OpenModelica講習中級 Modelica.Fluidライブラリ解説

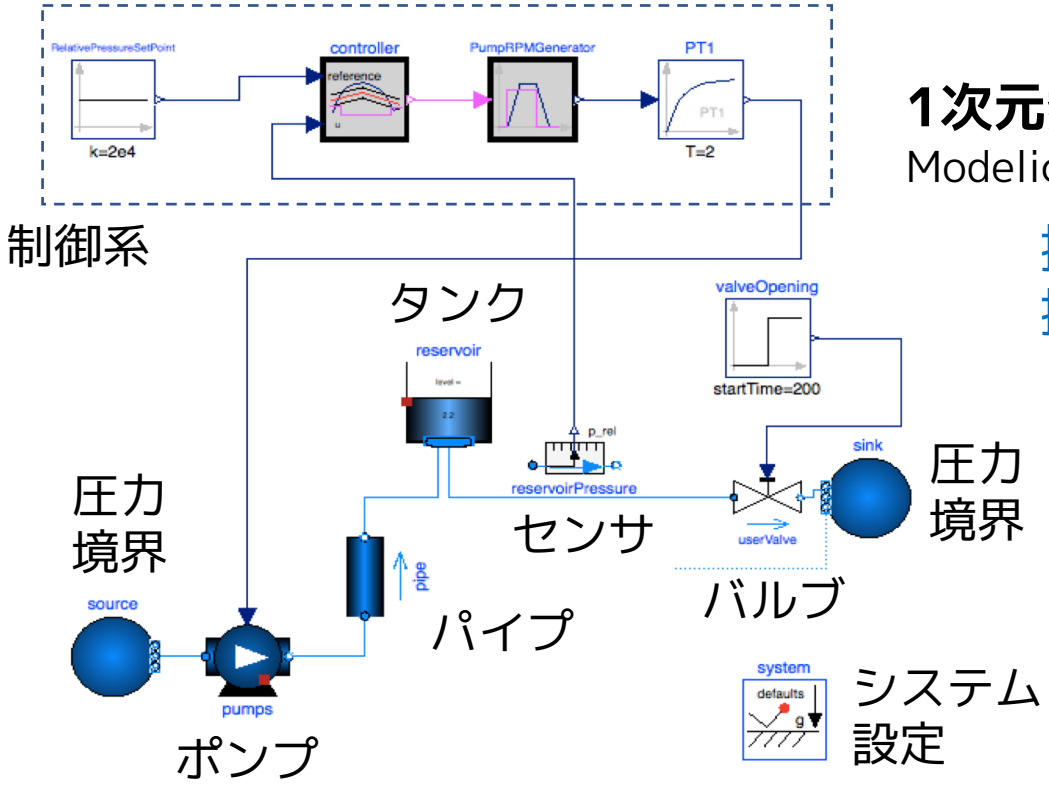
3. Modelica.Fluidライブラリ

2017年12月7日　田中 周(有限会社アマネ流研)

3. Modelica.Fluid ライブラリ

Modelica.Fluidライブラリの目標

- １次元熱流体システムモデルの**インターフェース**と**コンポーネント**の提供
- Modelicaによる流体モデルの実装方法の例示

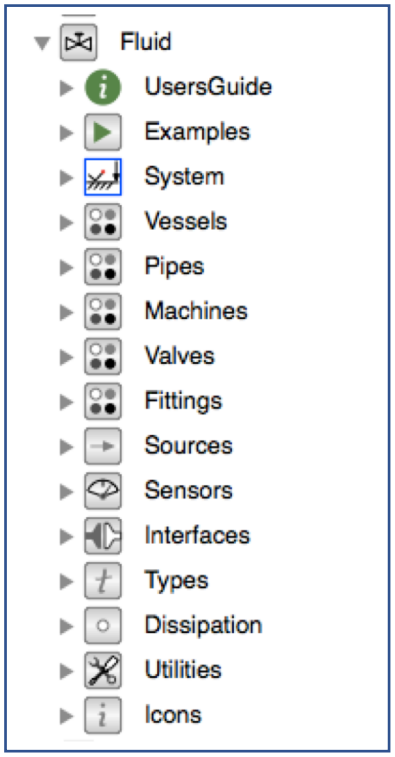


1次元熱流体システムモデルの例
Modelica.Fluid.Examples.PumpingSystem

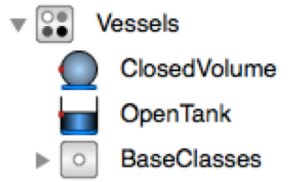
揚水ポンプで水をタンクに揚げて給水するシステム

Modelica.Fluid ライブラリ

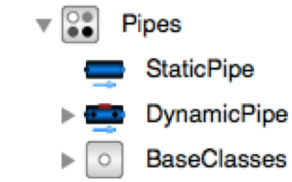
Modelica.Fluid のコンポーネント



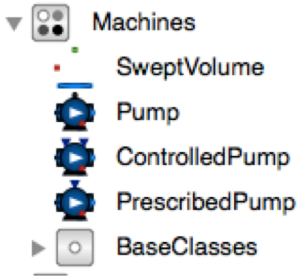
容器、タンクなど



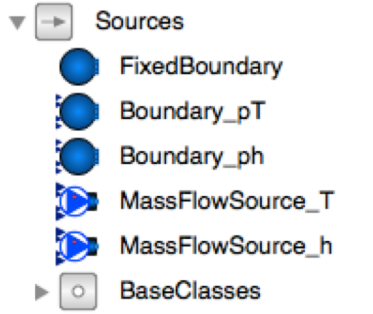
パイプ



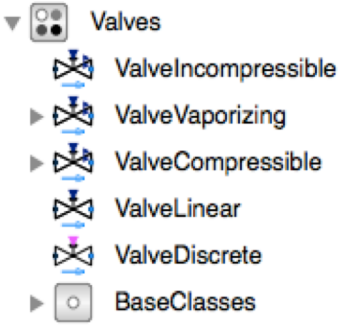
ポンプなど



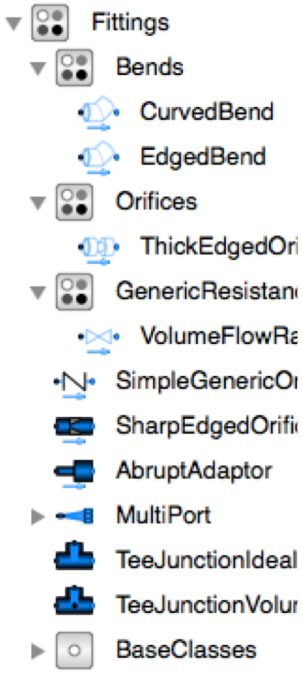
流量、圧力規定などの境界



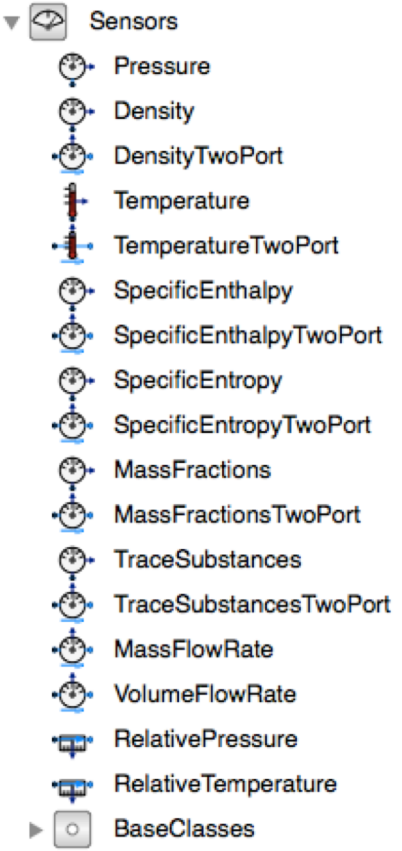
バルブ



ベンド、オリフィス、継手など



センサ



Examples

例題を通して、Modelica.Fluidライブラリの基本的概念を示す。

FluidExample1 FluidPort

- **WaterMix1** 熱湯と冷水を混ぜる
- **TraceSubstanceMix1** 塩素濃度の異なる水を混ぜる
- **GasMix1** メタンガスと空気を混ぜる

FluidExample2 volume モデル

- **HotRoom1** 部屋の冷たい空気を温かい空気に入れ替える
- **HotRoom2** 部屋を2つに分割する
- **WaterVolume1** 容器の冷水と熱湯を入れ替える
- **WaterVolume2** 容器出入り口の圧力損失を考慮する

Examples

FlowExample3 flow モデル

- **StaticPipeTest1** flow モデルと volume モデルを配置する
- **StaticPIpeTest2** flow モデルを直接つなぐ

FluidExample4 DynamicPipe

- **WaterExchange1〜4** 冷水の入ったパイプに熱湯を流し込む

FluidExample5 VesselPortsData

- **WaterTank** 水の入ったタンクにお湯を注ぎ込む

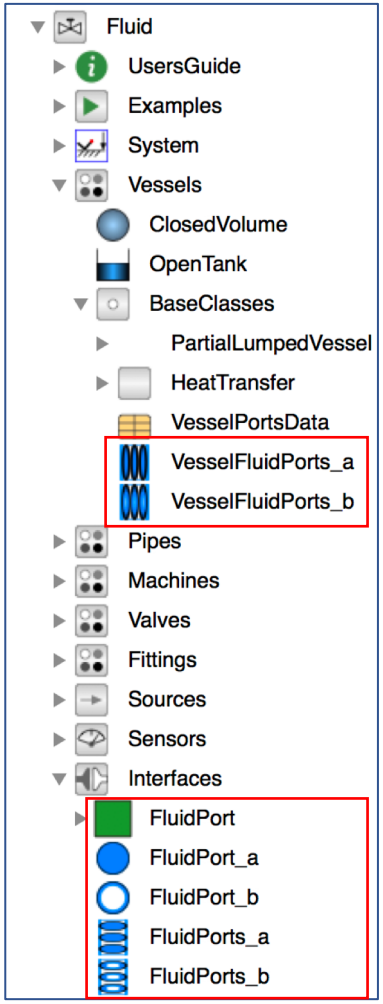
FluidExample6 HeatTransfer モデル

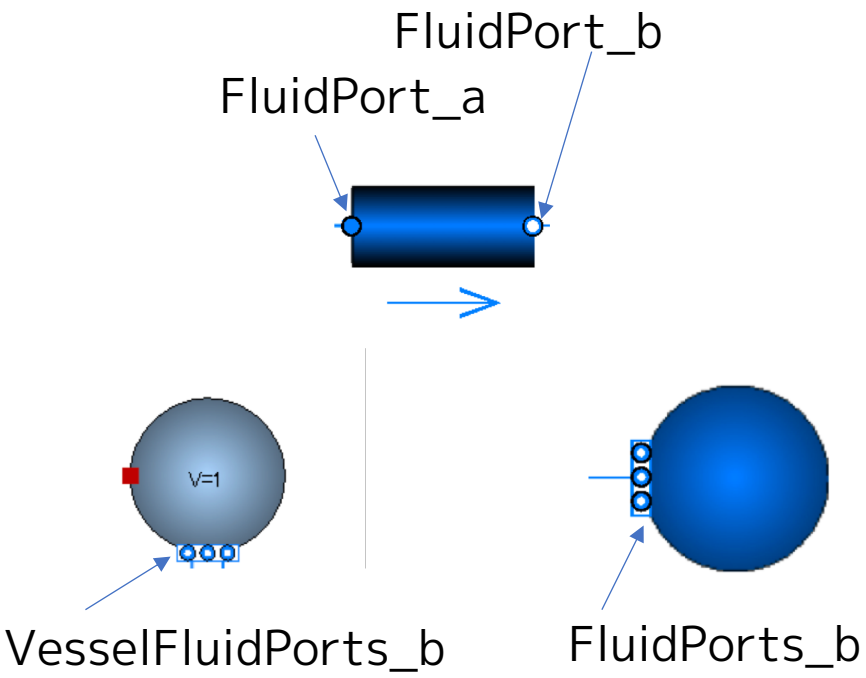
- **WaterHeating** パイプを加熱してタンク内の水を温める

Modelica.Fluid.Examples.IncompressibleFluidNetwork

- 非圧縮性流体のネットワークモデル

FluidExample1
FluidPort: 1次元流体モデルのコネクタ

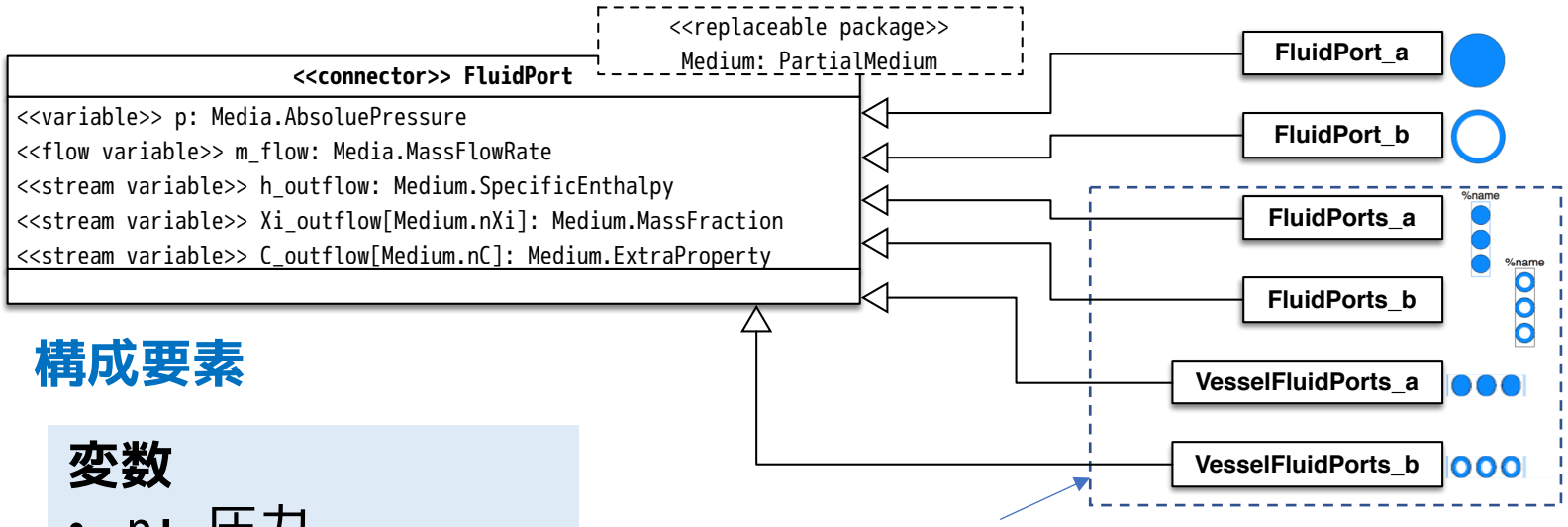




FluidPort

(1) FluidPortの継承関係と構成要素

中身は同じ！！



構成要素

変数
- p: 圧力

flow変数
- m_flow: 質量流量

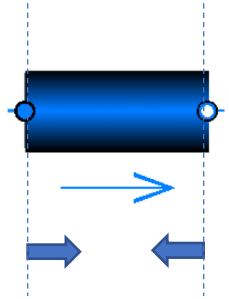
stream変数
- h_outflow: 比エンタルピ
- Xi_outflow: 成分物質の質量分率
- C_outflow: 付加的物質（微小物質)の濃度

