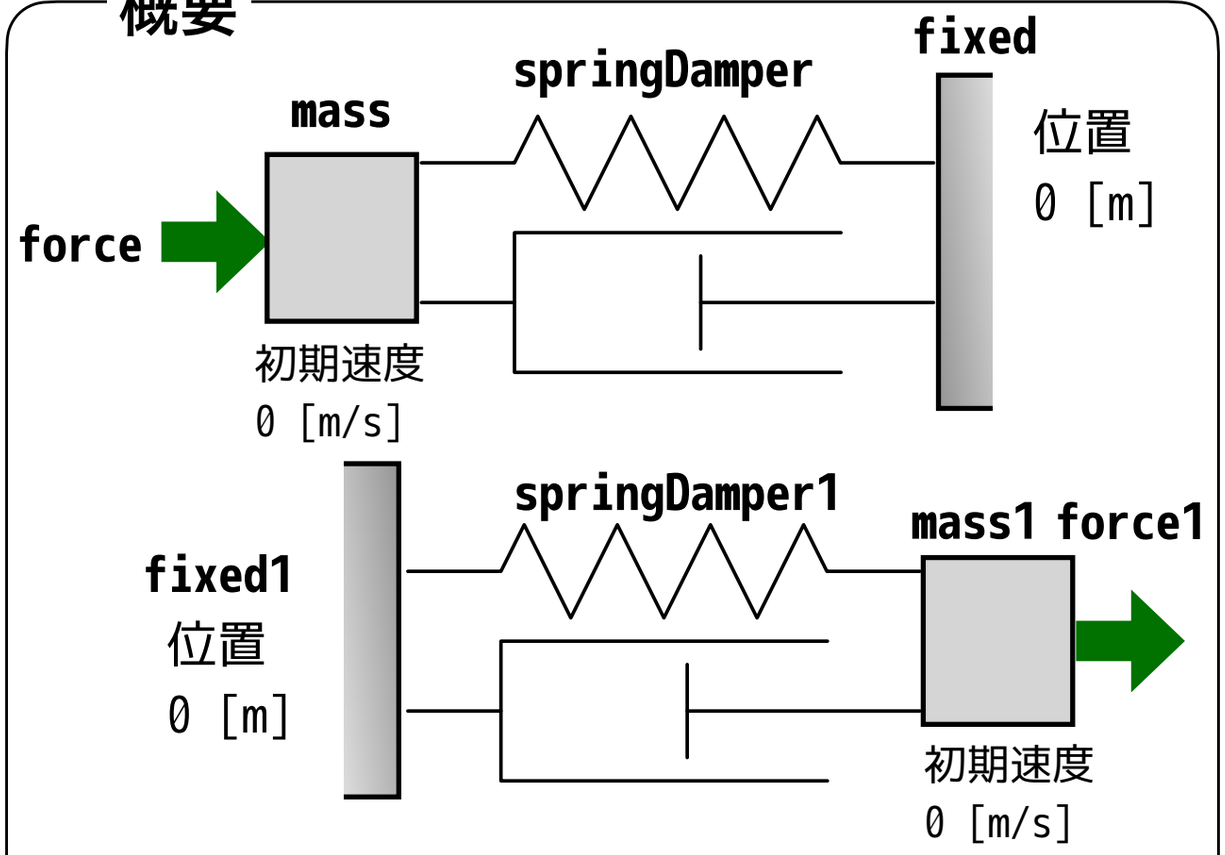


Example8.Mass1.s mass+spring+damper
Example5.Mass1.s mass+spring
Example6.Rod1.s rod+spring



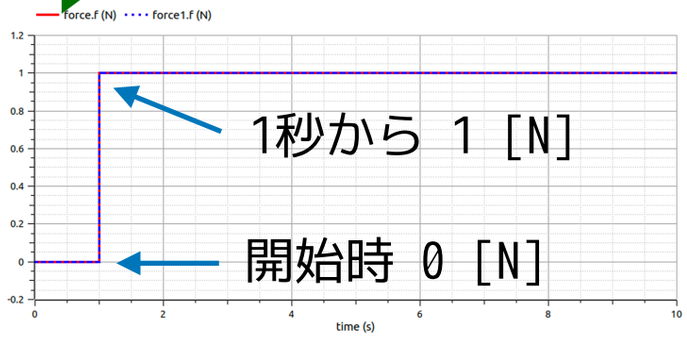
Example9 SpringDamper を使ってExample8を書き換える

概要



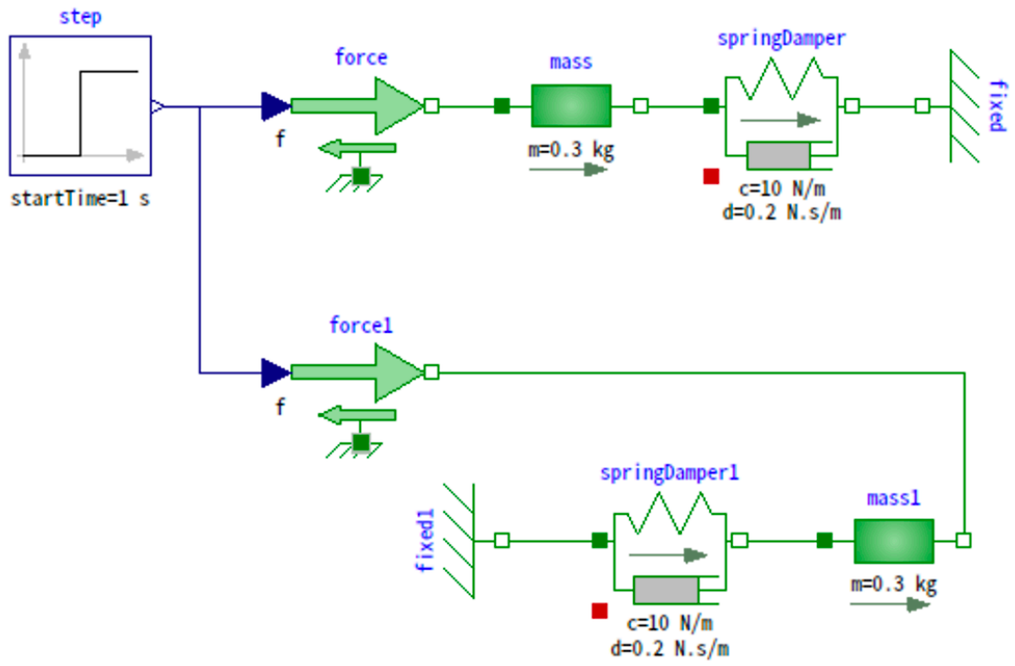
ダンピング係数 0.2 [N.s/m]　　ばね定数 10 [N/m]
自然長 3 [m]
初期長さ 3 [m]

物体に加える力



モデル

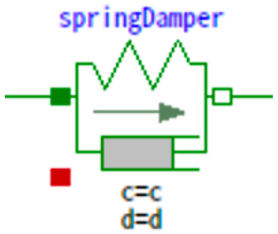
Example8 の spring, damper, spring1, damper1 を springDamper, springDamper1 に置き換える



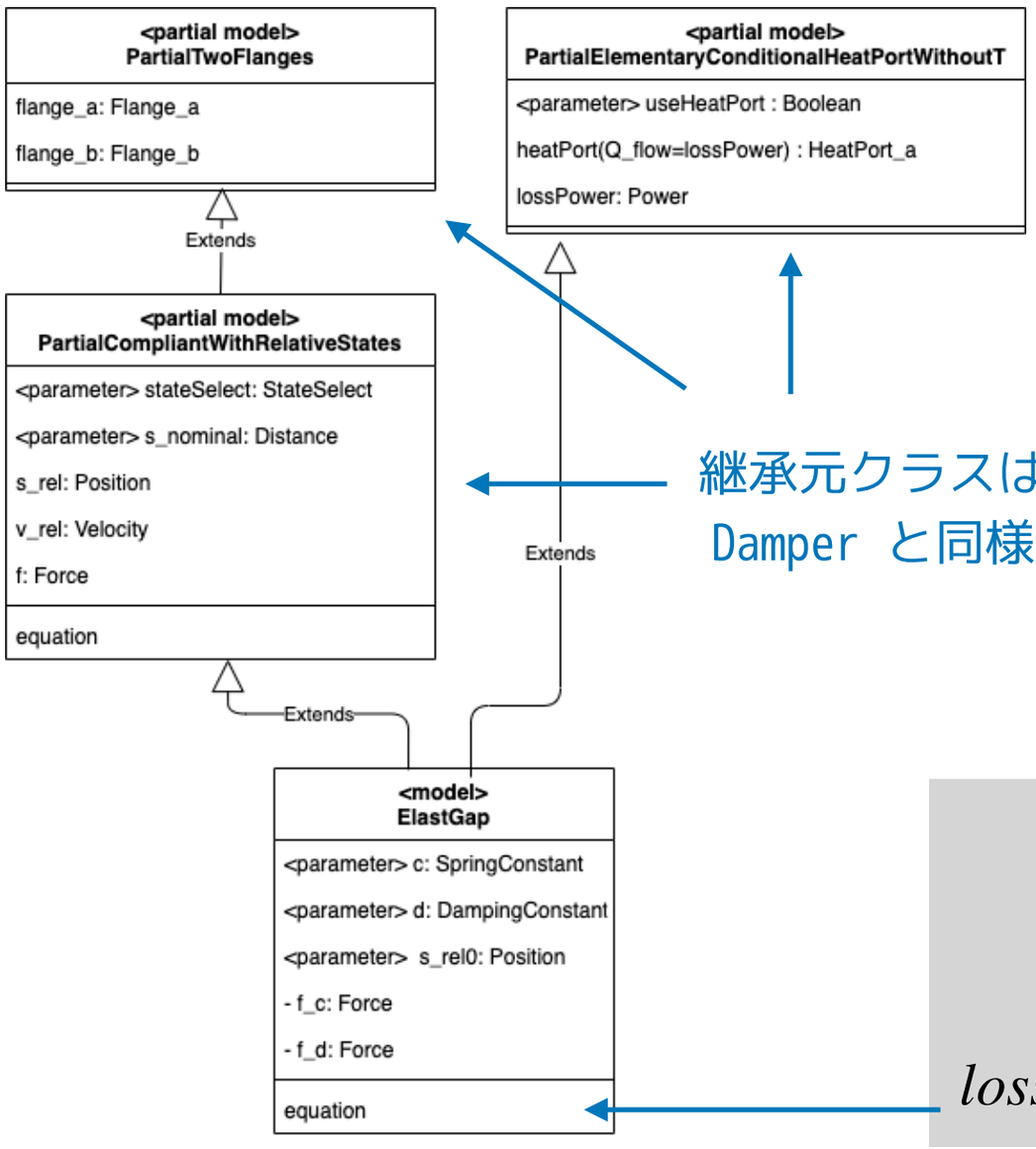
springDamper, springDamper1

```
c = 10 [N/m]     ばね定数
d = 0.2 [N.s/m]  ダンピング係数
s_rel0 = 3 [m]   自然長
s_rel.fixed = true    初期長さを設定する
s_rel.start = 3 [m]   初期長さ
```

SpringDamper ばねとダンパを並列接続したモデル



構成と方程式

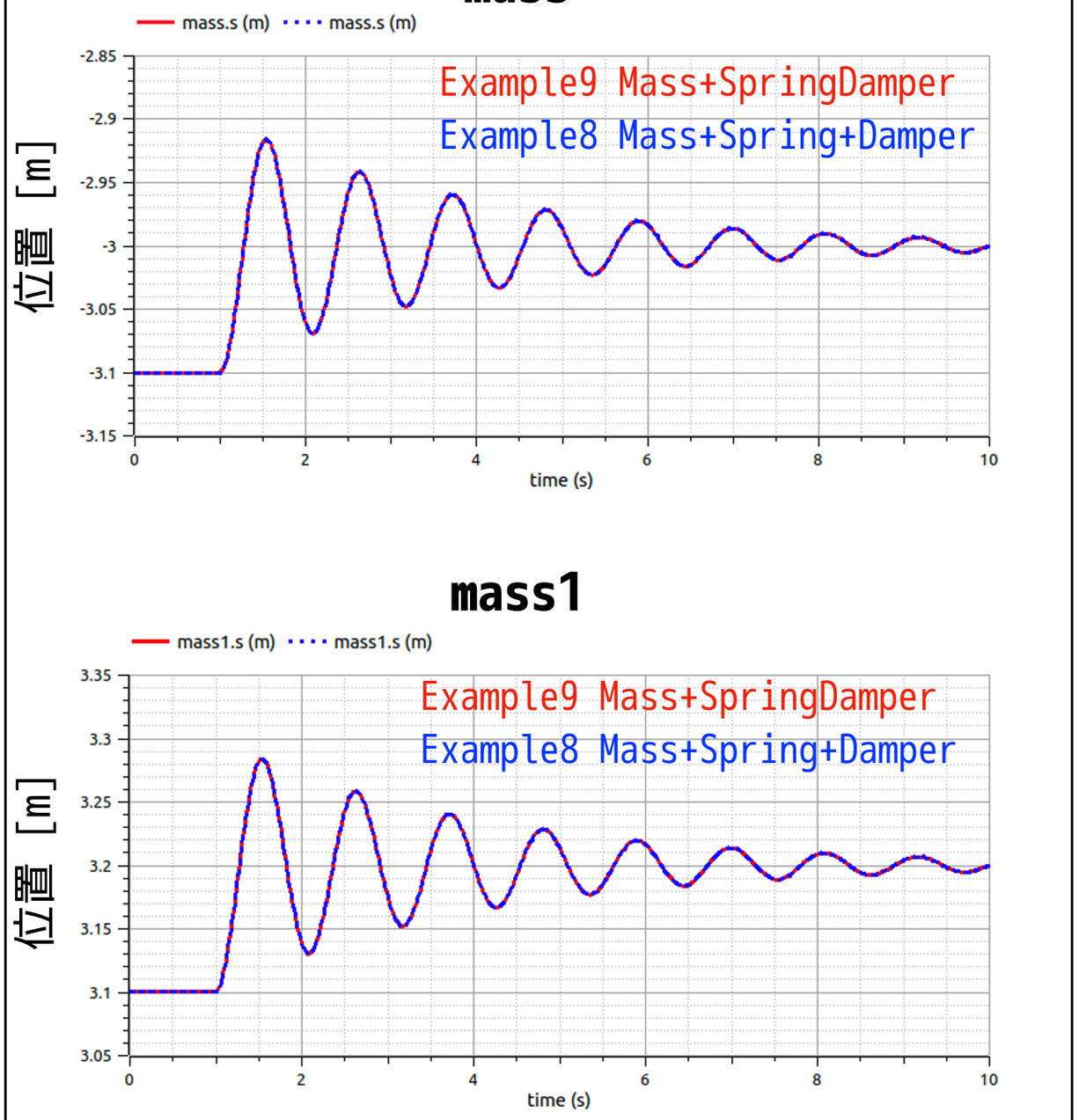


継承元クラスは
Damper と同様

パラメータ
s_rel0 ばねの自然長 [m]
c ばね定数 [N/m]
d ダンピング係数 [N.s/m]

$$f_c = c(s\_rel - s\_rel0)$$ ばねの弾力
$$f_d = d \cdot v\_rel$$ 抵抗力
$$f = f_c + f_d$$
$$lossPower = f_d \cdot v\_rel$$

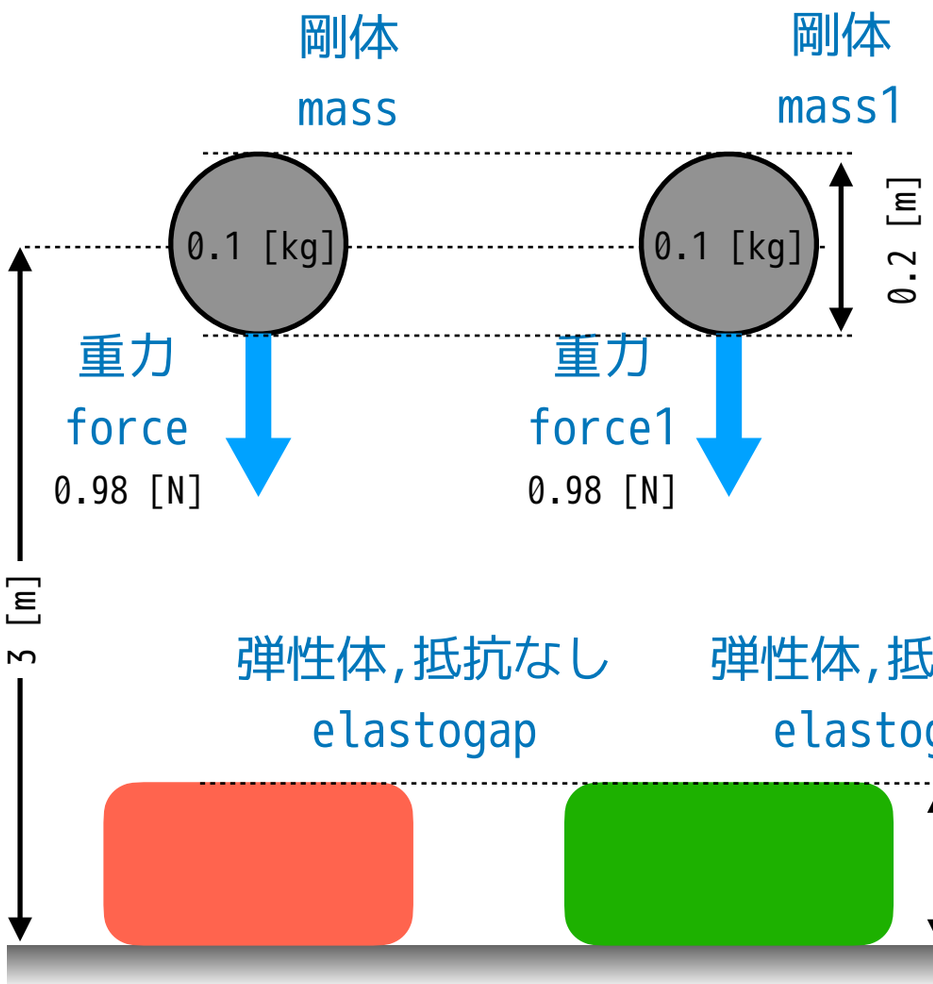
シミュレーション結果



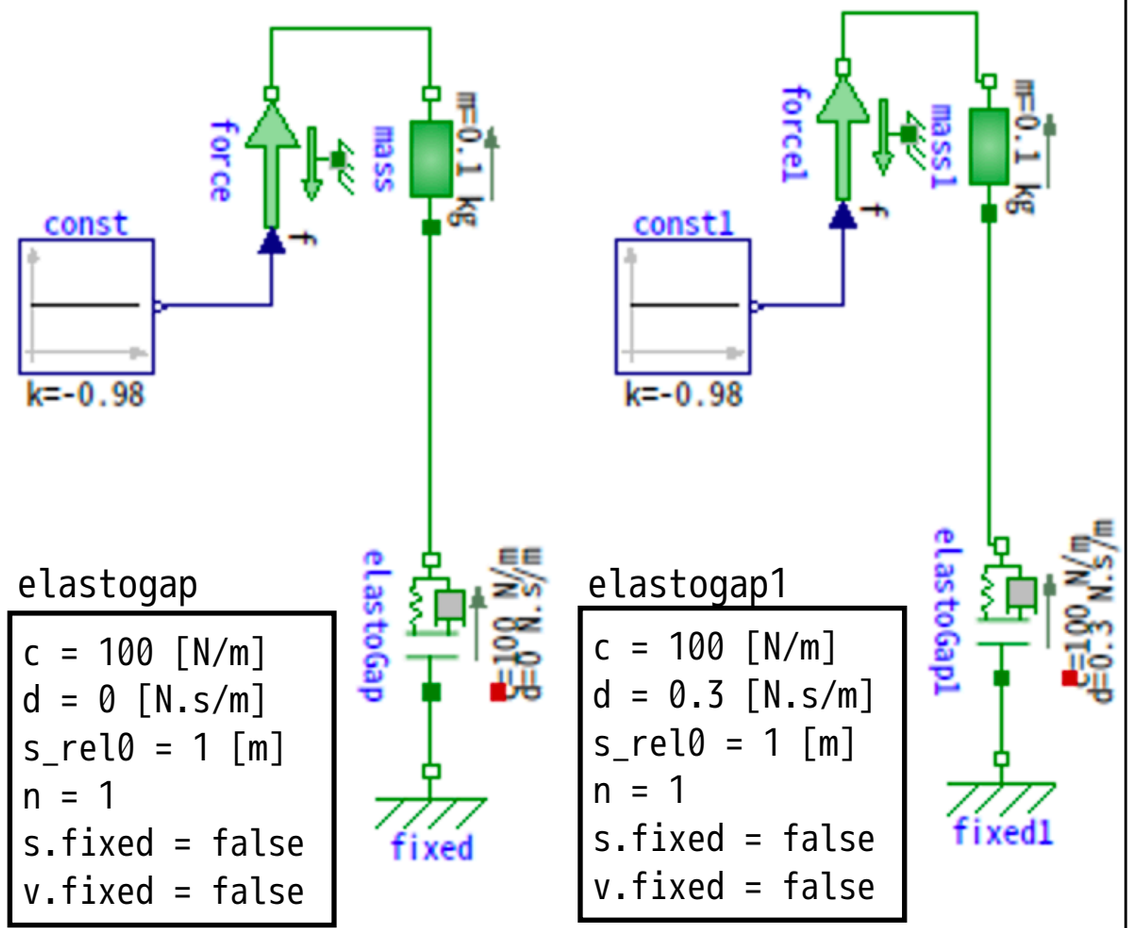
Example8と同じ結果になります。

Example10 弾性体に剛体を落とす

概要



モデル



elastogap

```
c = 100 [N/m]
d = 0 [N.s/m]
s_rel0 = 1 [m]
n = 1
s.fixed = false
v.fixed = false
```

elastogap1

```
c = 100 [N/m]
d = 0.3 [N.s/m]
s_rel0 = 1 [m]
n = 1
s.fixed = false
v.fixed = false
```

const, const1

```
k = -0.98 [N]
```

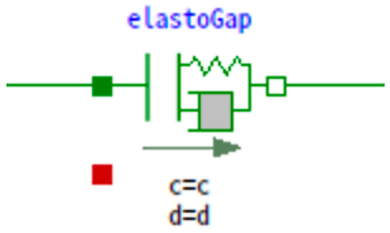
mass, mass1

```
m = 0.1 [kg]
L = 0.2 [m]
v.fixed = true     初期速度を設定する
v.start = 0 [m/s]  初期速度
s.fixed = true     初期位置を設定する
s.start = 3 [m]    初期位置
```

fixed, fixed1

```
s0 = 0 [m]
```

ElastoGap — 弾性体に接触したときの弾力と抵抗力のモデル



接触したとき（相対距離 s_rel が弾性体の厚さ s_rel0　未満になったとき）
弾力と速度に比例した抵抗力が発生します。

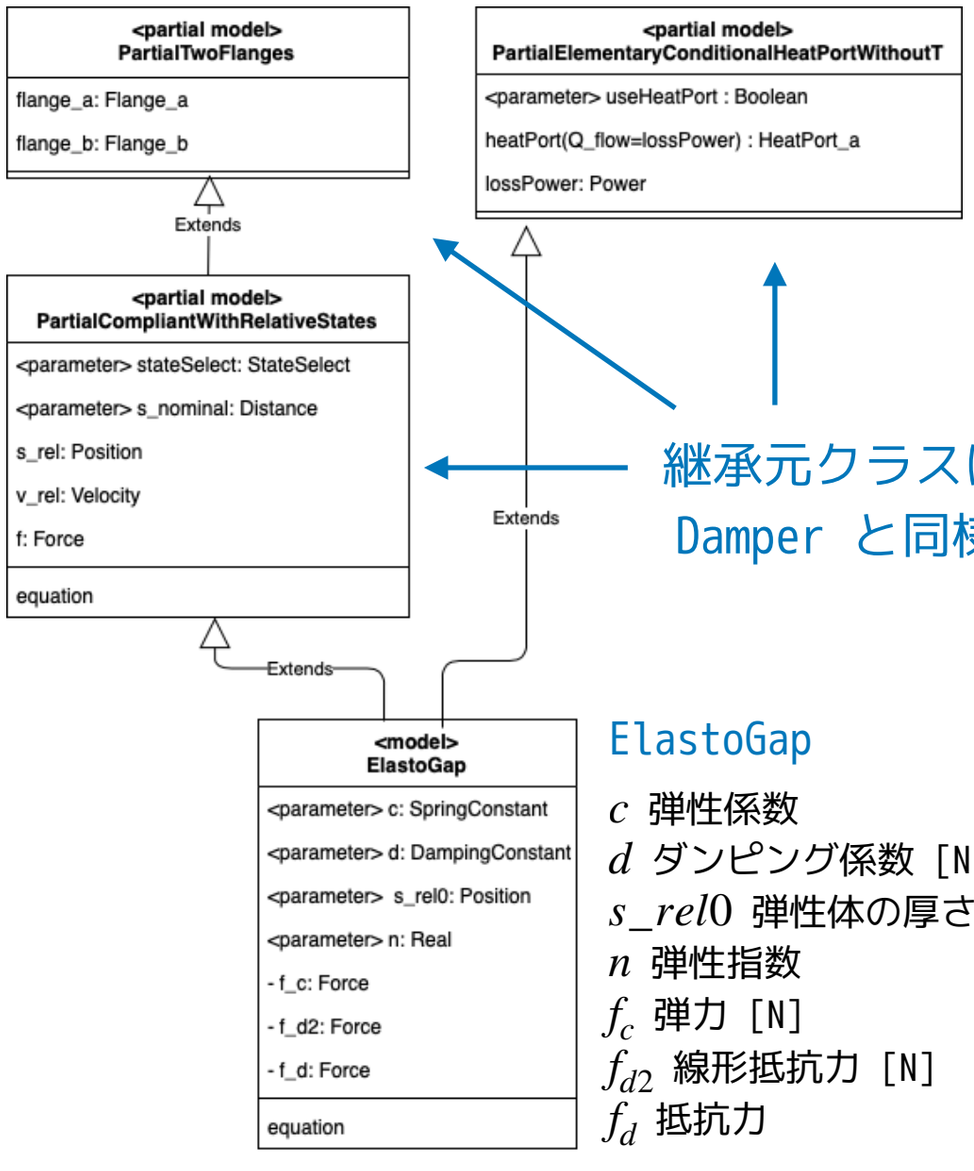
$$f = f_c + f_d$$

弾力　抵抗力

接触判定

$$contact = s\_rel < s\_rel0$$

構成と方程式



継承元クラスは
Damper と同様

ElastoGap

- c 弾性係数
- d ダンピング係数 [N.s/m]
- s_rel0 弾性体の厚さ [m]
- n 弾性指数
- f_c 弾力 [N]
- f_{d2} 線形抵抗力 [N]
- f_d 抵抗力

$$f_c = \text{smooth}_1 \begin{cases} -c\,|s\_rel - s\_rel0|^n, & contact \\ 0, & \text{else} \end{cases}$$

一階微分まで連続なことをソルバーに知らせるオペレータ。ソースコードでは smooth(1, …)

$$f_d = \text{smooth}_0 \begin{cases} \text{noevent} \begin{cases} f_c & f_{d2} < f_c \\ -f_c & f_{d2} > -f_c, \quad contact \\ f_{d2} & \text{else} \end{cases} \\ 0, & \text{else} \end{cases}$$

eventを発生させないオペレータ。ソースコードでは noevent(…)

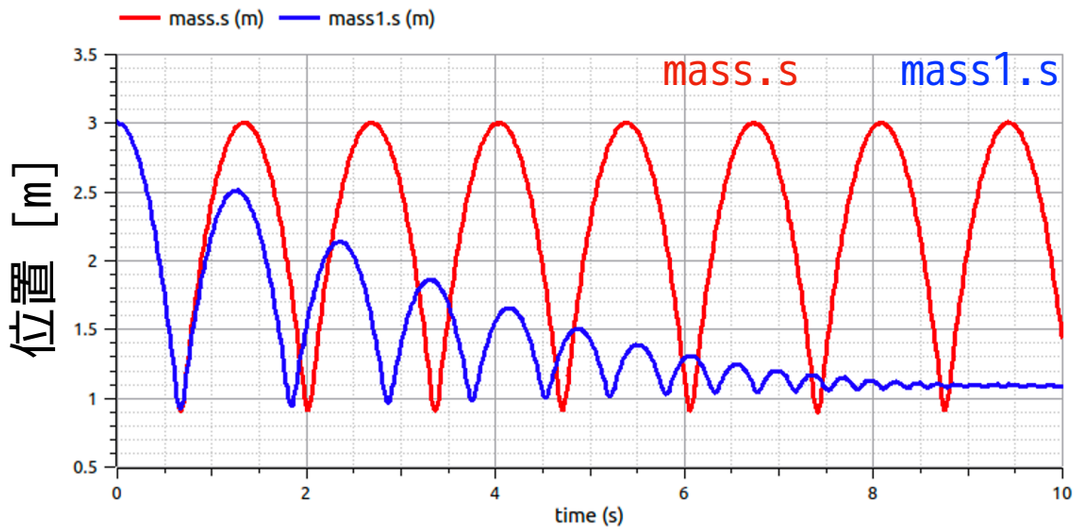
値が連続なことをソルバーに知らせるオペレータ。ソースコードでは smooth(0, …)

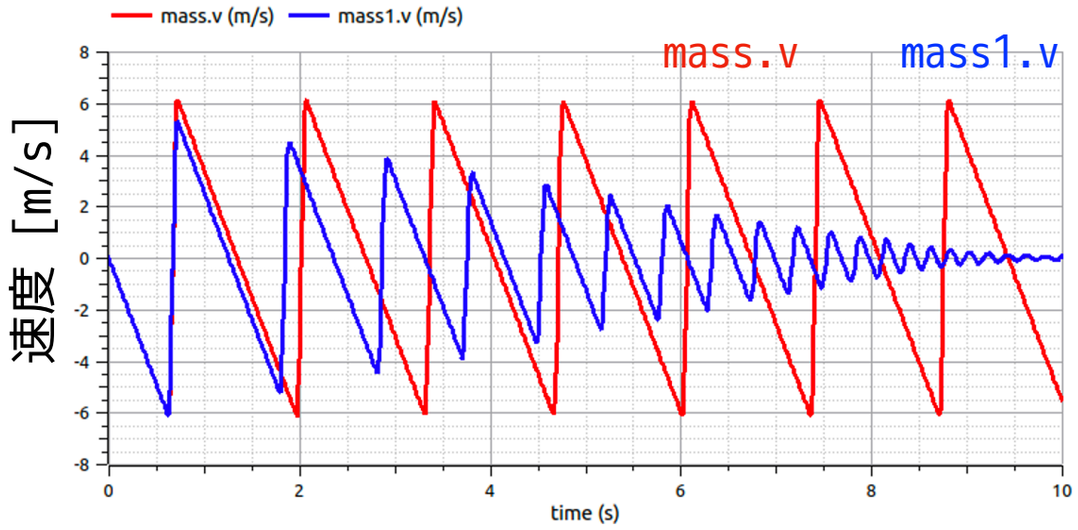
ダンピング係数
相対速度

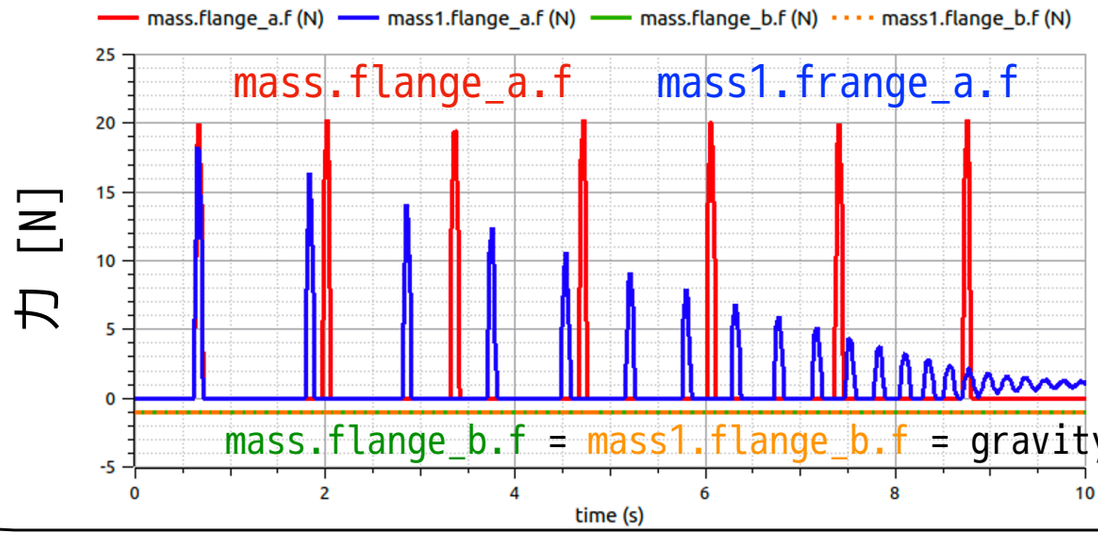
$$f_{d2} = \begin{cases} d \cdot v\_rel, & contact \\ 0, & \text{else} \end{cases}$$

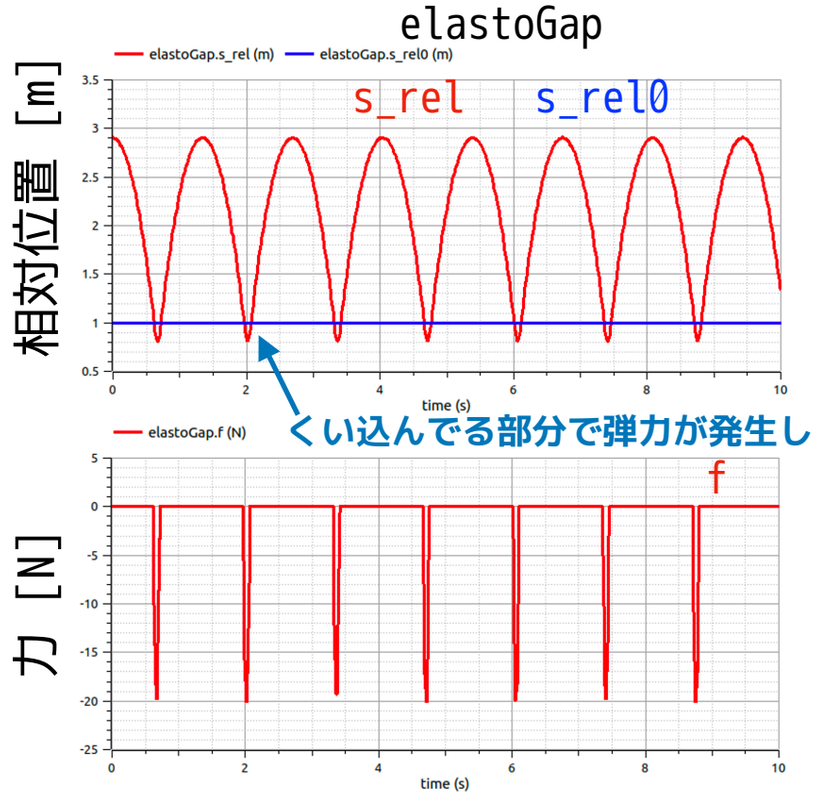
抵抗力の大きさが
弾力の大きさを
超えないように
調整する機能。

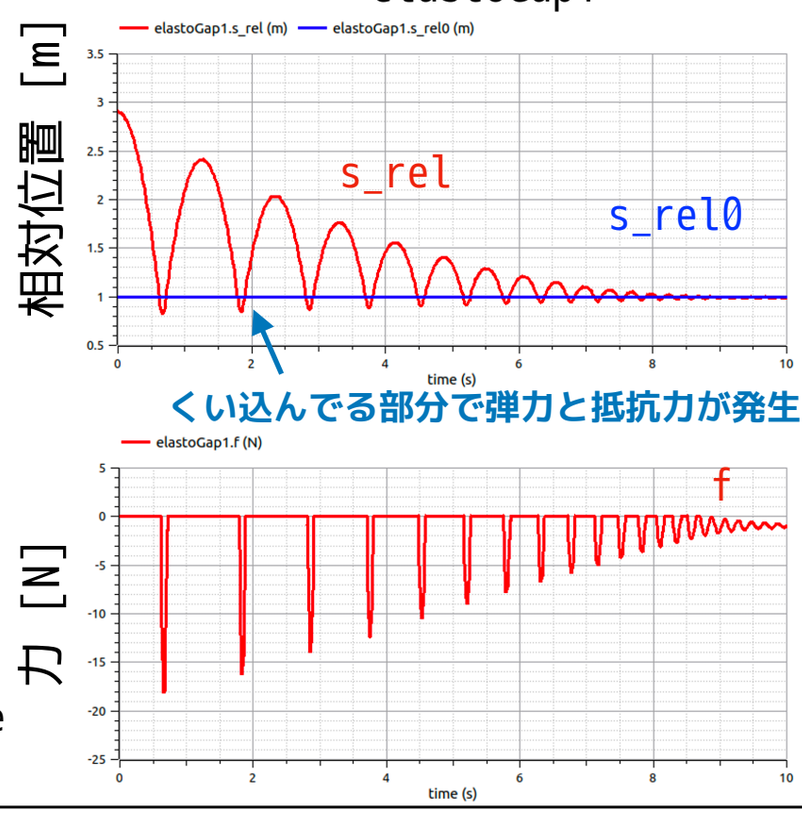
シミュレーション結果


mass.s (m)
mass1.s (m)
mass.s
mass1.s
位置 [m]
time (s)


mass.v (m/s)
mass1.v (m/s)
mass.v
mass1.v
速度 [m/s]
time (s)

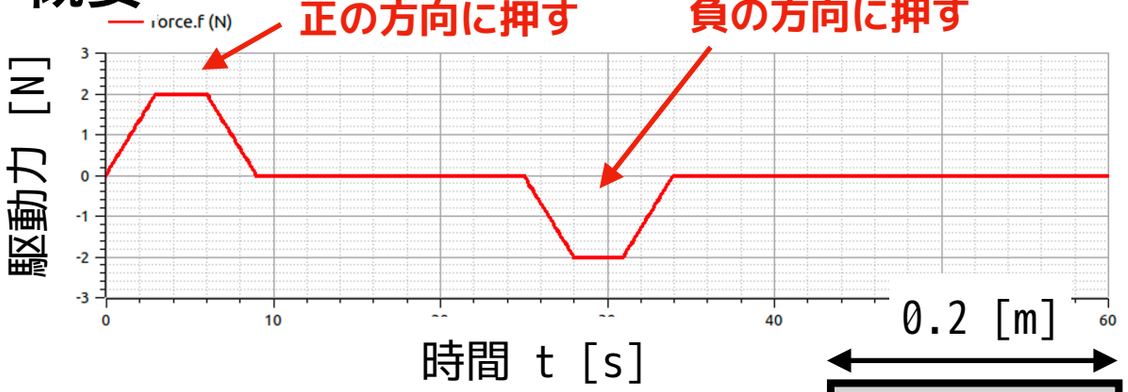

mass.flange_a.f (N)
mass1.flange_a.f (N)
mass.flange_b.f (N)
mass1.flange_b.f (N)
mass.flange_a.f
mass1.frange_a.f
力 [N]
time (s)
mass.flange_b.f = mass1.flange_b.f = gravity force

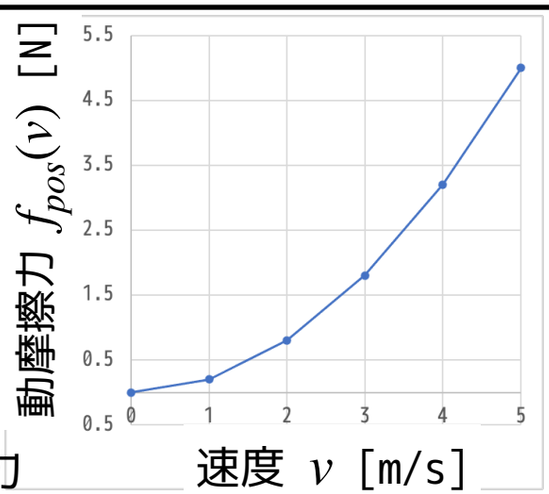

elastoGap
elastoGap.s_rel (m)
elastoGap.s_rel0 (m)
s_rel
s_rel0
相対位置 [m]
time (s)
くい込んでる部分で弾力が発生している
elastoGap.f (N)
f
力 [N]
time (s)


elastoGap1
elastoGap1.s_rel (m)
elastoGap1.s_rel0 (m)
s_rel
s_rel0
相対位置 [m]
time (s)
くい込んでる部分で弾力と抵抗力が発生している
elastoGap1.f (N)
f
力 [N]
time (s)

Example11 摩擦のある面上の物体を動かす

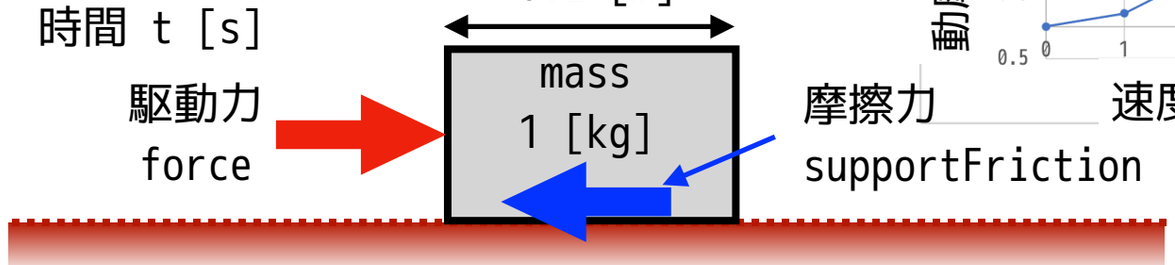
概要





v [m/s]	f_pos [N]
0	0.001
1	0.201
2	0.801
3	1.801
4	3.201
5	5.001

初期条件
位置 0 [m]
速度 0 [m/s]

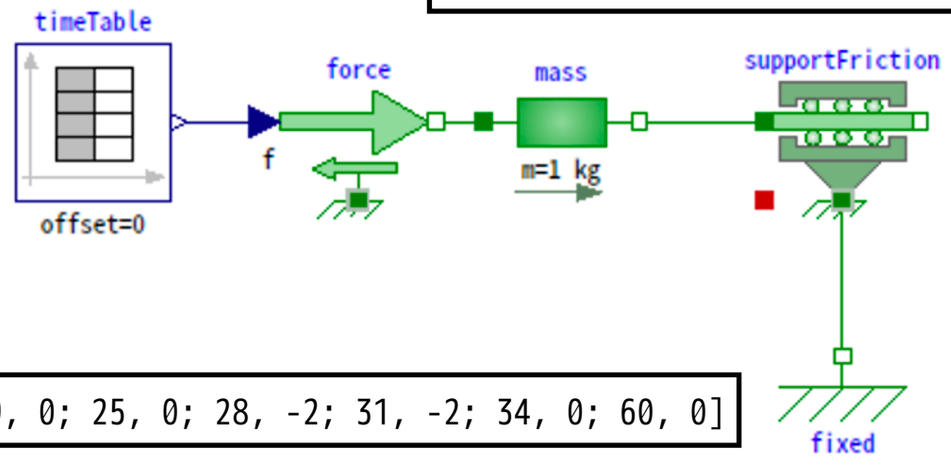


$$\text{peak} = \frac{\text{最大静止摩擦力}}{\text{v=0の動摩擦力}} = 500$$

モデル

Modelica.Blocks.Sources.TimeTable

t [s]	駆動力 [N]
0	0
3	2
6	2
9	0
25	0
28	-2
31	-2
34	0
60	0



supportFriction

```
f_pos = [0, 0.001; 1, 0.201; 2, 0.801; 3, 1.801; 4, 3.201; 5, 5.001]
peak = 500
mode.fixed = true
mode.start = Unknown
```

mass

```
m = 1 [kg]       質量
L = 0.2 [m]      長さ
v.fixed = true   初期速度を設定する
v.start = 0 [m] 初期速度
s.fixed = true   初期位置を設定する
s.start = 0 [m] 初期位置
```

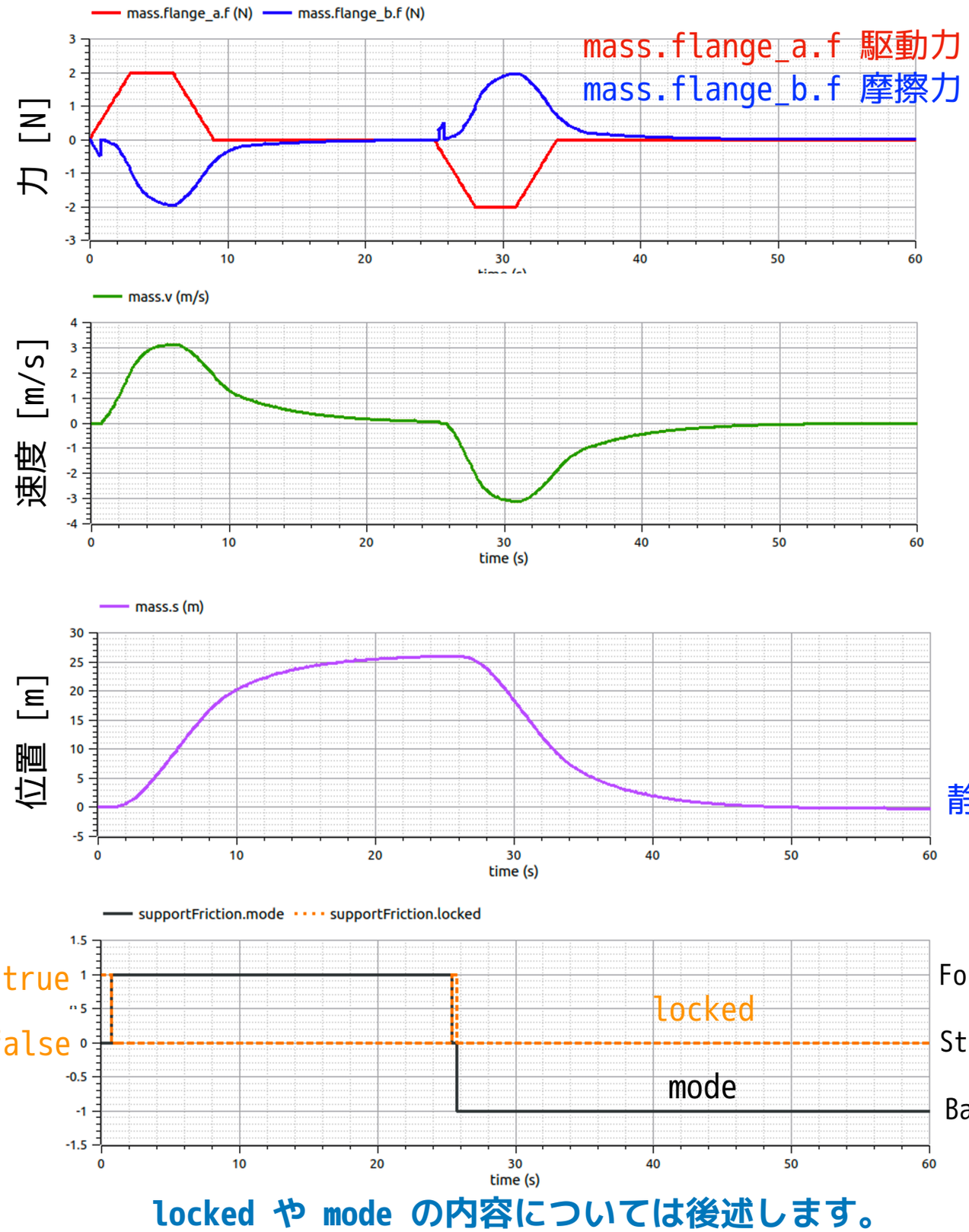
timeTable

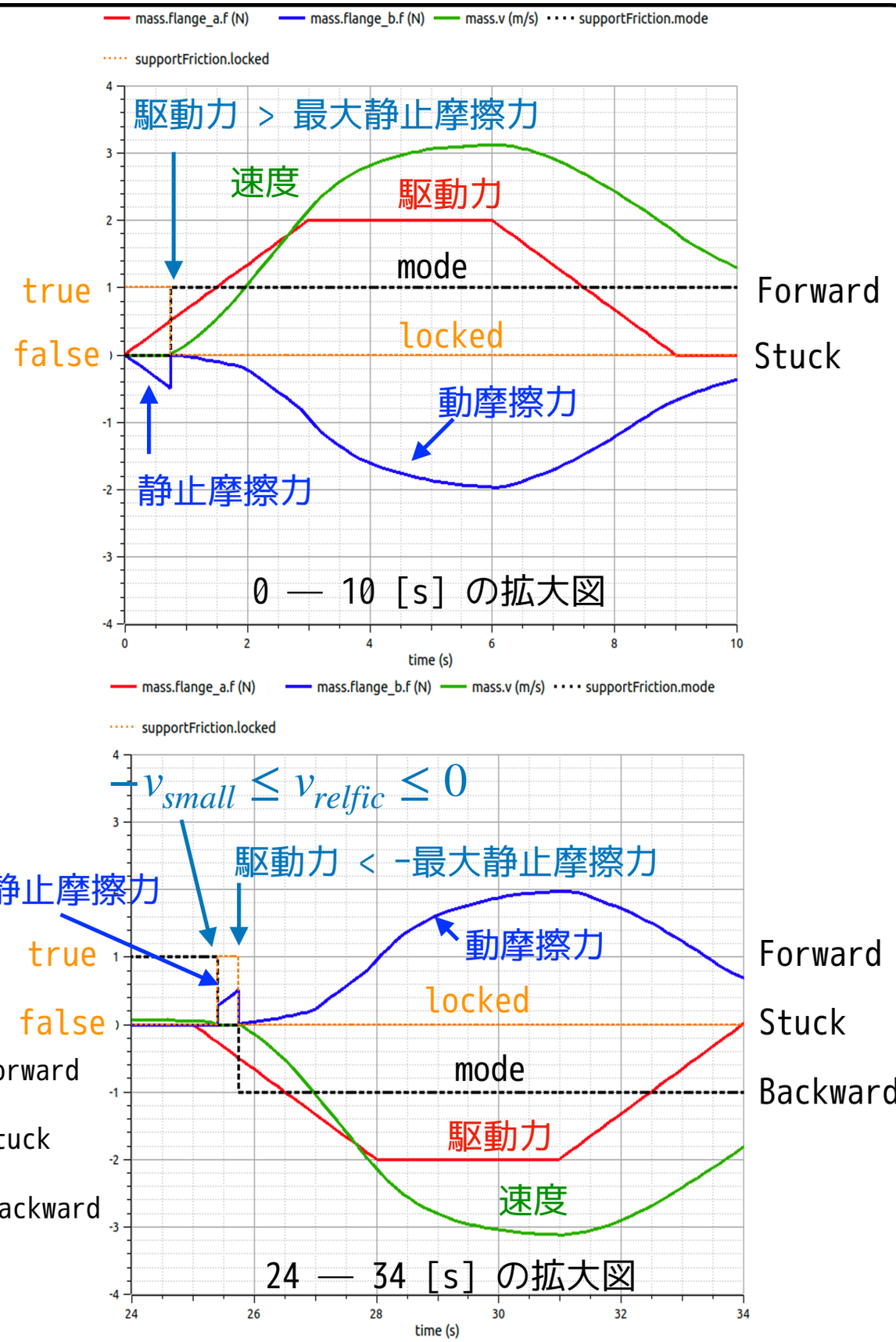
```
table = [0, 0; 3, 2; 6, 2; 9, 0; 25, 0; 28, -2; 31, -2; 34, 0; 60, 0]
```

fixed

```
s0 = 0  位置
```

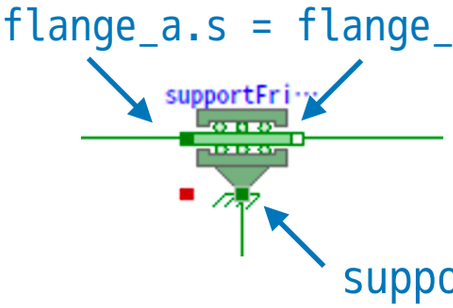
シミュレーション結果





locked や mode の内容については後述します。

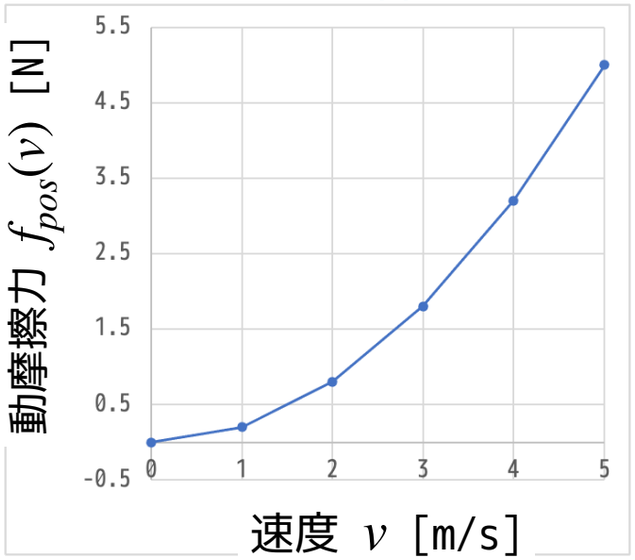
SupportFriction — 速度と運動状態に依存した摩擦力



support に対する flange の相対速度と運動状態に依存した摩擦力のモデル

設定例

```
f_pos = [0, 0.001; 1, 0.201; 2, 0.801; 3, 1.801; 4, 3.201; 5, 5.001];
peak = 500
```



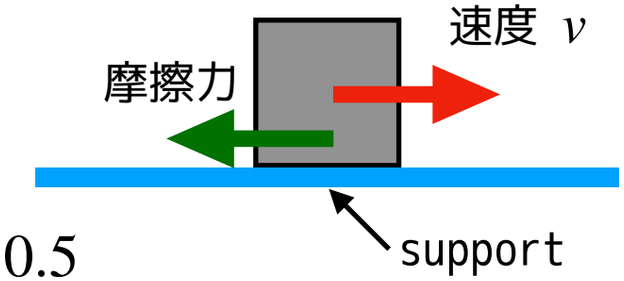
v [m/s]	f [N]
0	0.001
1	0.201
2	0.801
3	1.801
4	3.201
5	5.001

$v = 0$ の動摩擦擦力

$$f_0 = f_{pos}(0) = 0.001$$

最静止大摩擦力

$$f_{0\_max} = peak \cdot f_0 = 0.5$$



locked （滑っていない状態）では、加速度をゼロにするような静止摩擦力が計算される。

線形補間関数

$$f_{pos}(v) = \texttt{Modelica.Math.Vectors.interpolate(f\_pos[:, 1], f\_pos[:, 2], v, 1)}$$

運動状態はPartialFrictionモデルで場合分けする。

① 摩擦力　$f = \begin{cases} s_a \cdot unitForce & locked = true \\ f_{pos}(v), & startForward = true \\ -f_{pos}(-v), & startBackword = true \\ f_{pos}(v), & pre(mode) = Forward \\ -f_{pos}(-v), & else \end{cases}$

アクティブで以下のいずれでもない
前方に滑り始める条件を満たす状態
後方に滑り始める条件を満たす状態
直前が前方に滑っている状態
直前が後方に滑っている状態

$pre(mode) = Backward$ しかありえない。

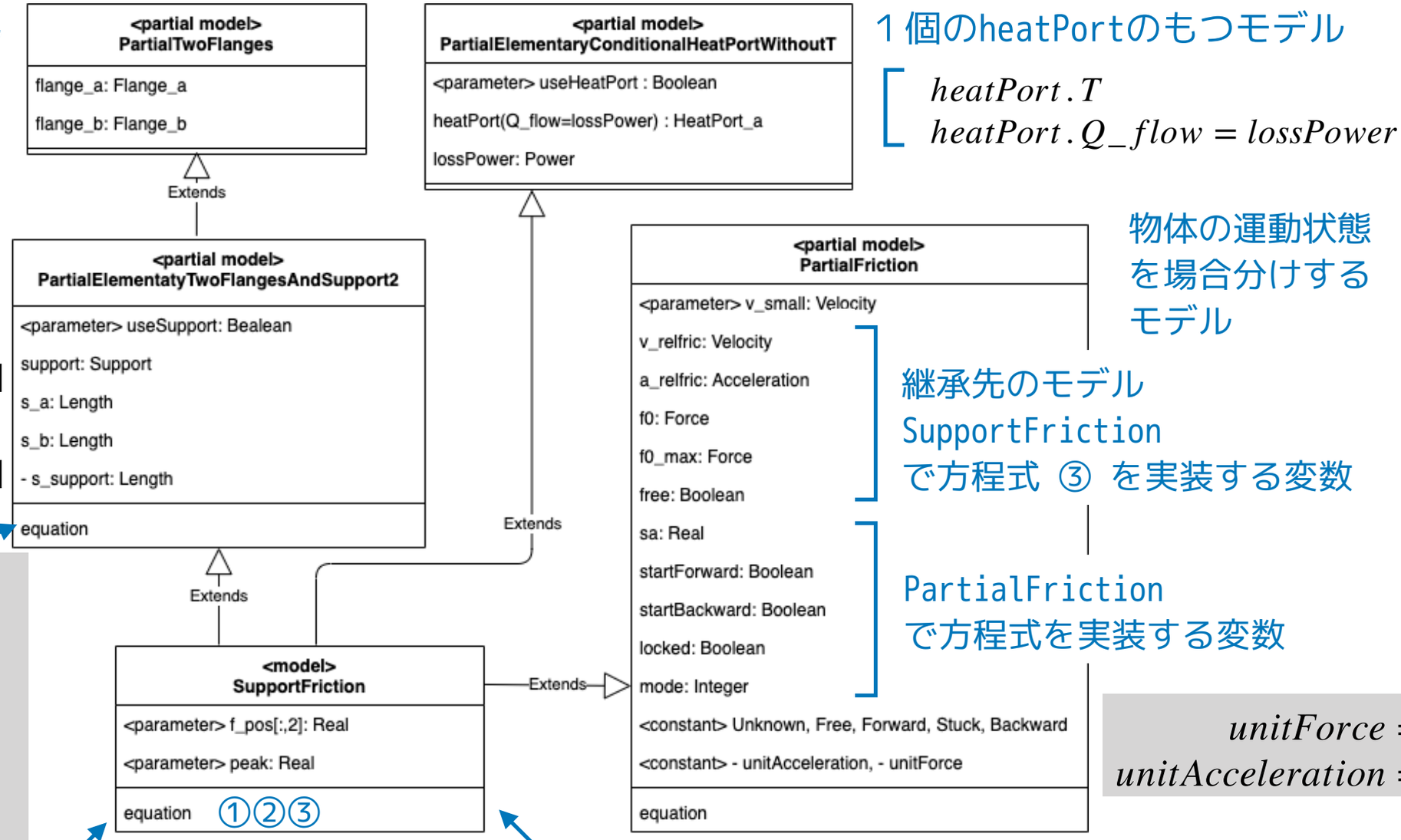
SupportFriction の構成と方程式

2個のFlangeをもつモデル

$$\begin{bmatrix} flange\_a.s & [m] \\ flange\_a.f & [N] \\ flange\_b.s & [m] \\ flange\_b.f & [N] \end{bmatrix}$$

1個のSupportをもつモデル

$$\begin{bmatrix} support.s = s\_support & [m] \\ support.f = -flange\_a.f & \\ \quad -flange\_b.f & [N] \end{bmatrix}$$



1 個のheatPortのもつモデル

$$\begin{bmatrix} heatPort.T \\ heatPort.Q\_flow = lossPower \end{bmatrix}$$

物体の運動状態を場合分けするモデル

継承先のモデル SupportFriction で方程式 ③ を実装する変数

PartialFriction で方程式を実装する変数

$$unitForce = 1$$
$$unitAcceleration = 1$$

$$s\_a = flange\_a.s - s_{suport}$$
$$s\_b = flange\_b.s - s_{support}$$
$$if\ not\ usrSupport\ then$$
$$\quad s_{support} = 0$$
$$end\ if$$

摩擦力を実装するモデル

②

$$s = flange\_a.s - s_{support}$$ 相対位置
$$flange\_a.s = flange\_b.s$$ 長さなし
$$flange\_a.f + flange\_b.f - f = 0$$ 力のバランス
$$lossPower = f \cdot v_{relfric}$$ 摩擦による仕事率（発熱量）

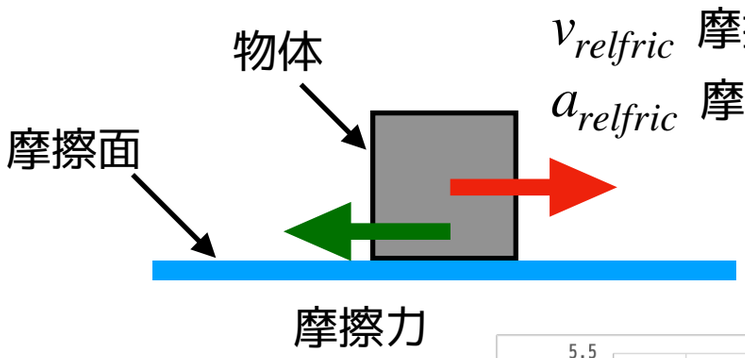
③

$$v_{relfric} = v = \frac{ds}{dt}$$ 相対速度
$$a_{relfric} = a = \frac{dv}{dt}$$ 相対加速度
$$f_0 = f_{pos}(0)$$ $v = 0$ の摩擦力
$$f_{0\_max} = peak \cdot f_0$$ 最大静止摩擦力
$$free = false$$ $free = false$ ときのアクティブ

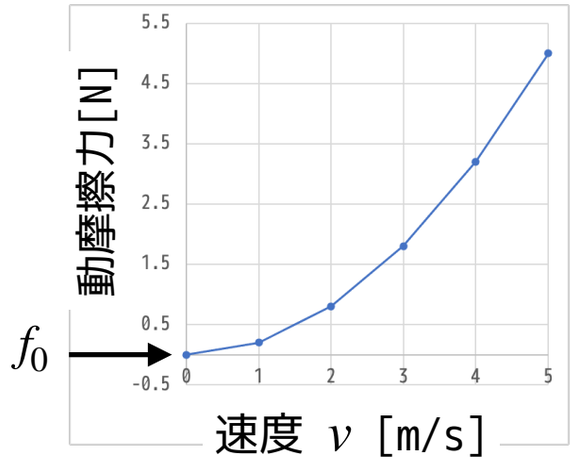
PartialFriction — 面の上をすべる物体の状態を場合分けする部分モデル

A このモデルを継承したモデルで方程式が実装される変数

$free$: $true$のとき摩擦要素が非アクティブになる
$v_{relfric}$: 摩擦面との間の速度 [m/s]
$a_{relfric}$: 摩擦面との間の加速度 [m/s2]
f_0: 速度ゼロで前方に滑っているときの摩擦力 [N]
f_{0_max}: 最大静止摩擦力 [N]



SupportFriction の方程式③



B このモデルで方程式が定義される変数

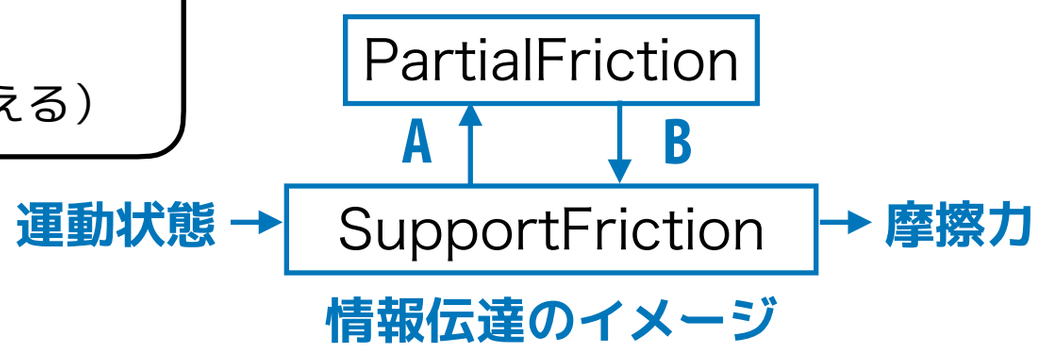
$startForward$: 速度ゼロで前方に滑り始めるとき true になる
$startBackword$: 速度ゼロで後方に滑り始めるときに true になる
$locked$: 速度ゼロで滑っていないとき true になる
s_a: path parameter と呼ばれる摩擦特性を表す無次元変数
$mode$: 摩擦の状態を表す整数型の変数（オペレータ pre が使える）

$mode$ のとり得る値

$Backword = -1$	速度が負。後方の滑っている。
$Stuck = 0$	速度がゼロ。
$Forward = 1$	速度が正。前方に滑っている。
$Free = 2$	摩擦要素が非アクティブ
$Unknown = 3$	不明（初期状態のみ）

継承先の SupportFriction モデルではこれらの変数を参照して摩擦力を定義する。

SupportFriction の方程式①



mode 運動状態を表す離散的状態変数

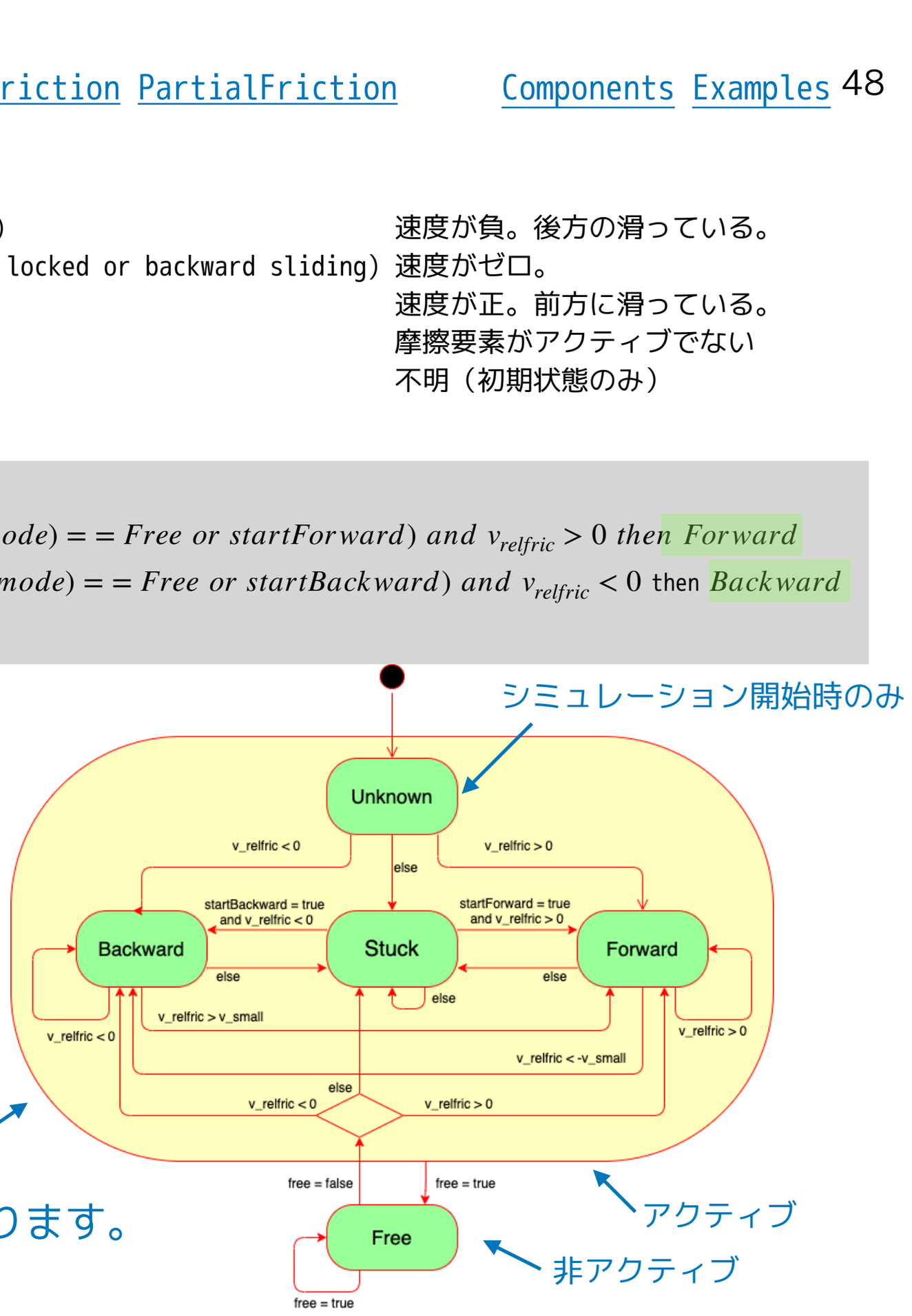
$Backward = -1$	v_relfric < 0 (backward sliding)	速度が負。後方の滑っている。
$Stuck = 0$	v_relfric = 0 (forward sliding, locked or backward sliding)	速度がゼロ。
$Forward = 1$	v_relfric > 0 (forward sliding)	速度が正。前方に滑っている。
$Free = 2$	Element is not active.	摩擦要素がアクティブでない
$Unknown = 3$	Value of mode is not known	不明（初期状態のみ）

計算式

$$
\begin{aligned}
mode = \ & if\ free\ then\ Free \\
& else\ (\ \ if\ (pre(mode) == Forward\ or\ pre(mode) == Free\ or\ startForward)\ and\ v_{relfric} > 0\ then\ Forward \\
& \quad else\ if\ (pre(mode) == Backward\ or\ pre(mode) == Free\ or\ startBackward)\ and\ v_{relfric} < 0\ \text{then}\ Backward \\
& \quad else\ Stuck\ )
\end{aligned}
$$

- $Free$ となるのはパラメータで $free = true$ とした場合のみです。
- $Unknown$ となるのはシミュレーション開始時のみ。シミュレーション中は $Forward, Stuck, Backward$ のいずれかになります。。
- 数値的計算では正確に $v_{relfric} = 0$ とはなりにくい。そのため $Stuck$ は、$Forward$ でも $Backward$ でもないという否定論理和的な定義になっています。

調査の結果、
mode の状態遷移はこんな感じになります。



startForward 前方に滑り始める条件

条件式

静止状態

Flangeの合力 > 最大静止摩擦力

前進し始めた直後に、
Flangeの合力 > 速度ゼロの動摩擦擦

$$startForward = pre(mode) == Stuck \ and \left(s_a > \frac{f_{0\_max}}{unitForce} \ or \ pre(startForward) \ and \ s_a > \frac{f_0}{unitForce} \right)$$
$$or \ pre(mode) == Backward \ and \ v_{relfric} > v_{small}$$
$$or \ initial() \ and \ (v_{relfric} > 0);$$

初期速度が正

後方に滑ってる状態（$v_{relfric} < 0$)から、
$v_{relfric} > v_{small}$ (> 0) の状態に変化した。

startBackward 後方に滑り始める条件

条件式

静止状態

Flangeの合力 < -最大静止摩擦力

後退し始めた直後に、
Flangeの合力 < -速度ゼロの動摩擦擦

$$startBackward = pre(mode) == Stuck \ and \left(s_a < -\frac{f_{0\_max}}{unitForce} \ or \ pre(startBackword) \ and \ s_a < -\frac{f_0}{unitForce} \right)$$
$$or \ pre(mode) == Forward \ and \ v_{relfric} < -v_{small}$$
$$or \ initial() \ and \ (v_{relfric} < 0)$$

初期速度が負

前方に滑ってる状態（$v_{relfric} > 0$)から、
$v_{relfric} < -v_{small}$ (< 0) の状態に変化した。

$locked$　モデルがアクティブ($not\ free$)で、滑ってる状態や滑り始める状態でないとき true になる

$$locked = not\ free\ and\ not\ (\ pre(mode) == Forward\ or\ startForward\ or\ pre(mode) == Backward\ or\ startBackward\ )$$

否定論理和的定義になっている

⇔ ド・モルガンの法則

$$not\ locked = free\ or\ (\ pre(mode) == Forward\ or\ startForward\ or\ pre(mode) == Backward\ or\ startBackward\ )$$

この５つの状態のどれにも当てはまらないとき $locked = true$ になります。

s_a　摩擦特性を表す path パラメータ

$$\frac{a_{relfric}}{unitAcceleration} = \begin{cases} 0, & locked \\ s_a, & free \\ s_a - \frac{f_{0\_max}}{unitForce}, & startForward \\ s_a + \frac{f_{0\_max}}{unitForce}, & startBackward \\ s_a - \frac{f_{0\_max}}{unitForce}, & pre(mode) = Forward \\ s_a + \frac{f_{0\_max}}{unitForce}, & else \end{cases}$$

$locked$ の式から $pre(mode) = Backward$ しかありえません。

これらの方程式を使って s_a を決定する。

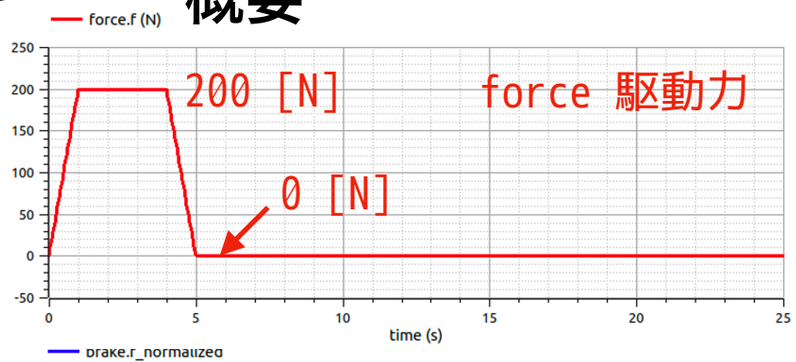
$$\Rightarrow s_a = \frac{a\_relfric}{unitAcceleration}$$
$$\Rightarrow s_a = \frac{a\_relfric}{unitAcceleration} + \frac{f_{0\_max}}{unitForce}$$
$$\Rightarrow s_a = \frac{a\_relfric}{unitAcceleration} - \frac{f_{0\_max}}{unitForce}$$
$$\Rightarrow s_a = \frac{a\_relfric}{unitAcceleration} + \frac{f_{0\_max}}{unitForce}$$
$$\Rightarrow s_a = \frac{a\_relfric}{unitAcceleration} - \frac{f_{0\_max}}{unitForce}$$

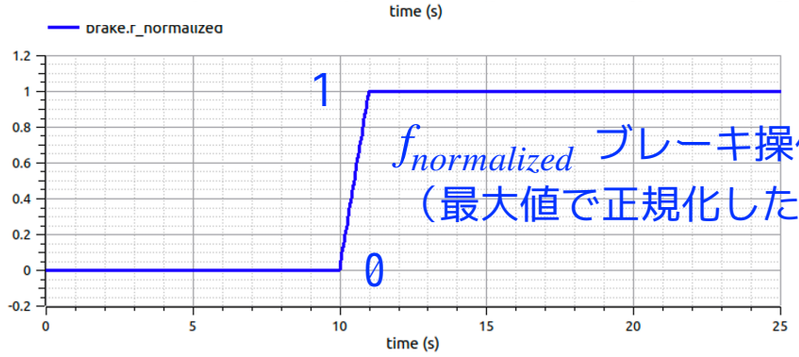
$locked = true$ の場合は方程式に s_a が存在しない。SupportFrictionの 場合は、式①のlockedの場合から静止摩擦力 f とFlange の合力がつり合うような s_a を計算する。

$$\Rightarrow s_a = \frac{f}{unitForce} = \frac{flange\_a.f + flange\_b.f}{unitForce}$$

Example12 ブレーキをかけて自転車を停止させる。

概要



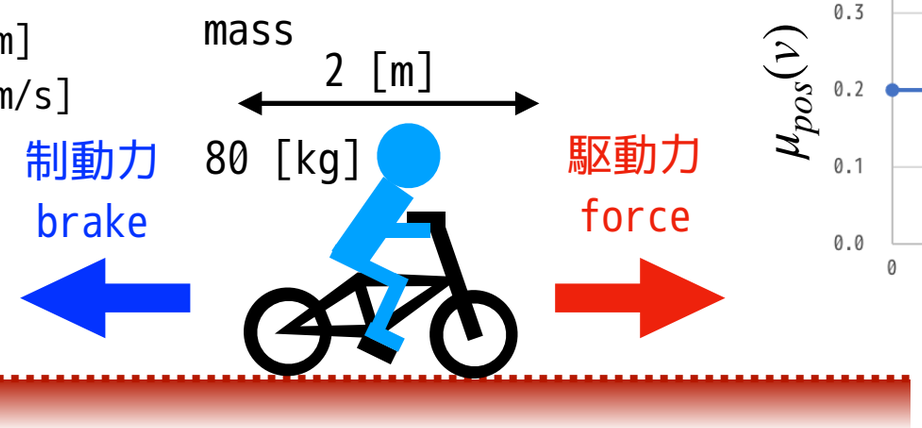


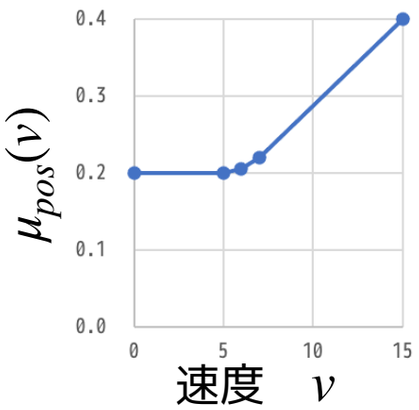
$f_{normalized}$ ブレーキ操作
（最大値で正規化した垂直抗力）

摩擦力　$f = c_{geo} \cdot \mu(v) \cdot f_n, \quad c_{geo} = 4$　　$v \geq 0$ の動摩擦係数 $\mu_{pos}(v)$

垂直抗力　$f_n = f_{n\_max} \cdot f_{normalized}, \quad f_{n\_max} = 300$ [N]

初期位置 0 [m]
初期速度 0 [m/s]



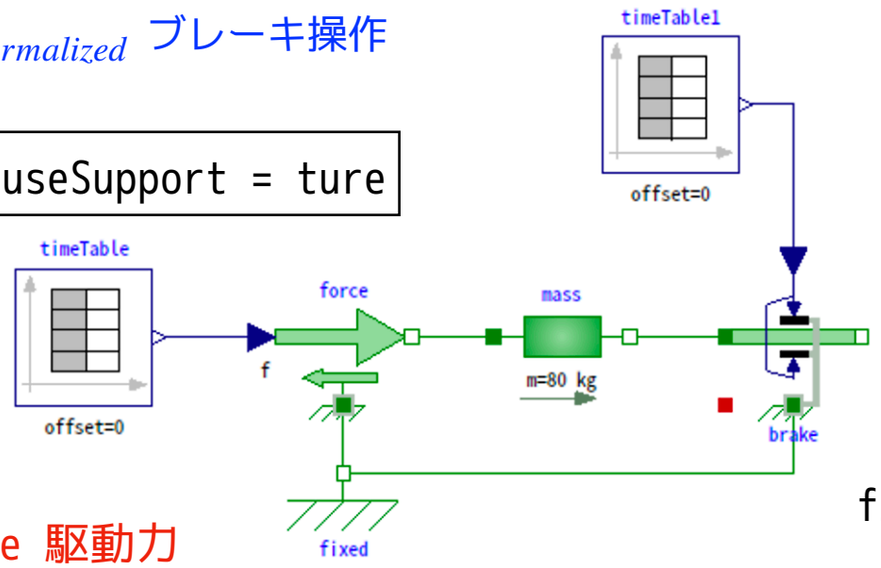


モデル

timeTable1　`table = [0, 0; 10, 0; 11, 1; 100, 1]`

$f_{normalized}$ ブレーキ操作

force　`useSupport = ture`



force 駆動力

timeTable　`table = [0, 0; 1, 200; 4, 200; 5, 0; 100, 0]`

brake

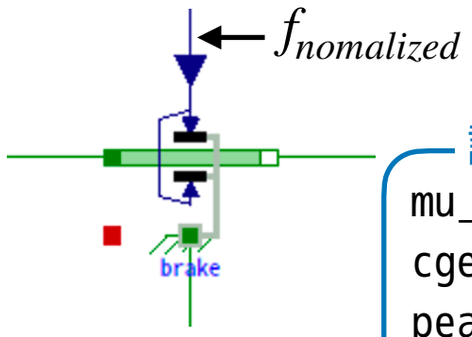
```
mu_pos = [0, 0.2; 5, 0.2; 6, 0.205; 7, 0.22; 15, 0.4]
peak = 2                最大静止摩擦力/(v=0の動摩擦力)
cgeo = 4                形状定数（geometry constant）
fn_max = 300            垂直抗力の最大値
mode.fixed = true       modeの初期値を設定する
mode.start = Unknown    modeの初期値
useSupport = true       support を使用する
```

mass

```
m = 80 [kg]         質量
L = 2 [m]           長さ
v.fixed = true      初期速度を設定する
v.start = 0 [m/s]   初期速度
s.fixed = true      初期位置を設定する
s.start = 0 [m]     初期位置
```

fixed　`s0 = 0 [m]`

Brake — 垂直抗力が制御できる摩擦モデル



垂直抗力が実数入力信号で制御できるブレーキのモデル

Flange の絶対速度 v に対する動摩擦係数 mu_pos を設定します。

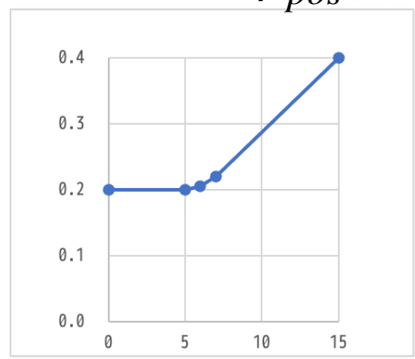
設定例

```
mu_pos = [0, 0.2; 5, 0.2; 6, 0.205; 7, 0.22; 15, 0.4]
cgeo = 4        形状定数
peak = 2
fn_max = 300    垂直抗力の最大値
```

$$peak = \frac{\text{最大静止摩擦力}}{\text{v=0の動摩擦力}}$$

v [m/s]	m_pos
0	0.200
5	0.200
6	0.205
7	0.220
15	0.400

動摩擦係数 $\mu_{pos}(v)$



摩擦力

①

$$f = \begin{cases} s_a \cdot unitForce, & locked = true \\ 0, & free = true \\ c_{geo} \cdot f_n \cdot \mu(v), & else \end{cases}$$

静止して滑りだす条件を満たしていない状態

アクティブでない状態

c_{geo} 形状定数（geometry constant）

$\mu(v)$ 摩擦係数（friction coefficient）

f_n 垂直抗力（normal force）

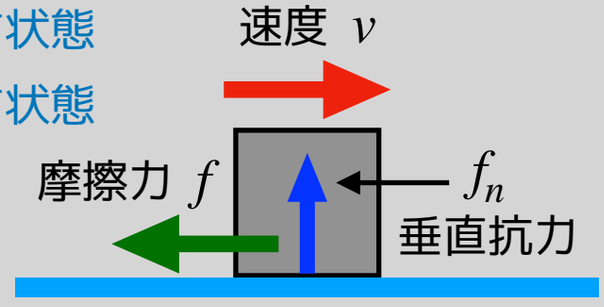
$$\mu(v) = \begin{cases} \mu_{pos}(v), & startForward = true \\ -\mu_{pos}(-v), & startBackward = true \\ \mu_{pos}(v), & pre(mode) = Forward \\ -\mu_{pos}(v), & else\ (pre(mode) = Backward) \end{cases}$$