ベクトル(配列)にして複数のFluidPortが集まったものとして使用する。

FluidPort

(2) Fluidの質量流量の向き

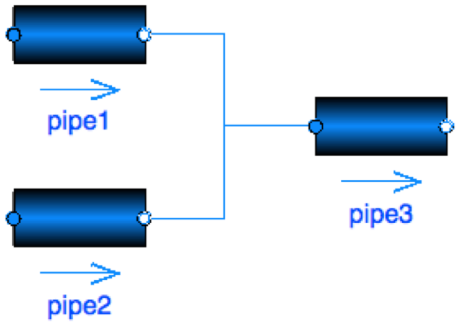
$\dot{m}$ = m_flow は、FluidPortから
コンポーネント内部に向かう向きが正



$\dot{m} > 0$

(3) FluidPortの接続による方程式

次のようにパイプを接続した場合を考える。



冗長なので変数などの表示を簡略化する。

p_i = pipe$i$.port_x.p
$\dot{m}_i$= pipe$i$.port_x.m_flow
$h_{outflow_i}$ = pipe$i$. port_x.h_outflow

x=a or b
i = 1,2,3

方程式

```
connect(pipe1.port_b, pipe3.port_a);
connect(pipe2.port_b, pipe3.port_a);
```

FluidPort

方程式の内容はこうなる。

pipe1

p_1

$h_1 = h_{outflow_1}$

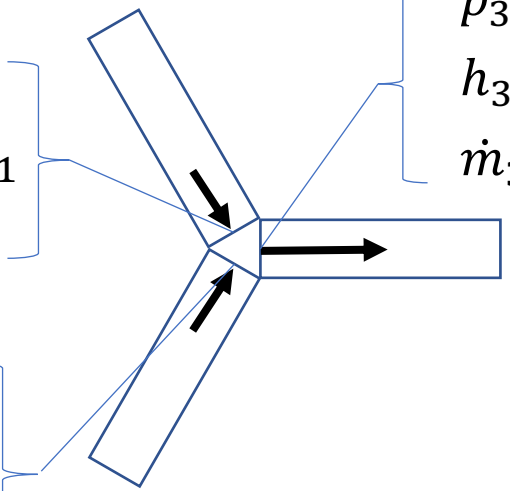
$\dot{m}_1 \leq 0$

pipe2

p_2

$h_2 = h_{outflow_2}$

$\dot{m}_2 \leq 0$



pipe3

p_3

$h_3 = inStream\left(h_{outflow_3}\right)$

$\dot{m}_3 > 0$

変数 p_i

$p_1 = p_2 = p_3$

flow変数 $\dot{m}_i$

$$\sum_i \dot{m}_i = \dot{m}_1 + \dot{m}_2 + \dot{m}_3 = 0$$

stream変数 $h_{outflow}, \boldsymbol{Xi_{outflow}}, \boldsymbol{C_{outflow}}$

$h_{outflow_i}$ $\quad \dot{m}_i \leq 0$

$$inStream\left(h_{outflow_i}\right) \equiv \frac{\sum_{j \neq i} \max\left(-\dot{m}_j, 0\right) \cdot h_{outflow_j}}{\sum_{j \neq i} \max\left(-\dot{m}_j, 0\right)} \quad \dot{m}_i > 0$$

流入時の変数の値を計算するビルトインオペレータ

FluidPort

(4) コンポーネント内部での stream 変数の扱い

port_h_outflow：比エンタルピ
port_x.Xi_outfow：質量分率
port_a.C_outflow：付加的物質濃度（微小物質濃度）

コンポーネント内部で計算した流出時の値を設定する。

A. コンポーネント内部に質量やエネルギーを蓄えない場合

StaticPipe のソースコードより抜粋

port_a から出て行く量は
port_b から入ってきた量、
port_b から出て行く量は
port_a から入ってきた量、...

```
equation
  ...
  0 = port_a.m_flow + port_b.m_flow;
  port_a.Xi_outflow = inStream(port_b.Xi_outflow);
  port_b.Xi_outflow = inStream(port_a.Xi_outflow);
  port_a.C_outflow = inStream(port_b.C_outflow);
  port_b.C_outflow = inStream(port_a.C_outflow);
  ...
  port_b.h_outflow = inStream(port_a.h_outflow) - system.g*height_ab;
  port_a.h_outflow = inStream(port_b.h_outflow) + system.g*height_ab;
  ...
end StaticPipe;
```

エンタルピは
パイプ両端の高低差
を考慮する。

FluidPort

B. コンポーネント内部に質量やエネルギーを蓄える場合

保存方程式の移流項は次のように表される。

エンタルピ移流項 $= \dot{m}_i \cdot actualStream(h_{outflow_i})$ [W]

stream変数

$$actualStream(h_{outflow_i}) \equiv \begin{cases} inStream\left(h_{outflow_i}\right), & \dot{m}_i > 0 \\ h_{outflow_i} & \dot{m} \leq 0 \end{cases}$$

流れの向きに依存して実際に流入出する変数の値を計算する
ビルトインオペレータ

参考 https://www.modelica.org/documents/ModelicaSpec32Revision1.pdf p.229-p.233

FluidPort

PartialLumpedVesselのソースコードより抜粋

```
ports[i].h_outflow  = medium.h;
ports[i].Xi_outflow = medium.Xi;
ports[i].C_outflow  = C;

ports_H_flow[i] = ports[i].m_flow * actualStream(ports[i].h_outflow)
"Enthalpy flow";
ports_E_flow[i] = ports[i].m_flow*(0.5*portVelocities[i]*portVelocities[i]
 + system.g*portsData_height[i])
"Flow of kinetic and potential energy";
ports_mXi_flow[i,:] = ports[i].m_flow * actualStream(ports[i].Xi_outflow)
"Component mass flow";
ports_mC_flow[i,:]  = ports[i].m_flow * actualStream(ports[i].C_outflow)
"Trace substance mass flow";
```

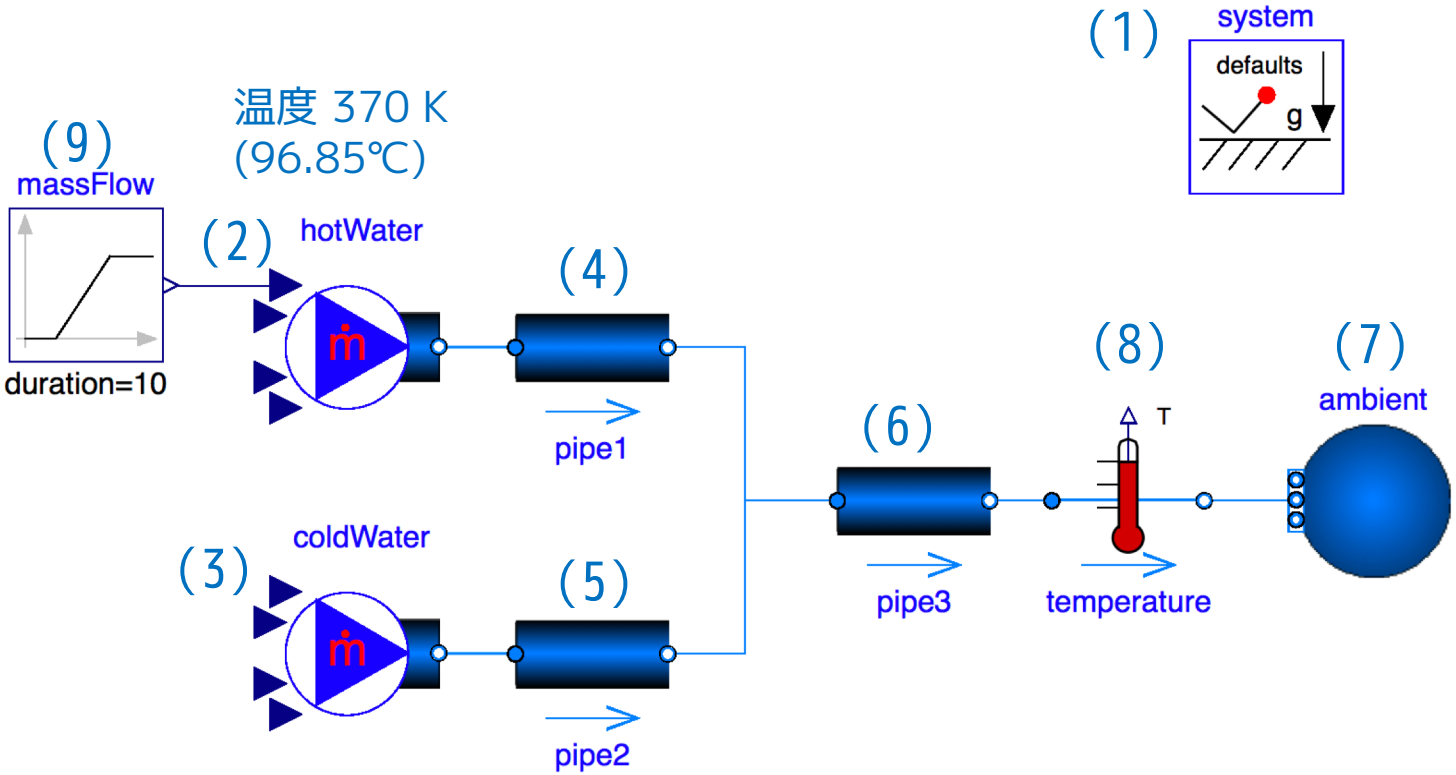
コンポーネント内部で保存式から計算した流出時の値を設定する。

エネルギー、成分の質量、付加的物質などの保存式の移流項の計算

WaterMix1

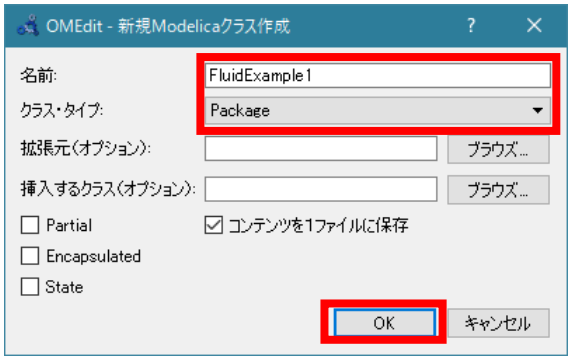
熱湯と冷水を混ぜる

FluidPort の機能を確かめる。**pipe1** に熱湯、**pipe2** に冷水が流れている。これを **pipe3** で混ぜ合わせるモデルを作る。

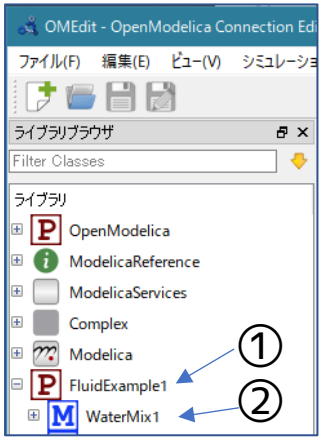


WaterMix1

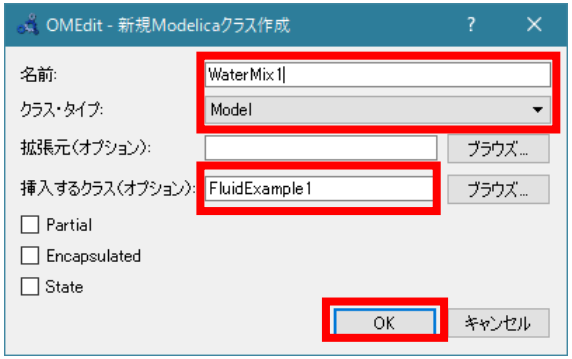
① ファイル >Modelicaクラス新規作成で package を作る。



名前: FluidExample1
クラス・タイプ: Package



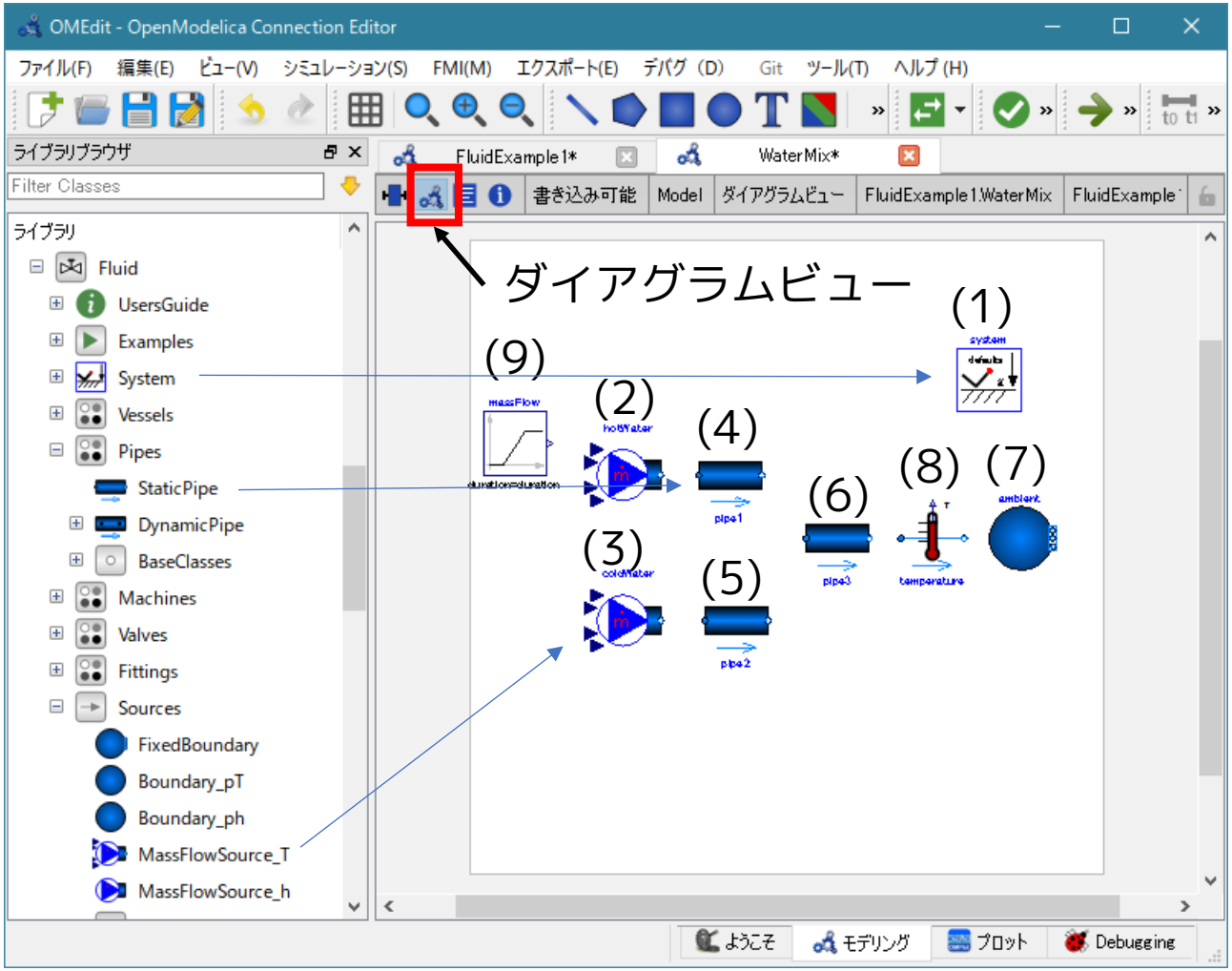
②ライブラリブラウザの FluidExample1を右クリックして「Modelicaクラス新規作成」を選び、model を作る。



名前: WaterMix1
クラス・タイプ: Model
挿入するクラス: FluidExample1

WaterMix1

③WaterMix1のダイアグラムビューに、次のスライドに示すコンポーネント (1)～(9) をドラッグアンドドロップする。

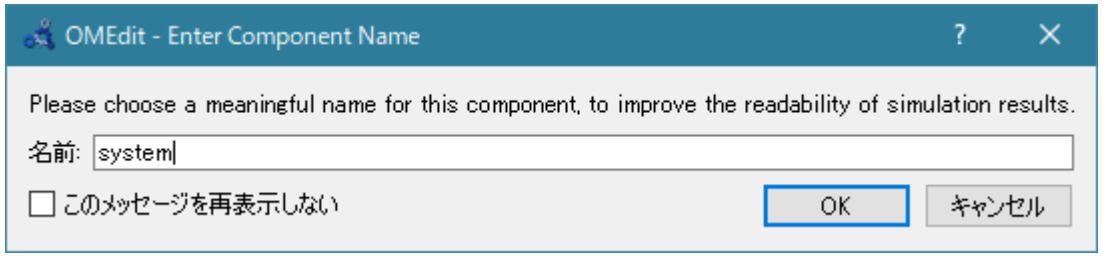


WaterMix1

配置するコンポーネントの名前とクラス

名前	クラス
(1)system:	Fluid.System
(2)hotWater:	Fluid.Sources.MassFlowSource_T
(3)coldWater:	Fluid.Sources.MassFlowSource_T
(4)pipe1:	Fluid.Pipes.StaticPipe
(5)pipe2:	Fluid.Pipes.StaticPipe
(6)pipe3:	Fluid.PIpes.StaticPipe
(7)ambient:	Fluid.Sources.FixedBoundary
(8)temperature:	Fluid.Sensors.TemperatureTwoPort
(9)massFlow:	Block.Sources.Ramp

コンポーネントをドロップするときに現れるダイアログで名前をつける。

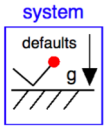


コンポーネントを右クリックして[属性]を選択しても名前を変更できる。

WaterMix1

④コンポーネントを右クリックして[パラメータ]を選択して設定する。

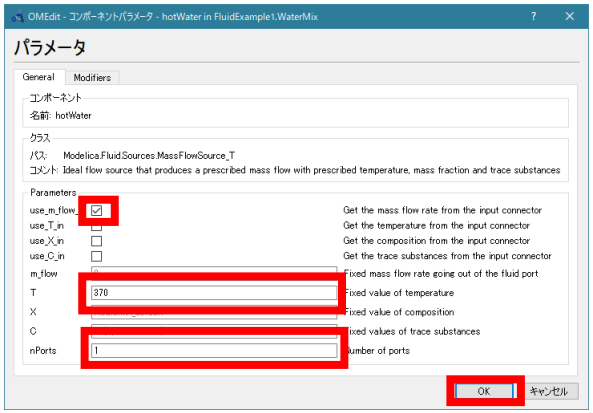
(1) Fluid.System

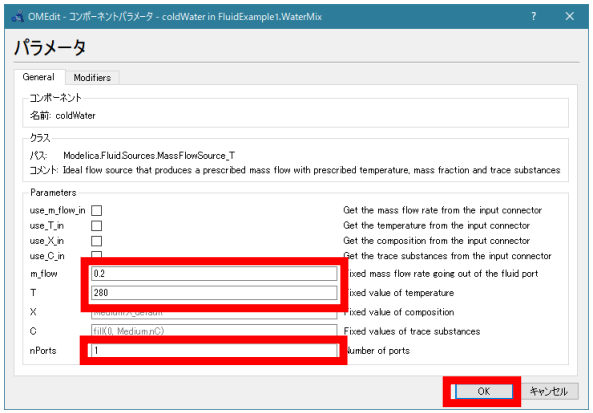


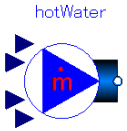
流体システムモデルで必ず配置する。

(1) system
- デフォルトのまま設定しない

(2)(3) Fluid.Sources.MassFlowSources_T







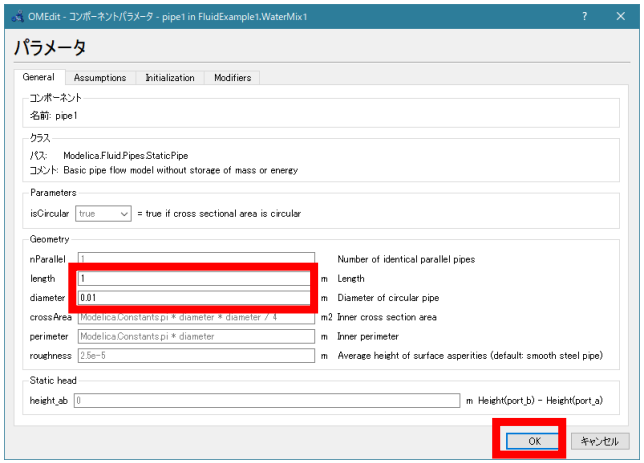
流量と温度を規定する。

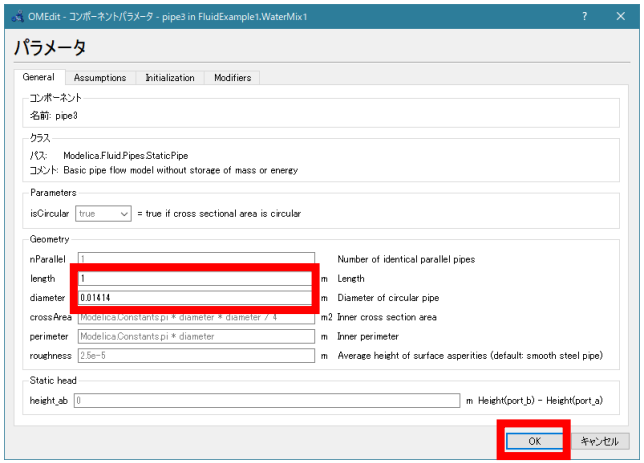
(2) hotWater
- use_m_flow_in をチェックする。
- T = 370 [K]
- nPorts = 1

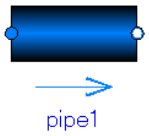
(3) coldWater
- m_flow = 0.2 [kg/s]
- T = 280 [K]
- nPorts = 1

WaterMix1

(4)(5)(6) Fluid.Pipes.StaticPipe







内部に質量やエネルギーを蓄えないパイプ
長さと内径を設定する。

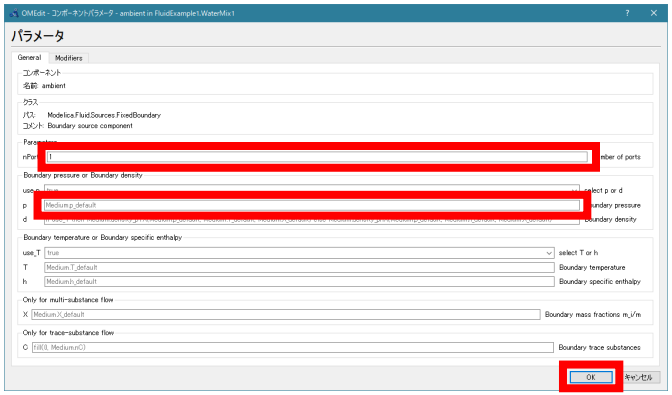
(4) pipe1, (5) pipe2
- length = 1 [m]
- diameter = 0.01 [m]

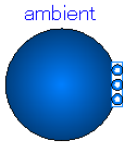
(6) pipe3
- length = 1 [m]
- diameter = 0.01414 [m]

水はpipe1とpipe2からpipe3に流れ込むので、pipe1 とpipe2の断面積の和がおおよそ pipe3の断面積になるようにした。

WaterMix1

(7) Fluid.Sources.FixedBoundary

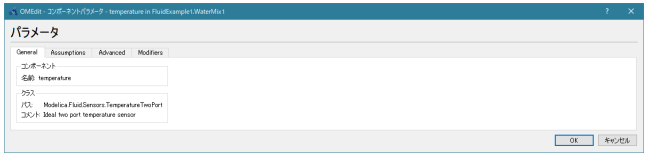


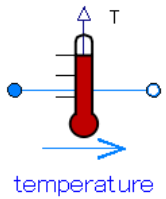


- 圧力または密度
- 温度またはエンタルピーを規定する

(7) ambient
- nPorts = 1
- p = 101325 [Pa]