前方に滑り始める条件を満たす状態

後方に滑り始める条件を満たす状態

直前が前方に滑っている状態

直前が後方に滑っている状態



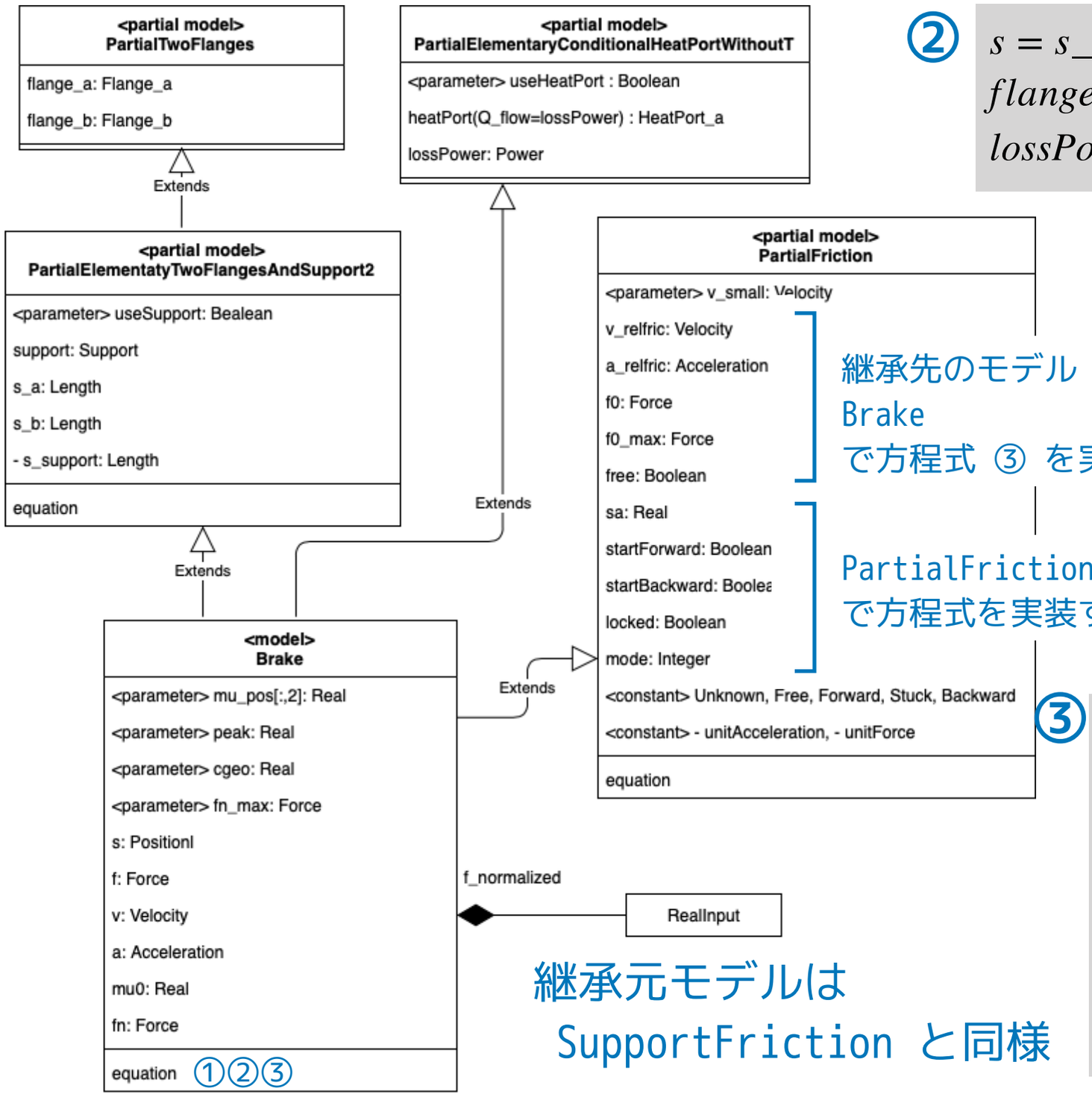
$$\mu_{pos}(v) = \text{Modelica.Math.Vectors.interpolate(mu\_pos[:, 1], mu\_pos[:, 2], v, 1)}$$

テーブルデータ mu_pos の線形補間関数

$$f_n = f_{n\_max} \cdot f_{normalized} \quad 0 \le f_{normalized} \le 1$$

実数入力信号$f_{normalized}$は
垂直抗力の最大値で正規化された値を入力する。

Brake の構成と方程式



② $s = s\_a = s\_b$　　flange_a, flange_b の絶対位置

$flange\_a.f + flange\_b.f - f = 0$　　力のバランス

$lossPower = f \cdot v\_relfric$　摩擦による仕事率（発熱量）

③

$v_{relfric} = v = \dfrac{ds}{dt}$　　絶対速度

$a_{relfric} = a = \dfrac{dv}{dt}$　　絶対加速度

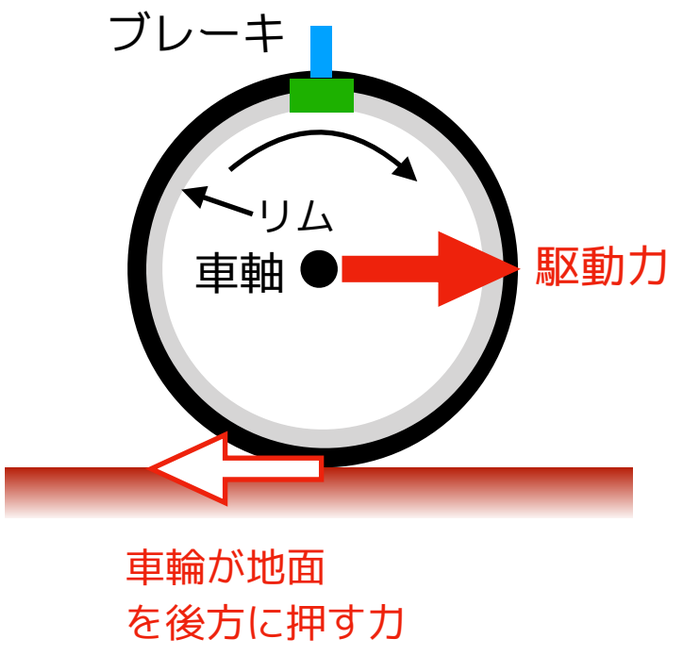
$f_0 = \mu_0 c_{geo} f_n,\ \mu_0 = \mu_{pos}(0)$　$v = 0$ の摩擦力

$f_{0\_max} = peak \cdot f_0$　　最大静止摩擦力

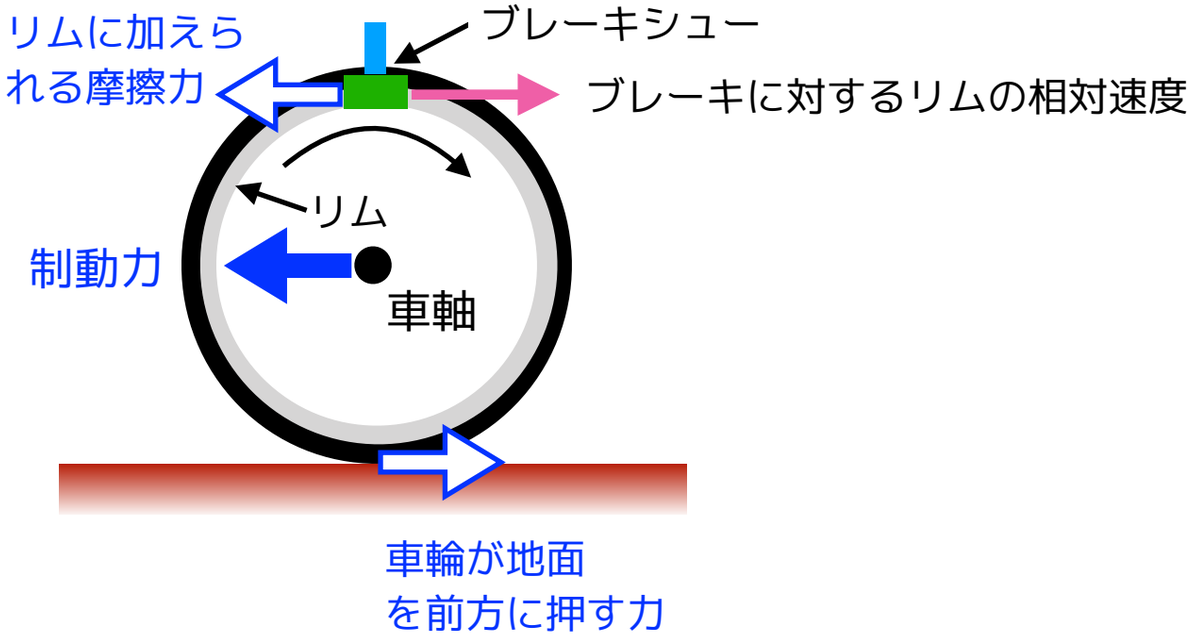
$free = f_n \le 0$　　$f_n < 0$ ときの非アクティブ

メカニズムの考察

本モデルでは、物体として自転車を想定しました。この場合、車体を加速させるのは人が車輪を回転させるトルクであり、減速させるのはブレーキと車輪のリムの間に発生する摩擦力です。これらが駆動力や制動力として作用するメカニズムを考えます。



- 車軸やブレーキは車体に固定されている。
- 車軸まわりの駆動トルクによって車輪が地面を後方に押す。
- 反作用として車軸は地面から前方に向かう駆動力を受ける。

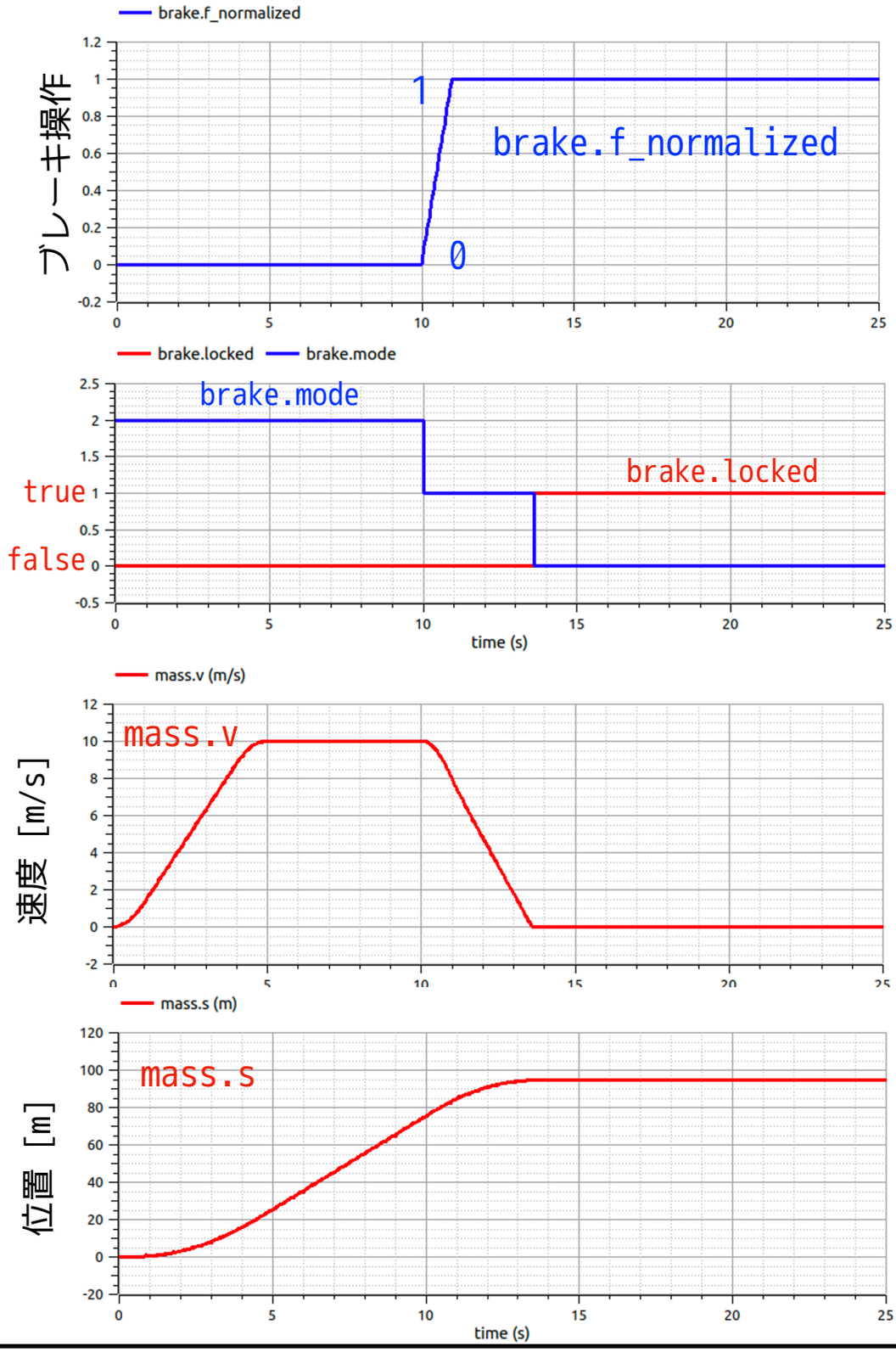


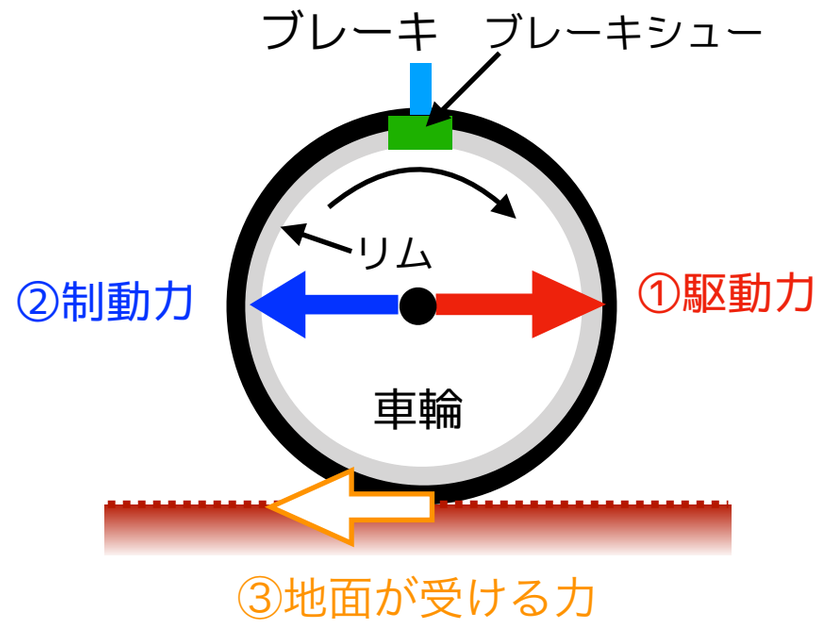
- ブレーキレバーを引くと、ブレーキシューが車輪のリムに押し付けられて、リムに相対速度と逆向きの摩擦力が加えられる。
- 車軸まわりのトルクバランスにより、車輪が地面を前方に押す力が生じる。
- 反作用として車体は地面から後方に向かう制動力を受ける。

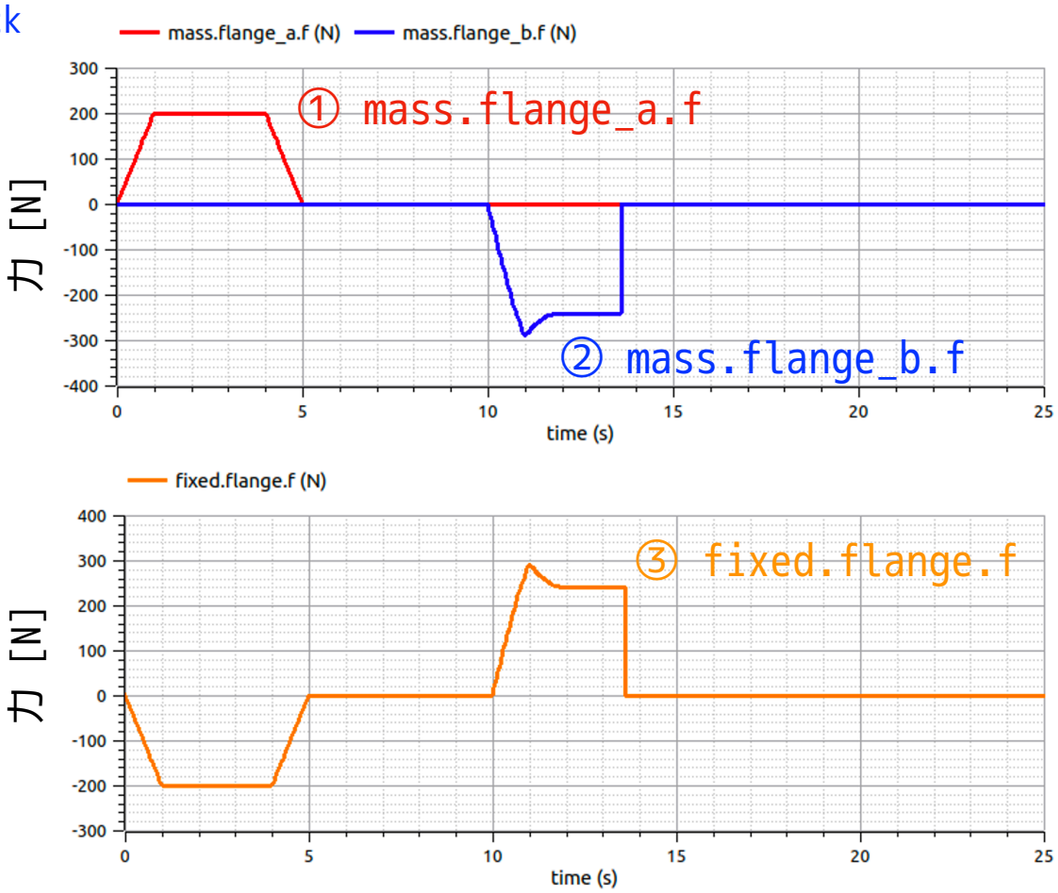
本モデルは以下を仮定しています。

- 車輪は滑らない。
- ブレーキにに対するリムの相対速度 = 車体のスピード
- リムに加えられる摩擦力の大きさ = 車輪が地面を前方に押す力の大きさ

シミュレーション結果

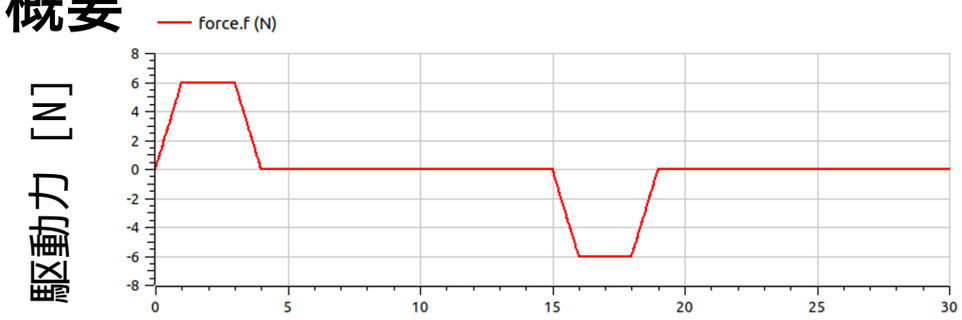

brake.f_normalized
ブレーキ操作
1
0
brake.f_normalized
brake.locked
brake.mode
brake.mode
brake.locked
true
false
Free
Forward
Stuck
time (s)
mass.v (m/s)
速度 [m/s]
mass.v
mass.s (m)
位置 [m]
mass.s
time (s)


ブレーキ
ブレーキシュー
リム
②制動力
①駆動力
車輪
③地面が受ける力


mass.flange_a.f (N)
mass.flange_b.f (N)
① mass.flange_a.f
② mass.flange_b.f
力 [N]
time (s)
fixed.flange.f (N)
③ fixed.flange.f
力 [N]
time (s)

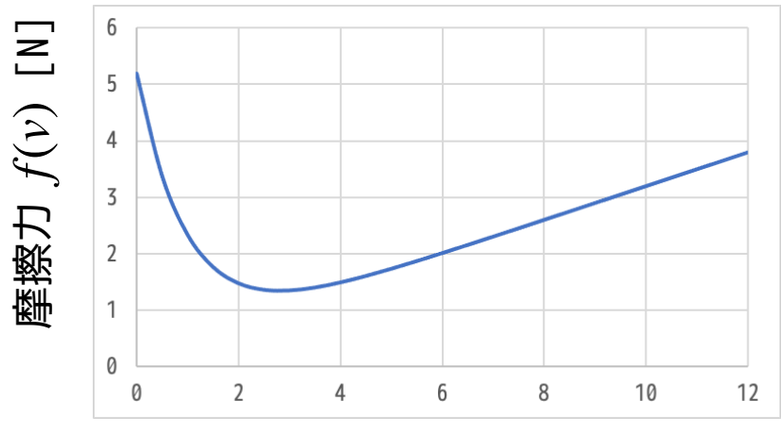
Example13 ストライベックの摩擦モデルを設定する

概要

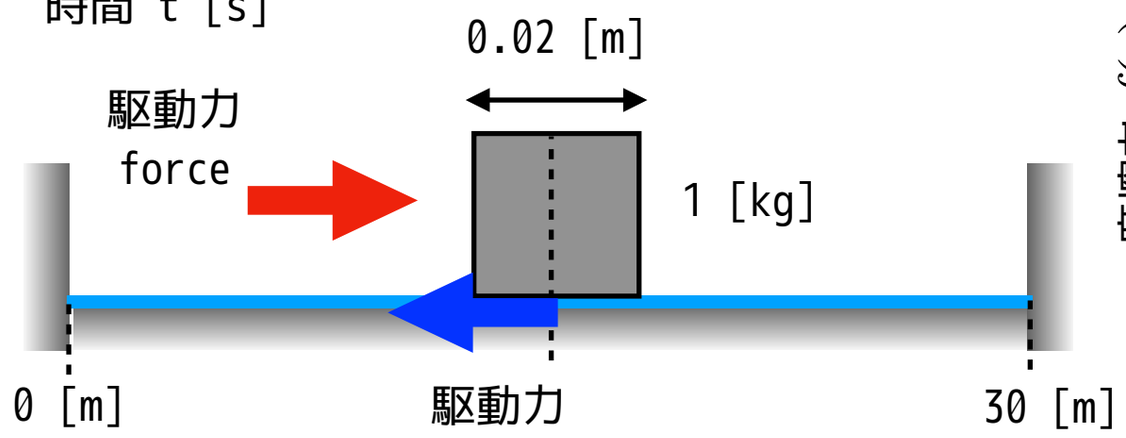


可動範囲と摩擦のある物体
massWithStopAndFriction

$$f(v) = F_{Coulomb} + F_{prop} \cdot v + F_{Stribeck} \cdot \exp(-fexp \cdot v)$$



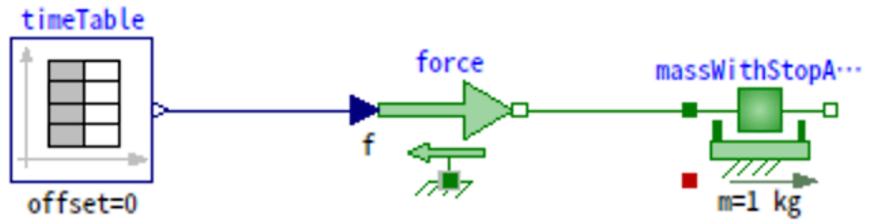
初期条件
位置 0 [m]
速度 0 [m/s]



モデル

timeTable

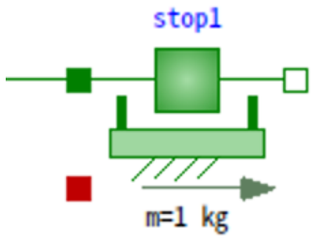
```
table = [0, 0; 1, 6; 3, 6; 4, 0; 15, 0; 16, -6; 18, -6; 19, 0; 30, 0]
```

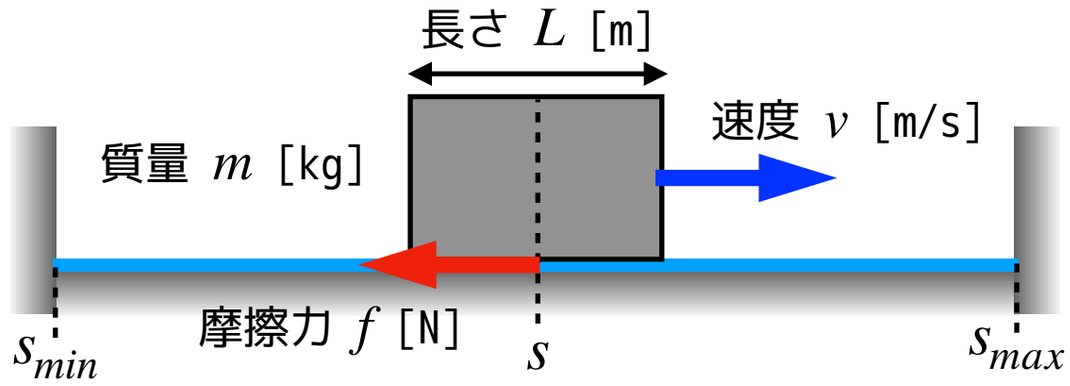


massWithStopAndFriction

```
m = 1 kg
F_prop = 0.3
F_Coulomb = 0.2
F_Stribeck =5
fexp = 1
L= 0.02
smax =30
smin= 0
```

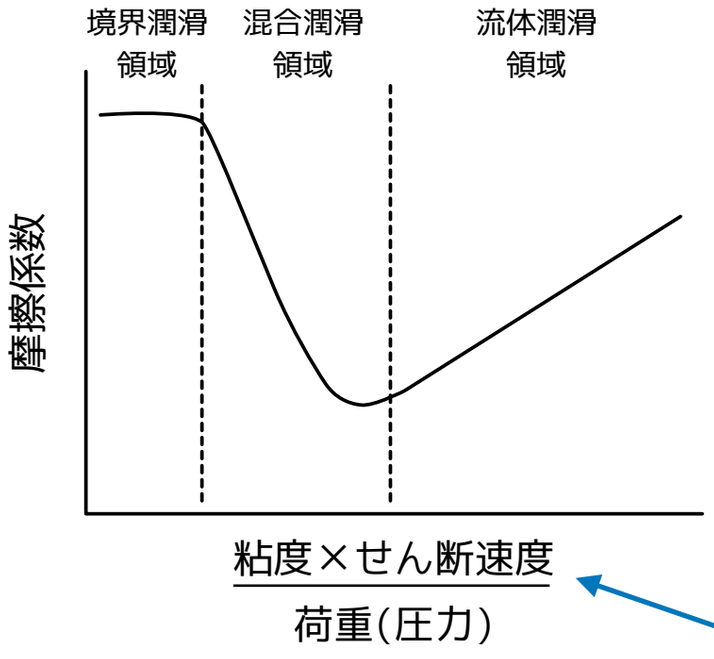
MassWithStopAndFriction — 摩擦力と可動範囲のある物体





可動範囲　$s_{min} + \frac{L}{2} \le s \le s_{max} - \frac{L}{2}$

摩擦力

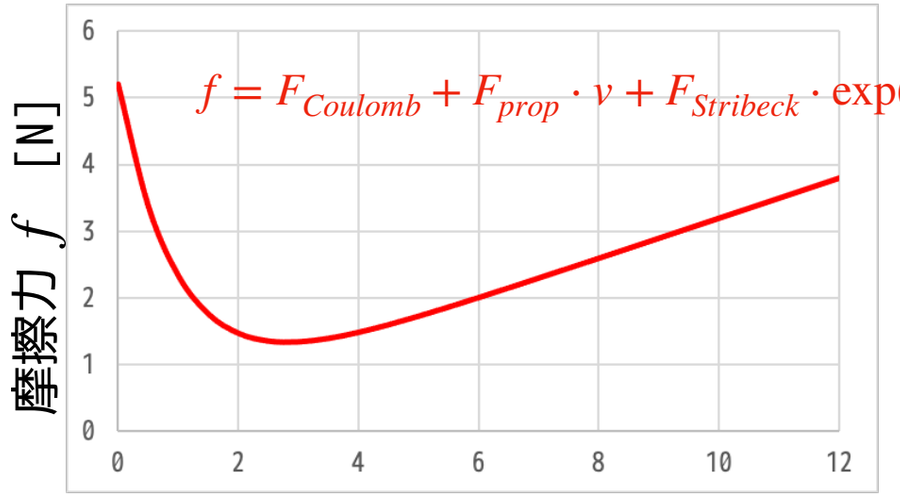


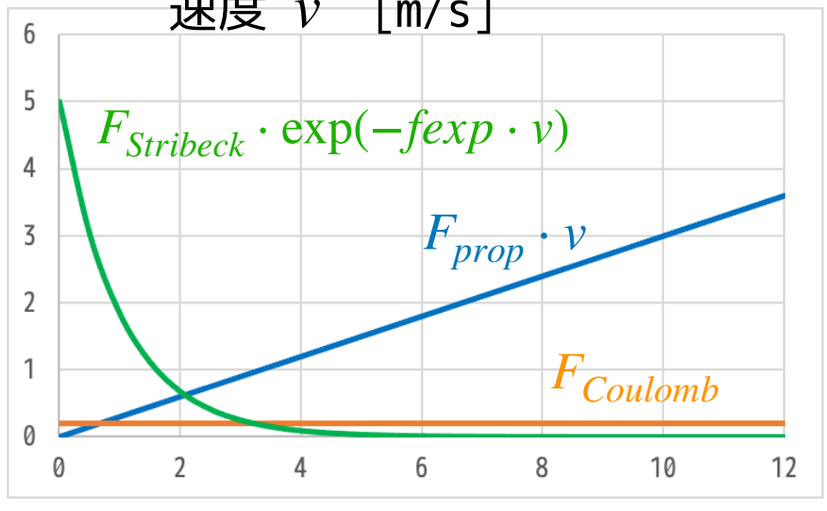
ストライベック曲線

モデル化

ベアリングの場合

粘度×回転数
荷重(圧力)





構成と方程式

2つのFlangeを持つモデル

$flange\_a.s$ [m]
$flange\_a.f$ [N]
$flange\_b.s$ [m]
$flange\_b.f$ [N]

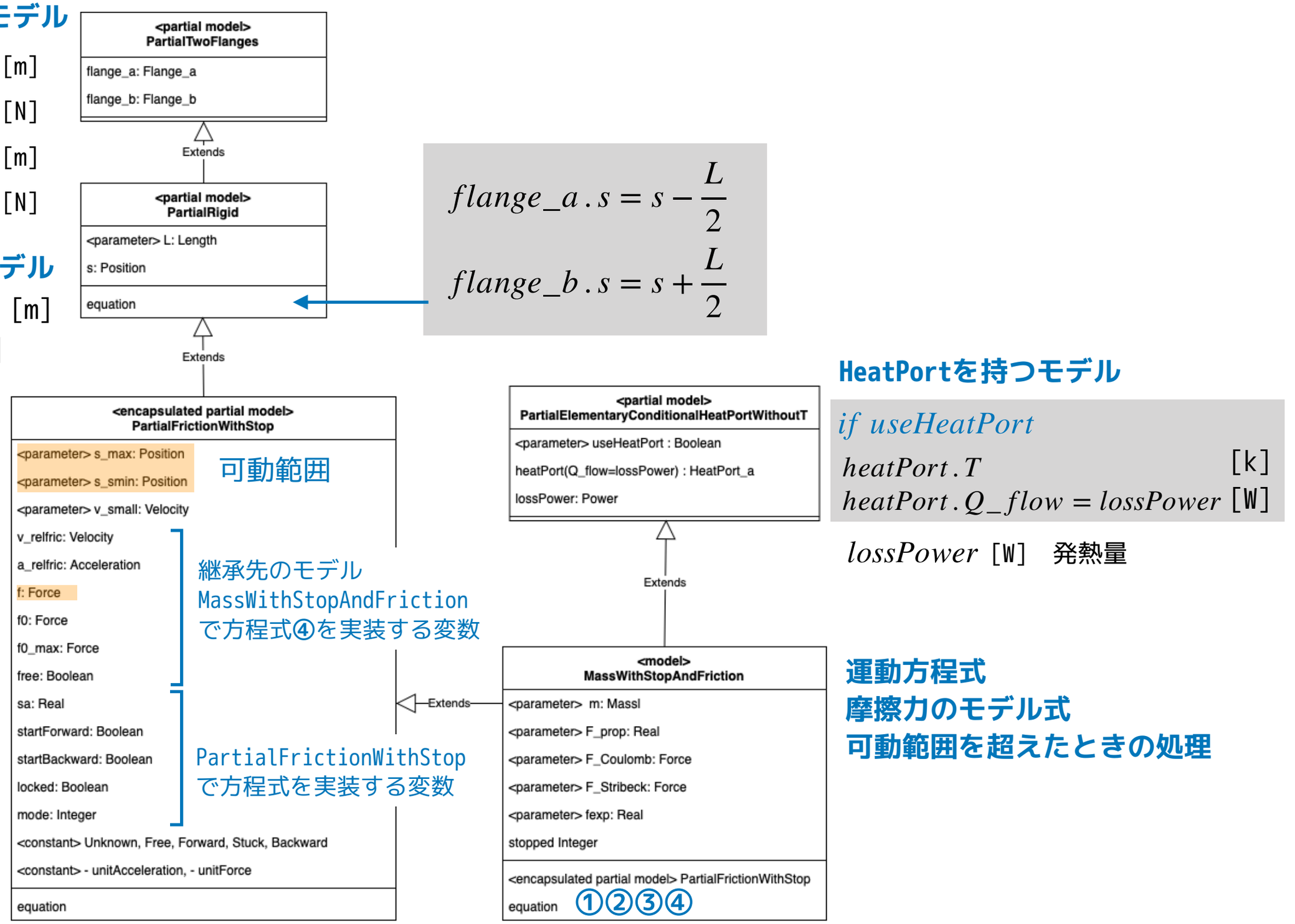
長さが変化しないモデル

s:物体の中心位置［m］
L: 物体の長さ［m］

$$flange\_a.s = s - \frac{L}{2}$$
$$flange\_b.s = s + \frac{L}{2}$$

物体の運動状態を場合分けするモデル

変数等の内容とふるまいは
PartialFriction
とほぼ同じですが、
可動範囲への対応が追加されています。



HeatPortを持つモデル

if useHeatPort
$heatPort.T$ [k]
$heatPort.Q\_flow = lossPower$ [W]

$lossPower$ [W]　発熱量

運動方程式
摩擦力のモデル式
可動範囲を超えたときの処理

MassWithStopAndFrictionの方程式

①運動方程式

$$v = \frac{ds}{dt}$$

$$a = \frac{dv}{dt}$$

$$0 = flange\_a.f + flange\_b.f - f - m \cdot \frac{dv}{dt}$$

②発熱量（熱損失）

$$lossPower = f \cdot v_{relfric}$$

③可動範囲を超えたときの、位置と速度の再初期化

判定条件

$$stopped = \begin{cases} -1, & s \le s_{min} + \frac{L}{2} \\ +1, & s \ge s_{max} + \frac{L}{2} \\ 0, & else \end{cases}$$

$$when\ stopped <> 0\ then$$

$$reinit(s,\ if\ stopped < 0\ then\ s_{min} + \frac{L}{2},\ else\ s_{max} - \frac{L}{2})$$

$$reinit(v,0)$$

$$end\ when$$

④摩擦力（PartialFrictionWithStopの継承先のモデルで実装する変数の方程式）

$f_0 = F_{Coulomb} + F_{Stribeck}$　$v = 0$ の動摩擦力

$f_{0\_max} \cdot 1.001$　最大静止摩擦力

$$free = f_0 \le 0\ and\ F_{prop} \le 0\ and\ s > s_{min} + \frac{L}{2}\ and\ s < s_{max} - \frac{L}{2}$$

$$v_{relfric} = v$$

$$a_{relfric} = a$$

$$f = \begin{cases} s_a \cdot unitForce & locked = true & \text{停止していて滑り始めない状態} \\ 0, & free = true & \text{非アクティブな状態} \\ F_{prop}v + F_{Coulomb} + F_{Stribeck}, & startForward = true & \text{前方に滑り始める条件を満たす状態} \\ F_{prop}v - F_{Coulomb} - F_{Stribeck}, & startBackword = true & \text{後方に滑り始める条件を満たす状態} \\ F_{prop}v + F_{Coulomb} + F_{Stribeck} \cdot \exp(-fexp \cdot |v|), & pre(mode) = Forward & \text{直前が前方に滑っている状態} \\ F_{prop}v - F_{Coulomb} - F_{Stribeck} \cdot \exp(-fexp \cdot |v|), & else\ (pre(mode) = Backward) & \text{直前が後方に滑っている状態} \end{cases}$$

PartialFrictionWithStop の方程式

変数のふるまいは、PartialFriction とほぼ同様なのでそちらを参考にしてください。変更されている部分に色をつけました。

mode 運動状態を表す離散的状態変数

可動範囲内のときだけ*Forward*や*Backward*になれる。
⇒可動範囲外では $mode = Stuck$ になる。

$$
\begin{aligned}
mode = \ & if\ free\ then\ Free \\
& else\ (\quad if\ (pre(mode) == Forward\ or\ pre(mode) == Free\ or\ startForward)\ and\ v_{relfric} > 0\ and\ s < (s_{max} - \frac{L}{2})\ then\ Forward \\
& \qquad else\ if\ (pre(mode) == Backward\ or\ pre(mode) == Free\ or\ startBackward)\ and\ v_{relfric} < 0\ and\ s > (s_{min} + \frac{L}{2})\ then\ Backward \\
& \qquad else\ Stuck\ )
\end{aligned}
$$

startForward 前方に滑り始める条件

位置が可動範囲上限を以上では前方に滑り始めない

$$
\begin{aligned}
startForward = \quad & pre(mode) == Stuck\ and \left(s_a > \frac{f_{0\_max}}{unitForce}\ and\ s < \left(s_{max} - \frac{L}{2} \right) \right. \\
& \left. or\ pre(startForward)\ and\ s_a > \frac{f_0}{unitForce}\ and\ s < \left(s_{max} - \frac{L}{2} \right) \right) \\
& or\ pre(mode) == Backward\ and\ v_{relfric} > v_{small} \\
& or\ initial()\ and\ (v_{relfric} > 0);
\end{aligned}
$$

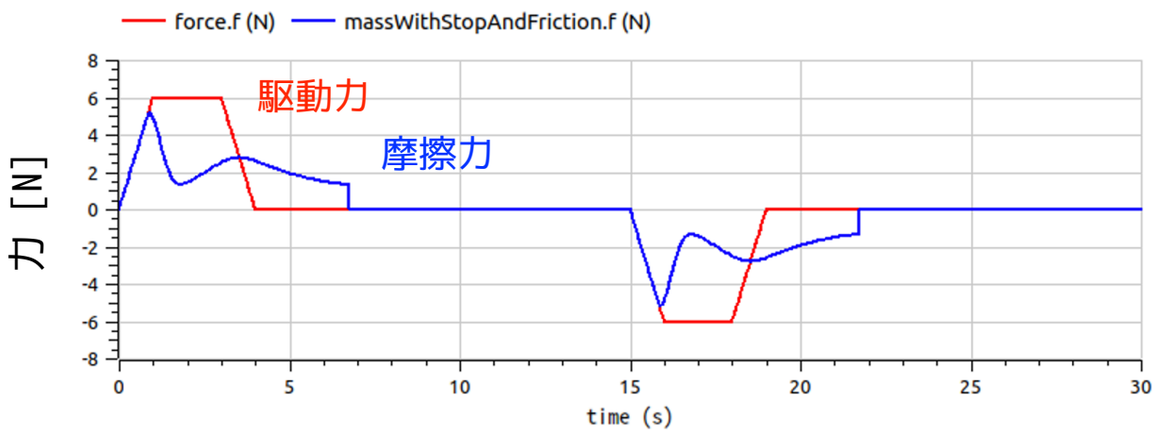
startBackward 後方に滑り始める条件

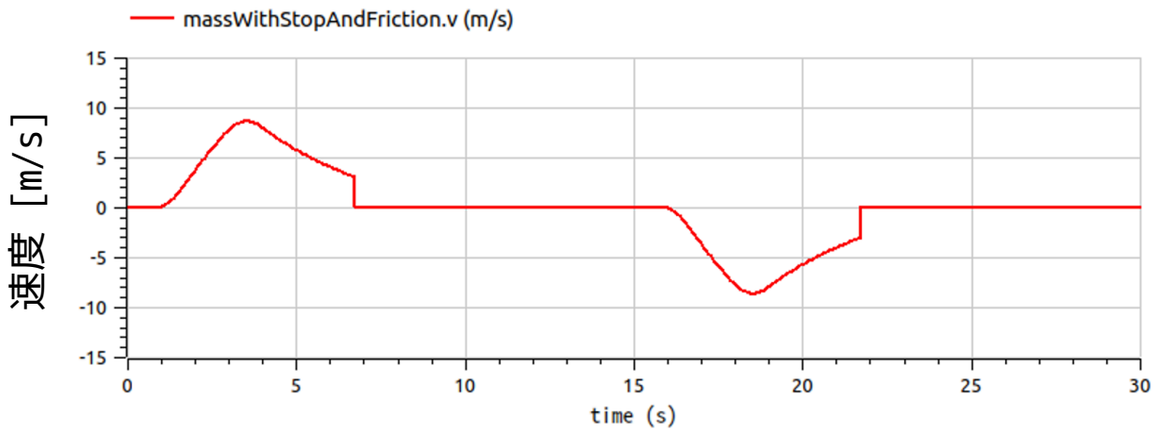
位置が可動範囲下限を以下では後方に滑り始めない

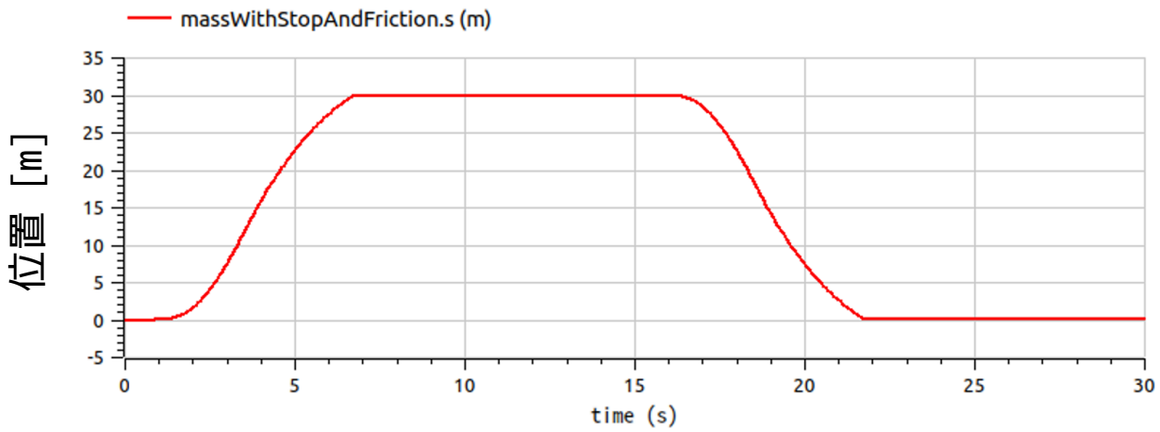
$$
\begin{aligned}
startBackward = \quad & pre(mode) == Stuck\ and \left(s_a < -\frac{f_{0\_max}}{unitForce}\ and\ s > \left(s_{min} + \frac{L}{2} \right) \right. \\
& \left. or\ pre(startBackword)\ and\ s_a < -\frac{f_0}{unitForce}\ and\ s > \left(s_{min} + \frac{L}{2} \right) \right) \\
& or\ pre(mode) == Forward\ and\ v_{relfric} < -v_{small} \\
& or\ initial()\ and\ (v_{relfric} < 0)
\end{aligned}
$$

$locked, s_a$ の式は PartialFrictionのlockedやsa と同様。

シミュレーション結果


force.f (N)
massWithStopAndFriction.f (N)
駆動力
摩擦力
力 [N]
time (s)

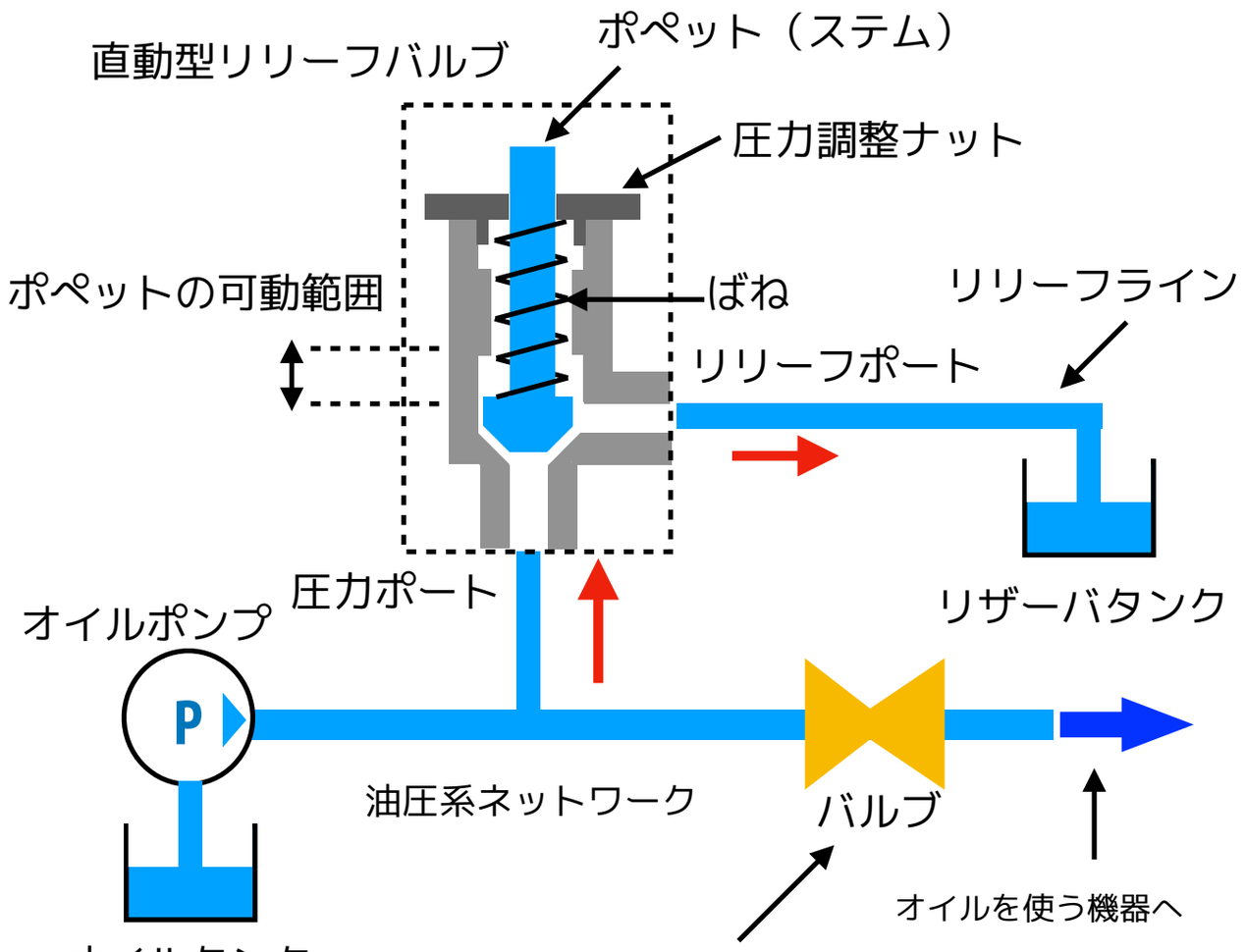

massWithStopAndFriction.v (m/s)
速度 [m/s]
time (s)


massWithStopAndFriction.s (m)
位置 [m]
time (s)

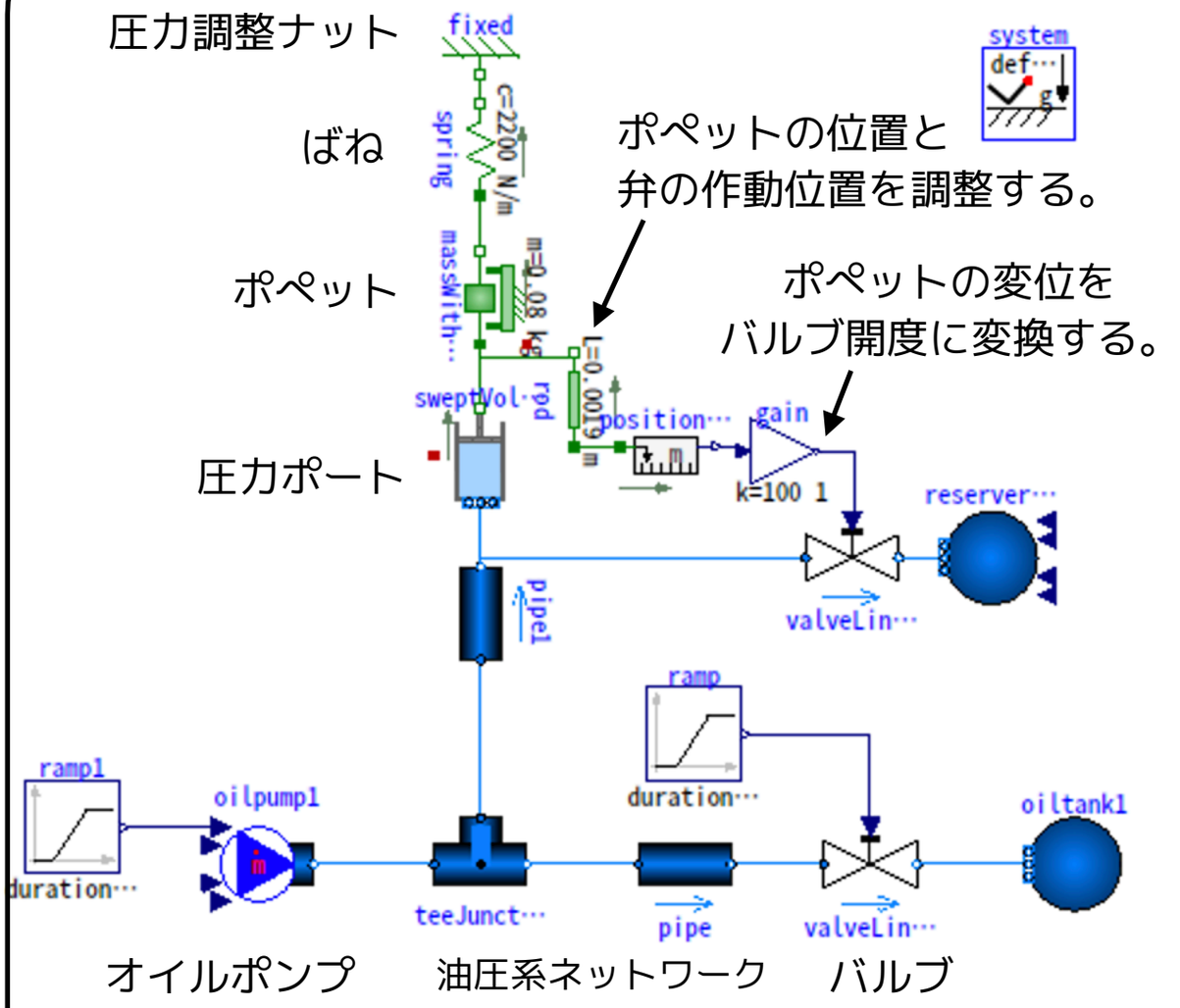
Example14 リリーフバルブをモデル化する。

- 油圧系ネットワークが正常な圧力（低圧）のときは、ばねによってポペットが圧力ポート側に押し付けられていて、リリーフバルブに流体は流れない。
- 油圧系ネットワークの圧力が異常に上昇すると、ポペットが押し上げられ、圧力ポートからリリーフポートに流体が流れ、圧力上昇が抑制される。

概要



モデル



Modelica.Fluid のモデルも使った応用例題です。

可動範囲のある物体として、ポペットをMassWithStopAndFrictionでモデル化しています。

ソースコードの最初の部分

```
model Example12
package Medium = Modelica.Media.CompressibleLiquids.LinearWater_pT_Ambient;
…
```

← 流体の種類を設定する

Modelica.Fluid のコンポーネントの共通設定

Medium: redeclare package Medium = Medium

摩擦に関する設定は
いいかげんです。😩

massWithStopAndFriction

```
m = 0.08 kg
F_prop = 0.1 N.s/m
F_Coulomb = 0.1 N
F_Stribeck =0.1 N
fexp = 0.1 s/m
L= 0.008 m
smax = 0.02 m
smin = 0.002
v.fixed = true
v.start = 0 m/s
s.fixed = true
s.start = 0.008 m
```

spring

```
c = 2200 N/m
s_rel0 = 0.0464 m
```

fixed

```
s0 = 0.046 m
```

rod

```
L = 0.0019 m
```

system

sweptVolume

```
pistonCrossAre = 5.027e-5 m2
clearance = 1e-6 m3
use_portData = false
```

positionSensor

gain

```
k = 100
```

valveLinear

```
m_flow_nominal = 0.33 kg/s
dp_nominal = 5e5 Pa
```

reservertank:Boundary_pT

```
p = 101325 Pa
```

pipe1

```
Length 0.01 m
diameter = 0.008 m
```

ramp

```
height = -0.95
duration = 0.05 s
offset = 1
startTime = 0.2 s
```

ramp1

```
height = 0.33 kg/s
duration = 0.05 s
offset = 0
startTime = 0.05 s
```

oilpump1:MassFlowSource_T

```
use_mflow_in = true
```

teejunctionVolume

```
v = 1e-6 m3
```

pipe

```
Length = 2 m
diameter = 0.008 m
```

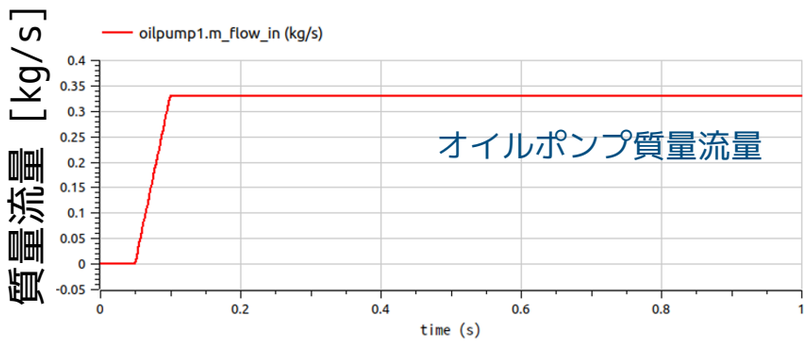
valvelinear1

```
m_flow_nominal = 0.33 kg/s
dp_nominal = iameter = 2e5 Pa
```

oiltank1:FixedBoundary

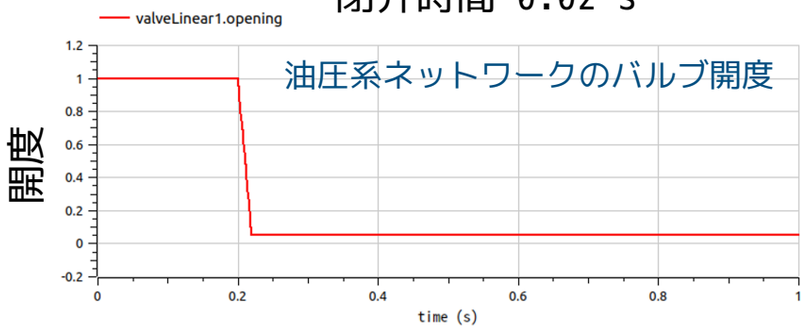
```
useP = true
p = 101325 Pa
```

シミュレーション結果

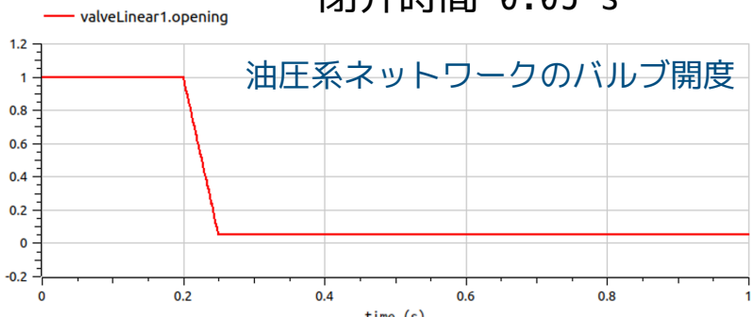


- **オイルポンプ流量は0.1秒以後 0.33 kg/s に固定しています。**
- **油圧系ネットワークのバルブの閉弁時間を長くすると圧力上昇が遅くなり、リリーフバルブの応答の様子が変化します。**

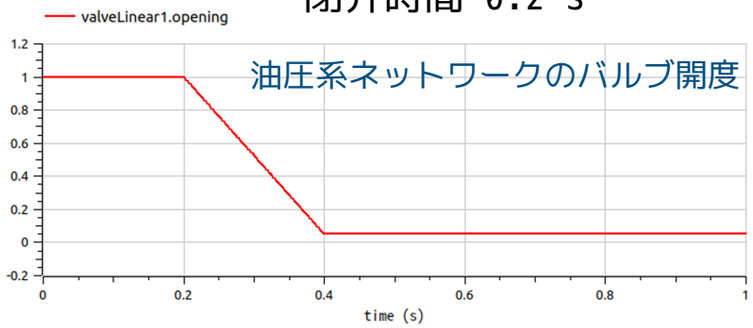
閉弁時間 0.02 s

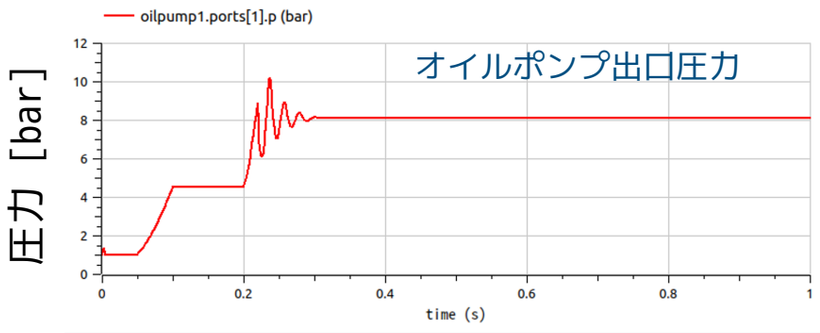


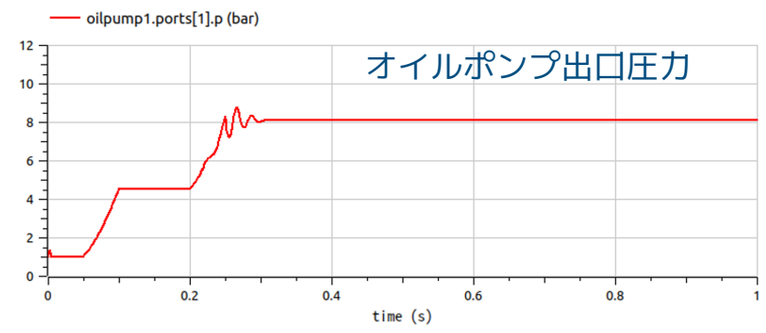
閉弁時間 0.05 s

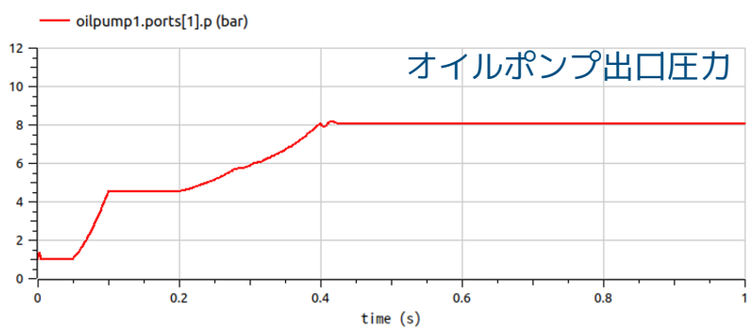


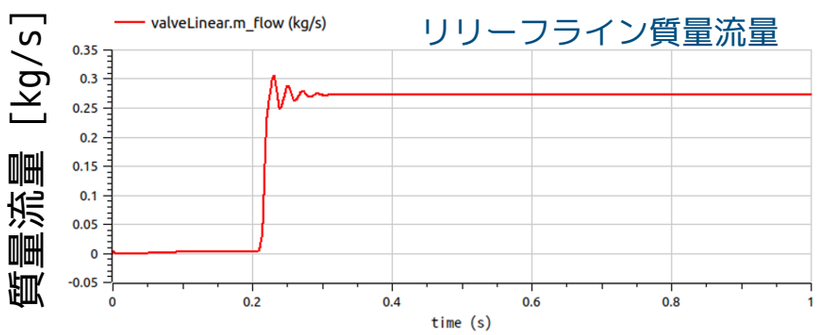
閉弁時間 0.2 s

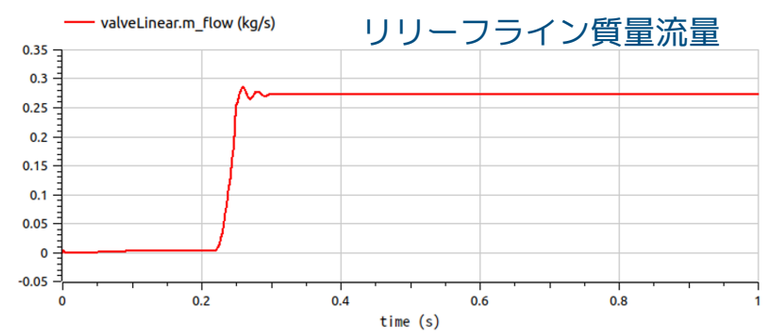


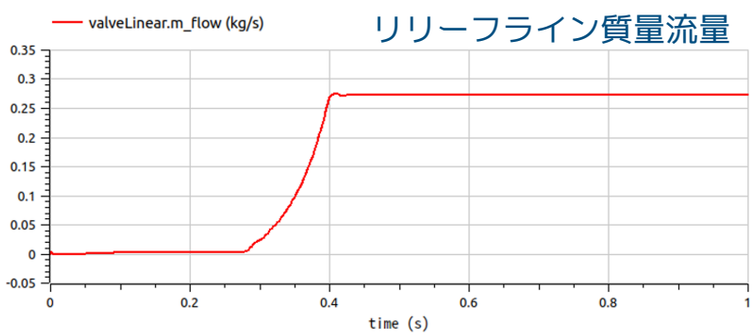






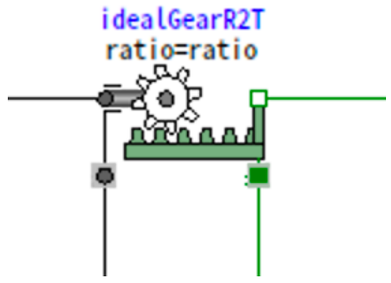




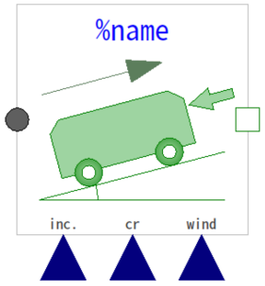


車両関係のコンポーネント

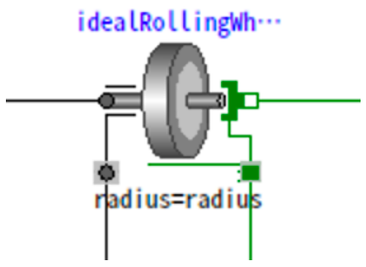
IdealGearR2T



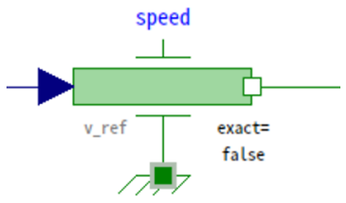
Vehicle



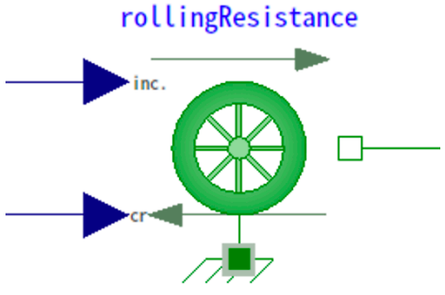
IdealRollingWheel



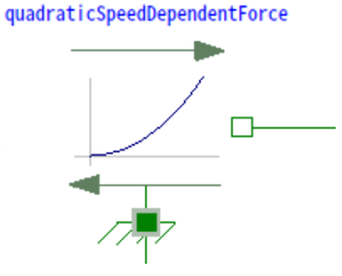
Speed



RollingResistance



QuadraticSpeedDependentForce

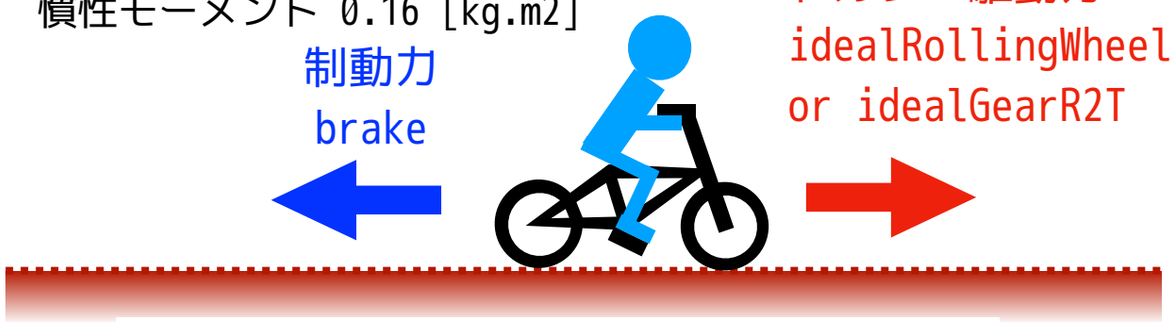


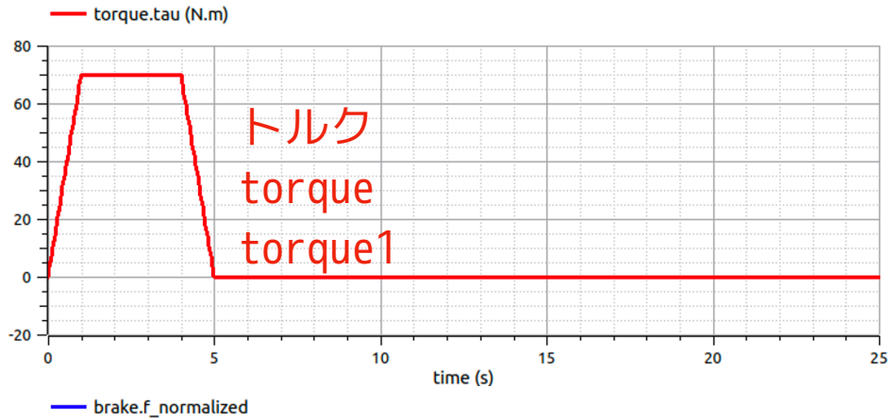
Example15 車輪にトルクを加えて自転車を駆動する

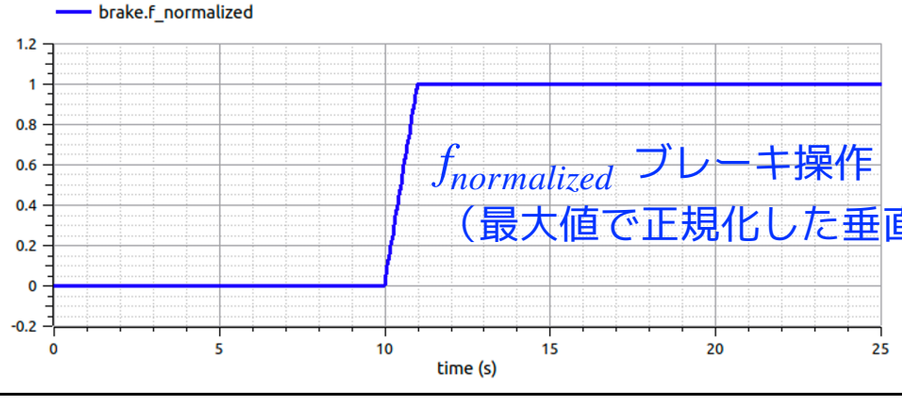
Example12で車体の駆動力を加える代わりに、車輪のトルクをの加えるモデルを作成します。
トルクと駆動力の変換に IdelaRollingWheel を使うモデルと IdealGeraR2T を使うるモデルを作ります。
車輪の慣性モーメントをコンポーネント Inertia　で表します。

概要

車輪
半径 0.33 [m]
トランスミッション比 3.03 [rad/m]
慣性モーメント 0.16 [kg.m2]







モデル

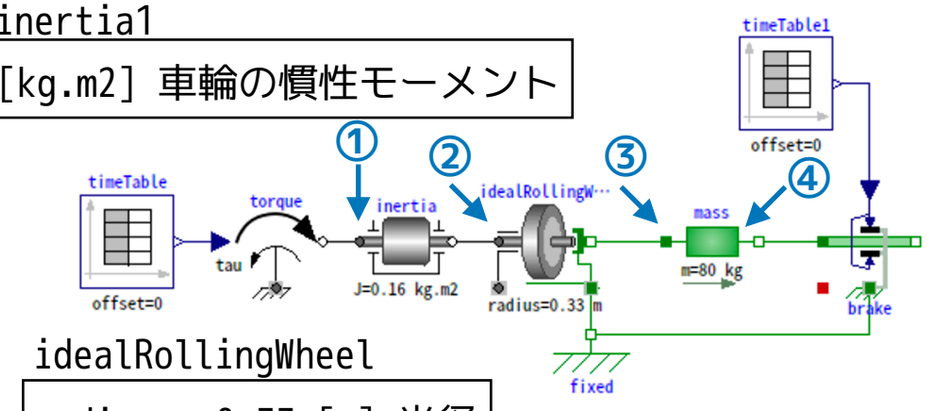
Example12 の force の代わりに
idealRollingWheel または idealGearR2T を使用し、
それぞれTorqueとInertiaを追加します。

timeTable, timeTable2

```
table = [0, 0; 1, 70; 4, 70; 5, 0; 100, 0]
```

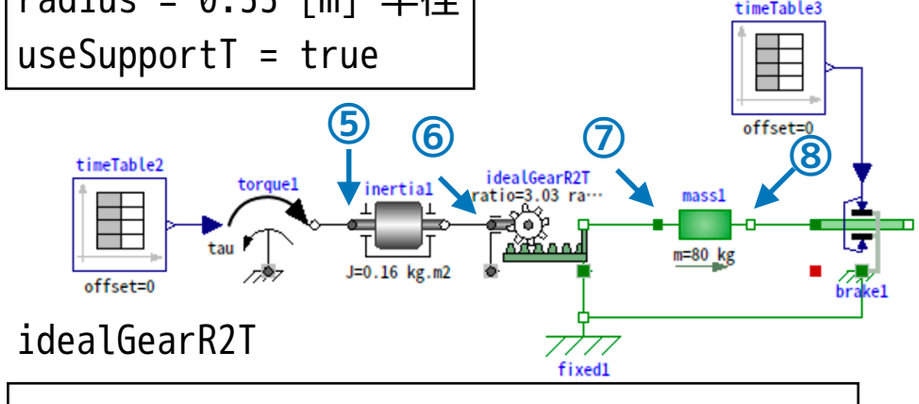
inertia, inertia1

```
J = 0.16 [kg.m2] 車輪の慣性モーメント
```



idealRollingWheel

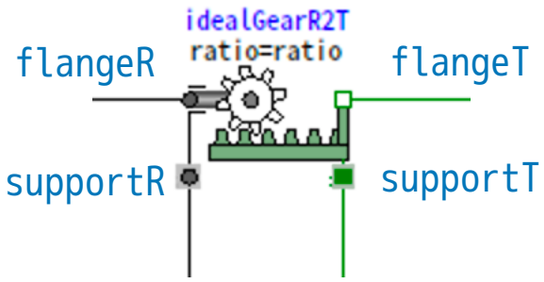
```
radius = 0.33 [m] 半径
useSupportT = true
```

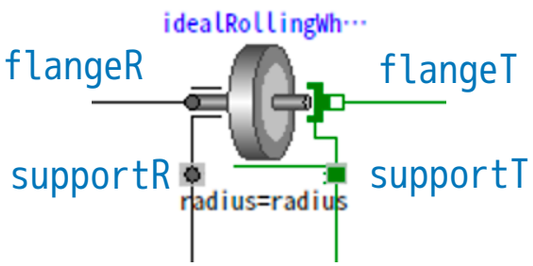


idealGearR2T

```
ratio = 3.03 [rad/m] トランスミッション比
useSupportT = true
```

IdealGearR2T, IdealRollingWheel — 回転と並進の変換





- Modelica.Mechanics.Rotational（回転系）のコンポーネントと Modelica.Mechanics.Translational（並進系）のコンポーネントを接続するためのコンポーネント。
- 回転系の flangeR を並進系の flangeT に変換する。
- 回転の基準角度 $internalSupportR.\varphi$ や並進の基準位置 $internalSupport.s$ を設定するために、内蔵されたコンポーネント fixedR, fixedT を使うか、supportR, supportT に接続した外部のコンポーネントを使うかをパラメータ useSupportR, useSupportT で選択できます。

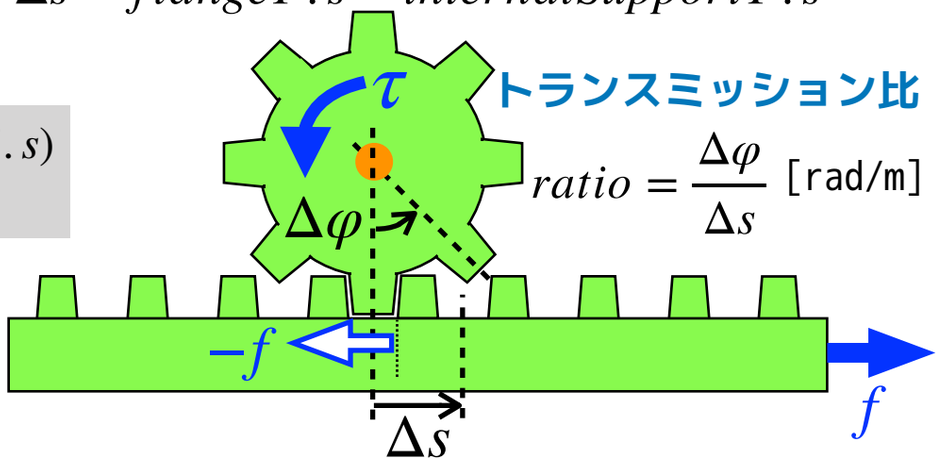
トルク $\tau = flangeR.\tau$　**回転角** $\Delta\varphi = flangeR.\varphi - internalSupportR.\varphi$

力 $-f = flangeT.f$　**変位** $\Delta s = flangeT.s - internalSupportT.s$

IdealGearR2T の方程式

① $(flangeR.\varphi - internalSupportR.\varphi) = ratio \cdot (flangeT.s - internalSupportT.s)$
$0 = ratio \cdot frangeR.\tau + flangeT.f$

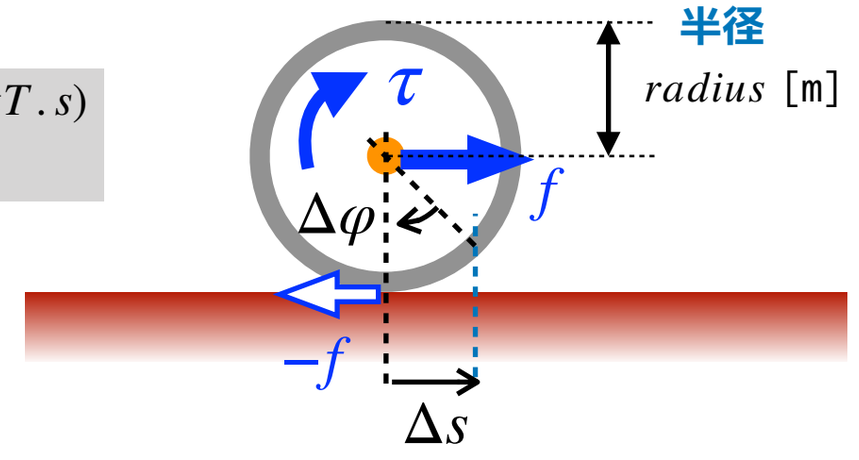
$$\Delta\varphi = ratio \cdot \Delta s \qquad f = ratio \cdot \tau$$



IdealRollingWheel の方程式

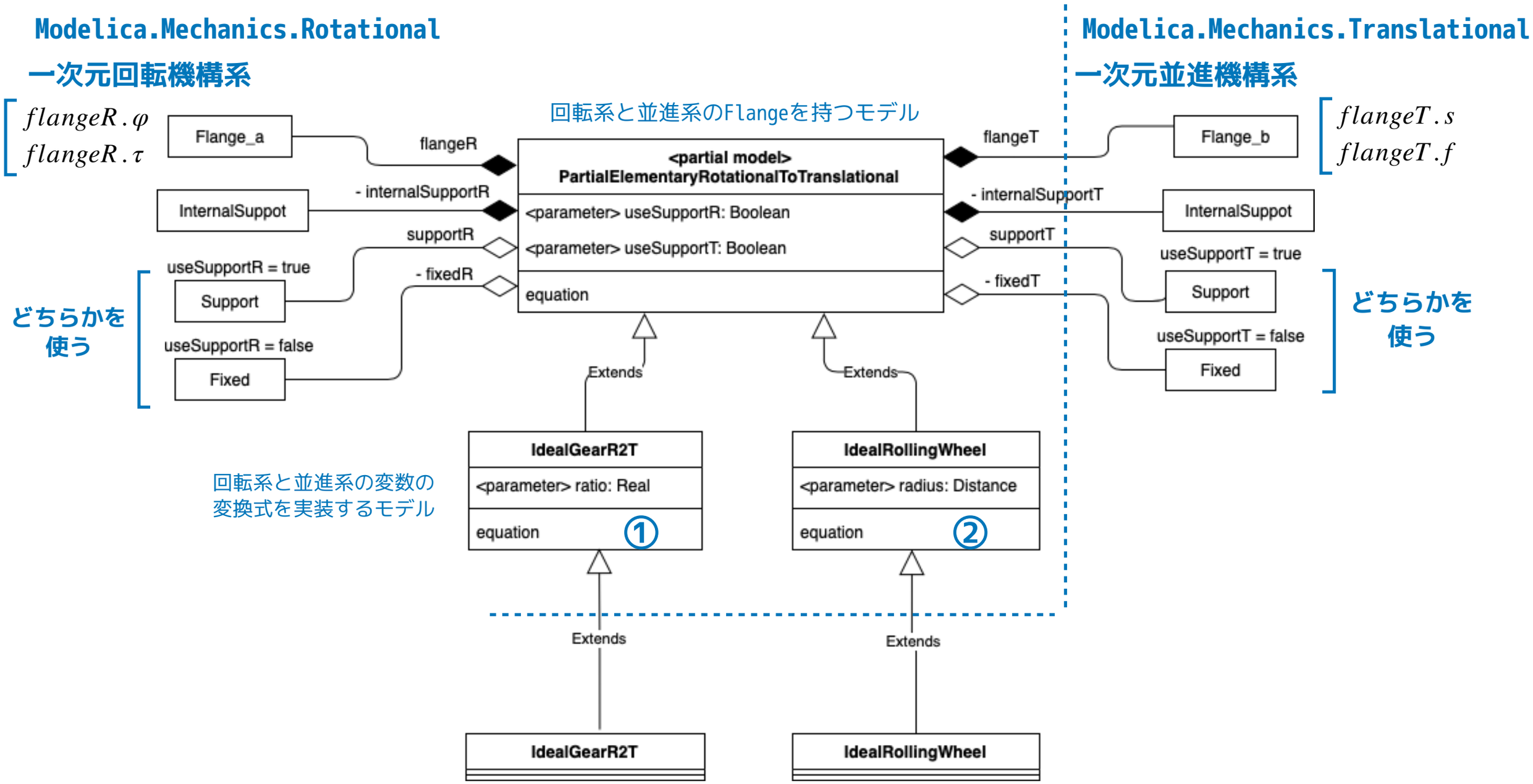
② $(flangeR.\varphi - internalSupportR.\varphi) \cdot radius = (flangeT.s - internalSupportT.s)$
$0 = radius \cdot frangeT.f + flangeR.\tau$

$$\Delta s = radius \cdot \Delta\varphi \qquad \tau = radius \cdot f$$



IdealGearR2T と IdealRollingWheel の構成と方程式

Modelica.Mechanics.Translational.Components の IdealGearR2T と IdealRollingWheel は、
Modelica.Mechanics.Rotational.Components にある同名のコンポーネントを拡張（継承）して作られている。



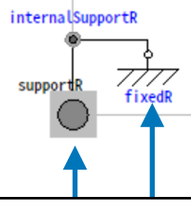
PartialElementaryRotationalTranslational 回転系と並進系の Flange を持つモデル

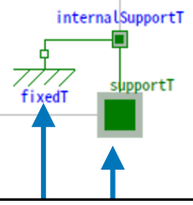
一次元回転機構系 $\begin{bmatrix} flangeR.\varphi \\ flangeR.\tau \end{bmatrix}$

$\begin{bmatrix} flangeT.s \\ flangeT.f \end{bmatrix}$ 一次元並進機構系

$internalSupportR.\varphi$ 基準角度
$internalSupportR.\tau = -flangeR.\tau$

$internalSupportT.s$ 基準位置
$internalSupportT.f = -flangeT.f$






```
Rotational.Components.Fixed fixedR if not useSupportR
Rotational.Interfaces.Support supportR if useSupportR
```

```
Translational.Components.Fixed fixedT if not useSupportT
Translational.Interfaces.Support supportT if useSupportT
```