(8) Fluid.Sensors.TemperatureTwoPort



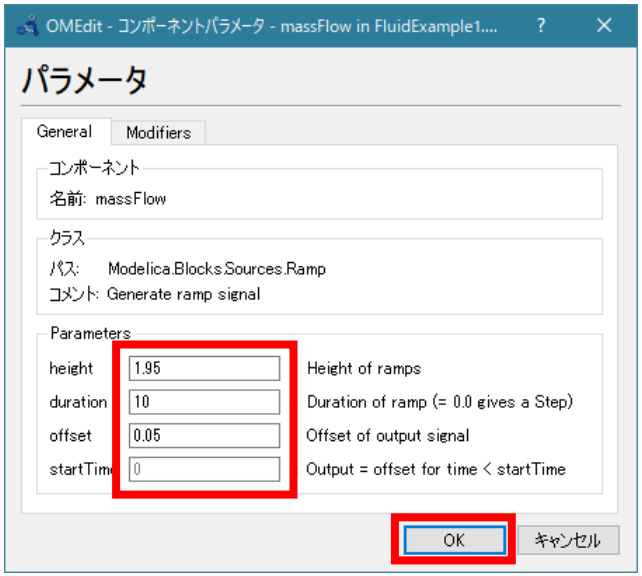


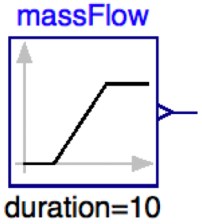
温度センサー

(8) temperature
- デフォルト設定

WaterMix1

(9) Blocks.Sources.Ramp

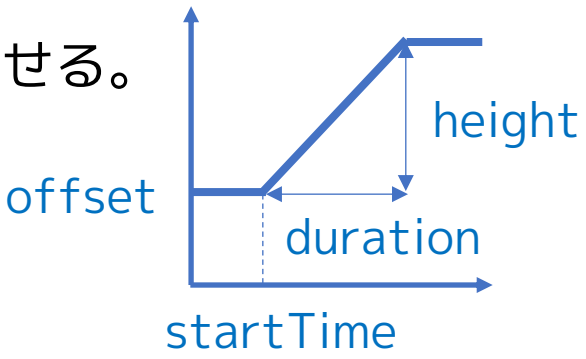




出力がランプ関数的に変化する信号源

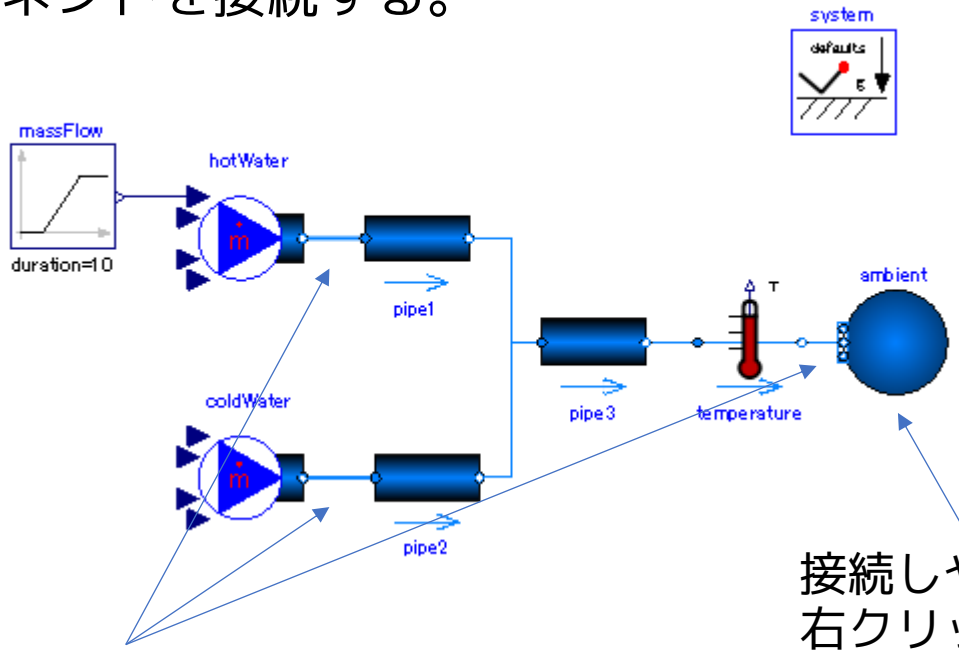
(9) massFlow
- height = 1.95
- duration = 10 [s]
- offset = 0.05
- startTime = 0 [s]

低濃度側の水の質量流量を、10秒間で
0.05 kg/s から 2.00 kg/s まで変化させる。

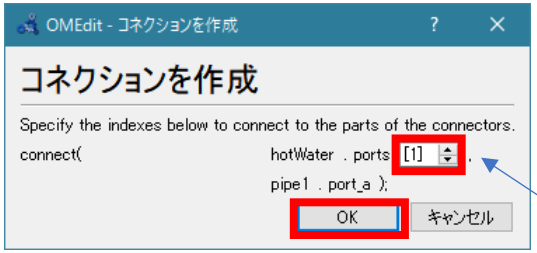


WaterMix1

⑤コンポーネントを接続する。



接続しやすいように
右クリックして[横反転]を
選択する。



hotWater、coldWater、ambientはコネクタfluidPortが配列になっている。接続時に表示されるダイアログで **[1]** (1番目の要素)を選ぶ。

fluidPortの配列の要素1個に対して1本の接続線だけをつなぐ。

WaterMix1

⑥ FluidExample1のテキストビューに切り替えてコードを編集する。

太字の部分を編集する。

交換可能ローカルパッケージとしてMediumという名前の物性パッケージを宣言する。

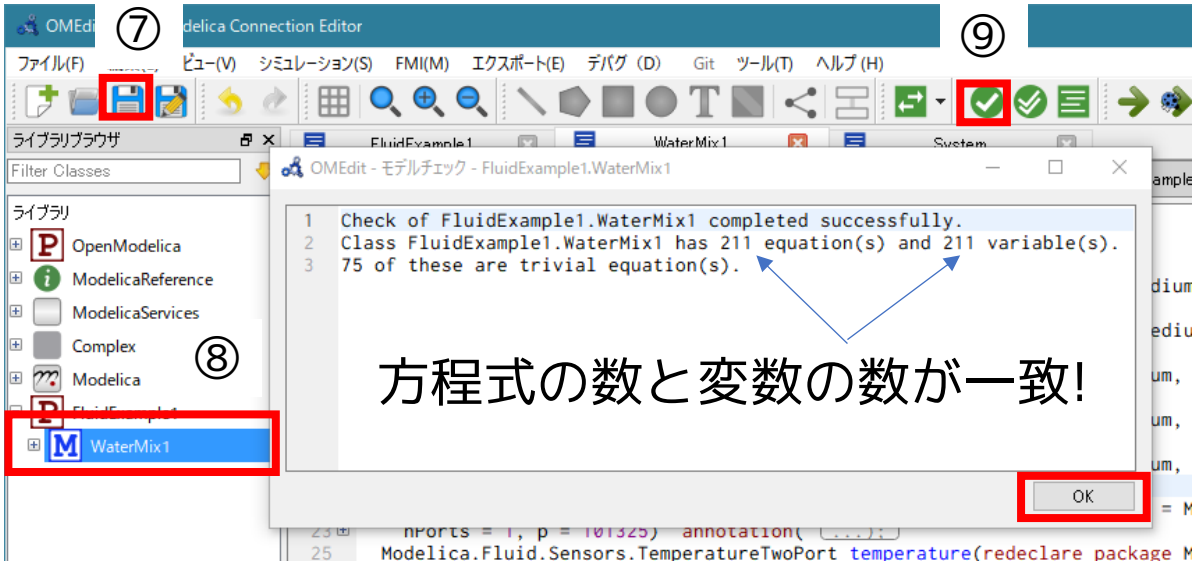
使用するコンポーネントでも再宣言する。

```
package FluidExample1
  import Modelica.Media;

model WaterMix1
  replaceable package Medium = Media.Water.StandardWater;                                    (1)
  inner Modelica.Fluid.System system annotation( ...);
  Modelica.Fluid.Sources.MassFlowSource_T hotWater(redeclare package Medium = Medium,        (2)
    T = 370, m_flow = 0.2, nPorts = 1, use_m_flow_in = true)  annotation( ...);
  Modelica.Fluid.Sources.MassFlowSource_T coldWater(redeclare package Medium = Medium,       (3)
    T = 280, m_flow = 0.2, nPorts = 1) annotation( ...);
  Modelica.Fluid.Pipes.StaticPipe pipe1(redeclare package Medium = Medium,                   (4)
    diameter = 0.01, length = 1)  annotation( ...);
  Modelica.Fluid.Pipes.StaticPipe pipe2(redeclare package Medium = Medium,                   (5)
    diameter = 0.01, length = 1)  annotation( ...);
  Modelica.Fluid.Pipes.StaticPipe pipe3(redeclare package Medium = Medium,                   (6)
    diameter = 0.01414, length = 1)  annotation( ...);
  Modelica.Fluid.Sources.FixedBoundary ambient(redeclare package Medium = Medium,            (7)
    nPorts = 1)  annotation( ...);
  Modelica.Fluid.Sensors.TemperatureTwoPort temperature(redeclare package Medium = Medium)   (8)
    annotation( ...);
  Modelica.Blocks.Sources.Ramp massFlow(duration = 10, height = 1.95, offset = 0.05,         (9)
    startTime = 0)  annotation( ...);
```

WaterMix1

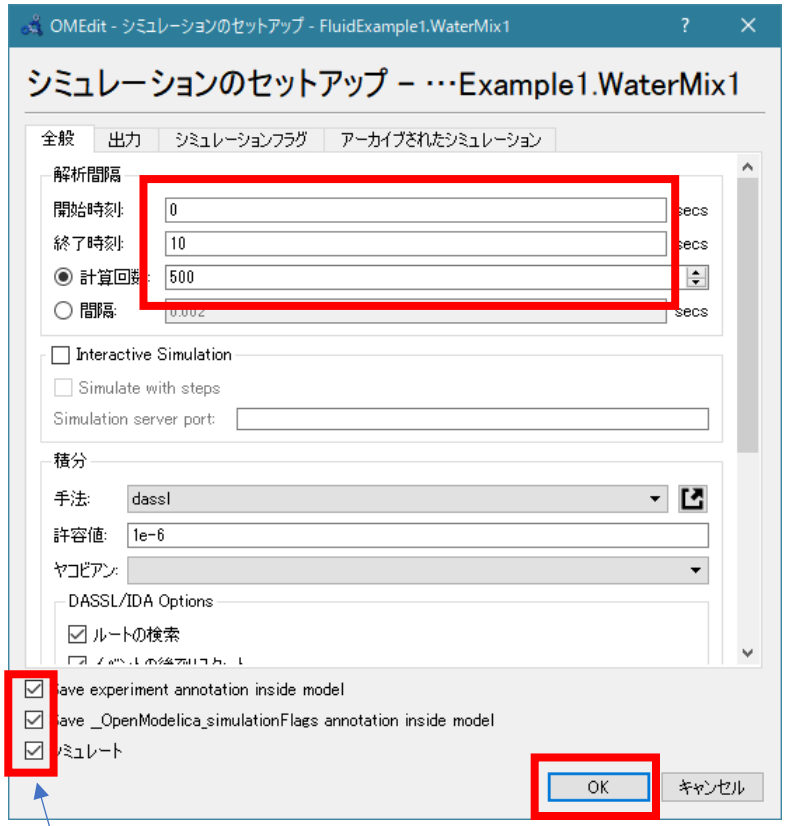
```
equation
    connect(massFlow.y, hotWater.m_flow_in) annotation( ...);
    connect(coldWater.ports[1], pipe2.port_a) annotation( ...);
    connect(hotWater.ports[1], pipe1.port_a) annotation( ...);
    connect(pipe2.port_b, pipe3.port_a) annotation( ...);
    connect(pipe1.port_b, pipe3.port_a) annotation( ...);
    connect(temperature.port_b, ambient.ports[1]) annotation( ...);
    connect(temperature.port_a, pipe3.port_b) annotation( ...);
end WaterMix1;

  annotation( ...);
end FluidExample1;
```

⑦保存する。
⑧WaterMix1をダブルクリックしてアクティブにし、
⑨[モデルチェック]ボタンをクリックする。
うまくいけば右の画面が表示される。



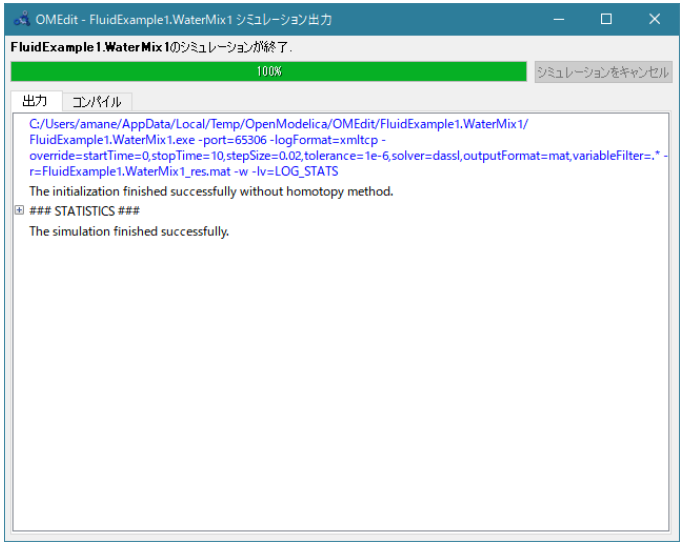
WaterMix1

⑩シミュレーション＞シミュレーションのセットアップ



- 開始時刻 0 [sec]
- 終了時刻 10 [sec]
- 計算回数 500

シミュレーションが成功すれば、
この画面が表示される。

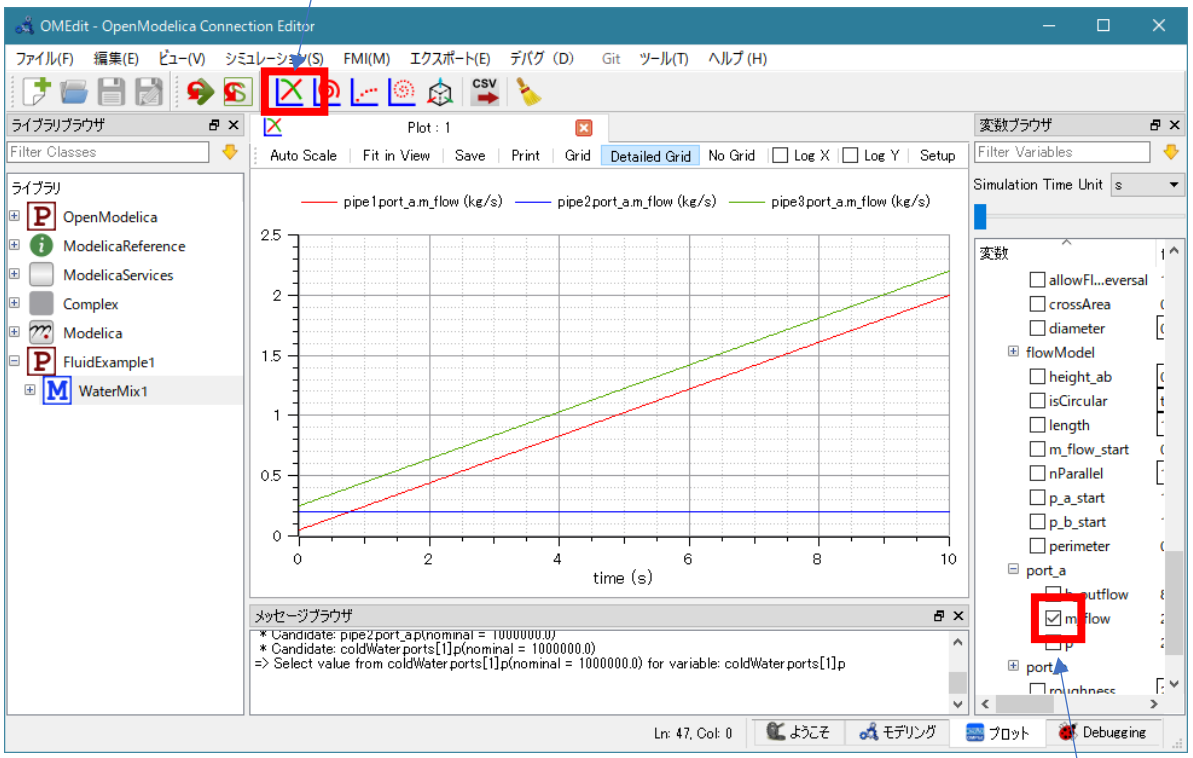


これをチェックすると計算条件がモデル内に保存される。
シミュレートをチェックすると[OK]ボタンでシミュレーションが
スタートする。

WaterMix1

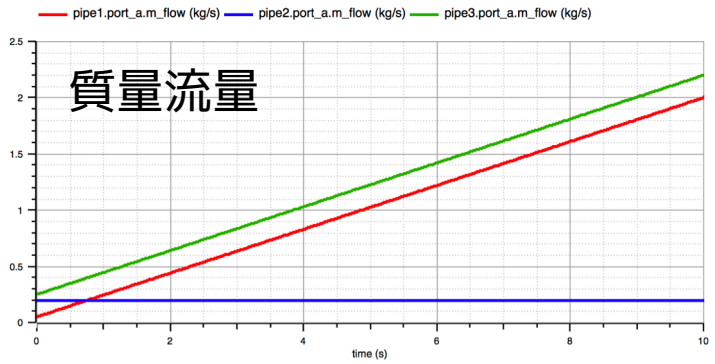
シミュレーション結果のプロット方法

① 新規プロットウインドウを生成する。



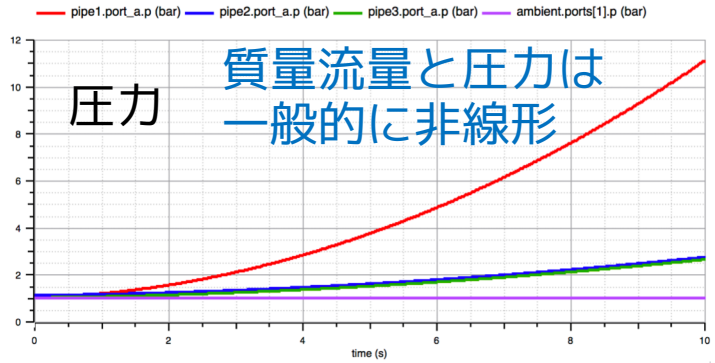
② プロットする変数をチェックする

WaterMix1



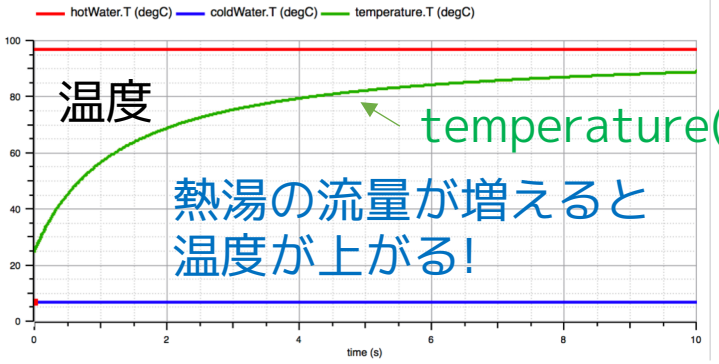
pipe3
pipe1

pipe2



pipe1入口

pipe2入口
pipe3入口
雰囲気



hotWater

coldWate
r

プロットする変数

- pipe1.port_a.m_flow
- pipe2.port_a.m_flow
- pipe3.port_a.m_flow

- pipe1.port_a.p
- pipe2.port_a.p
- pipe3.port_a.p
- ambient.ports[1].p

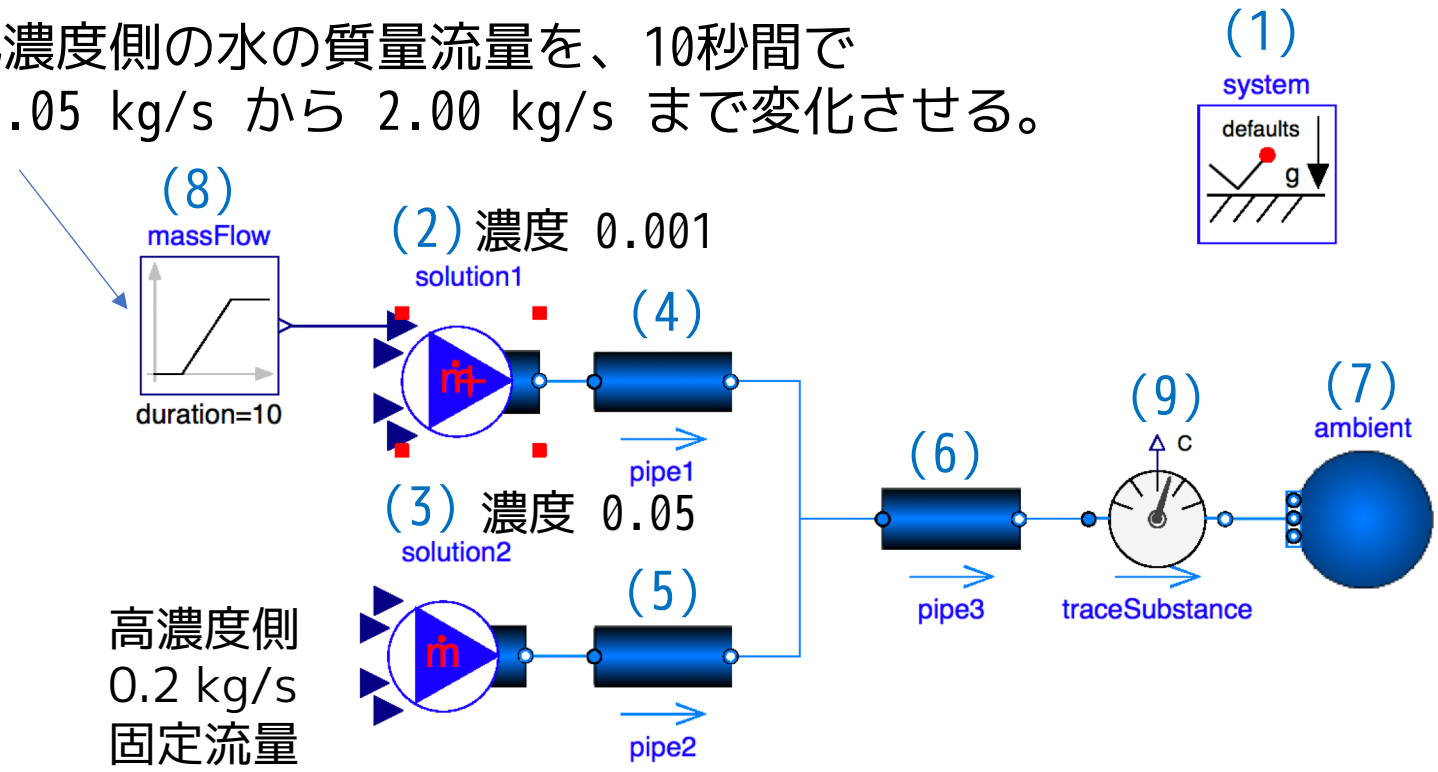
- hotWater.T
- coldWater.T
- temperature.T

TraceSubstanceMix1

塩素濃度の異なる水を混ぜる

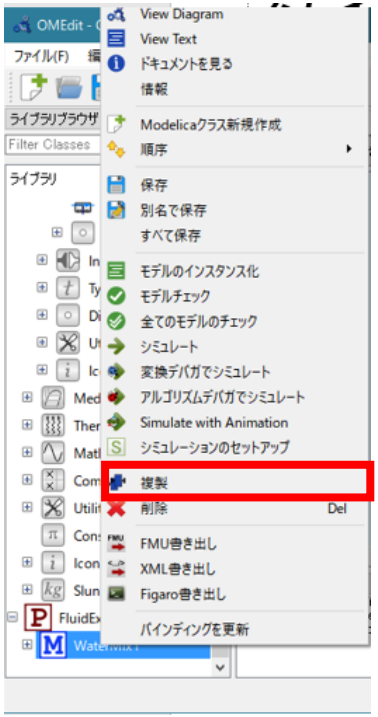
extraProperties (付加的物質)として塩素(chlorine)を定義し、
FluidPortで高濃度の水と低濃度の水を合流させる。

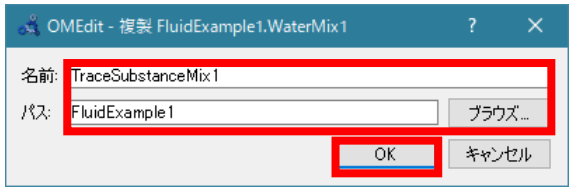
低濃度側の水の質量流量を、10秒間で
0.05 kg/s から 2.00 kg/s まで変化させる。



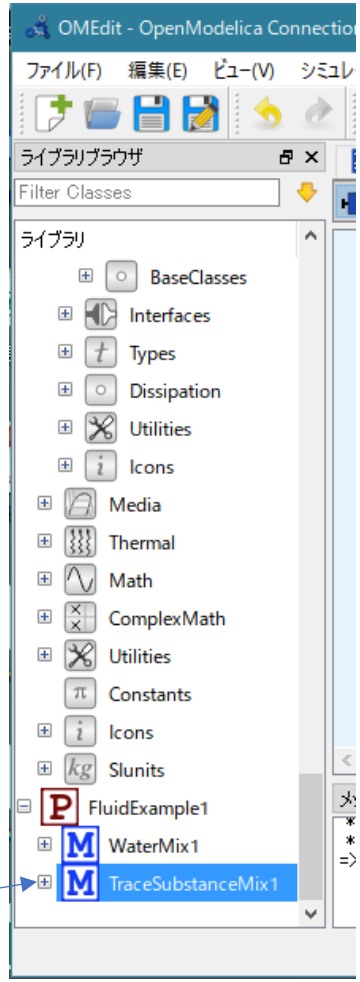
TraceSubstanceMix1

①ライブラリブラウザのWaterMix1を右クリックして複製を選択し、TraceSubstanceMix1 を作成する。





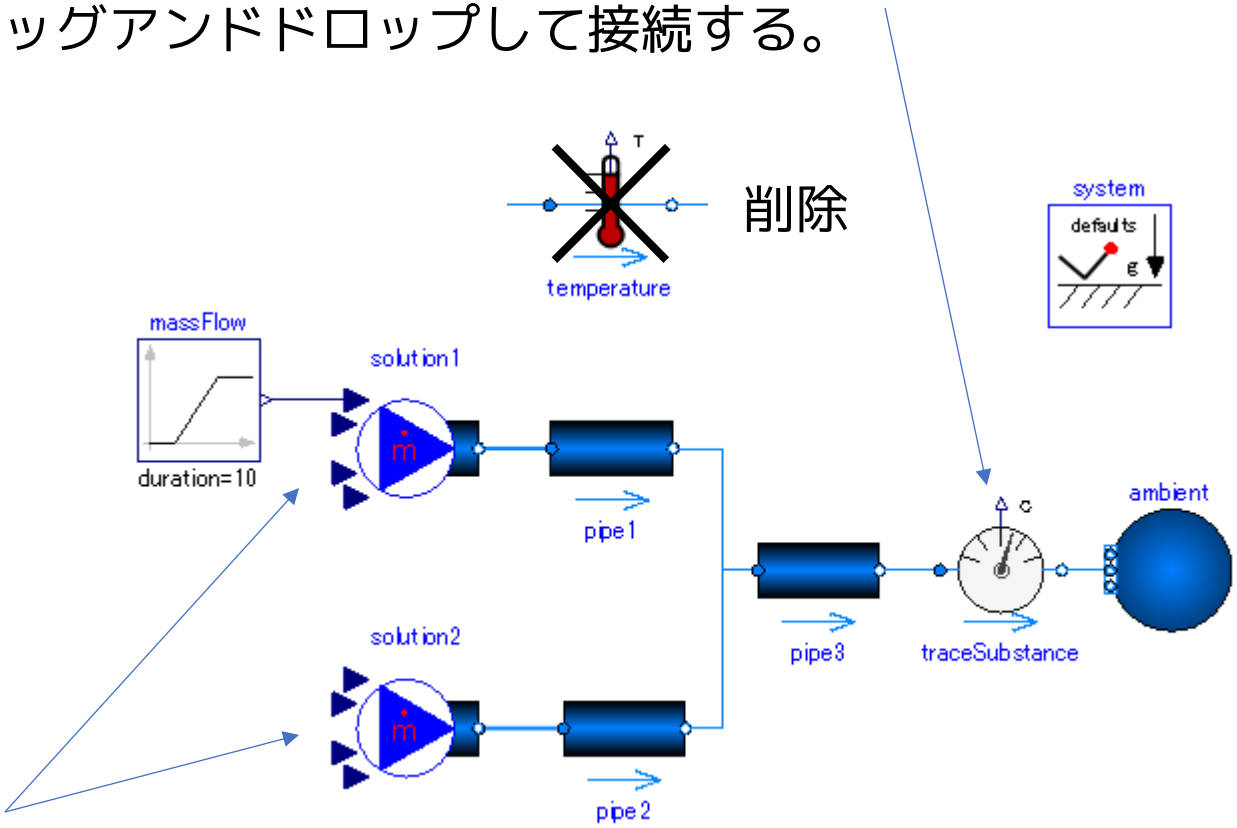
- 名前 TraceSubstanceMix1
- パス FluidExample1



TraceSubstanceMix1

TraceSubstanceMix1

②temperature を右クリックして[削除]を選択して削除し、代わりに
Fluid.Sensors.TraceSubstancesTwoPort
をドラッグアンドドロップして接続する。