基準位置の選択方法

useSupportR = true
基準角度を$supportR$で設定します。

$$supportR.\varphi = internalSupportR.\varphi$$
$$supportR.\tau = internalSupportR.\tau$$

useSupportR = false
基準角度を$fixedR$で設定します。

$$fixedR.\varphi = internalSupportR.\varphi = 0$$
$$fixedR.\tau = -internalSupportR.\tau$$

useSupportT = true
基準位置を$supportT$で設定します。

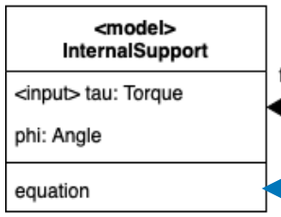
$$supportT.s = internalSupportT.s$$
$$supportT.f = internalSupportT.f$$

useSupportT = false
基準位置を$fixedT$で設定します。

$$fiexdT.s = internalSupportT.s = 0$$
$$fixedT.f = -internalSupportT.f$$

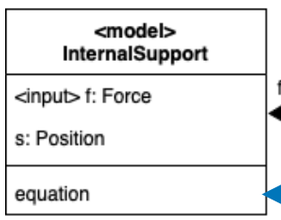
InternalSupport

Modelica.Mechanics.Rotational.Interfaces.InternalSupport



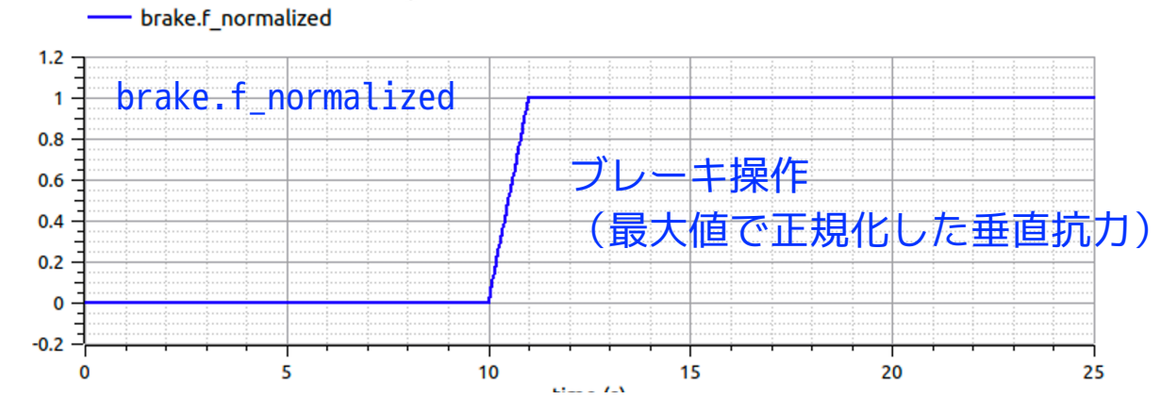
$$flange.\tau = \tau$$
$$flange.\varphi = \varphi$$

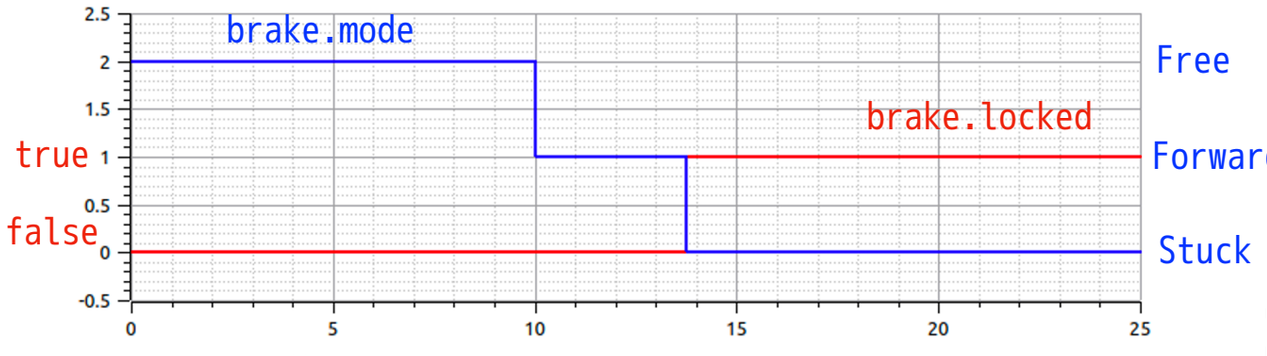
Modelica.Mechanics.Translational.Interfaces.InternalSupport

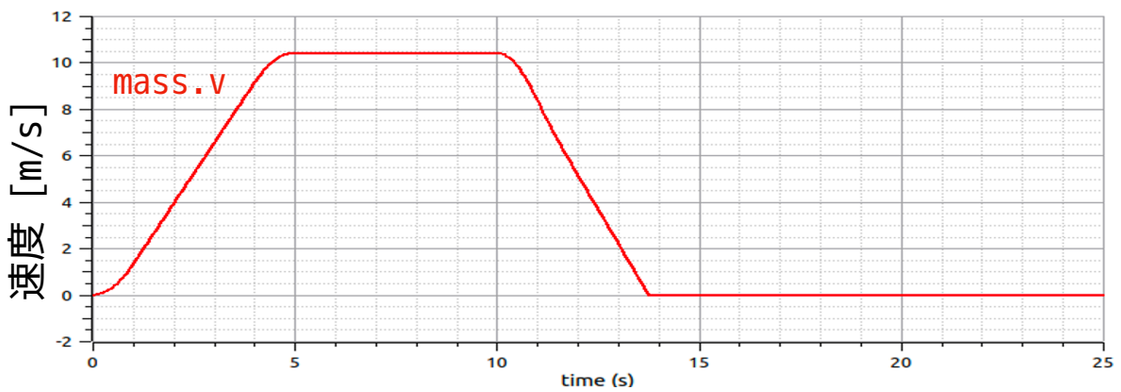


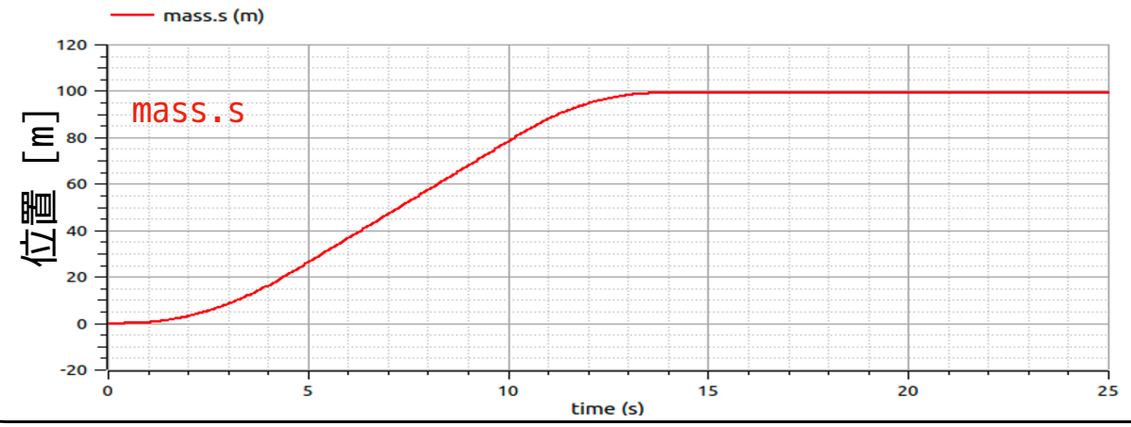
$$flange.f = f$$
$$flange.s = s$$

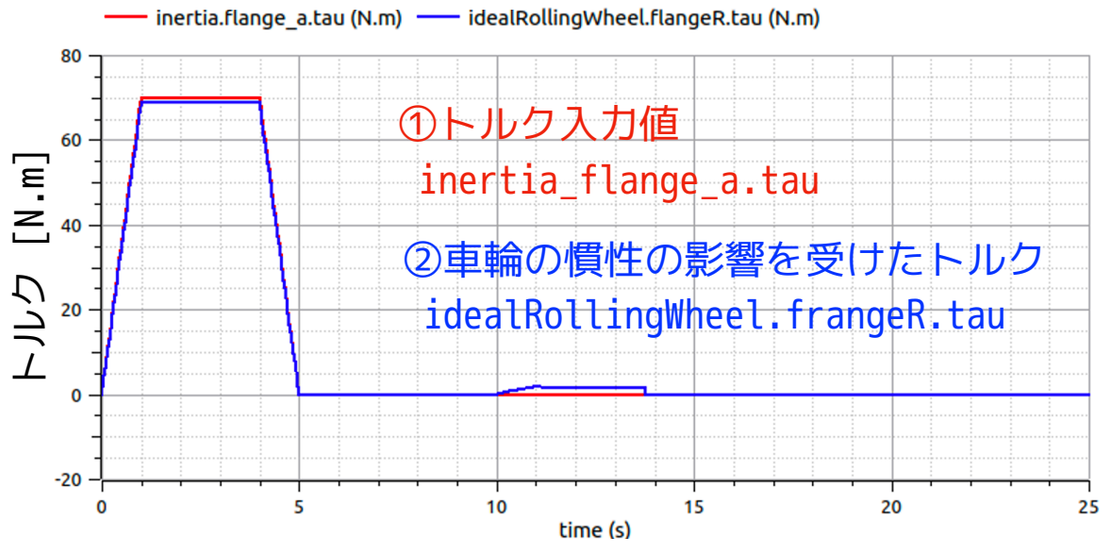
IdealRollingWheel を使ったモデルのシミュレーション結果

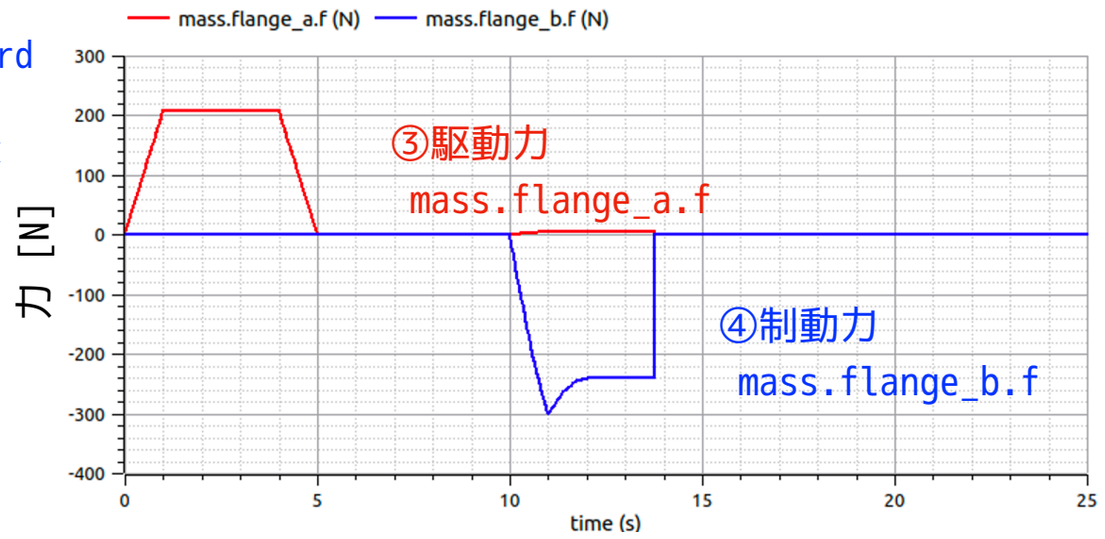


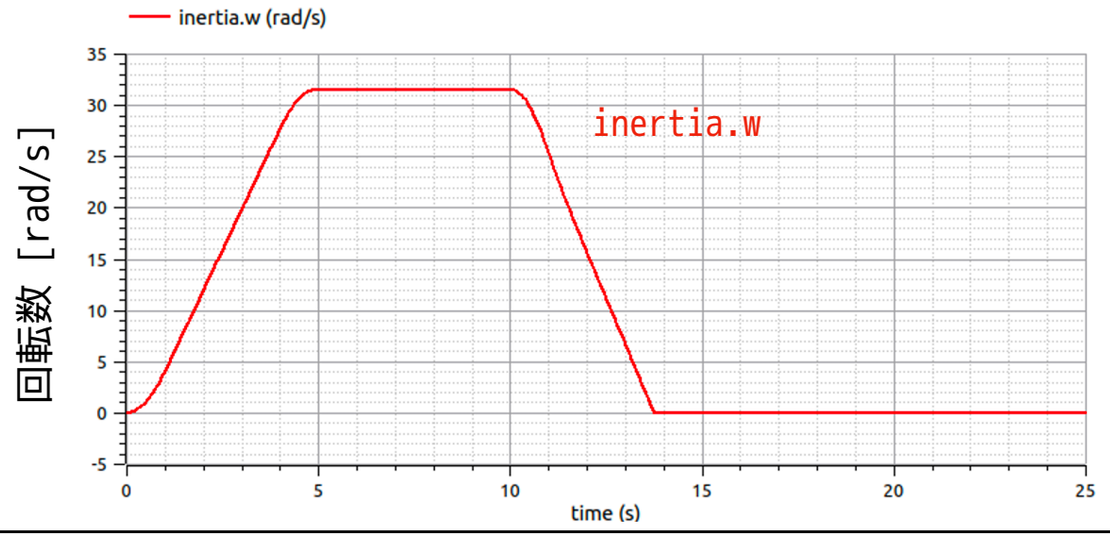




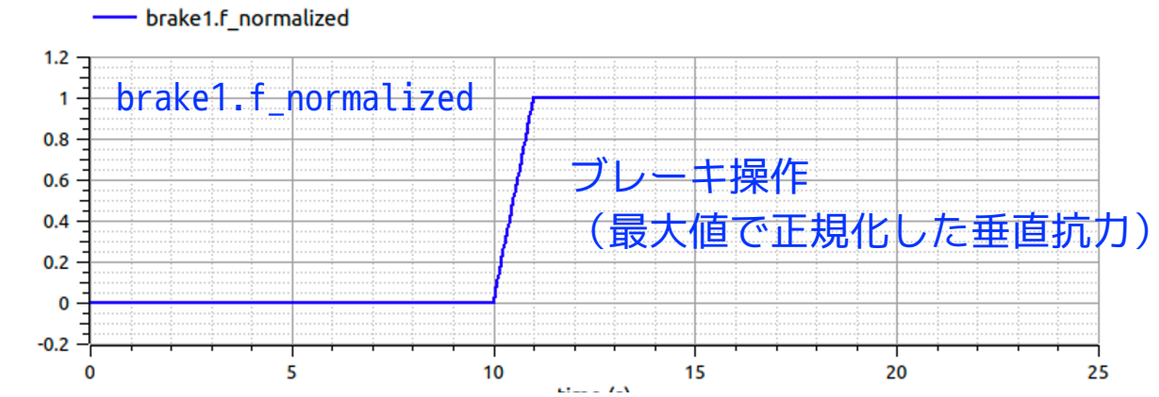


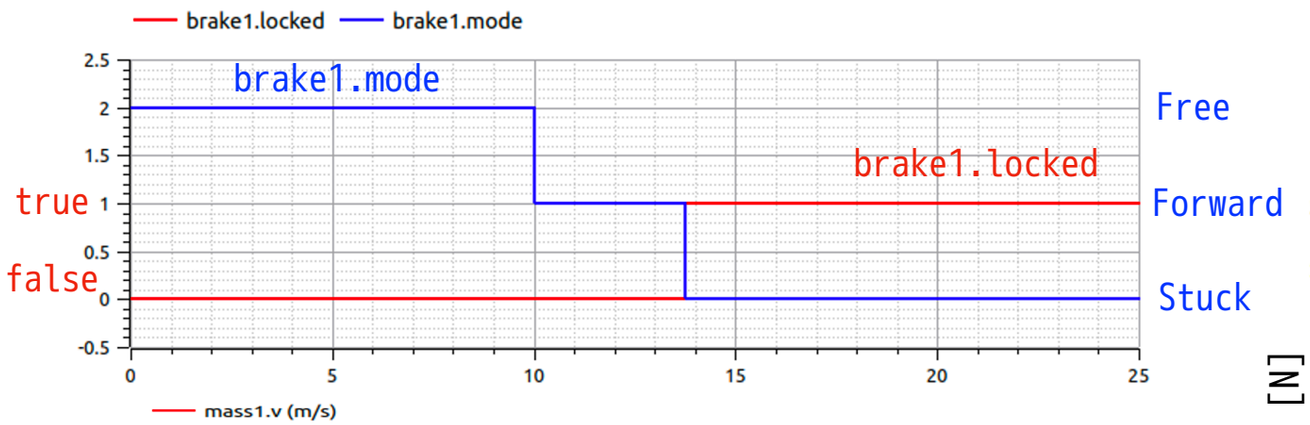


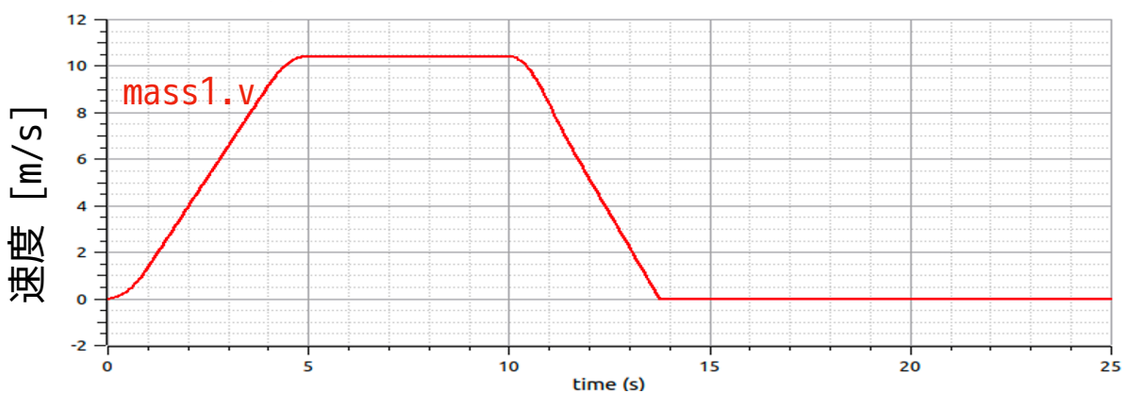


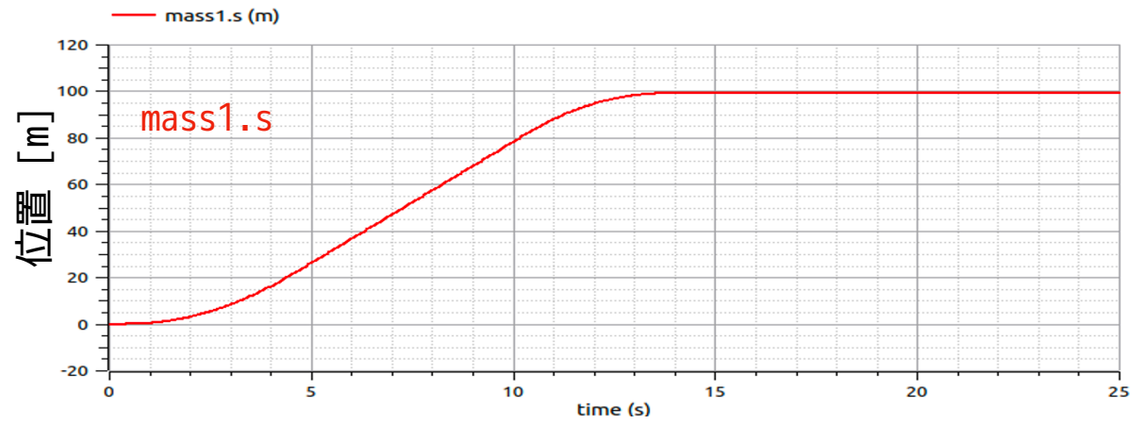


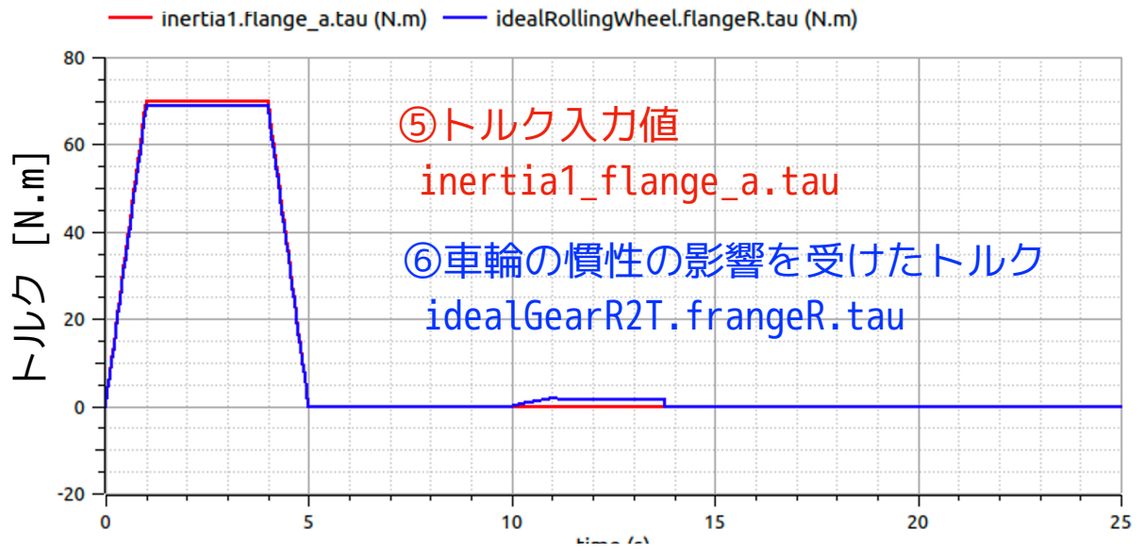
IdealGearR2T を使ったモデルのシミュレーション結果

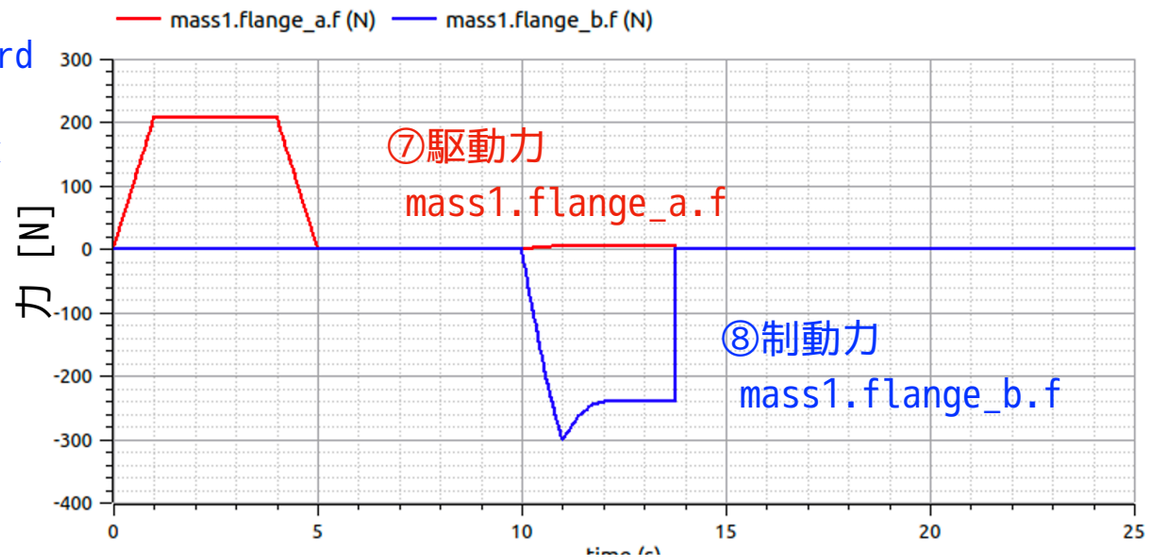


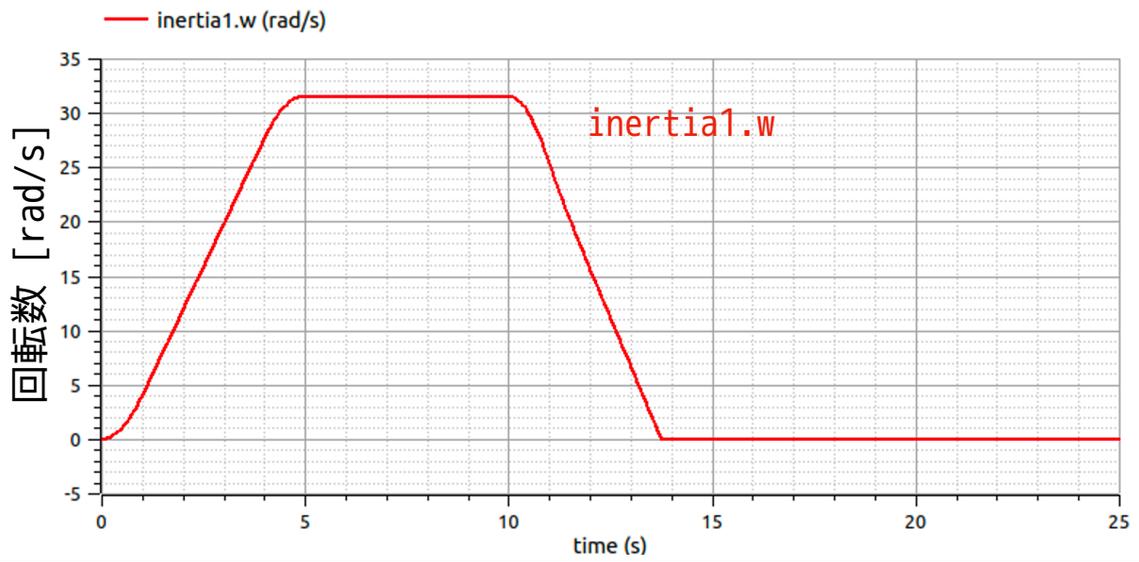










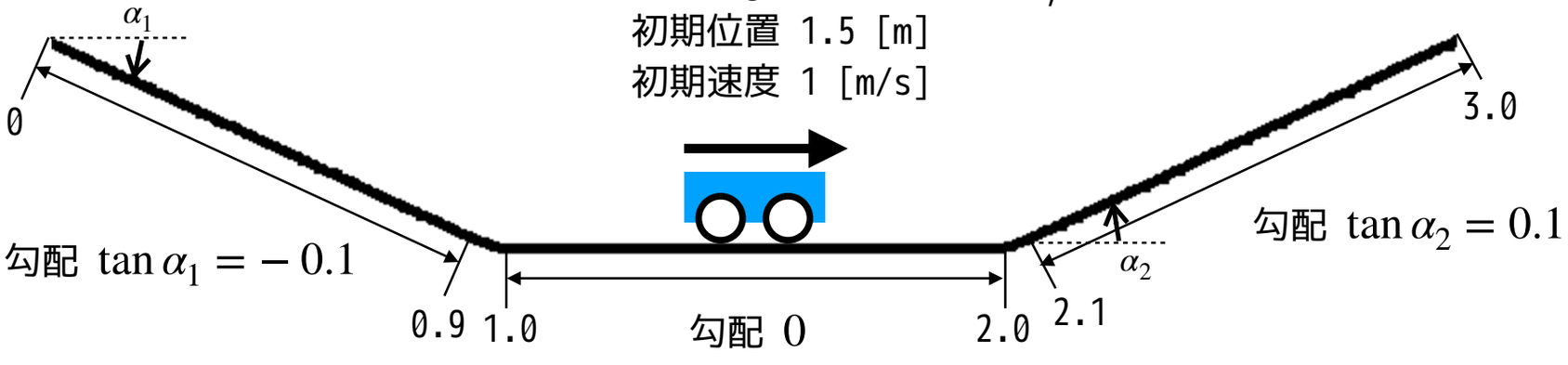


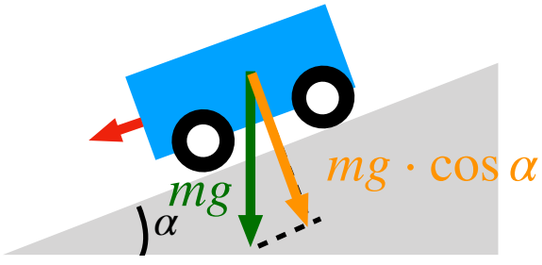
Example16 坂道で台車を転がす。

概要

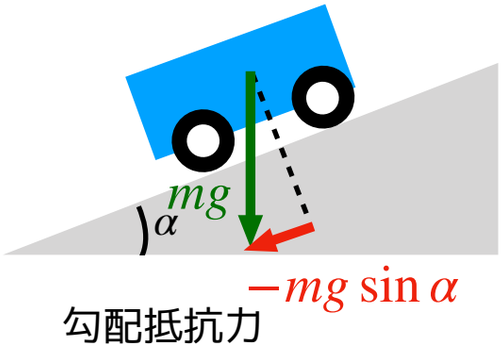
質量 m = 1 [kg]
重力 mg = 9.8 [N]
初期位置 1.5 [m]
初期速度 1 [m/s]

転がり抵抗係数
$c_r = 0.005 \ or \ 0.002$





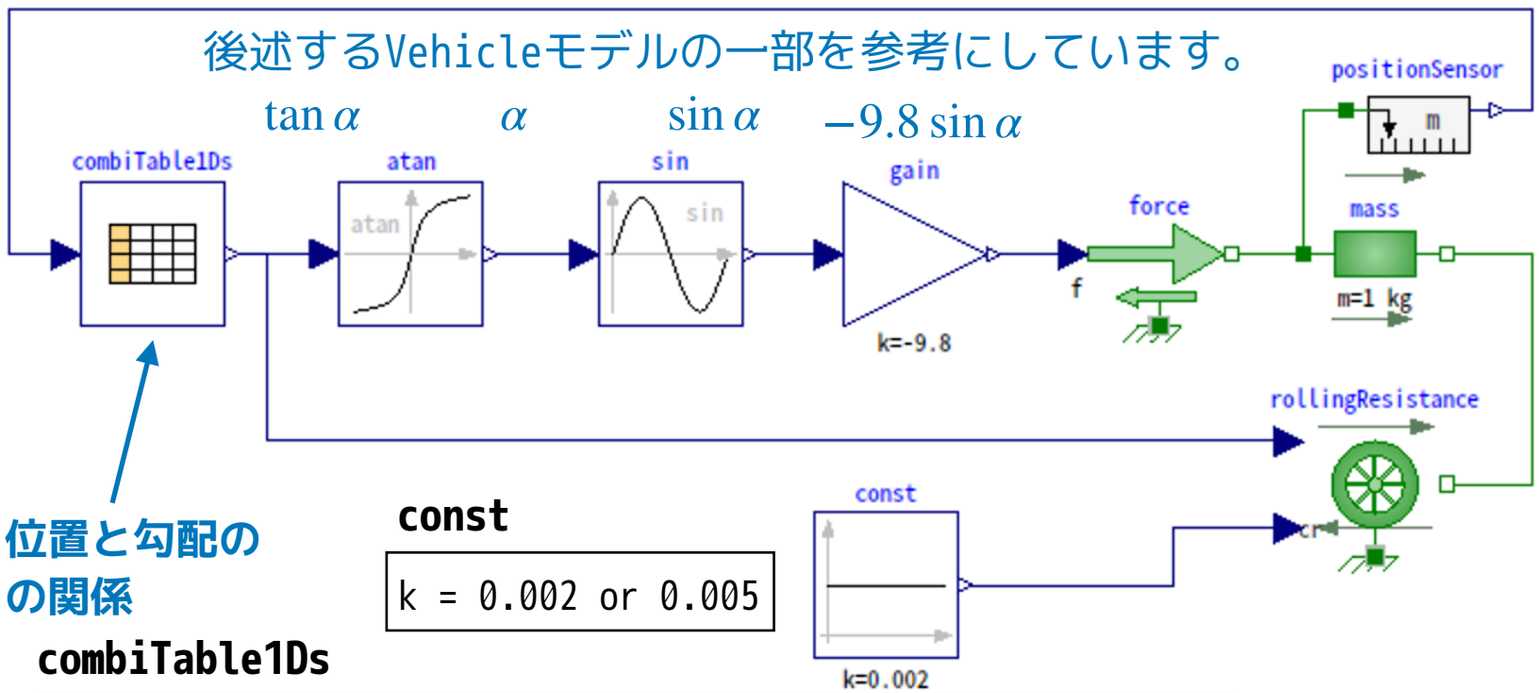
転がり抵抗力
$f_{Roll} = -Cr \cdot mg \cdot \cos\alpha$



勾配抵抗力
$f_{Grav} = -mg \sin\alpha$

モデル

後述するVehicleモデルの一部を参考にしています。



位置と勾配の
の関係

combiTable1Ds

```
table = [0, -0.1; 0.9, -0.1; 1, 0; 2, 0; 2.1, 0.1; 3, 0.1]
```

const

```
k = 0.002 or 0.005
```

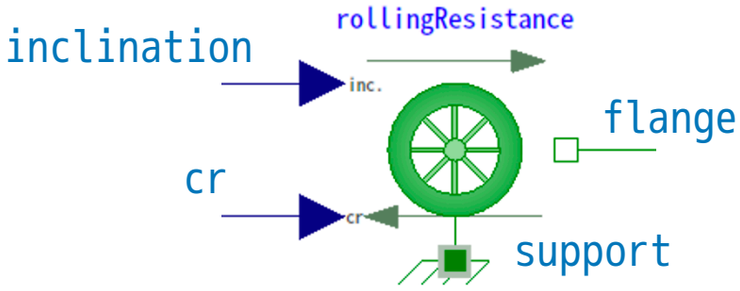
mass

```
m = 1 [kg]
L = 0 [m]
v.fixed = ture
v.start = 1 [m/s]
s.fixed = true
s.start = 1.5 [m]
```

rollingRegistance

```
fweight = 9.8 [N]
useCRInput = true
useInclinationInput =true
```

RollingResistance — 転がり抵抗力のモデル

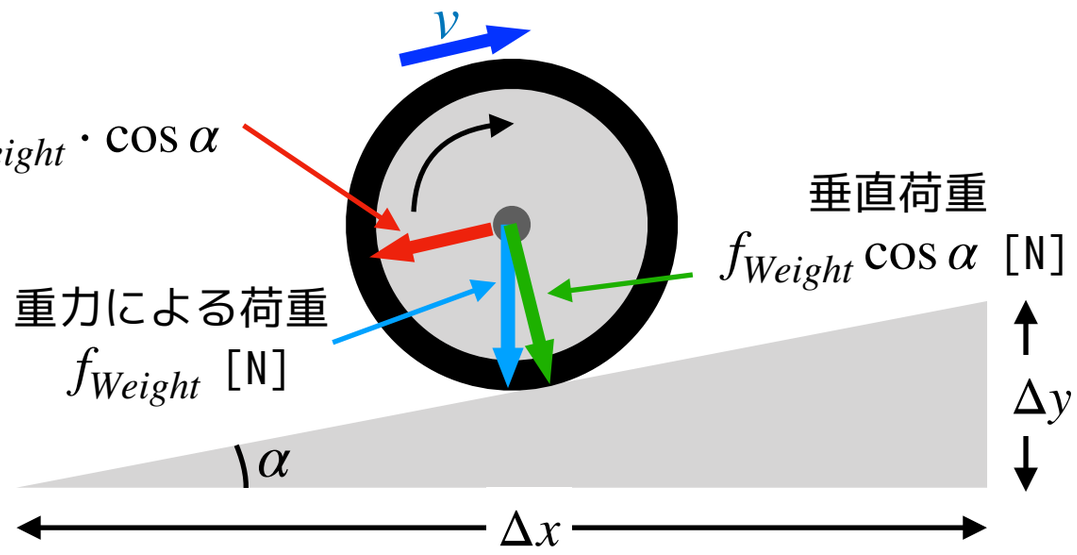


$$inclination = \tan\alpha = \frac{\Delta y}{\Delta x}$$ 勾配

Cr 転がり抵抗係数

転がり抵抗力

$$f = -Cr \cdot F_{Weight} \cdot \cos\alpha$$



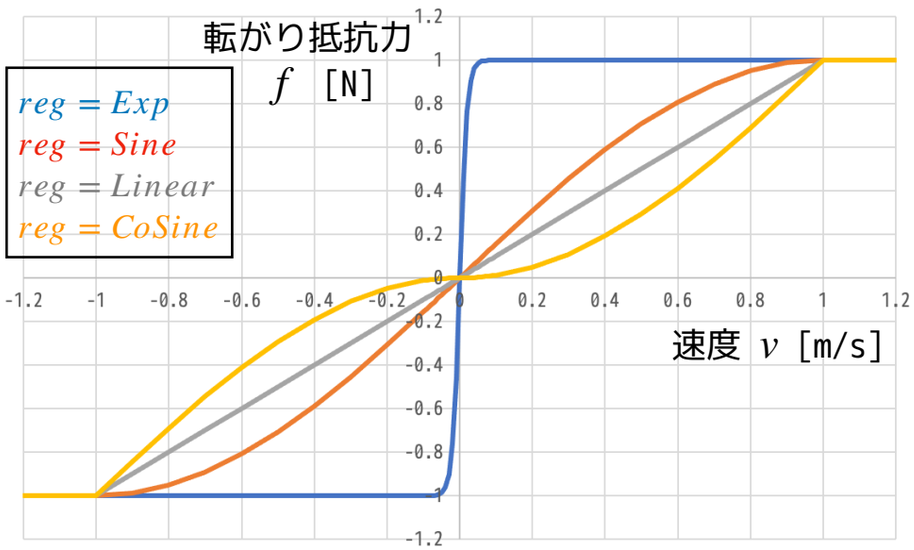
転がり抵抗力は速度と逆向きなので $v = 0$ で不連続になる。$|v| < v_0$ でregularization を行う。

① $v = \frac{ds}{dt}$

$$f_{nominal} = -Cr_{internal} \cdot f_{Weight} \cos(\arctan(inclination_{internal}))$$

$$f = \begin{cases} -f_{nominal}\left(\frac{2}{1+\exp(\frac{-v}{0.01v0})} - 1\right), & reg = Exp \\ -f_{nominal} \cdot \mathrm{smooth}_1 \begin{cases} \mathrm{sign}(v), & |v| \geq v_0 \\ \sin\left(\frac{\pi}{2}\frac{v}{v_0}\right), & else \end{cases}, & reg = Sine \\ -f_{nominal} \cdot \begin{cases} \mathrm{sign}(v), & |v| \geq v_0 \\ \frac{v}{v_0}, & else \end{cases}, & reg = Linear \\ -f_{nominal} \cdot \begin{cases} \mathrm{sign}(v), & |v| \geq v_0 \\ \mathrm{sign}(v)\left(1 - \cos\left(\frac{\pi}{2}\frac{v}{v_0}\right)\right), & else \end{cases} & else\ (reg = CoSine) \end{cases}$$

$|v| < v_0$ の場合の regularization
（速度の符号が変化するときの、
転がり抵抗力の不連続性を避ける数値的処理）



$v_0 = 1.0$ [m/s], $f_{nominal} = -1$ [N] とした場合の例

RollingResistance の構成と方程式

Flange と Support を持つモデル →

s [m]　　位置

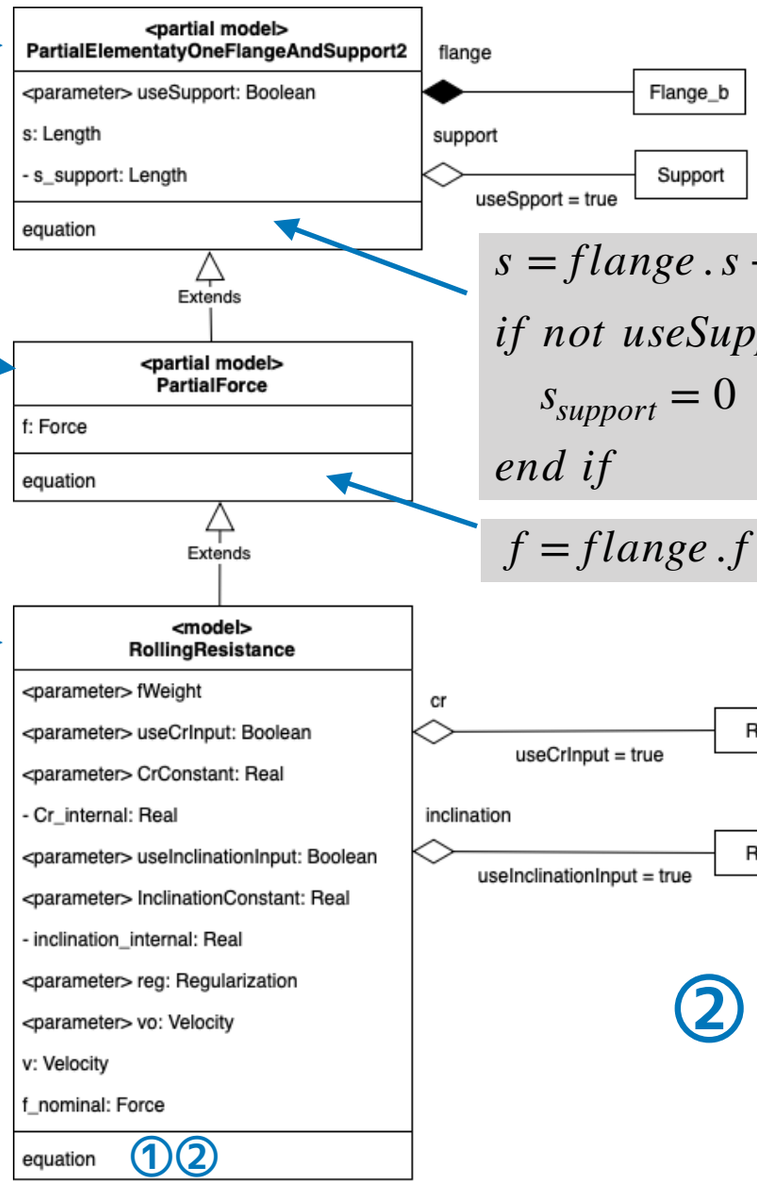
$s_{support}$ [m] 基準位置

力を表すベースモデル →

f [N] flangeに加えられる力

継承先のモデルで
f の方程式を実装する。

転がり抵抗力のモデル →



$$\begin{bmatrix} flange.s & [m] \\ flange.f & [N] \end{bmatrix}$$

$$\begin{aligned} &if\ useSupport \\ &\quad support.s = s_{support} \\ &\quad supoort.f = -flange.f \end{aligned}$$

$$\begin{aligned} &s = flange.s - s_{support} \\ &if\ not\ useSupport \\ &\quad s_{support} = 0 \\ &end\ if \end{aligned}$$

$$f = flange.f$$

転がり抵抗係数（実数入力信号）

$$\begin{aligned} &if\ useCrInput \\ &\quad cr = Cr_{internal} \end{aligned}$$

勾配（実数入力信号）

$$\begin{aligned} &if\ useInclinationInput \\ &inclination = inclination\_internal \end{aligned}$$

②

$$\begin{aligned} &if\ not\ useCrInput\ then \\ &\quad Cr_{internal} = CrConstant \\ &end\ if \\ &if\ not\ useInclinationInput\ then \\ &\quad inclination_{internal} = inclinationConstant \\ &end\ if \end{aligned}$$

変数

v [m/s]　　速度

$f_{nominal}$ [N] 転がり摩擦力

パラメータ

f_{Weight} [N] 重力による荷重

reg　regularization の選択肢

v_0　regularizationを行う流速の上限値

$useCrInput$

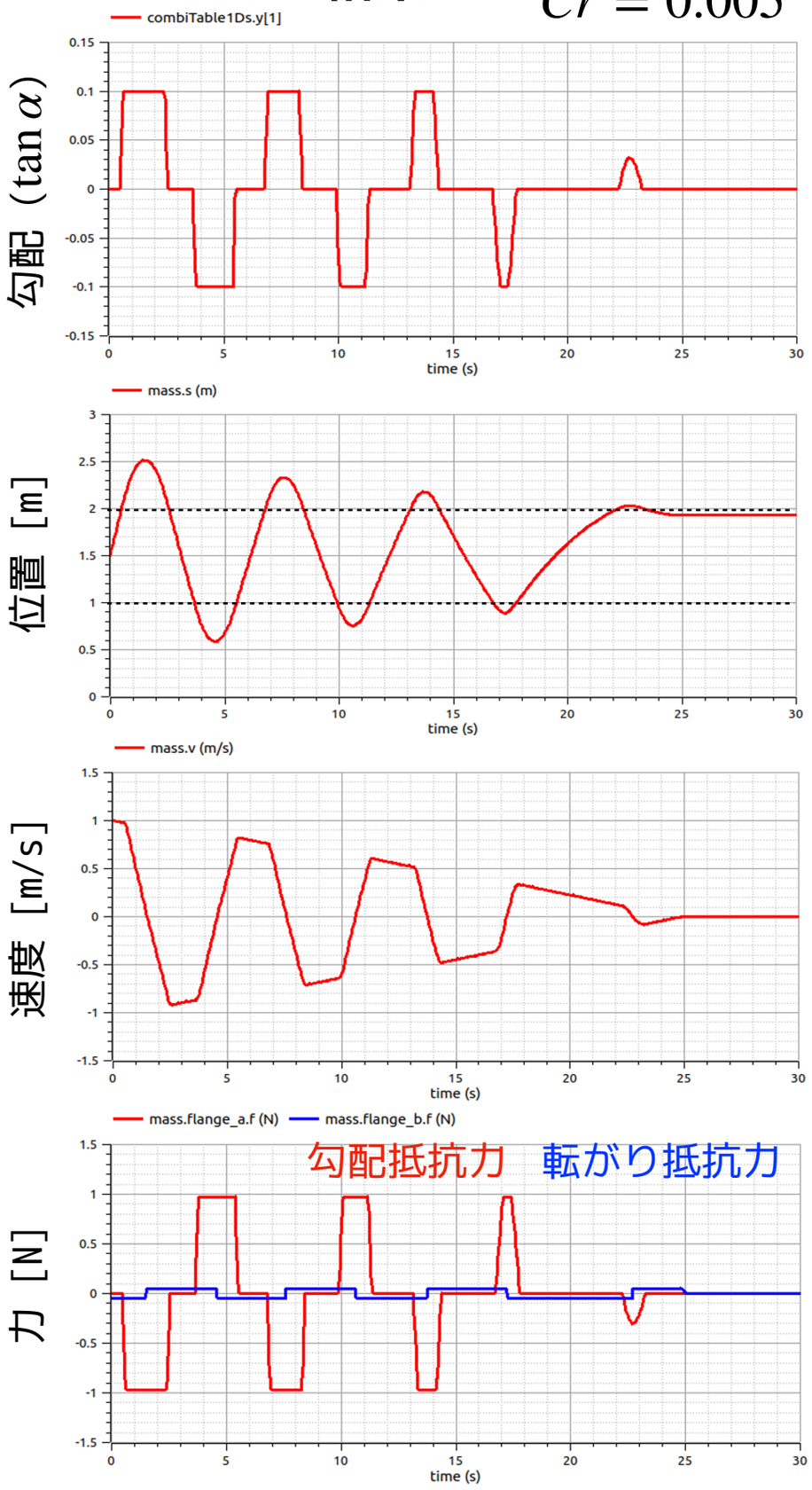
$useInclinationInput$

$CrConstant$ 転がり抵抗係数

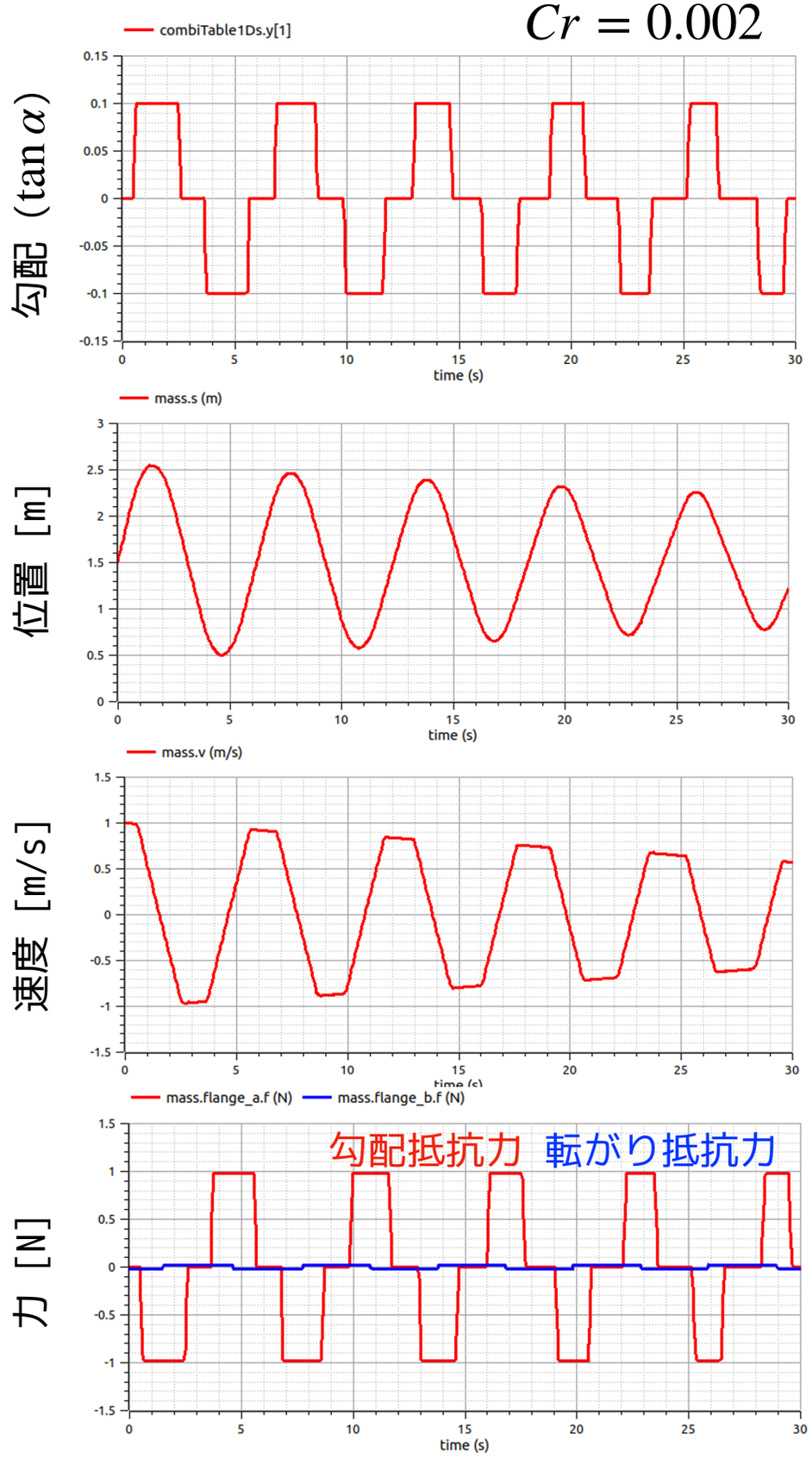
$inclinationConstant$ 勾配

シミュレーション結果

$Cr = 0.005$



$Cr = 0.002$

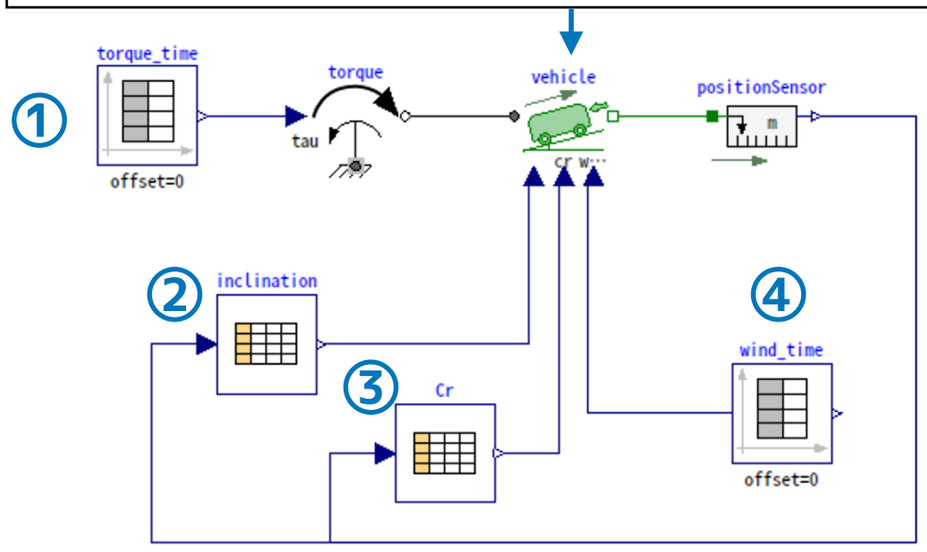


Example17

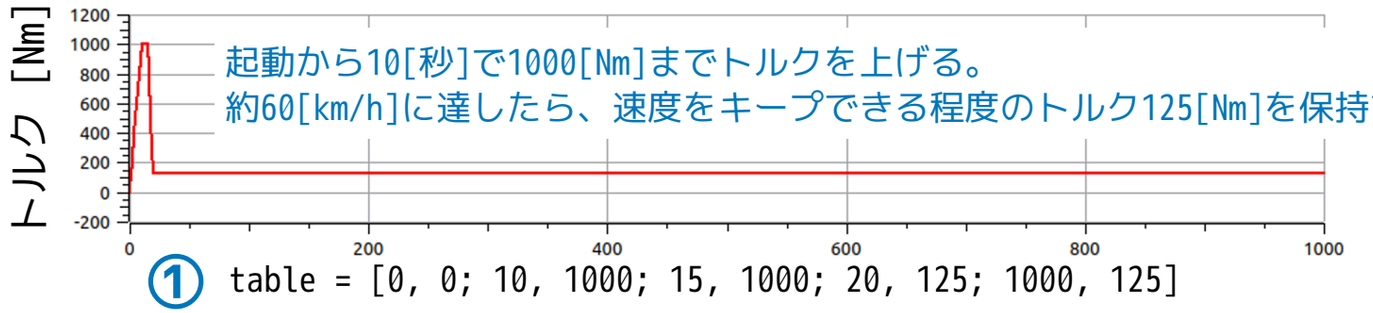
自動車を走らせる。

モデル

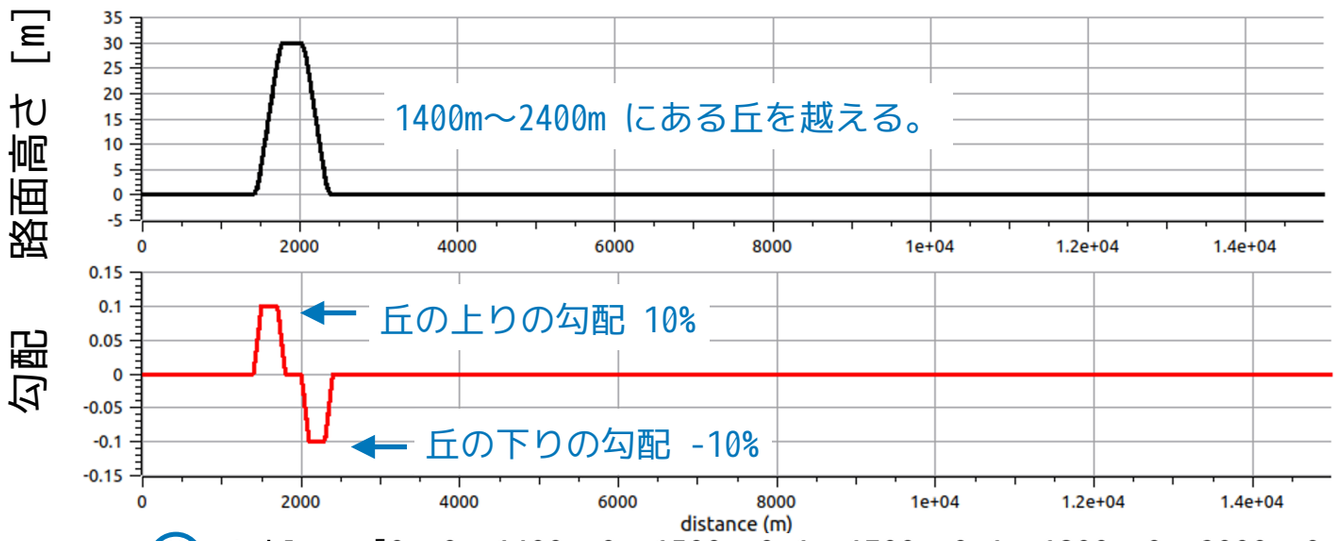
```
m = 1200    [kg]     総重量
J = 108     [kg.m2]  慣性モーメント
R = 0.3     [m]      車輪半径
s.fixed = ture       位置を初期化する
s.start = 0 [km]     初期位置
v.fixed = ture       速度を初期化する
v.start = 0 [km/h]   初期速度
A = 2.8     [m2]     前面投影面積
Cd = 0.35            空気抵抗係数
rho = 2.354 [kg/m3]  空気密度(25℃,1atm)
useWindInput = true  風速を信号入力する
useCrInput = true    転がり抵抗係数を信号入力する
useInclinationInput = ture   勾配を信号入力する
```



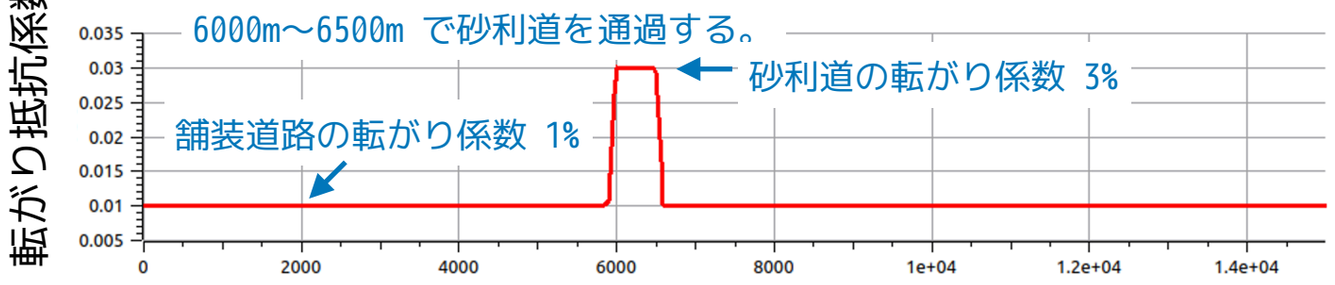
走行条件



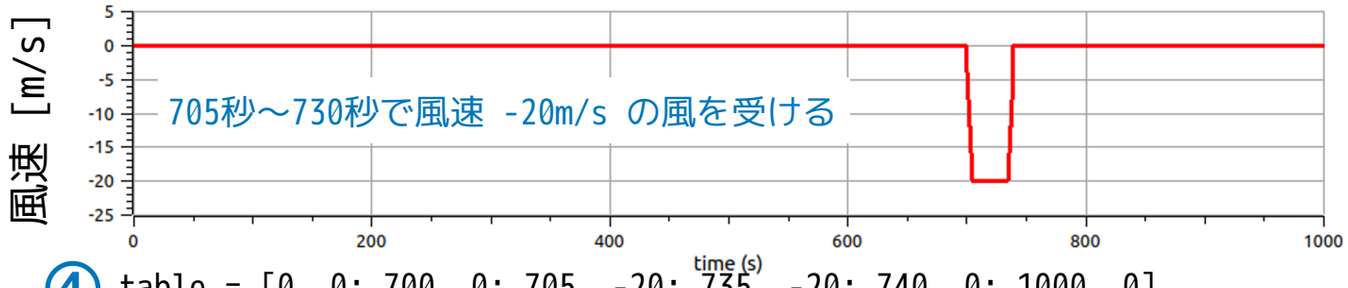
① table = [0, 0; 10, 1000; 15, 1000; 20, 125; 1000, 125]



② table = [0, 0; 1400, 0; 1500, 0.1; 1700, 0.1; 1800, 0; 2000, 0;
2100, -0.1; 2300, -0.1; 2400, 0; 15000, 0]

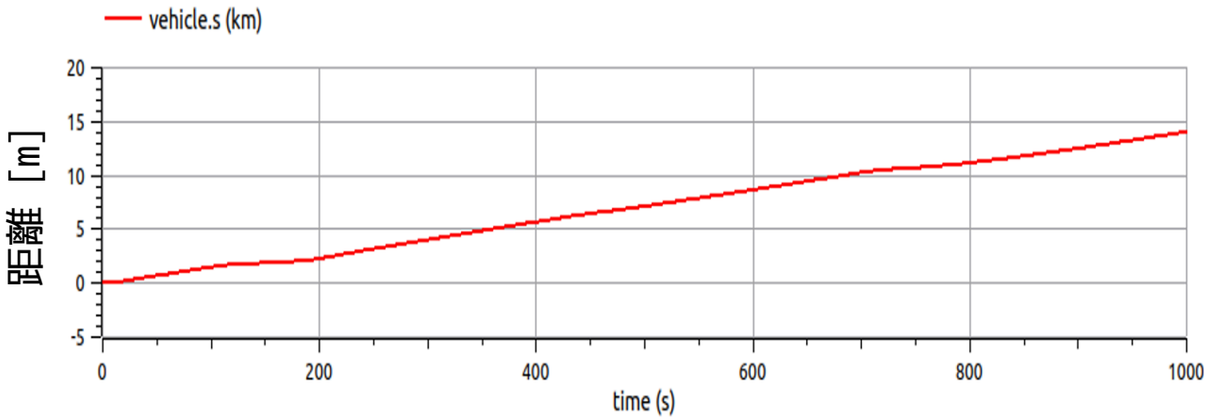


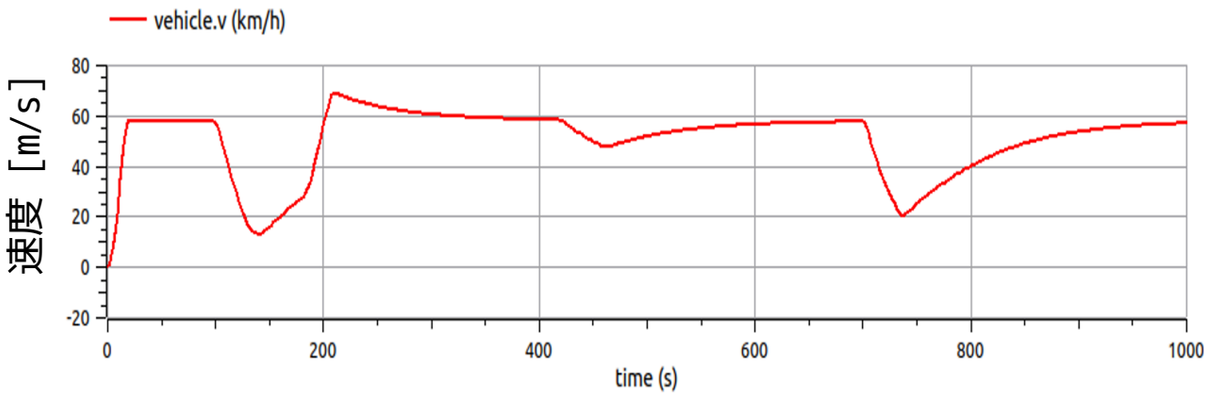
③ table = [0, 0.01; 5900, 0.01; 6000, 0.03; 6500, 0.03; 6600, 0.01; 15000, 0.01]

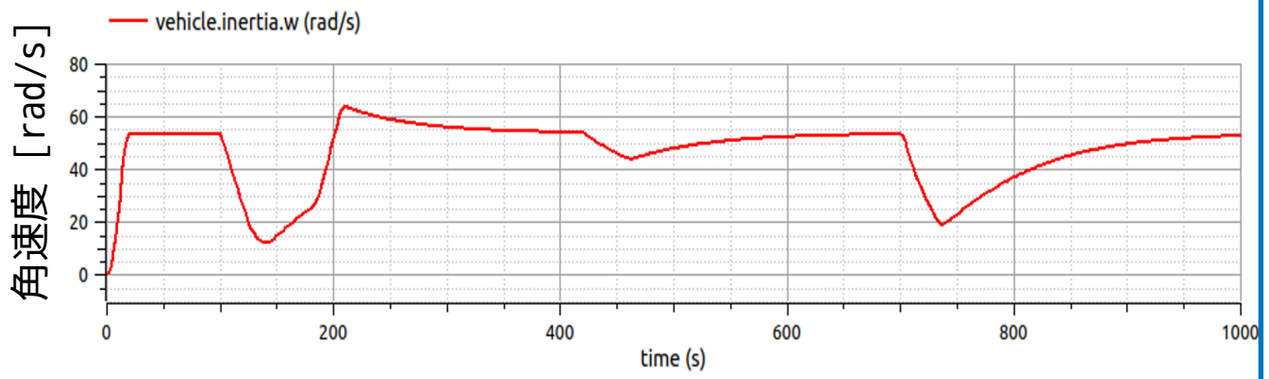


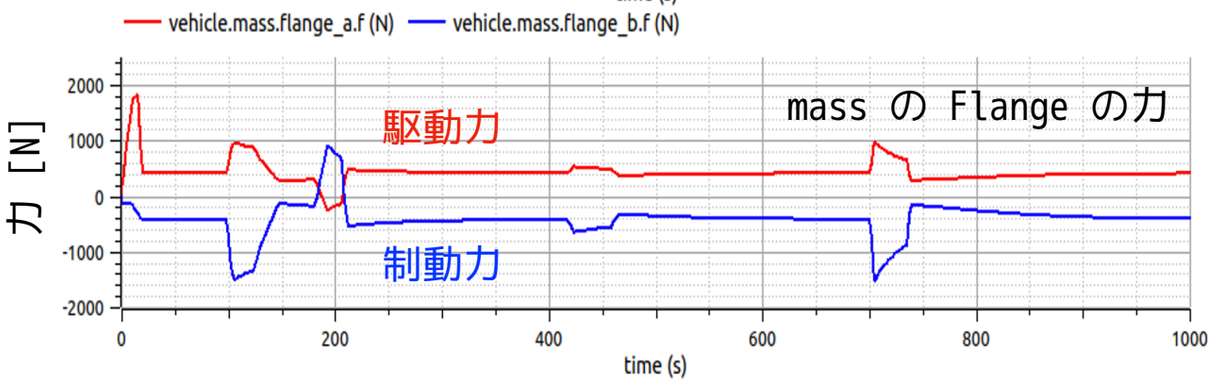
④ table = [0, 0; 700, 0; 705, -20; 735, -20; 740, 0; 1000, 0]

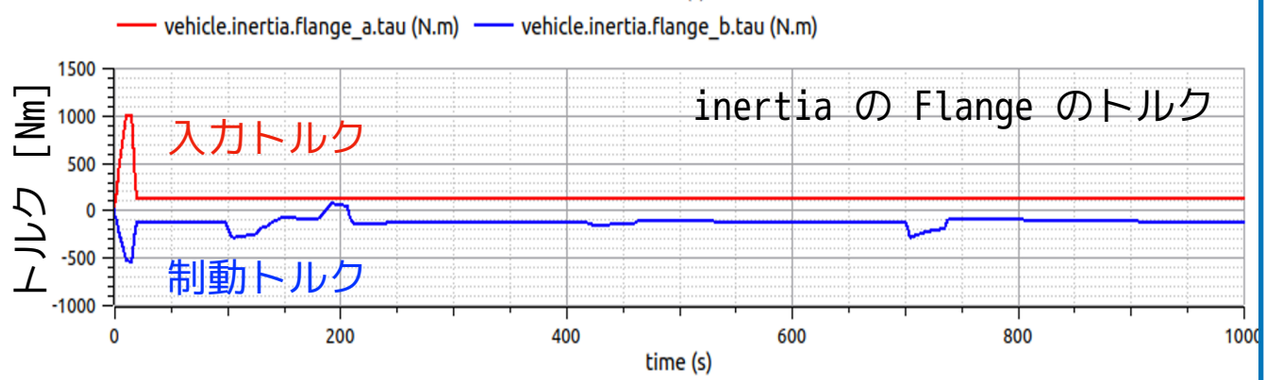
シミュレーション結果

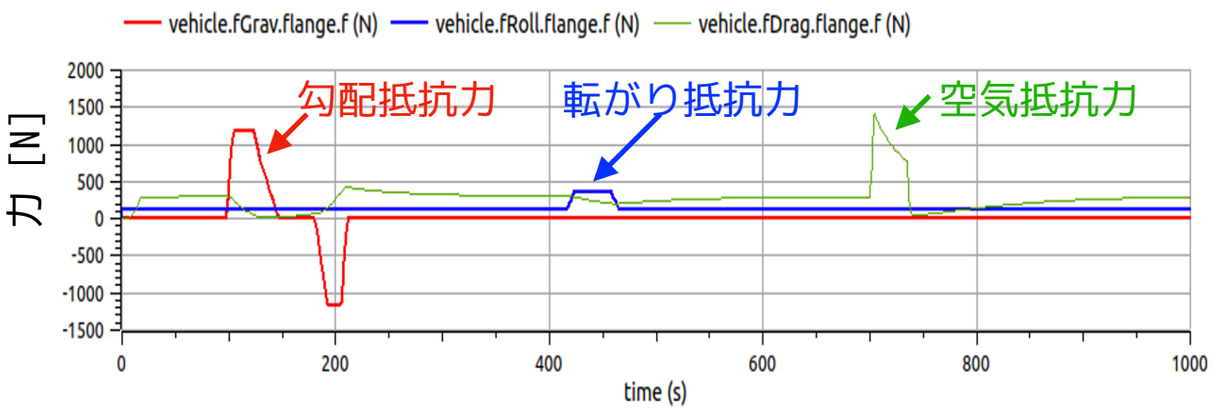

vehicle.s (km)
距離 [m]
time (s)


vehicle.v (km/h)
速度 [m/s]
time (s)

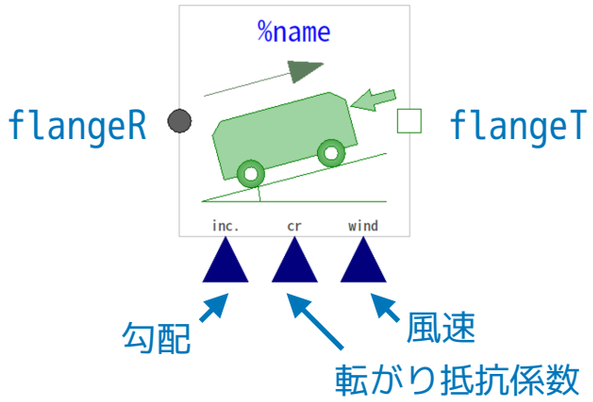

vehicle.inertia.w (rad/s)
角速度 [rad/s]
time (s)

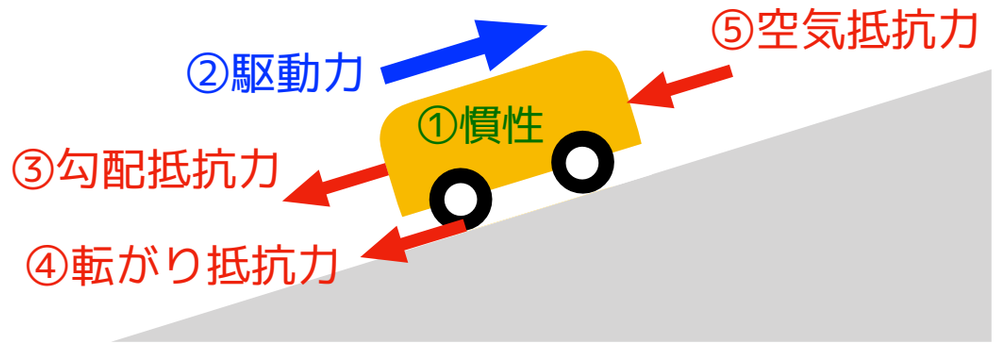

vehicle.mass.flange_a.f (N)
vehicle.mass.flange_b.f (N)
駆動力
制動力
mass の Flange の力
力 [N]
time (s)


vehicle.inertia.flange_a.tau (N.m)
vehicle.inertia.flange_b.tau (N.m)
入力トルク
制動トルク
inertia の Flange のトルク
トルク [Nm]
time (s)

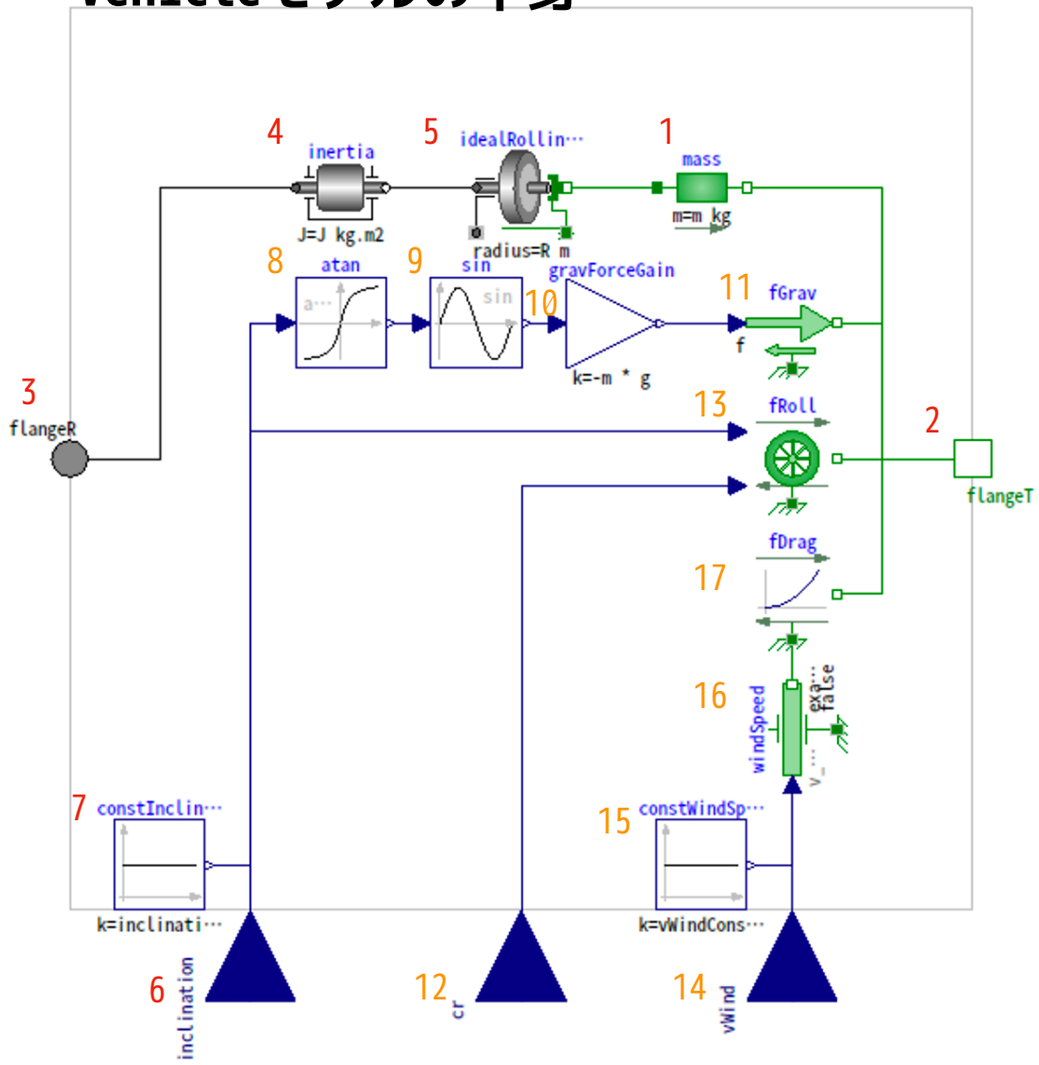

vehicle.fGrav.flange.f (N)
vehicle.fRoll.flange.f (N)
vehicle.fDrag.flange.f (N)
勾配抵抗力
転がり抵抗力
空気抵抗力
力 [N]
time (s)

Vehicle - トルクで駆動する車両の簡易モデル





Vehicleモデルの中身



①車体慣性 Inertia

1	mass: Mass	
2	flangeT: Flange_b	Translational.Interfaces

②駆動力 driving force

3	frangeR: Flange_a	Rotational.Interfaces
4	inertia(J=J): Inertia	Rotational.Components
5	idealRollingWheel: IdealRollingWheel	

③勾配抵抗力 Inclination registance

6	inclination: RealInput	useInclinationInput = true
7	constInclination: Constant	useInclinationInput = false
8	atan: Atan	
9	sin: Sin	
10	gravForceGain:Gain	
11	fGrav: Force	

④転がり抵抗力 Rolling registance

12	cr: RealInput	useCRInput = true
13	fRoll: RollingRegistance	

⑤空気抵抗力 Drag registance

14	vWind: RealInput	useWindInput = true
15	constWindSpeed: Constant	useWindInput = false
16	windSpeed: Speed	
17	fDrag: QuadraticSpeedDependentForce	

Vehicle のパラメータ、変数、定数

<parameter>

①車体慣性

m [kg]	総重量
g = Constants.g_n [m/s2]	重力加速度

②駆動力

J [kg.m2]	車輪の慣性モーメント
R [m]	車輪半径

⑤空気抵抗力

A [m2]	車両前面投影面積
Cd	空気抵抗係数
rho [kg/m3]	空気密度
useWindInput	true で風速を信号入力する
vWindConstant [m/s]	風速

④転がり抵抗力

useCrInput	true で転がり抵抗係数を信号入力する
CrConstant	転がり抵抗係数
vReg [m/s]	v < vReg でregularizationを行う

③勾配抵抗力

useInclinationInput	true で勾配を信号入力する
inclinationConstant	勾配（$\tan\alpha$）

<variable>

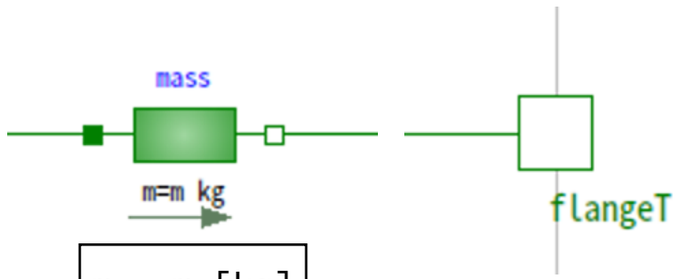
s = mass.s [m]	位置
v = mass.v [m/s]	速度
a = mass.a [m/s2]	加速度

<constant>

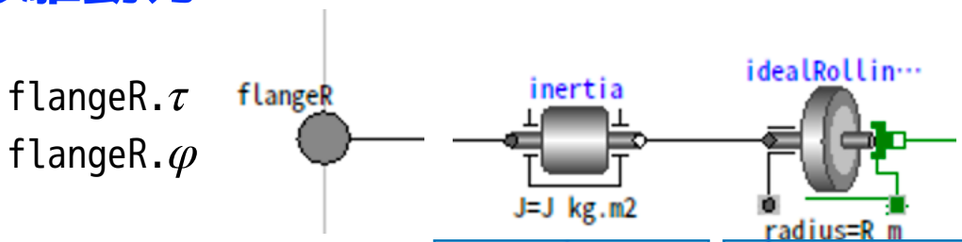
- vRef = 1 [m/s]	風速の基準値

モデル内容を調べます。

①車体慣性



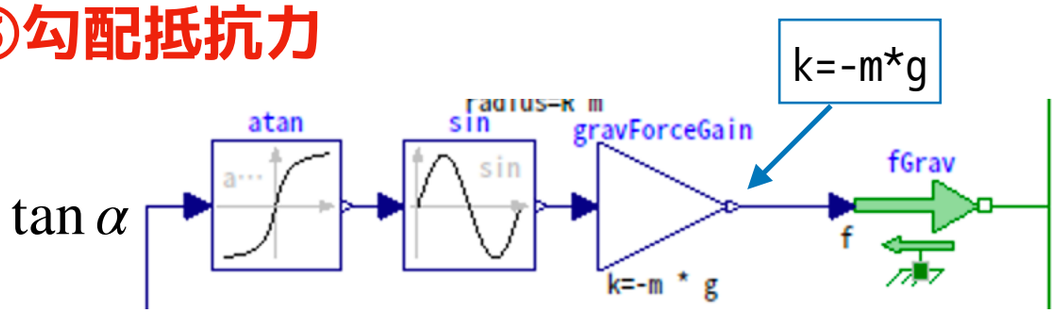
②駆動力



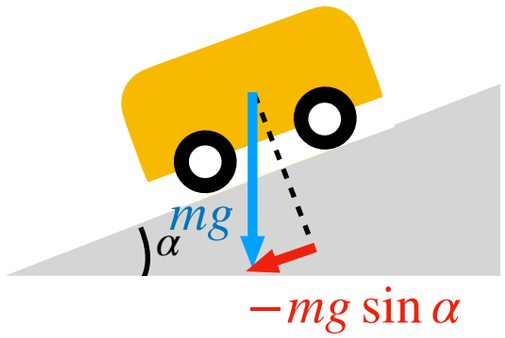
$$\Rightarrow \tau = frangeR.\tau - J\frac{d^2}{dt^2}flangeR.\varphi$$

$$f = \frac{\tau}{R}$$

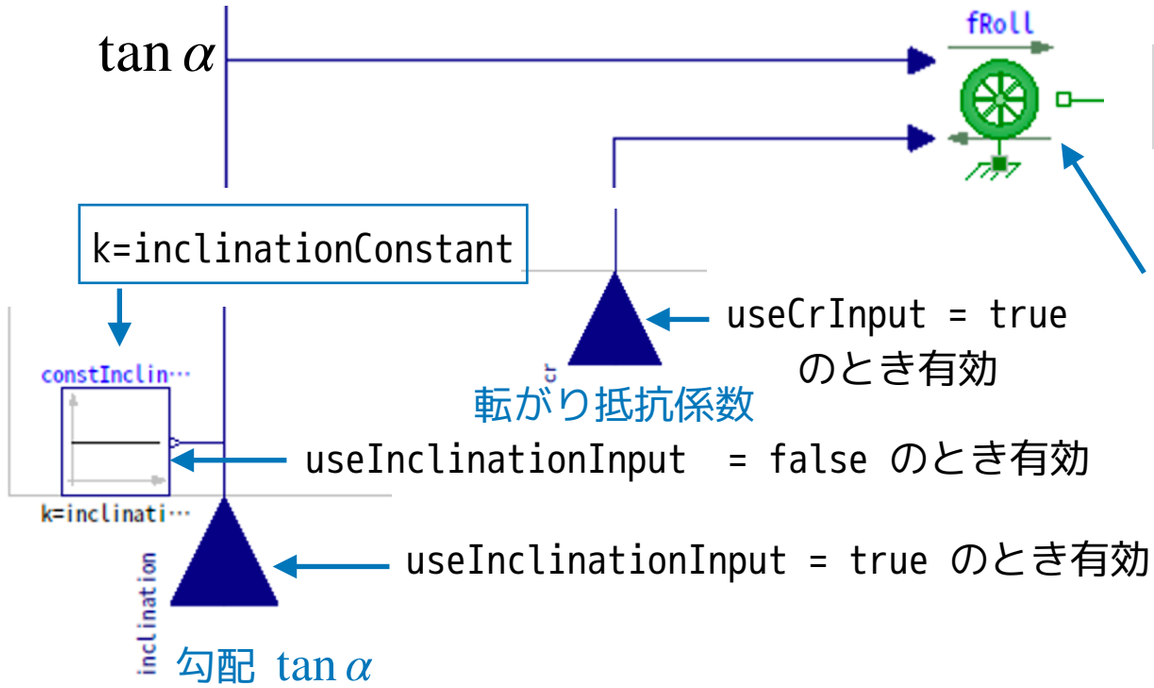
③勾配抵抗力



$$\Rightarrow f_{Grav} = -mg\sin\alpha$$



④転がり抵抗力



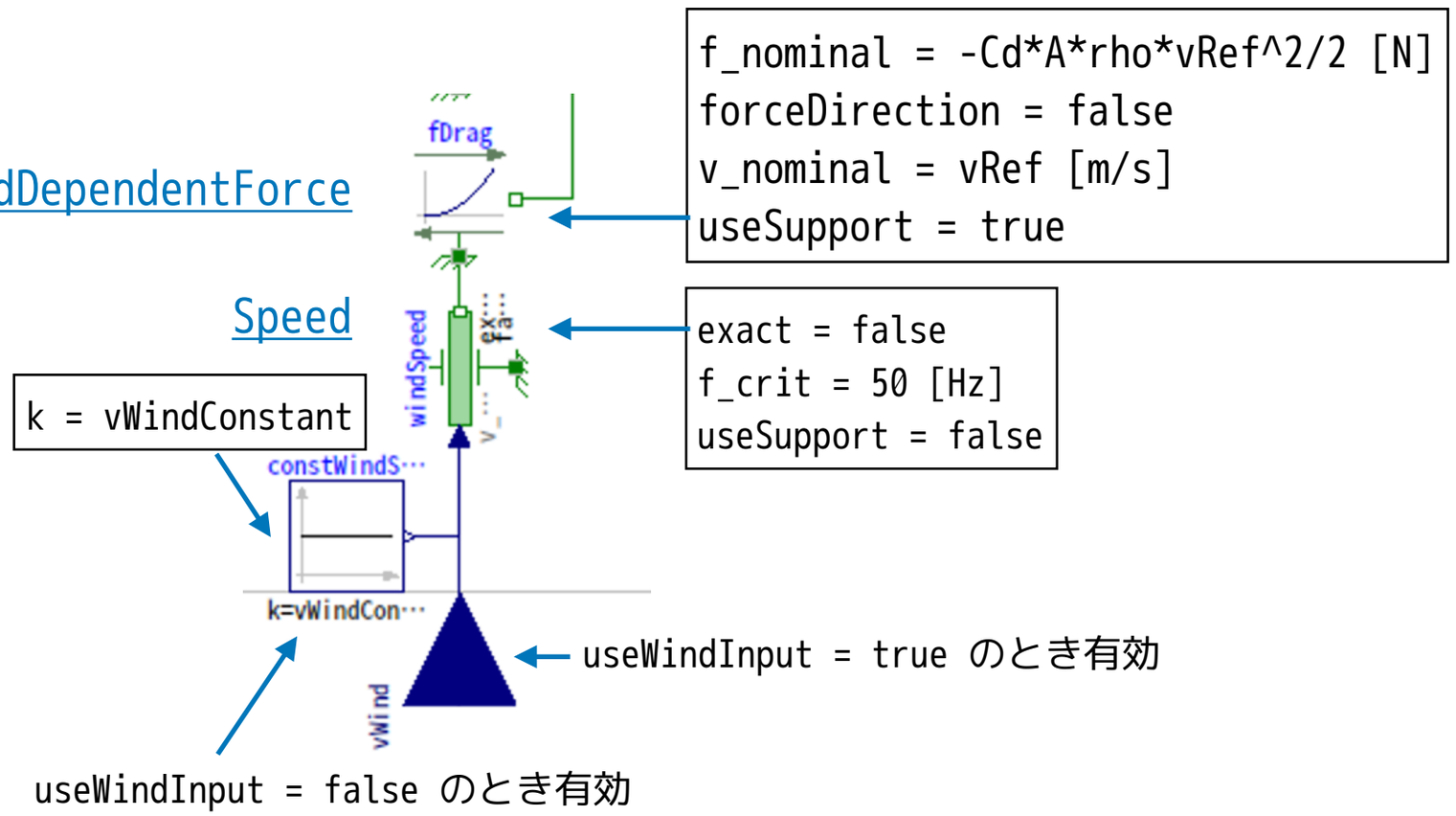
$$f_{Roll} = Cr \cdot F_{Weight} \cdot \cos\alpha$$