③hotWaterとcoldWaterを右クリックして属性を選び、それぞれ solution1, solution2 にリネームする。

TraceSubstanceMix1

④コードを編集する。

濃度の設定

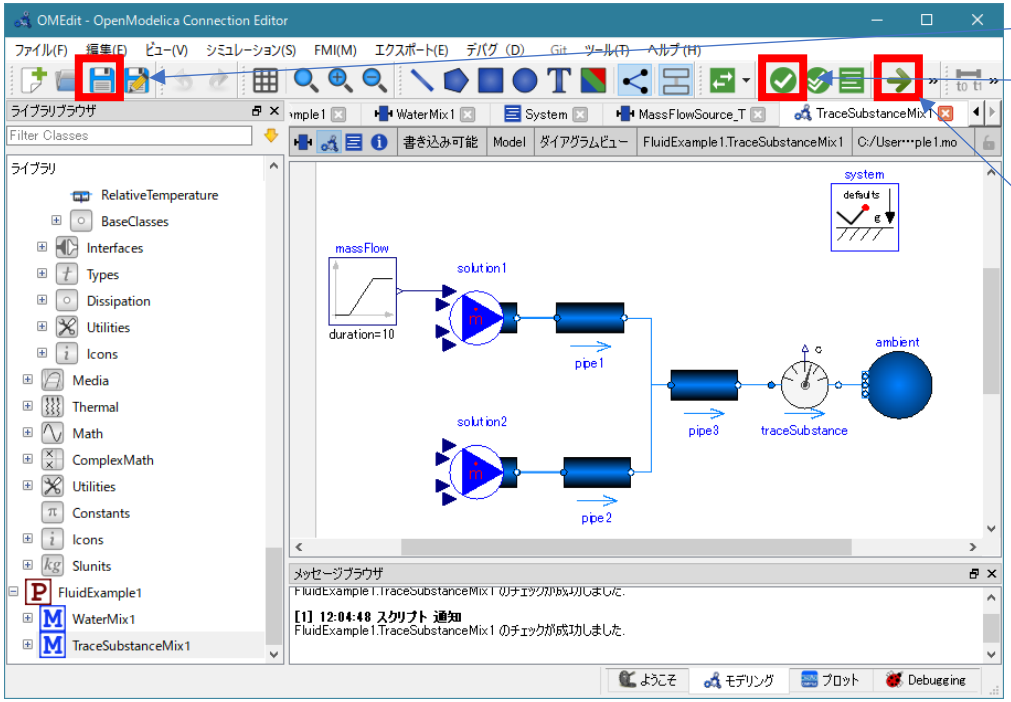
extraPropertiesの設定

```
model TraceSubstanceMix1
  replaceable package Medium = Media.Water.StandardWater(extraPropertiesNames={"chlorine"},
    C_nominal={0.005});
  inner Modelica.Fluid.System system annotation( ...);                                    (1)
  Modelica.Fluid.Sources.MassFlowSource_T solution1(redeclare package Medium = Medium,    (2)
    C = {0.001}, T = 293.15, nPorts = 1, use_m_flow_in = true) annotation( ...);
  Modelica.Fluid.Sources.MassFlowSource_T solution2(redeclare package Medium = Medium,    (3)
    C = {0.05}, T = 293.15, m_flow = 0.2, nPorts = 1) annotation( ...);
  Modelica.Fluid.Pipes.StaticPipe pipe1(redeclare package Medium = Medium,                (4)
    diameter = 0.01, length = 1) annotation( ...);
  Modelica.Fluid.Pipes.StaticPipe pipe2(redeclare package Medium = Medium,                (5)
    diameter = 0.01, length = 1) annotation( ...);
  Modelica.Fluid.Pipes.StaticPipe pipe3(redeclare package Medium = Medium,                (6)
    diameter = 0.01414, length = 1) annotation( ...);                                     (7)
  Modelica.Fluid.Sources.FixedBoundary ambient(redeclare package Medium = Medium, nPorts = 1)
    annotation( ...);
  Modelica.Blocks.Sources.Ramp massFlow(duration = 10, height = 1.95, offset = 0.05,      (8)
    startTime = 0) annotation( ...);
  Modelica.Fluid.Sensors.TraceSubstancesTwoPort traceSubstance(                           (9)
    redeclare package Medium = Medium, substanceName = "chlorine")  annotation( ...);
```

濃度センサで表示する物質の設定

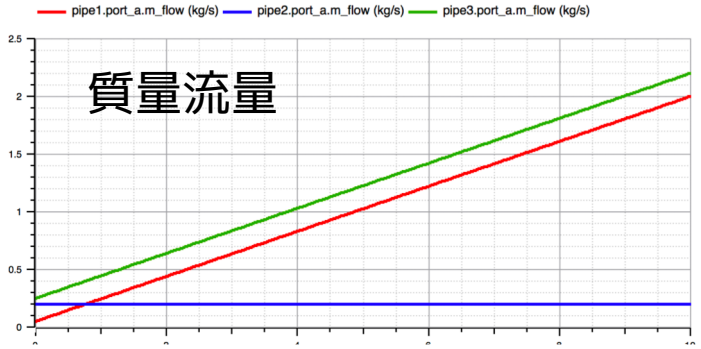
TraceSubstanceMix1

```
 equation
    connect(solution1.ports[1], pipe1.port_a) annotation( ...);
    connect(massFlow.y, solution1.m_flow_in) annotation( ...);
    connect(traceSubstance.port_a, pipe3.port_b) annotation( ...);
    connect(ambient.ports[1], traceSubstance.port_b) annotation( ...);
    connect(solution2.ports[1], pipe2.port_a) annotation( ...);
    connect(pipe2.port_b, pipe3.port_a) annotation( ...);
    connect(pipe1.port_b, pipe3.port_a) annotation( ...);
end TraceSubstanceMix1;
```



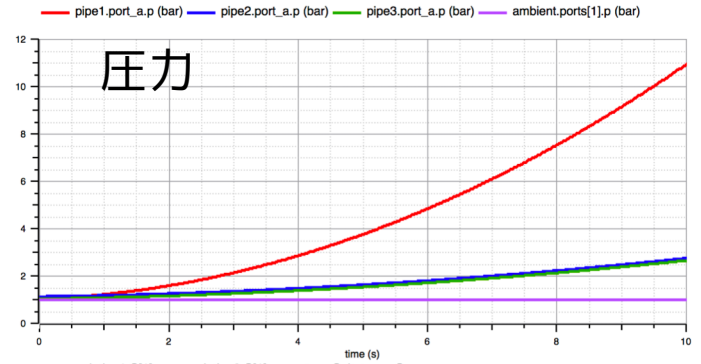
④保存する。
⑤チェックモデルを実行する。
⑥シミュレーションを実行する。

TraceSubstanceMix1　シミュレーション結果



pipe3
pipe1

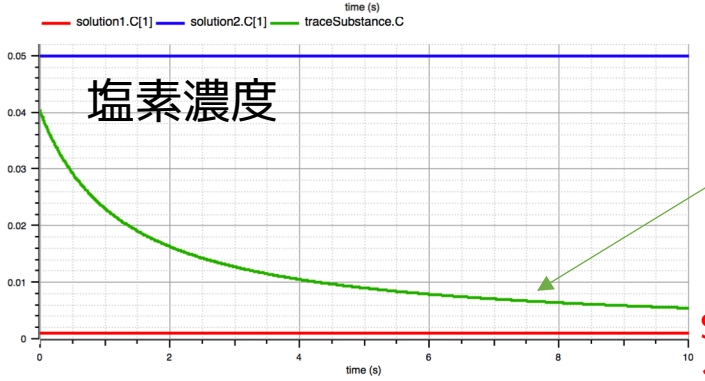
pipe2



pipe1入口

pipe2入口
pipe3入口

雰囲気



solutions 2

traceSubstance (濃度センサ)

solutions 1

プロットする変数

- pipe1.port_a.m_flow
- pipe2.port_a.m_flow
- pipe3.port_a.m_flow

- pipe1.port_a.p
- pipe2.port_a.p
- pipe3.port_a.p
- ambient.ports[1].p

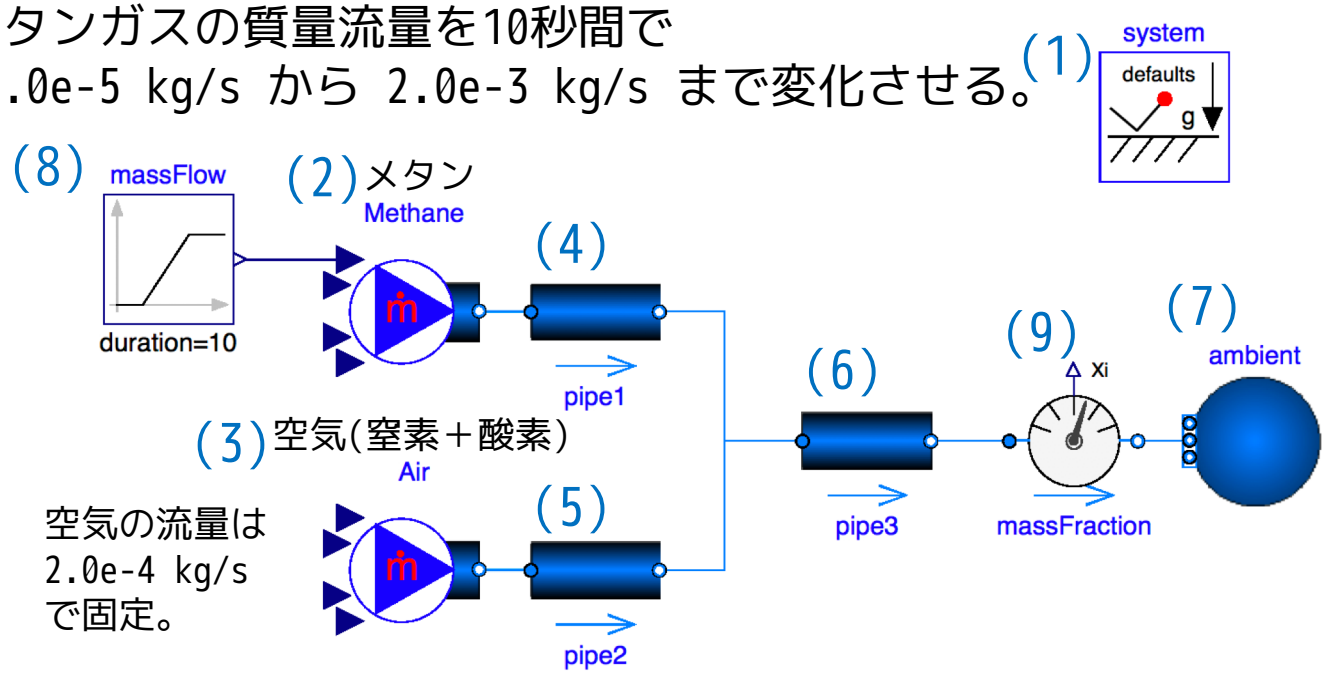
- solution1.C[1]
- solution2.C[1]
- traceSubstance.C

低濃度の水の流量が増えると濃度が小さくなる!

GasMix1
メタンガスと空気を混ぜる

メタンガスと空気の混合状態が表現できる物性モデル(Mediumパッケージ)を作成し、メタンガスと空気が FluidPort で混合するモデルを作成する。

メタンガスの質量流量を10秒間で
5.0e-5 kg/s から 2.0e-3 kg/s まで変化させる。



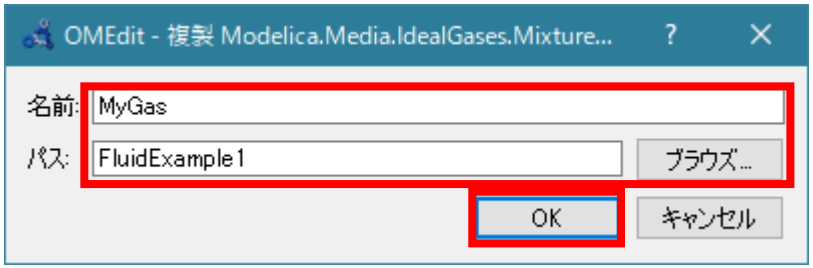
GasMix1

手順
1. メタンガスと空気の混合物を表す MyGas を作る。
2. 質量分率センサを修正する。
3. システムモデルを作る。

GasMix1

1. メタンガスと空気の混合物を表す MyGas を作る

① Modelica.Media.IdealGases.MixtureGases.CombustionAirを右クリックして複製を選び、MyGasという名前でパス FluidExample1 にコピーする。



- 名前 MyGas
- パス FluidExample1

② MyGasのスコープの問題を解決するために FluidExample1 の最初の方に次のような import 文を付け加える。

FluidExample1 の最初の部分

```
package FluidExample1
  import Modelica.Media;
  import Modelica.Media.IdealGases.Common;

  model WaterMix1
  ...
```

GasMix1

③ MyGas を次のように編集して、メタンガスと空気の混合ガスを表すように変更する。

MyGas (メタンガスと空気の混合気体)

```
package MyGas "Nitrogen and Air"
   extends Common.MixtureGasNasa(mediumName = "MyGas",
   data = {Common.SingleGasesData.CH4, Common.SingleGasesData.N2, Common.SingleGasesData.O2},
       fluidConstants = {Common.FluidData.CH4, Common.FluidData.N2, Common.FluidData.O2},
       substanceNames = {"Methane", "Nitrogen", "Oxygen"},
       reference_X = {0.5, 0.4, 0.1});
    annotation( ...);
end MyGas;
```

和が1.0になるようにする。

GasMix1

２. 質量分率センサの修正

Fluid.Sensors.MassFractionsTwoPort (質量分率センサ)はバグがあるので、FluidExample1にコピーして MassFractionsTwoPort1 を作成し、修正して使用する。

```
 model MassFractionsTwoPort1 "Ideal two port sensor for mass fraction"
    extends Modelica.Fluid.Sensors.BaseClasses.PartialFlowSensor;
    extends Modelica.Icons.RotationalSensor;
    Modelica.Blocks.Interfaces.RealOutput Xi "Mass fraction in port medium"
annotation( ...);
    parameter String substanceName = "water" "Name of mass fraction";
  protected
    parameter Integer ind(fixed = false) "Index of species in vector of independent
mass fractions";
  initial algorithm
    ind := -1;
    for i in 1:Medium.nX loop
      if Modelica.Utilities.Strings.isEqual(Medium.substanceNames[i], substanceName)
then
        ind := i;
      end if;
以下略
```

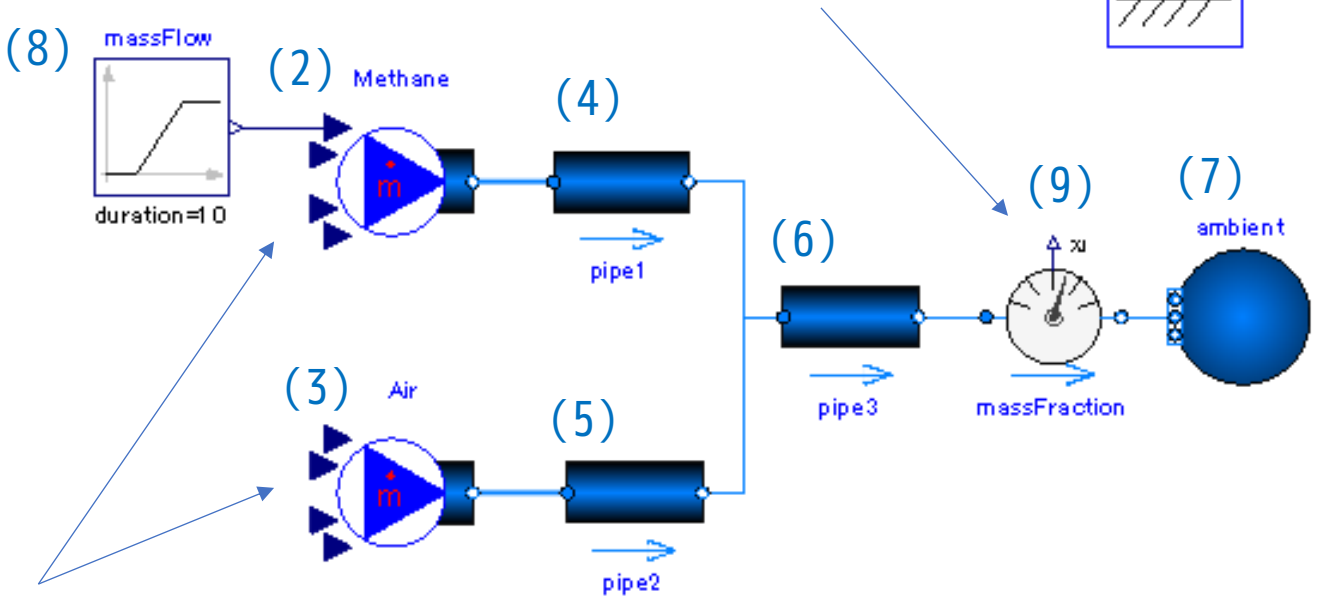
Medium.nC を Medium.nX に修正する。

GasMix1

3. システムモデルの作成

① WaterMixを複製して、GasMix1という名前のモデルを作る。

② temperature を削除し、代わりに MassFractionTwoPort1 を配置して接続する。



③ MethaneとAirにリネームする。

GasMix1

④ GasMix1のコードを編集する。

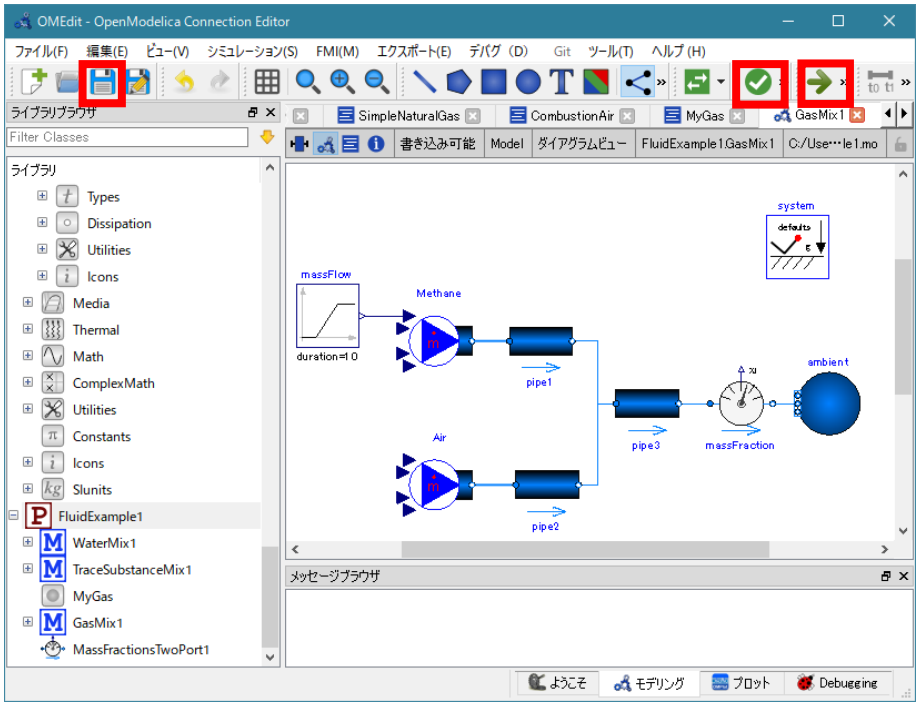
空気側は、成分が窒素と酸素だけになるように設定する。

メタン側は、成分がメタンだけとなるように設定する。

```
model GasMix1
  replaceable package Medium = MyGas;
  inner Modelica.Fluid.System system annotation( ...);                                    (1)
  Modelica.Fluid.Sources.MassFlowSource_T Methane(redeclare package Medium = Medium,      (2)
    T = 293.15, X = {1.0, 0.0, 0.0}, nPorts = 1, use_m_flow_in = true) annotation( ...);
  Modelica.Fluid.Sources.MassFlowSource_T Air(redeclare package Medium = Medium,          (3)
    X = {0.0, 0.8, 0.2}, T = 293.15, m_flow = 2.0e-4, nPorts = 1) annotation( ...);
  Modelica.Fluid.Pipes.StaticPipe pipe1(redeclare package Medium = Medium,                (4)
    diameter = 0.01, length = 1) annotation( ...);
  Modelica.Fluid.Pipes.StaticPipe pipe2(redeclare package Medium = Medium,                (5)
    diameter = 0.01, length = 1) annotation( ...);
  Modelica.Fluid.Pipes.StaticPipe pipe3(redeclare package Medium = Medium,                (6)
    diameter = 0.01414, length = 1) annotation( ...);                                     (7)
  Modelica.Fluid.Sources.FixedBoundary ambient(redeclare package Medium = Medium, nPorts = 1)
    annotation( ...);
  Modelica.Blocks.Sources.Ramp massFlow(duration = 10, height = 1.95e-3, offset = 5.0e-5,  (8)
    startTime = 0) annotation( ...);
  FluidExample3.MassFractionsTwoPort1 massFraction(redeclare package Medium = Medium,     (9)
    substanceName = Medium.substanceNames[1]) annotation( ...);
```

GasMix1

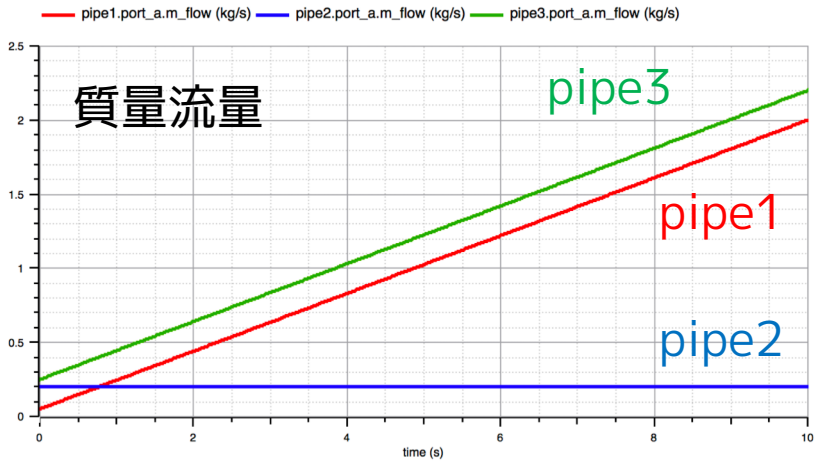
```
equation
   connect(Air.ports[1], pipe2.port_a) annotation( ...);
   connect(massFraction.port_b, ambient.ports[1]) annotation( ...);
   connect(massFraction.port_a, pipe3.port_b) annotation( ...);
   connect(Methane.ports[1], pipe1.port_a) annotation( ...);
   connect(massFlow.y, Methane.m_flow_in) annotation( ...);
   connect(pipe2.port_b, pipe3.port_a) annotation( ...);
   connect(pipe1.port_b, pipe3.port_a) annotation( ...);
 end GasMix1;
```

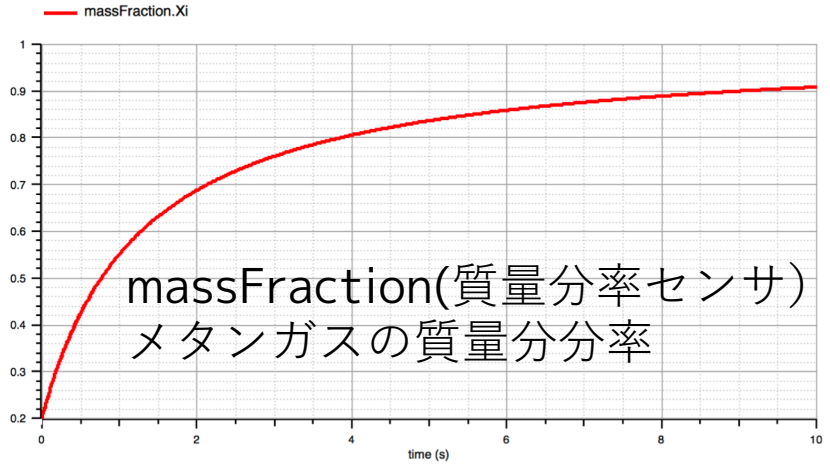


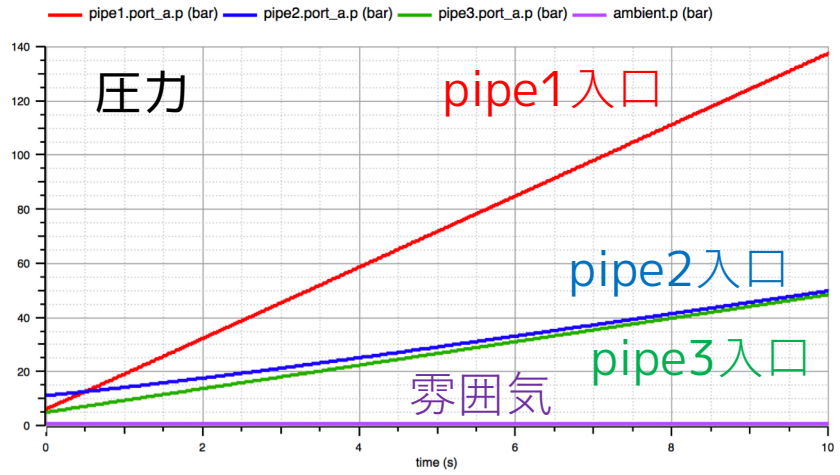
⑤保存する。
⑥チェックモデルを実行する。
⑦シミュレーションを実行する。

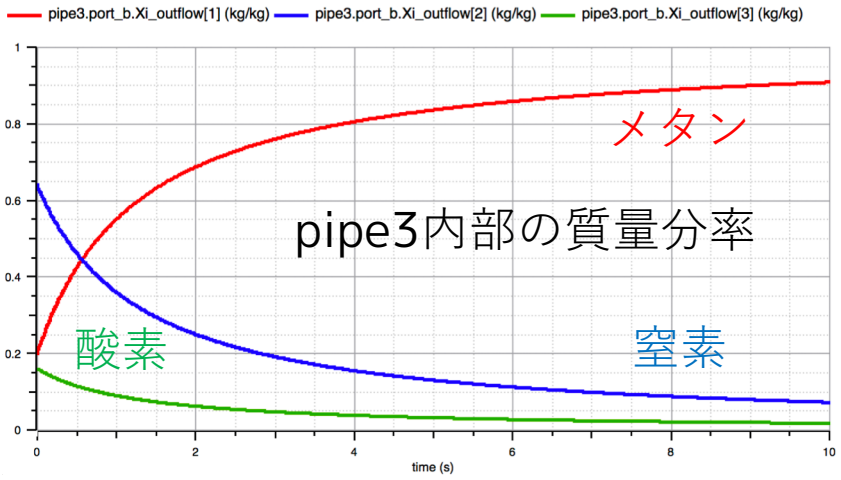
GasMix1

メタンの流量が増えるとメタンの質量分率が増大する!