但し、力の向きは速度 v と逆になる。
$|v| < v_{Reg}$のときregularizationを行う。

```
fweight = m*g [N]
useCrInput = useCrInput
CrConstant = CrConstant
usecInclinationInput = useInclinationInput
inclinationConstant = inclinationConstant
reg = Linear
v0 = vReg
```

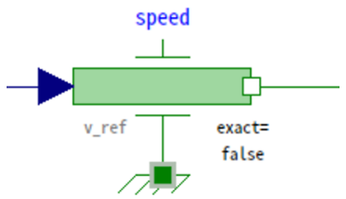
⑤空気抵抗力

$$f_{Drag} = C_D \rho A (v - v_{wind})^2$$

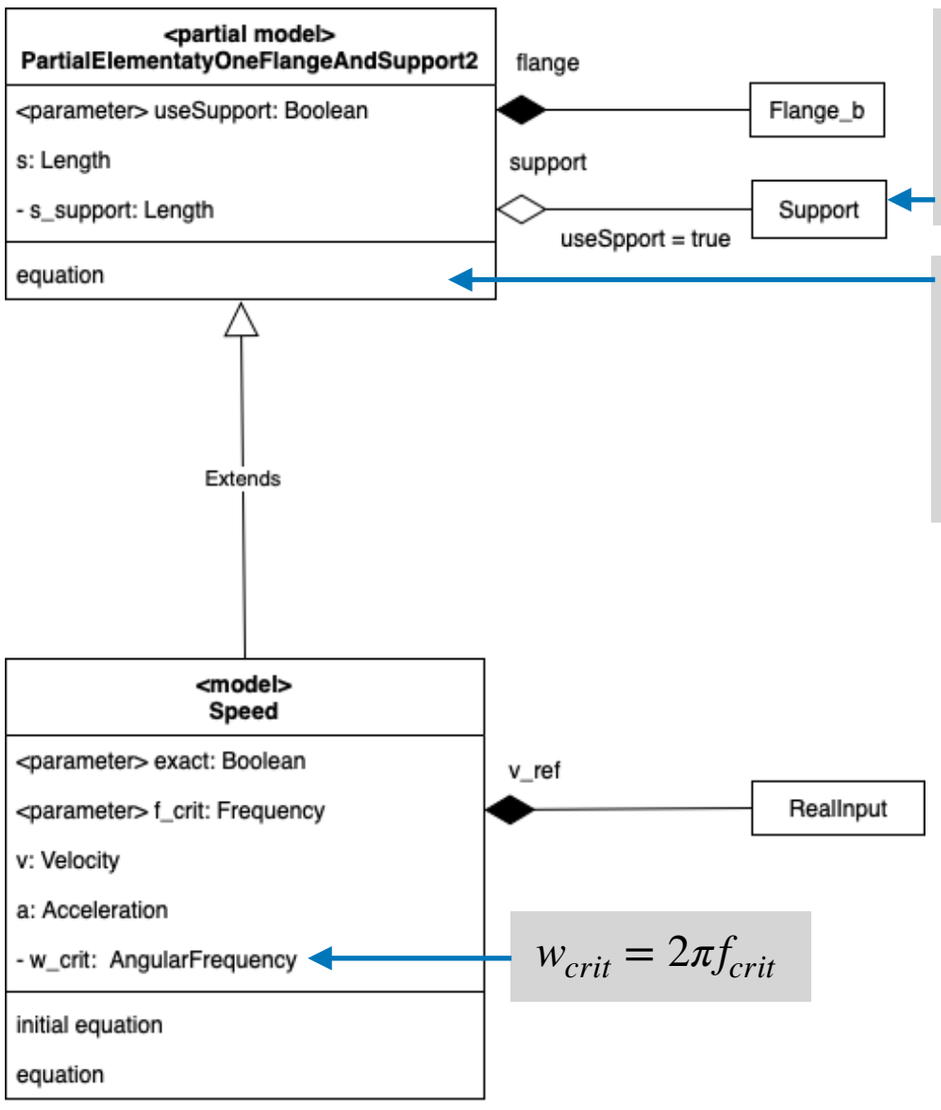


Flange の速度を規定する Speed コンポーネントと、
速度の自乗に比例する力を出力する QuadraticSpeedDependentForce
が使用されています。これらの中身も調べます。

speed — 速度を設定するモデル



- 実数入力信号 v_{ref} でflangeの速度を設定するモデル。
- v_{ref}が解析的関数で微分可能な場合は、exact = true する。flange の速度がそのまま v_{ref} となる。
- exact = false とした場合は、v_{ref} に一時遅れフィルターを適用し、v_{ref}とvの差に比例する加速度を与えてflangeの速度が v_{ref} に近づくようにする。



if useSupport

$$support.s = s_{support}$$

$$supoort.f = -flange.f$$

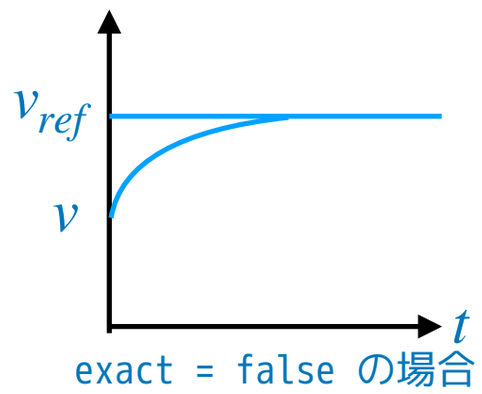
$$s = flange.s - s_{support}$$

if not useSupport

$$s_{support} = 0$$

end if

$$w_{crit} = 2\pi f_{crit}$$



exact = false の場合

initial equation

if not exact then

$$v = v_{ref}$$

end if

equation

$$v = \frac{ds}{dt}$$

if exact then

$$v = v_{ref}$$

$$a = 0$$

else

$$a = \frac{dv}{dt}$$

$$a = (v_{ref} - v) \cdot w_{crit}$$

end if

StateSelect 状態変数選択

Modelicaのシミュレーション環境は、どの変数を状態変数として微分代数方程式系を構成するか自動的に判断します。
Real型変数の stateSelect パラメータは、ソルバーがその変数を状態変数として扱うプライオリティを制御します。

プライオリティ（低 ▲ 高 ▼）

```
type StateSelect = enumeration(
  never "Do not use as state at all.” ,
  avoid "Use as state, if it cannot be avoided (but only if variable appears
         differentiated and no other potential state with attribute default,
         prefer, or always can be selected).” ,
  default "Use as state if appropriate, but only if variable appearsdifferentiated.” ,
  prefer "Prefer it as state over those having the default value
         (also variables can be selected, which do not appear differentiated). ",
  always "Do use it as a state.”
);
```

Modelica Lanuage Specification 3.5, 4.8.7.1, p.57 https://modelica.org/documents/MLS.pdf

変数 s や v の stateSelectパラメータの設定

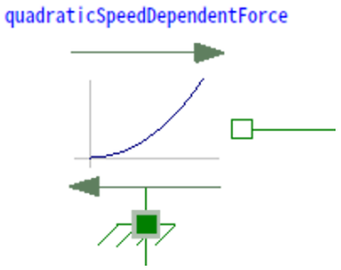
PartialElementaryOneFlangeAndSupport2 を継承する部分

```
  extends
    Modelica.Mechanics.Translational.Interfaces.PartialElementaryOneFlangeAndSupport2(
     s(
      start=0,
      fixed=true,
      stateSelect=StateSelect.prefer));
```

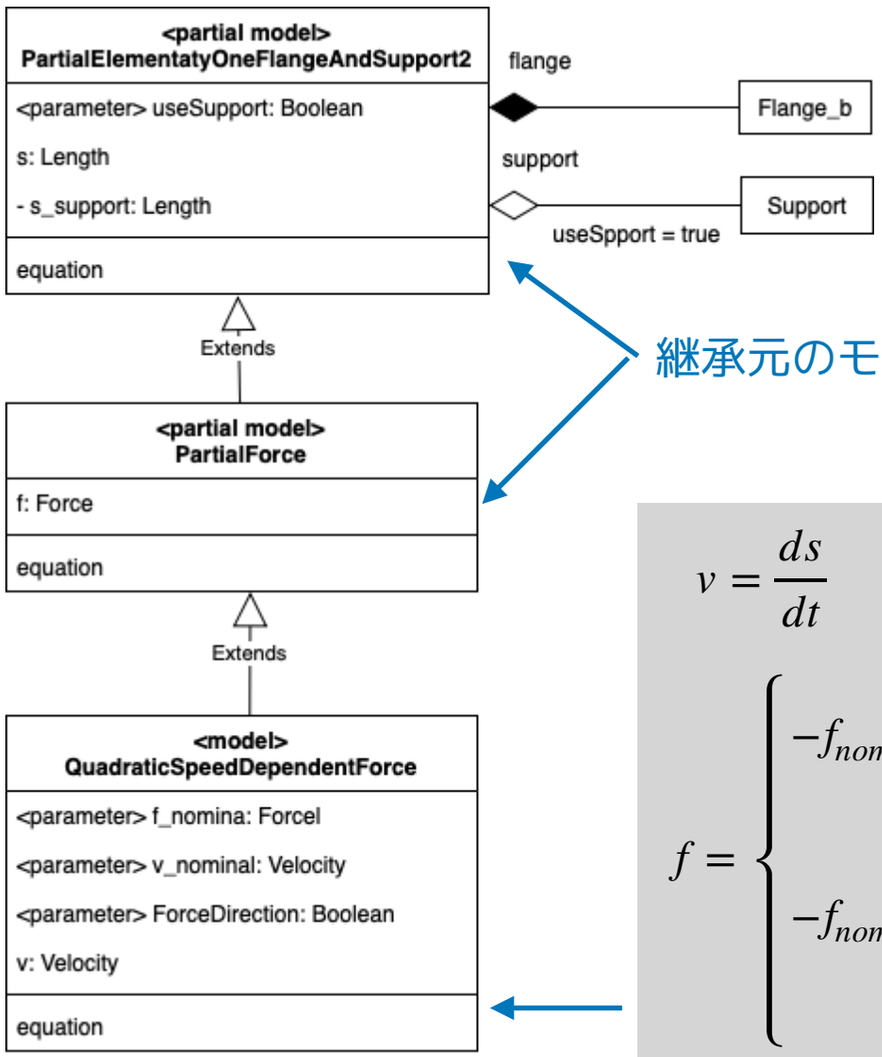
速度 v を宣言する部分

```
  SI.Velocity v(stateSelect=if exact then StateSelect.default else
        StateSelect.prefer) "Absolute velocity of flange";
```

QuadraticSpeedDependentForce — 速度の自乗に比例した力



- 速度 v の自乗に比例した力のモデル
- 速度 v は useSupport = false のときは flange の絶対速度、useSupport = true のときは flange と support の速度差となる。
- 速度 $v_{nominal}$ のときの力 $f_{nominal}$ を設定する。



パラメータ

$v_{nominal}$ [m/s] 基準となる速度

$f_{nominal}$ [N] $v = v_{nominal}$ のときの力

$ForceDirection$ true のとき力の向きが速度に依存しなくなる

継承元のモデルはRollingResistanceと同様

$$v = \frac{ds}{dt}$$

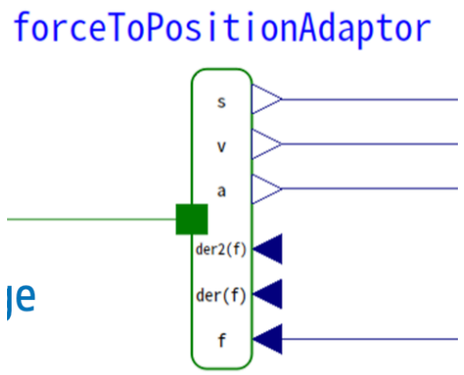
$$f = \begin{cases} -f_{nominal}\left(\frac{v}{v_{nominal}}\right)^2, & ForceDirection = true \\ -f_{nominal} \cdot smooth_1 \begin{cases} \left(\frac{v}{v_{nominal}}\right)^2, & v \geq 0 \\ -\left(\frac{v}{v_{nominal}}\right)^2, & else \end{cases}, & ForceDirection = false \end{cases}$$

その他のコンポーネント

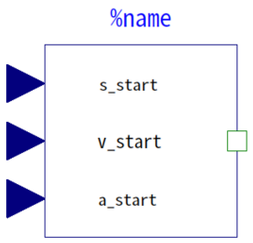
その他のコンポーネント
RelativeStates 相対位置、相対速度を状態変数にするモデル
InitializeFlange 入力信号でFlangeを初期化するモデル
GeneralForceToPositionAdaptor Flangeを入力出力信号に変換する(1)
GeneralPositionToForceAdaptor Flangeを入力出力信号に変換する(2)

RelativeStates

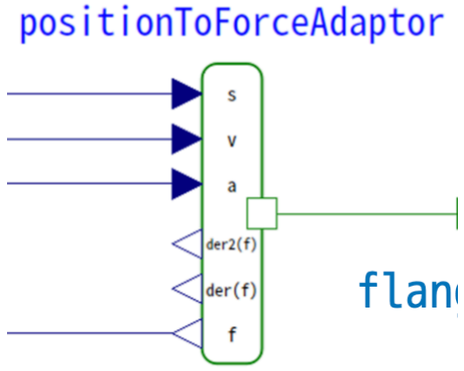
GeneralForceToPositionAdaptor



InitializeFlange

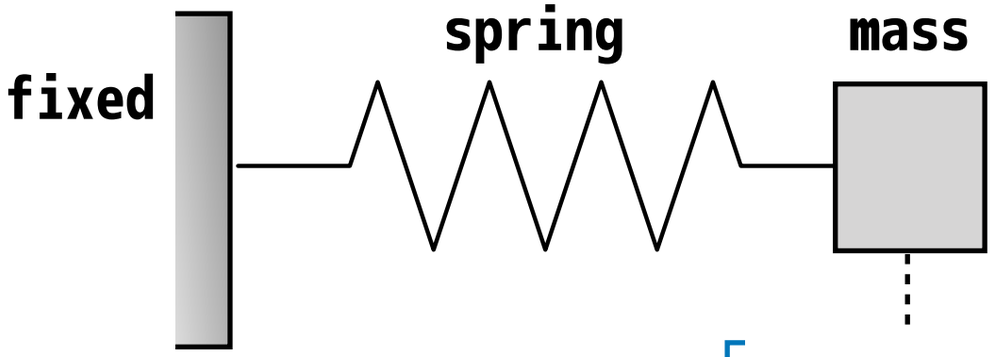


GeneralPositionToForceAdaptor

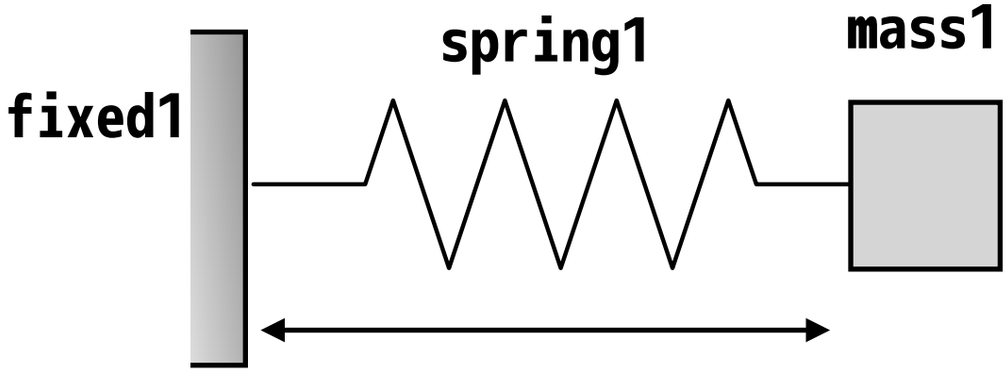


Example18 ばねの長さや伸びる速さを状態変数にする。

概要



状態変数
- s [m] 絶対位置
- v [m/s] 絶対速度



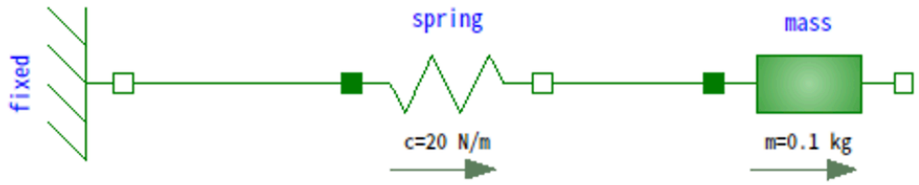
状態変数
- s_{rel} [m] 相対位置(ばねの長さ)
- v [m/s] 相対速度(ばねが伸びる速さ)

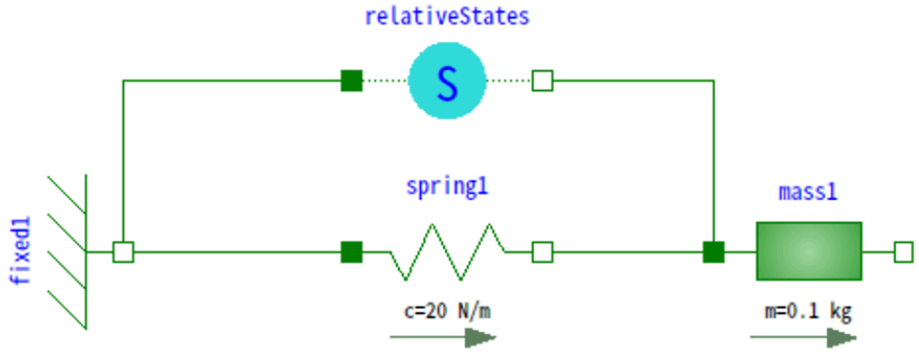
固定位置
0 [m]

ばね
ばね定数 20 [N/m]
自然長 3 [m]
初期長さ 3[m]

物体
質量 0.1[kg]
長さ 0.1 [m]
初期速度 2 [m/s]

モデル





fixed, fixed1

```
s0 = 0 [m]
```

spring, spring1

```
c = 20 [N/m]
s_rel0 = 3 [m]
s_rel.fixed = true
s_rel.start = 3 [m]
```

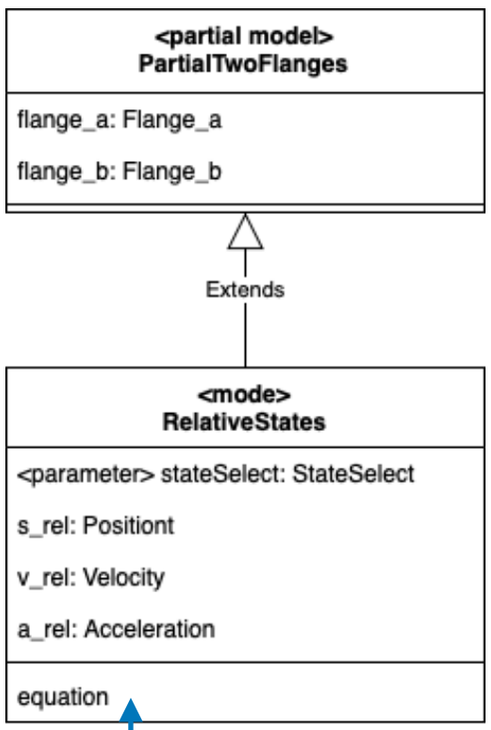
mass, mass1

```
m = 0.1
L = 0.1
v.fixed = true
v_start = 2 [m/s]
```

初期長さは、
relativeStates.s_rel
で設定することもできます。

RelativeStates — 相対位置、相対速度を状態変数にするモデル

慣性質量 Mass などを使ったモデルでは、通常、質量のある物体の絶対位置 s や絶対速度 v が微分代数方程式の状態変数となる。RelativeStatesを接続すると、これらの変数の代わりにFlange間の相対位置 s_{rel} と相対速度 v_{rel} を状態変数にすることができる。



$$s_{rel} = flange\_b.s - flange\_a.s$$
$$v_{rel} = \frac{ds_{rel}}{dt}$$
$$a_{rel} = \frac{dv_{rel}}{dt}$$
$$flange\_a.f = 0$$
$$flange\_b.f = 0$$

Mass のパラメータと変数の宣言部分

```
model Mass "Sliding mass with inertia"
  parameter SI.Mass m(min=0, start=1) "Mass of the sliding mass";
  parameter StateSelect stateSelect=StateSelect.default
    "Priority to use s and v as states" annotation (Dialog(tab="Advanced"));
  extends Translational.Interfaces.PartialRigid(L=0,s(start=0, stateSelect=
          stateSelect));
  SI.Velocity v(start=0, stateSelect=stateSelect)
    "Absolute velocity of component";
  SI.Acceleration a(start=0) "Absolute acceleration of component";
```

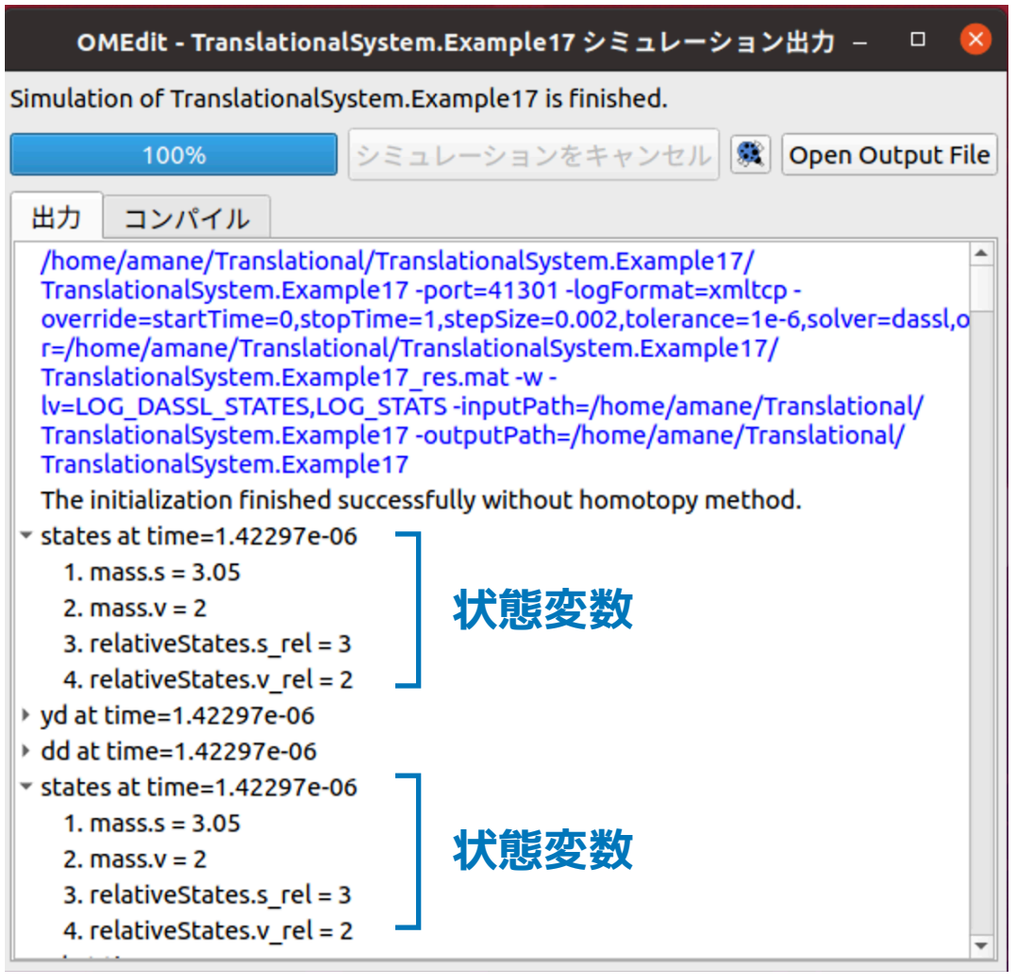
RelativeStates のパラメータと変数の宣言部分

```
  parameter StateSelect stateSelect=StateSelect.prefer
    "Priority to use the relative angle and relative speed as states";
  SI.Position s_rel(start=0, stateSelect=StateSelect.prefer)
    "Relative position used as state variable";
  SI.Velocity v_rel(start=0, stateSelect=StateSelect.prefer)
    "Relative velocity used as state variable";
  SI.Acceleration a_rel(start=0) "Relative angular acceleration";
```

このコンポーネントを接続すると、Modelica のソルバーは、**stateSelect** パラメータのプライオリティが高い s_{rel} や v_{rel} を状態変数として選択する。

確認方法

シミュレーション＞シミュレーションのセットアップ
シミュレーションフラグでLOG_DASSL_STATESをチェック



massでは、mass.s と mass.v が状態変数になっています。
mass1では、relativestates.s_rel, relativestates.v_rel が状態変数になっています。

Example19 入力信号でFlangeの初期条件を設定する。

モデル

初期位置 **const**

k = -3.2 m

初期速度 **const1**

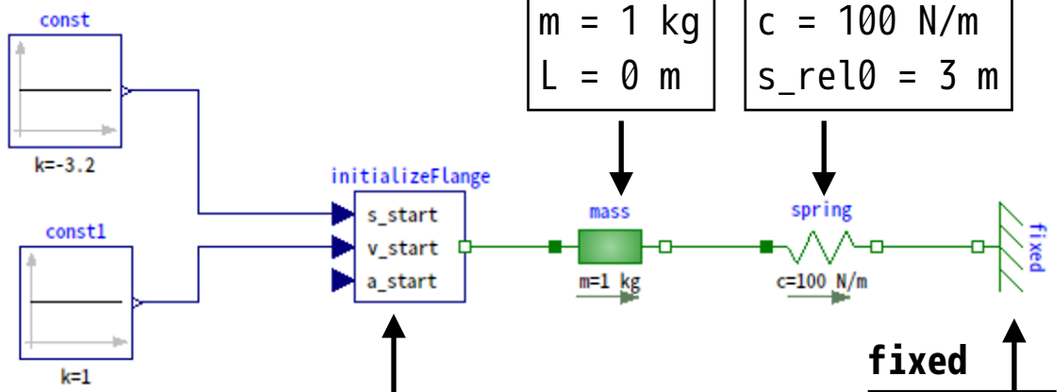
k = 1 m/s

mass

m = 1 kg
L = 0 m

spring

c = 100 N/m
s_rel0 = 3 m



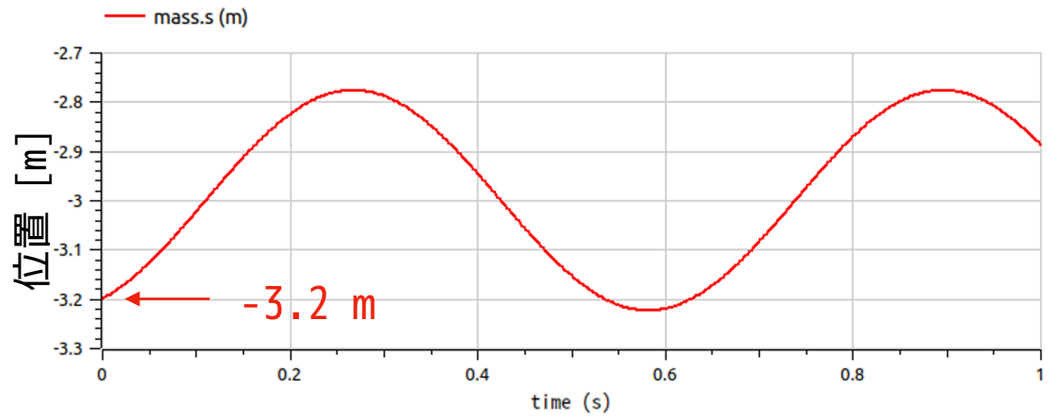
fixed

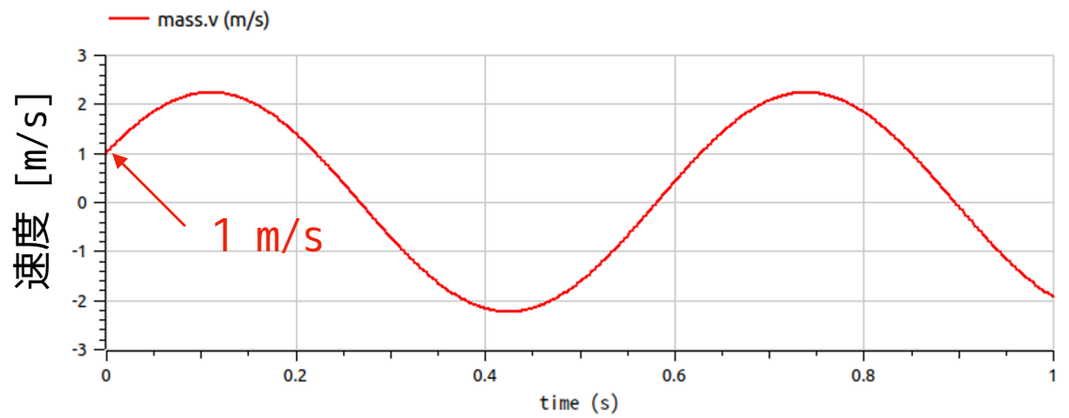
s0 = 0 m

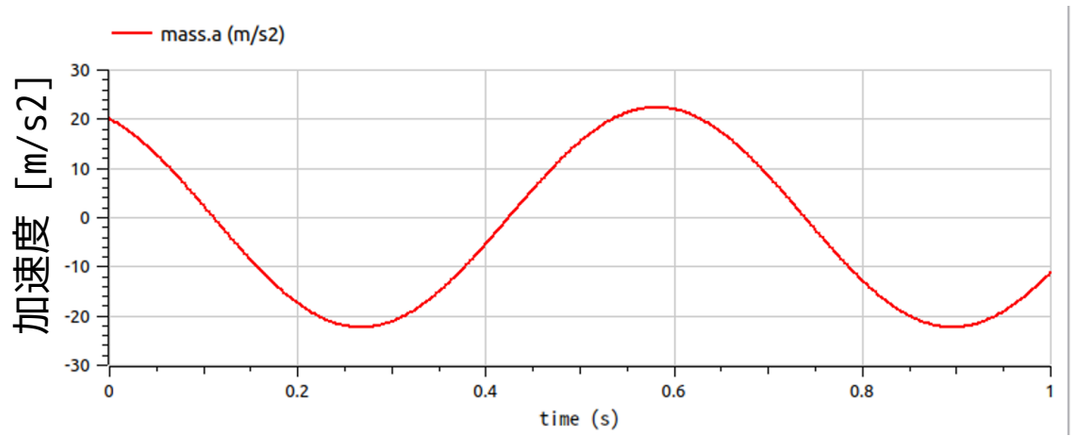
initializeFlange

```
use_s_start = true
use_v_start = true
use_a_start = false
stateSelect = StateSelect.default
```

シミュレーション結果

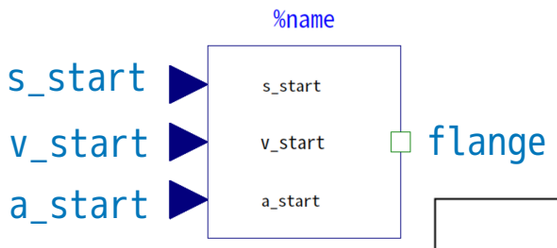




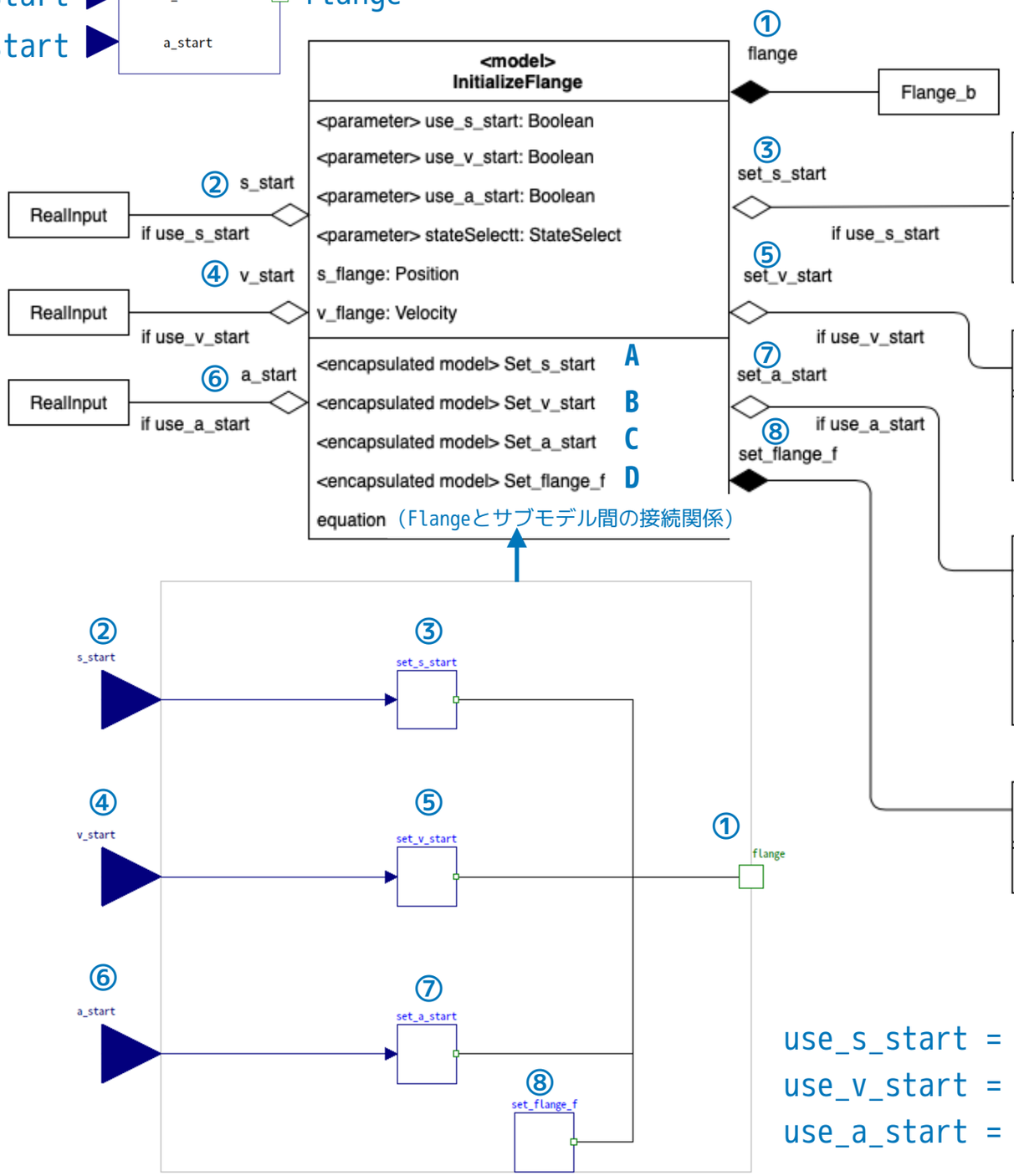


入力信号で設定した初期値になっています。

InitializeFlange — 実数入力信号でFlangeを初期化するモデル



接続されたFlangeの位置、速度、加速度の初期値を、実数入力信号 s_start, v_start, a_start で設定できるモデル。



initial equation セクションによる初期化

initial equation

$$flange.s = s\_start$$

equation

$$flange.f = 0$$

initial equation

$$\frac{d\ flange.s}{dt} = v\_start$$

equation

$$flange.f = 0$$

initial equation

$$\frac{dv}{dt} = a\_start$$

equation

$$flange.f = 0$$

equation

$$flange.f = 0$$

use_s_start = true で位置を初期化する。
use_v_start = ture で速度を初期化する。
use_a_start = ture で加速度を初期化する。

initial equation による初期化

Modelica.Mechanics.Translational.Components.InitializeFlange の一部
カプセル化(encapsulated)されたローカルモデル Set_s_start の定義部分

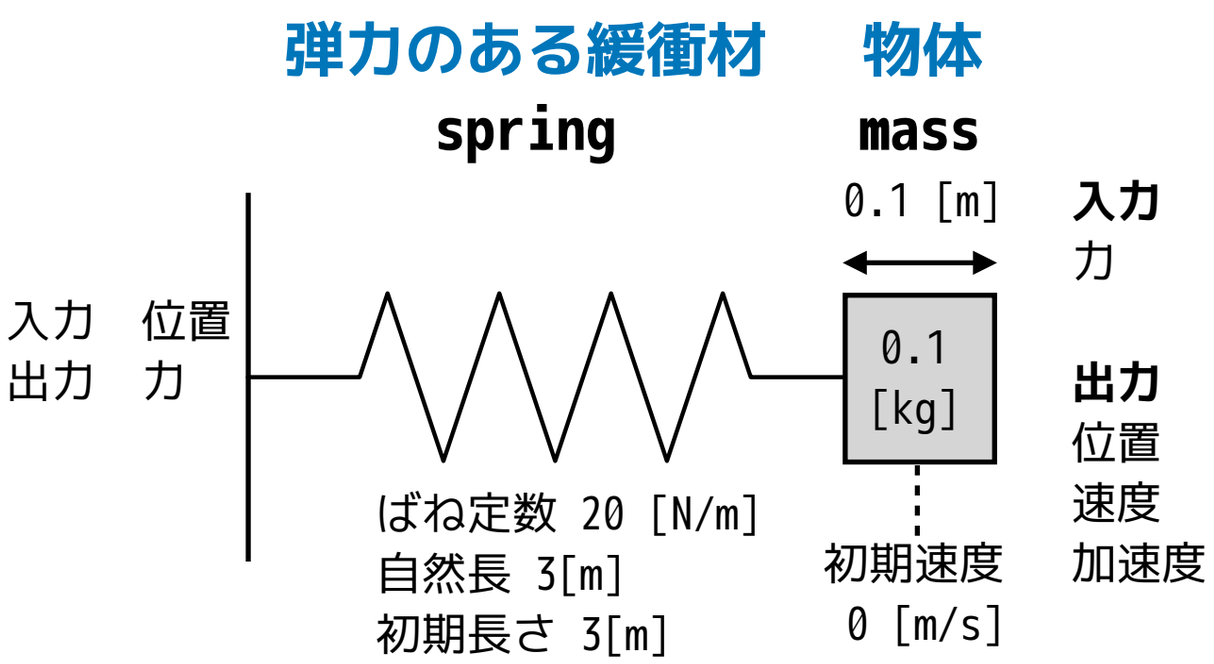
```
encapsulated model Set_s_start "Set s_start"
  import Modelica;
  extends Modelica.Blocks.Icons.Block;
  Modelica.Blocks.Interfaces.RealInput s_start(unit="m") "Start position"
    annotation (HideResult=true, Placement(transformation(extent={{-140,-20},
            {-100,20}})));

  Modelica.Mechanics.Translational.Interfaces.Flange_b flange annotation (
     Placement(transformation(extent={{90,-10},{110,10}})));
initial equation
  flange.s = s_start;
equation
  flange.f = 0;

end Set_s_start;
```

initial equation セクションで変数を初期化することができます。

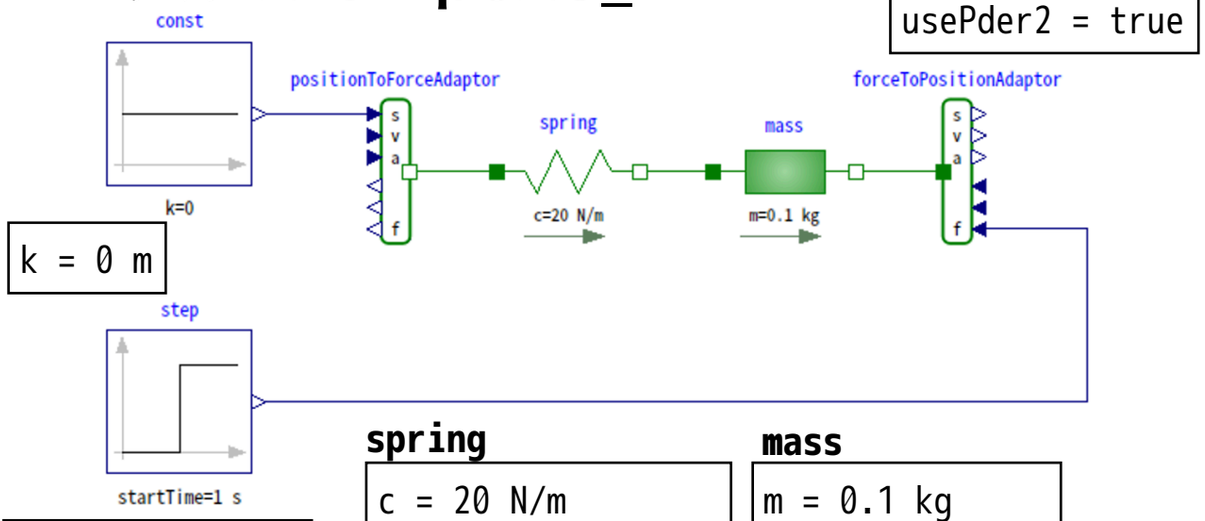
equation セクション

Example20 入出力ブロック系モデルに変換する。

概要



モデル Exampe20_1



usePder = true
usePder2 = true

k = 0 m

height = 1
offset = 0
startTime = 1 s

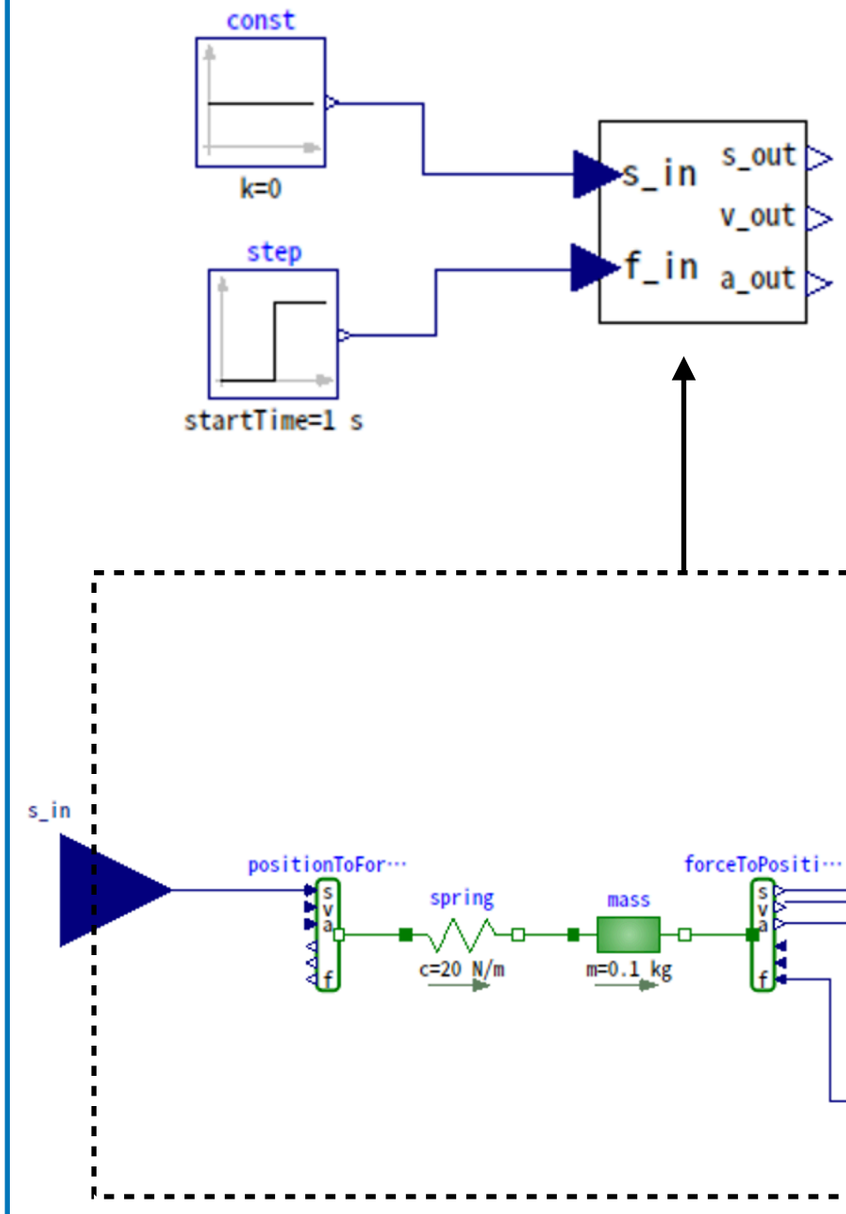
spring

c = 20 N/m
s_rel0 = 3 m
s_rel.fixed = true
s_rel.start = 3 m

mass

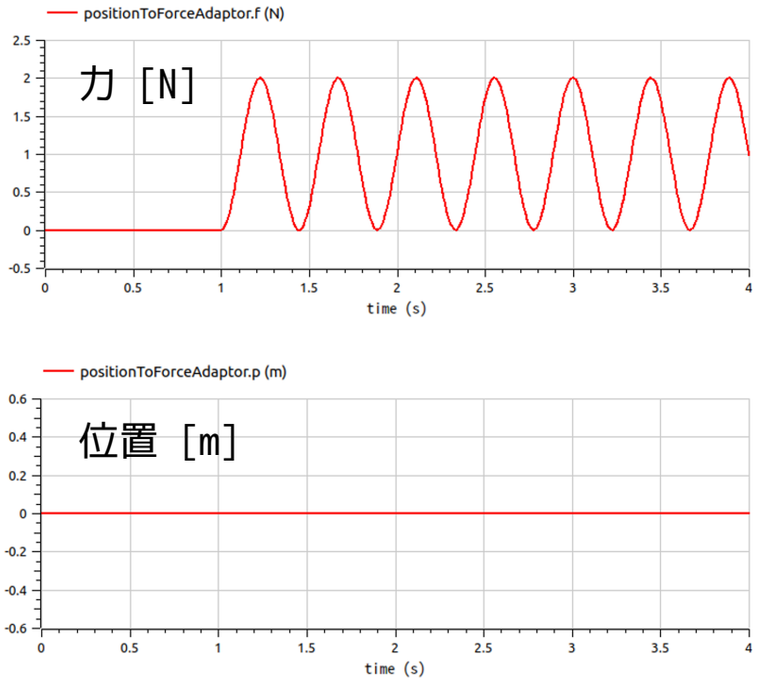
m = 0.1 kg
L = 0.1 m
v.fixed = true
v.start = 0 m/s

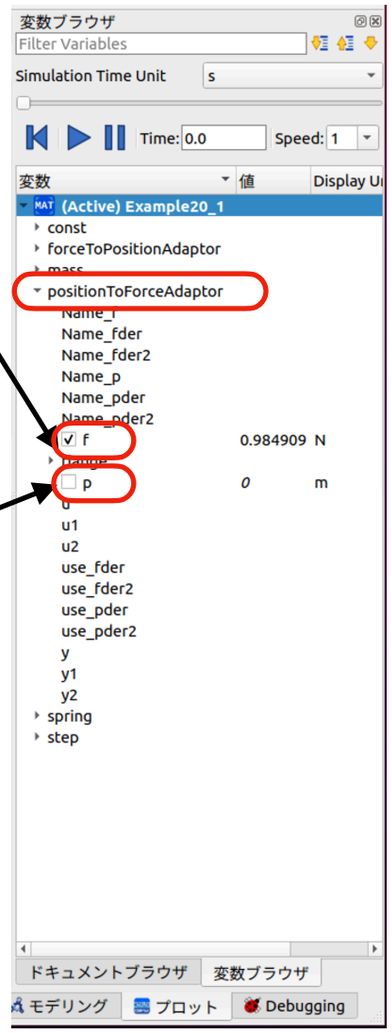
モデル Exampe20_2

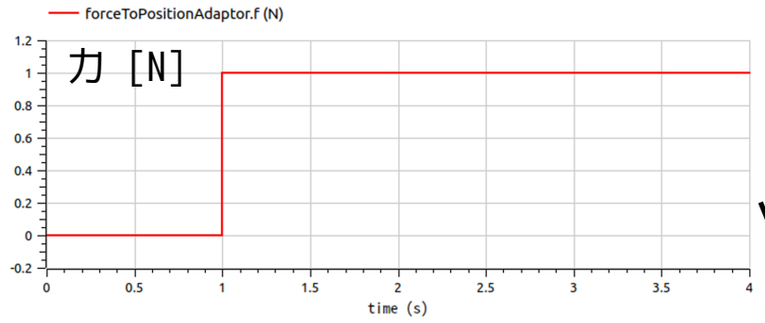


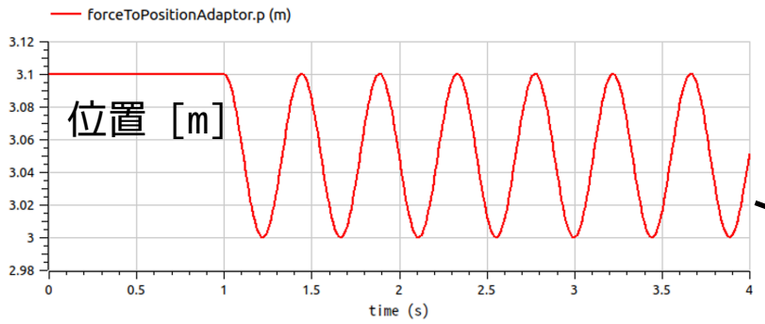
コンポーネント化

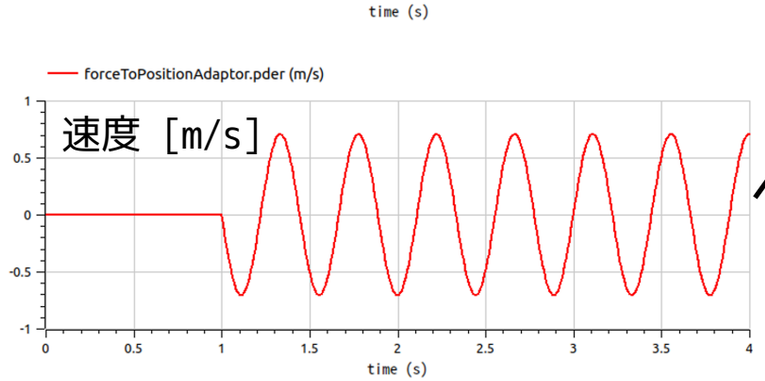
シミュレーション結果

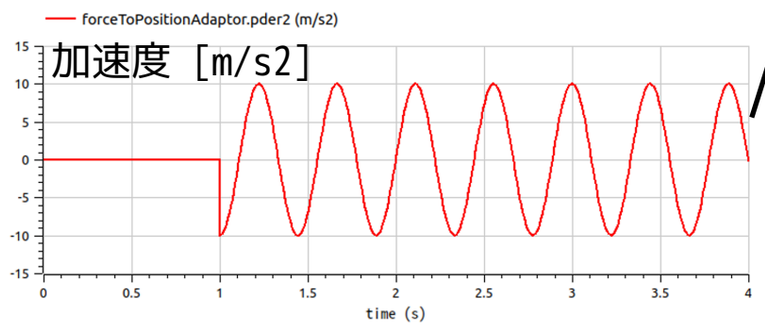


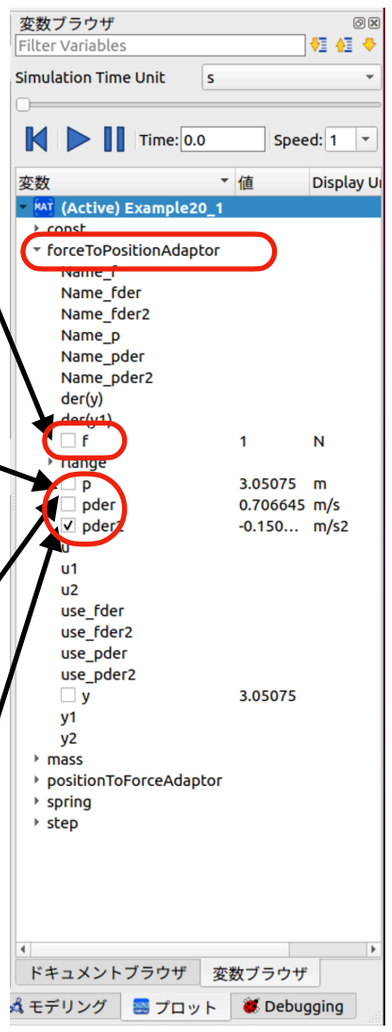






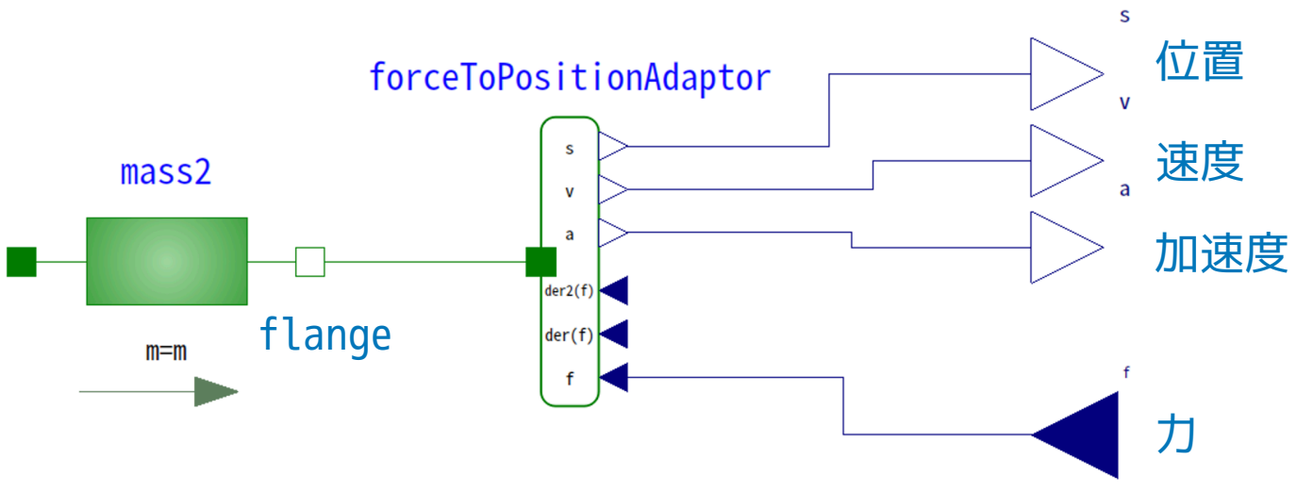






注目する入出力がアダプターに集中するので便利です！

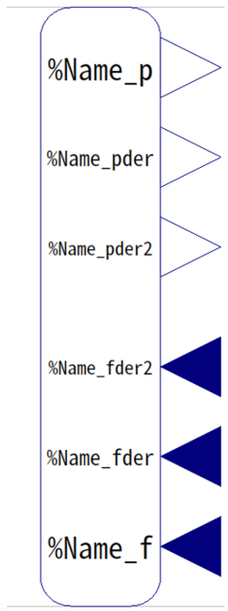
GeneralForceToPositionAdaptor — Flangeを入出力信号に変換する (1)



- 一次元並進機構系の Flange を、入力が力、出力が位置、速度、加速度となるような実数入出力コネクタ(RealInput, RealOutput)の組み合わせに変換する。
- パラメータ use_pder, use_pder2 で、それぞれ速度、加速度の出力の有無を制御できる。

Modelica.Blocks.Interfaces.Adapters.FlowToPotentialAdapter

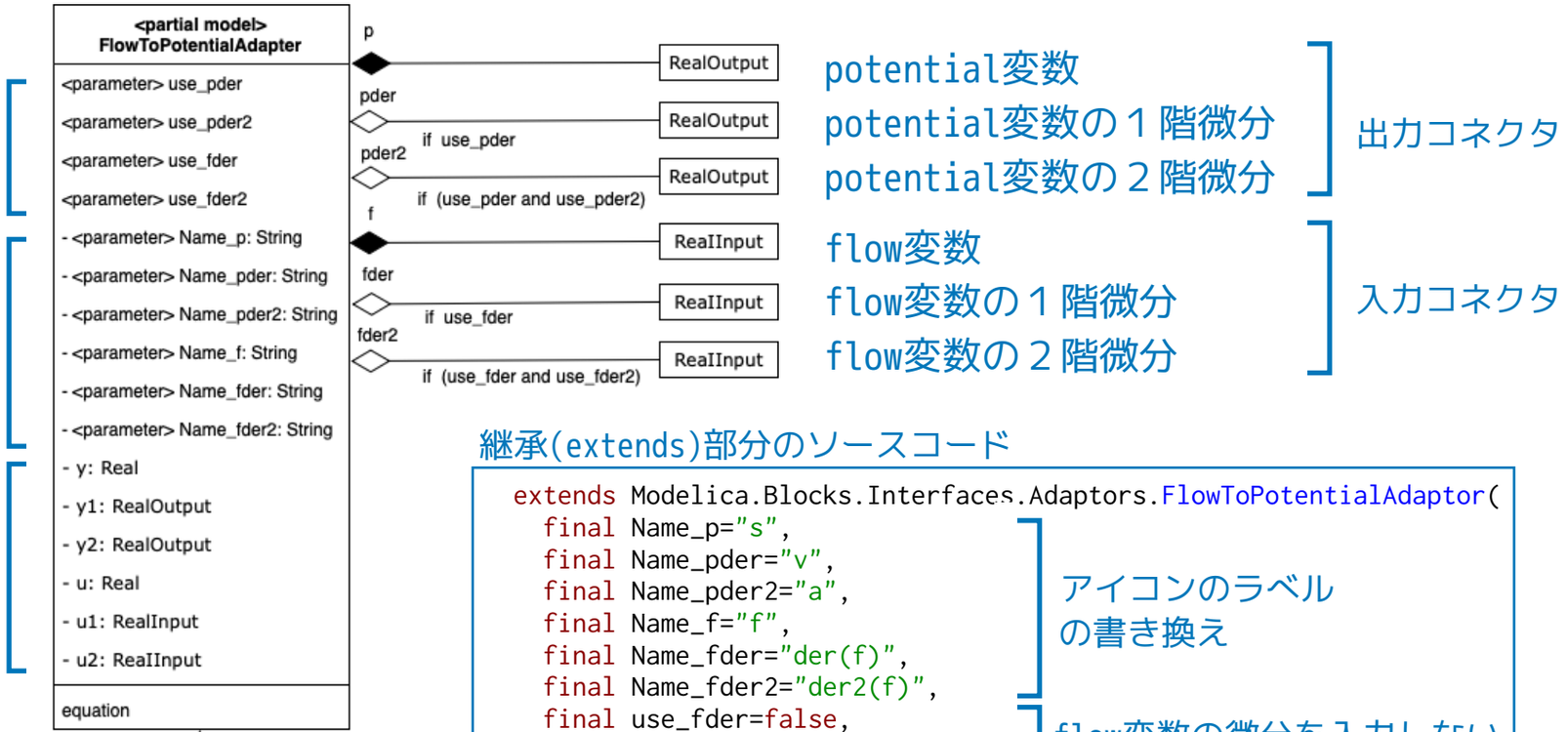
flow変数を入力してpotential変数を出力するアダプタのベースモデル



入出力変数の
１階微分や２階微分のコネクタを
使うかどうかを決めるパラメータ

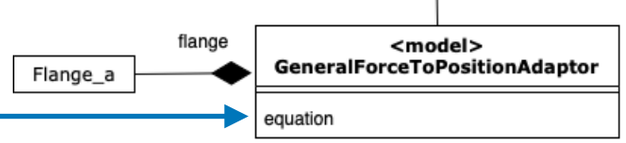
アイコンに表示されるラベル
の文字列

継承先のモデルで使う入出力
変数



GeneralForceToPositionAdapter

$$y = flange.s$$
$$u = flange.f$$



継承(extends)部分のソースコード

```
extends Modelica.Blocks.Interfaces.Adaptors.FlowToPotentialAdaptor(
  final Name_p="s",
  final Name_pder="v",
  final Name_pder2="a",
  final Name_f="f",
  final Name_fder="der(f)",
  final Name_fder2="der2(f)",
  final use_fder=false,
  final use_fder2=false,
  final p(unit="m"),
  final pder(unit="m/s"),
  final pder2(unit="m/s2"),
  final f(unit="N"),
  final fder(unit="N/s"),
  final fder2(unit="N/s2"));
```

アイコンのラベル
の書き換え

flow変数の微分を入力しない

単位の設定

Modelica.Blocks.Interfaces.Adapters.FlowToPotentialAdapterの方程式の内容

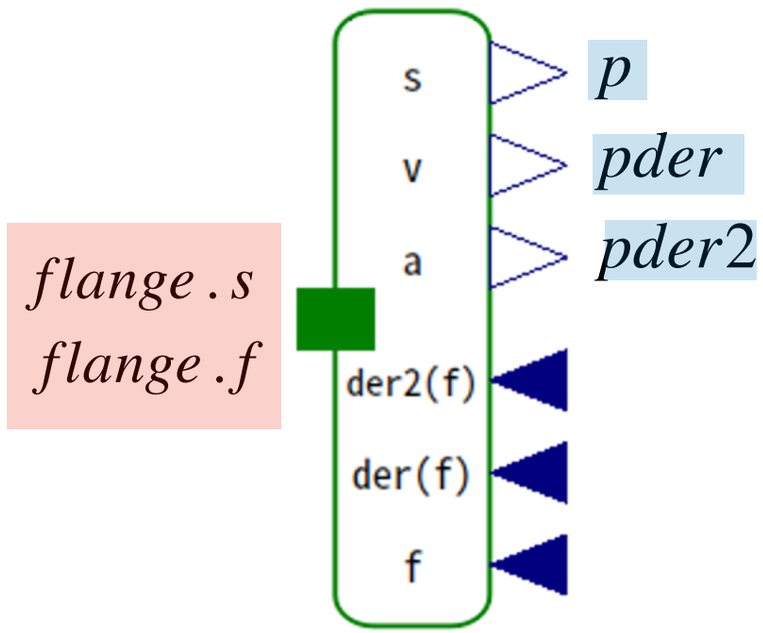
(1) 出力コネクタ $p, pder, pder2$ と出力変数 y, y_1, y_2 の関係

$$y = p$$

$$y_1 = \begin{cases} \frac{dp}{dt}, if\ use\_pder \\ 0, \quad else \end{cases}$$

$$y_2 = \begin{cases} \frac{dy_1}{dt}, if\ (use\_pder\ and\ use\_pder2) \\ 0, \quad else \end{cases}$$

$$connect(y_1, pder)$$
$$connect(y_2, pder2)$$



パラメータによる場合分け

$use\_pder = false, use\_pder2 = false$ のとき

$$p = y = flange.s$$
$$pder = y_1 = 0$$
$$pder2 = y_2 = 0$$

$use\_pder = true, use\_pder2 = false$ のとき

$$p = y = flange.s$$
$$pder = y_1 = \frac{dp}{dt}$$
$$pder2 = y_2 = 0$$

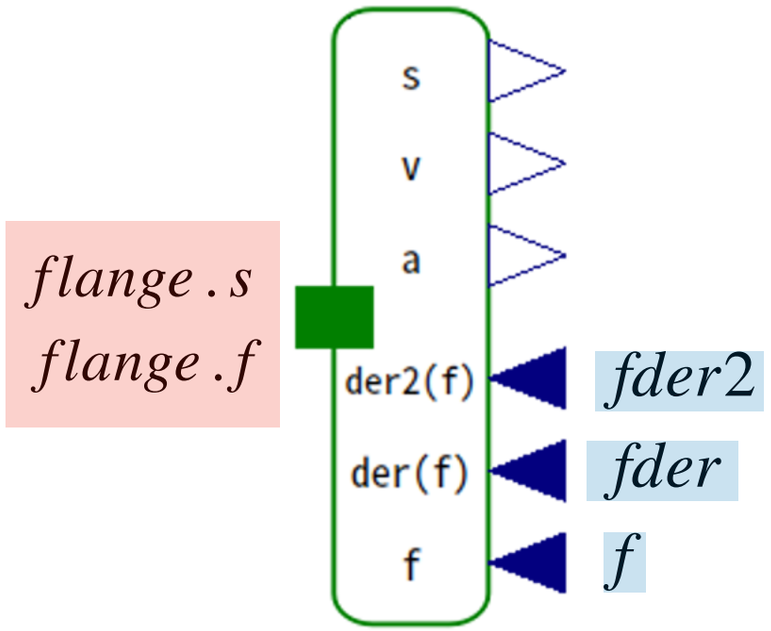
$use\_pder = true, use\_pder2 = true$ のとき

$$p = y = flange.s$$
$$pder = y_1 = \frac{dp}{dt}$$
$$pder2 = y_2 = \frac{dy_1}{dt}$$

(2) 入力コネクタ $f, fder, fder2$ と入力変数 u, u_1, u_2の関係

$$u = \begin{cases} state2(\{f, u_1, u_2\}, time), & if\ (use\_fder\ and\ use\_fder2) \\ state1(\{f, u_1\}, time), & if\ (use\_fder\ and\ not\ use\_fder2) \\ f & else \end{cases}$$

```
if use_fder then
   connect(fder, u1)
else
   u1 = 0
end if

if (use_fder and use_fder2) then
   connect(fder2,u2)
else
   u2 = 0
end if
```



パラメータによる場合分け

$use\_fder = false, use\_fder2 = false$ のとき

$flange.f = u = f$

$u1 = 0$

$u2 = 0$

$use\_fder = true, use\_fder2 = false$ のとき

$flange.f = state1(\{f, u_1\}, time) = f$

$\frac{d}{dt}(flange.f) = state1der1(\{f, u_1\}, time) = u_1 = fder$

$u2 = 0$

$use\_fder = true, use\_fder2 = true$ のとき

$flange.f = state2(\{f, u_1, u_2\}, time) = f$

$\frac{d}{dt}(flange.f) = state2der1(\{f, u_1, u_2\}, time) = u_1 = fder$

$\frac{d^2}{dt^2}(flange.f) = state2der2(\{f, u_1, u_2\}, time) = u_2 = fder2$

オレンジのマーカーをつけた関数の内容は次のスライドに示します。

annotationによる関数の微分の設定

Modelica.Blocks.Interfaces.Adaptors.Functions.state1

```
function state1 "Return state (with one derivative)"
  extends Modelica.Icons.Function;
  input Real u[2] "Required values for state and der(s)";
  input Real dummy
    "Just to have one input signal that should be differentiated to avoid possible problems in the Modelica tool (is not used)";
  output Real s;
algorithm
  s := u[1];
  annotation (derivative(noDerivative=u) = state1der1,
      InlineAfterIndexReduction=true);
end state1;
```

← state1 の微分が state1der1 になる設定

Modelica.Blocks.Interfaces.Adaptors.Functions.state1der1

```
function state1der1 "Return 1st derivative (der of state1)"
  extends Modelica.Icons.Function;
  input Real u[2] "Required values for state and der(s)";
  input Real dummy
    "Just to have one input signal that should be differentiated to avoid possible problems in the Modelica tool (is not used)";
  input Real dummy_der;
  output Real sder1;
algorithm
  sder1 := u[2];
  annotation (InlineAfterIndexReduction=true);
end state1der1;
```

この設定の詳細は以下を参照してください。

https://build.openmodelica.org/Documentation/ModelicaReference.Annotations.derivative.html

Modelica Language Specification ver 3.5, p.171, https://modelica.org/documents/MLS.pdf

Modelica.Blocks.Interfaces.Adaptors.Functions.state2

```
function state2 "Return state (with two derivatives)"
  extends Modelica.Icons.Function;
  input Real u[3] "Required values for state and der(s)";
  input Real dummy
    "Just to have one input signal that should be differentiated to avoid possible problems in the Modelica tool (is not used)";
  output Real s;
algorithm
  s := u[1];
  annotation (derivative(noDerivative=u) = state2der1,
      InlineAfterIndexReduction=true);
end state2;
```

← state2 の微分が state2der1 になる設定

Modelica.Blocks.Interfaces.Adaptors.Functions.state2der1

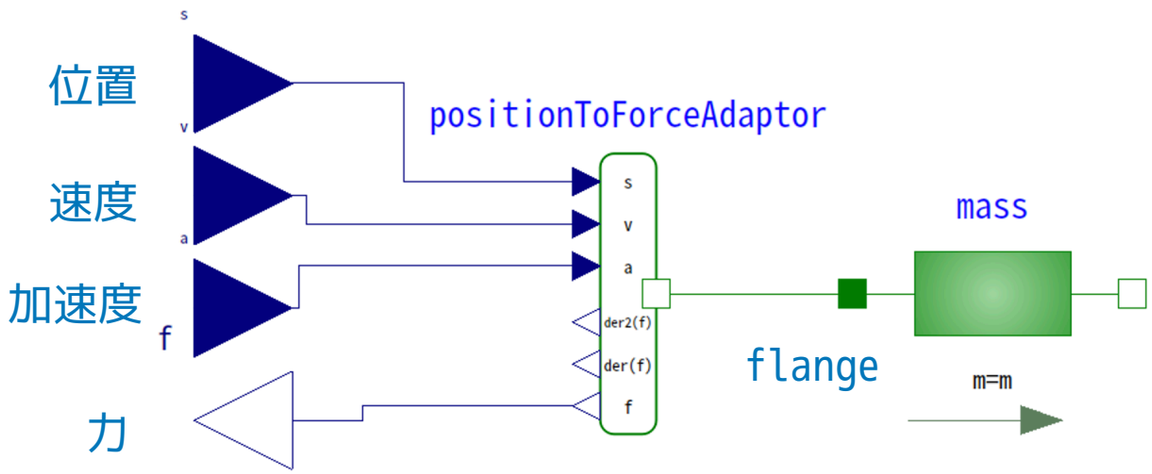
```
function state2der1 "Return 1st derivative (der of state2)"
  extends Modelica.Icons.Function;
  input Real u[3] "Required values for state and der(s)";
  input Real dummy
    "Just to have one input signal that should be differentiated to avoid possible problems in the Modelica tool (is not used)";
  input Real dummy_der;
  output Real sder1;
algorithm
  sder1 := u[2];
  annotation (derivative(noDerivative=u, order=2) = state2der2,
      InlineAfterIndexReduction=true);
end state2der1;
```

← state2der1 の微分が state2der2 になる設定

Modelica.Blocks.Interfaces.Adaptors.Functions.state2der1

```
function state2der2 "Return 2nd derivative (der of state2der1)"
  extends Modelica.Icons.Function;
  input Real u[3] "Required values for state and der(s)";
  input Real dummy
    "Just to have one input signal that should be differentiated to avoid possible problems in the Modelica tool (is not used)";
  input Real dummy_der;
  input Real dummy_der2;
  output Real sder2;
algorithm
  sder2 := u[3];
  annotation (InlineAfterIndexReduction=true);
end state2der2;
```

GeneralPositionToForceAdapter — Flangeを入出力信号に変換する(2)



- 一次元並進機構系の Flange を、入力が位置、速度、加速度など、出力が力となるような実数入出力コネクタ(RealInput, RealOutput)の組み合わせに変換する。
- パラメータ use_pder, use_pder2 で、それぞれ速度、加速度の入力の有無を設定できる。

入力コネクタ
- potential変数
- potential変数の１階微分
- potential変数の２階微分

出力コネクタ
- flow変数
- flow変数の１階微分
- flow変数の２階微分

継承(extends)部分のソースコード

```
extends Modelica.Blocks.Interfaces.Adaptors.PotentialToFlowAdaptor(
  final Name_p="s",
  final Name_pder="v",
  final Name_pder2="a",
  final Name_f="f",
  final Name_fder="der(f)",
  final Name_fder2="der2(f)"
  final use_fder=false,
  final use_fder2=false,
  final p(unit="m"),
  final pder(unit="m/s"),
  final pder2(unit="m/s2"),
  final f(unit="N"),
  final fder(unit="N/s"),
  final fder2(unit="N/s2"));
```

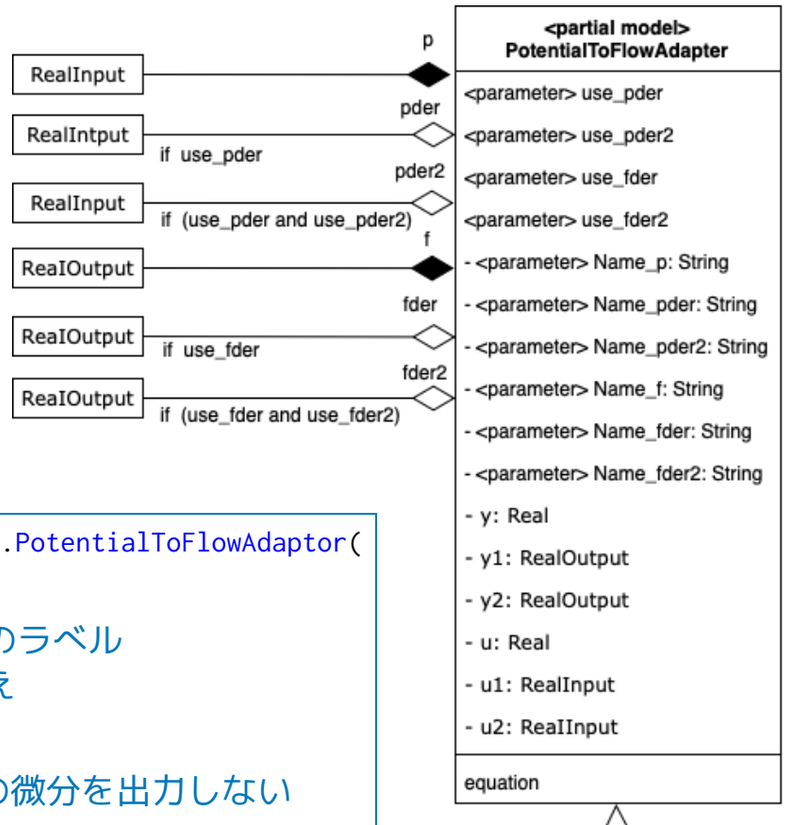
アイコンのラベルの書き換え

flow変数の微分を出力しない

単位の設定

Modelica.Blocks.Interfaces.Adapters.PotentialoFlowAdapter

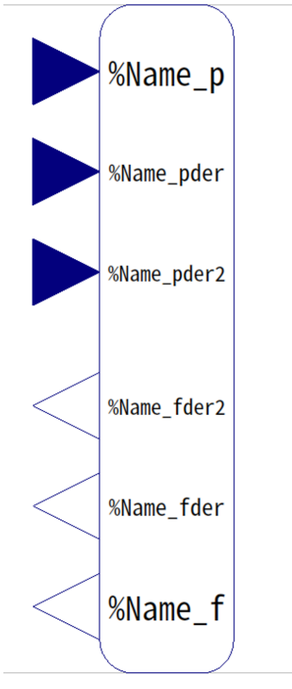
potential変数を入力してflow変数を出力するアダプタのベースモデル



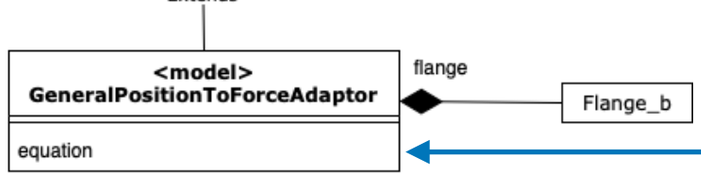
入出力変数の１階微分や２階微分のコネクタを使うかどうかを決めるパラメータ

アイコンに表示されるラベルの文字列

継承先のモデルで使う入出力変数



GeneralForceToPositionAdapter



$$y = flange.f$$
$$u = flange.s$$

Modelica.Blocks.Interfaces.Adapters.PotentialToFlowAdapterの方程式の内容

出力コネクタ $f, fder, fder2$ と出力変数 y, y_1, y_2 の関係

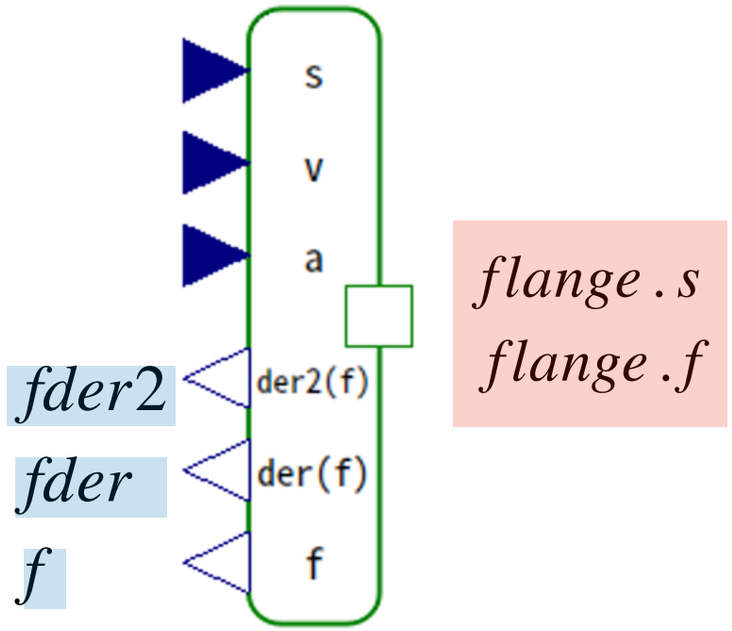
$$y = -f$$

$$y_1 = \begin{cases} -\frac{dy}{dt}, if\ use\_fder \\ 0, \quad else \end{cases}$$

$$y_2 = \begin{cases} -\frac{dy_1}{dt}, if\ (use\_fder\ and\ use\_fder2) \\ 0, \quad else \end{cases}$$

$$connect(y_1, fder)$$

$$connect(y_2, fder2)$$



パラメータによる場合分け

$use\_fder = false, use\_fder2 = false$ のとき

$$-f = y = flange.f$$

$$fder = y_1 = 0$$

$$fder2 = y_2 = 0$$

$use\_pder = true, use\_pder2 = false$ のとき

$$-f = y = flange.f$$

$$fder = y_1 = -\frac{dy}{dt}$$

$$fder2 = y_2 = 0$$

$use\_pder = true, use\_pder2 = true$ のとき

$$-f = y = flange.f$$

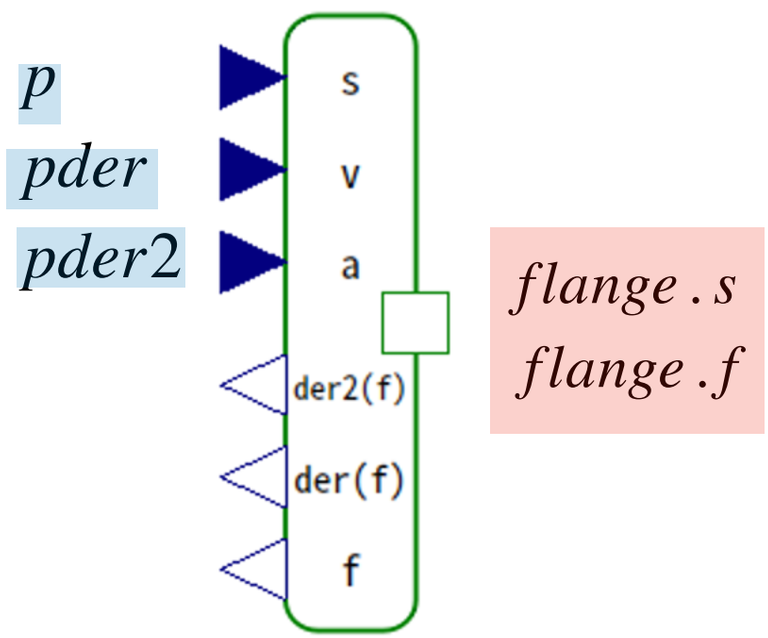
$$fder = y_1 = -\frac{dy}{dt}$$

$$fder2 = y_2 = -\frac{dy_1}{dt}$$

入力コネクタ $p, pder, pder2$ と入力変数 u, u_1, u_2の関係

$$u = \begin{cases} state2(\{p, u_1, u_2\}, time), & if\ (use\_pder\ and\ use\_pder2) \\ state1(\{p, u_1\}, time), & if\ (use\_pder\ and\ not\ use\_pder2) \\ p & else \end{cases}$$

```
if use_pder then
  connect(pder, u1)
else
  u1 = 0
end if

if (use_pder and use_pder2) then
  connect(pder2,u2)
else
  u2 = 0
end if
```



パラメータによる場合分け

$use\_pder = false, use\_pder2 = false$ のとき

$flange.s = u = p$

$u1 = 0$

$u2 = 0$

$use\_fder = true, use\_fder2 = false$ のとき

$flange.s = state1(\{p, u_1\}, time) = p$

$\frac{d}{dt}(flange.s) = state1der1(\{p, u_1\}, time) = u_1 = pder$

$u2 = 0$

$use\_fder = true, use\_fder2 = true$ のとき

$flange.s = state2(\{p, u_1, u_2\}, time) = p$

$\frac{d}{dt}(flange.s) = state2der1(\{p, u_1, u_2\}, time) = u_1 = pder$

$\frac{d^2}{dt^2}(flange.s) = state2der2(\{p, u_1, u_2\}, time) = u_2 = pder2$

オレンジのマーカーをつけた関数の内容は annotationによる関数の微分の設定に 示します。