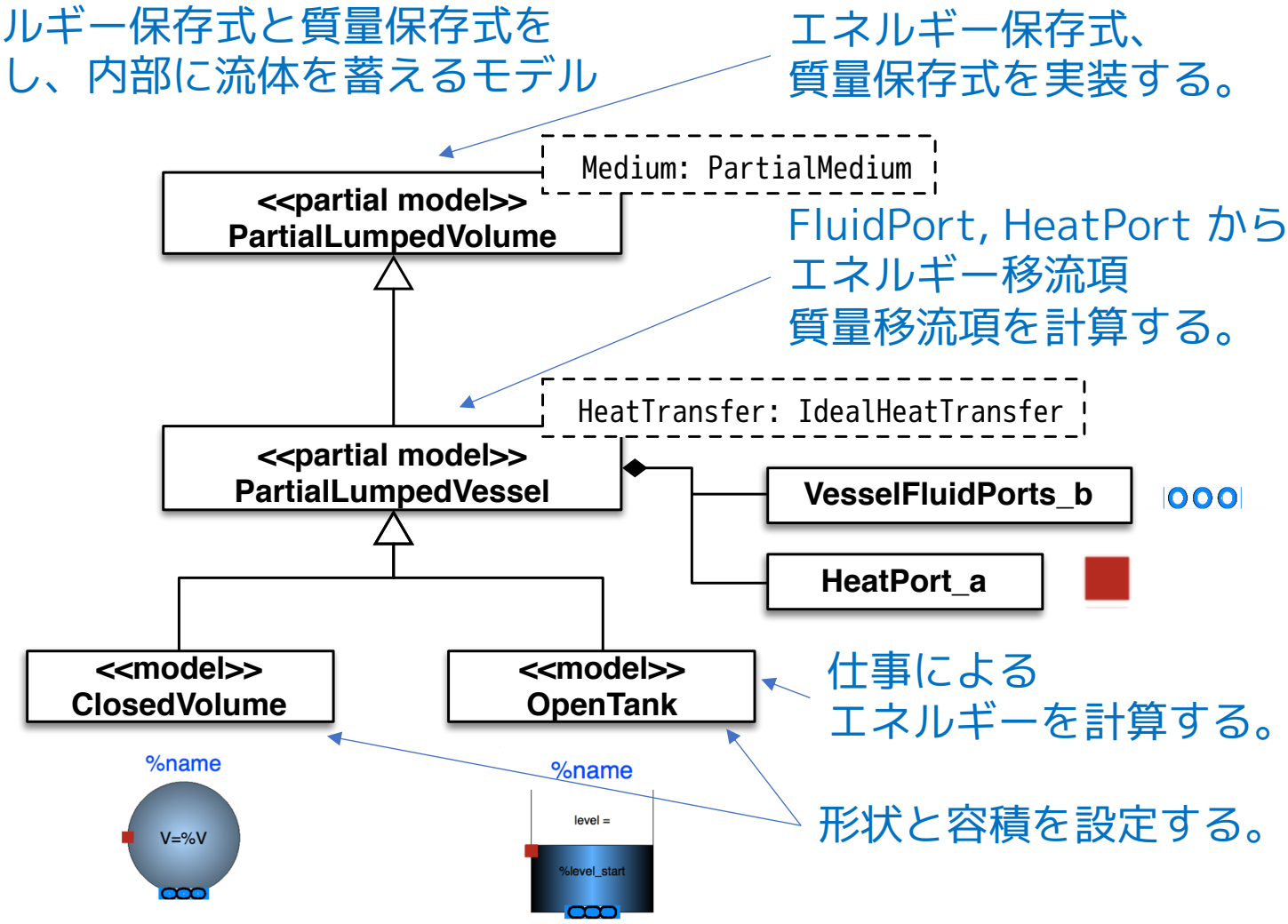




FluidExample2

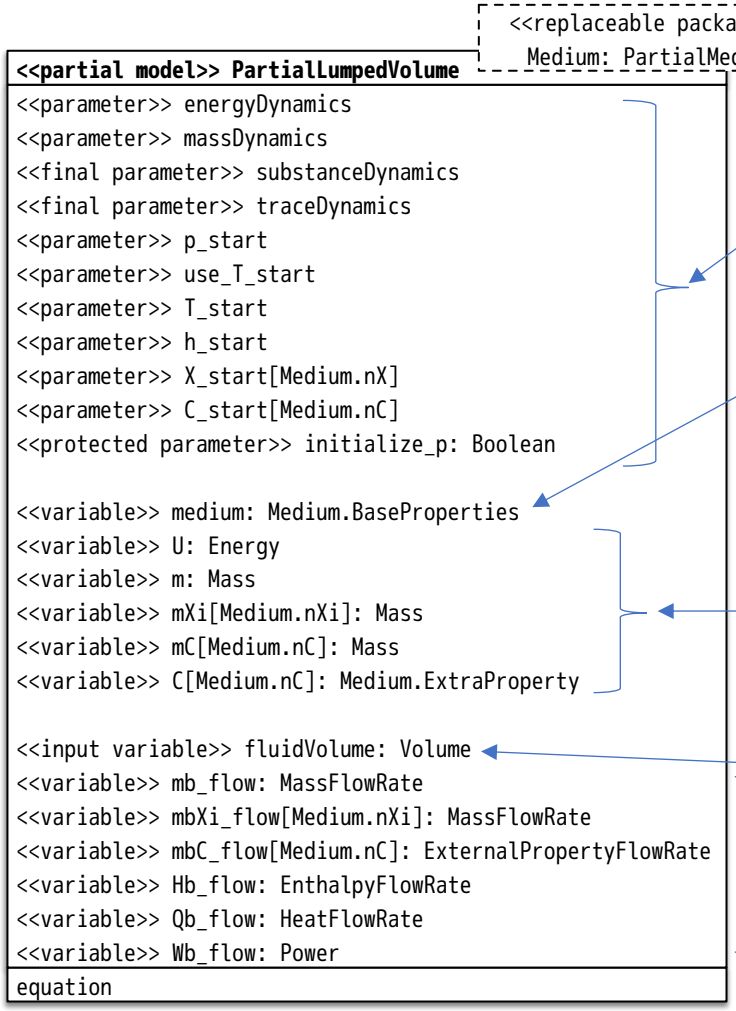
volume モデル

エネルギー保存式と質量保存式を
実装し、内部に流体を蓄えるモデル



volume モデル

PartialLumpedVolume volumeモデルのベースモデル



定常/非定常
初期条件を設定する
パラメータ

medium: BasePropertiesモデル

容器内の状態変数
U: エネルギー
m: 質量
$\boldsymbol{mX_i}$: 成分の質量
$\boldsymbol{mC}$: 微小物質の質量

$fluidVolume$: 入力変数の容積

継承先のモデルで方程式を定義する
エンタルピ移流項、質量移流項
伝熱項、仕事の項

volume モデル

PartialLumpedVolume の方程式(1)

```
equation
  assert(not (energyDynamics<>Dynamics.SteadyState and
massDynamics==Dynamics.SteadyState) or Medium.singleState,
         "Bad combination of dynamics options and Medium not conserving mass if
fluidVolume is fixed.");

  // Total quantities
  m = fluidVolume*medium.d;
  mXi = m*medium.Xi;
  U = m*medium.u;
  mC = m*C;

  // Energy and mass balances
  if energyDynamics == Dynamics.SteadyState then
    0 = Hb_flow + Qb_flow + Wb_flow;
  else
    der(U) = Hb_flow + Qb_flow + Wb_flow;
  end if;

  if massDynamics == Dynamics.SteadyState then
    0 = mb_flow;
  else
    der(m) = mb_flow;
  end if;
```

energyDynamics や massDynamicsなどの設定で方程式や初期条件を切り替える。

$$m = fluidVolume \cdot \rho \quad \text{容器内の全質量}$$
$$\boldsymbol{mX_i} = m\,\boldsymbol{X_i} \quad \text{成分毎の全質量}$$
$$U = m\,u \quad \text{全内部エネルギー}$$
$$\boldsymbol{mC} = m\,\boldsymbol{C} \quad \text{微小物質の全質量}$$

エネルギー保存式

$$\begin{cases} 0 = Hb_{fow} + Qb_{flow} + Wb_{flow} & \text{定常} \\ \dfrac{dU}{dt} = Hb_{flow} + Qb_{flow} + Wb_{flow} & \text{非定常} \end{cases}$$

質量保存式

$$\begin{cases} 0 = mb_{flow} & \text{定常} \\ \dfrac{dm}{dt} = mb_{flow} & \text{非定常} \end{cases}$$

volume モデル

PartialLumpedVolume の方程式(2)

```
if substanceDynamics == Dynamics.SteadyState then
  zeros(Medium.nXi) = mbXi_flow;
else
  der(mXi) = mbXi_flow;
end if;

if traceDynamics == Dynamics.SteadyState then
  zeros(Medium.nC)  = mbC_flow;
else
  der(mC_scaled) = mbC_flow./Medium.C_nominal;
end if;
mC = mC_scaled.*Medium.C_nominal;
```

成分物質の質量保存式

$$\begin{cases} \mathbf{0} = \boldsymbol{mbXi}_{flow} & \text{定常} \\ \dfrac{d\,\boldsymbol{mXi}}{dt} = \boldsymbol{mbXi}_{flow} & \text{非定常} \end{cases}$$

正規化した微小物質の質量保存式

$$\begin{cases} \mathbf{0} = \boldsymbol{mbC}_{flow} & \text{定常} \\ \dfrac{d\,\boldsymbol{mC}_{scaled}}{dt} = \dfrac{\boldsymbol{mbC}_{flow}}{Medium.C_{nominal}} & \text{非定常} \end{cases}$$

微小物質の濃度正規化

$$\boldsymbol{mC} = \boldsymbol{mC}_{scaled} \cdot Medium.C_{nominal}$$

volume モデル

PartialLumpedVolume の初期化(1) (一部コメント文を省略しています)

```
initial equation
  // initialization of balances
  if energyDynamics == Dynamics.FixedInitial then
    /* ... */
    if Medium.ThermoStates == IndependentVariables.ph or
       Medium.ThermoStates == IndependentVariables.phX then
       medium.h = h_start;
    else
       medium.T = T_start;
    end if;
  elseif energyDynamics == Dynamics.SteadyStateInitial then
    /* ... */
    if Medium.ThermoStates == IndependentVariables.ph or
       Medium.ThermoStates == IndependentVariables.phX then
       der(medium.h) = 0;
    else
       der(medium.T) = 0;
    end if;
  end if;
```

$$h = h_{start} \text{ or } T = T_{start}$$

$$\frac{dh}{dt} = 0 \text{ or } \frac{\mathrm{d}T}{dt} = 0$$

volume モデル

PartialLumpedVolume の初期化(2)

```
 if massDynamics == Dynamics.FixedInitial then
    if initialize_p then
      medium.p = p_start;
    end if;
  elseif massDynamics == Dynamics.SteadyStateInitial then
    if initialize_p then
      der(medium.p) = 0;
    end if;
  end if;

  if substanceDynamics == Dynamics.FixedInitial then
    medium.Xi = X_start[1:Medium.nXi];
  elseif substanceDynamics == Dynamics.SteadyStateInitial then
    der(medium.Xi) = zeros(Medium.nXi);
  end if;

  if traceDynamics == Dynamics.FixedInitial then
    mC_scaled = m*C_start[1:Medium.nC]./Medium.C_nominal;
  elseif traceDynamics == Dynamics.SteadyStateInitial then
    der(mC_scaled) = zeros(Medium.nC);
  end if;

  annotation ( ...);
end PartialLumpedVolume;
```

$$p = p_{start}$$

$$\frac{dp}{dt} = 0$$

$$\boldsymbol{X_i = X_{start}}$$

$$\frac{\boldsymbol{dX_i}}{\boldsymbol{dt}} = 0$$

$$\boldsymbol{mC_{scaled}} = m\,\frac{\boldsymbol{C_{start}}}{C_{nominal}}$$

$$\frac{d\,\boldsymbol{mC_{scaled}}}{dt} = \boldsymbol{0}$$

volume モデル

PartialLumpedVessel

i: FluidPort の番号
j: 成分物質の番号
k: 付加的物質の番号

質量移流項

$$mb_{flow} = \sum_i m\_flow[i] \ , i = 1, \dots, nPorts$$

$$mbXi_{flow}[\mathrm{j}] = \sum_i \left\{ m_{flow[i]} \cdot actualStream\left(Xi_{outflow}[i,j]\right) \right\}, j = 1, \dots, nX_i$$

$$mbC_{flow}[\mathrm{k}] = \sum_i \left\{ m_{flow[i]} \cdot actualStream\left(C_{outflow}[i,k]\right) \right\} \ , k = 1, \dots, nC$$

エネルギー移流項

$$Hb_{flow} = \sum_i \left\{ m_{flow[i]} \cdot \ actualStream\left(h_{outflow_i}\right) \right\}$$

FluidPort からの
エンタルピ移流

$$+ \sum_i \left\{ m_{flow}[i] \cdot (0.5\, v[i]^2 + gh[i]) \right\}$$

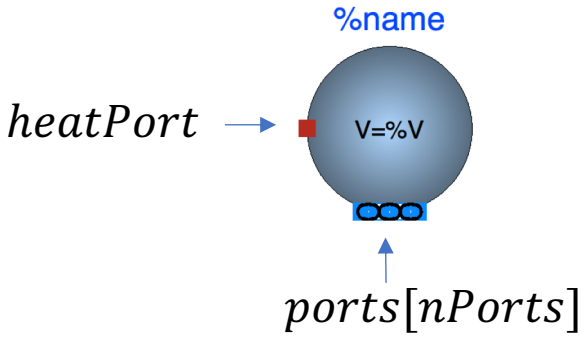
運動エネルギーと
ポテンシャルエネルギー

$$Qb_{flow} = heatTransfer.Q\_flows[1]$$

HeatPort からの伝熱量

volume モデル

ClosedVolume 固定サイズで出入口のある容器



形状パラメータ

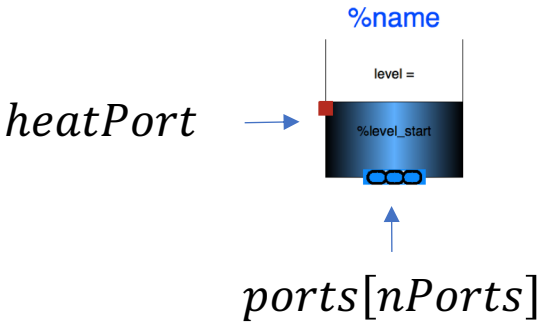
- V [m3]: 容積

容積　$V = \frac{4}{3}\pi r^3$

伝熱面積　$A = 4\pi r^2$

fluidVolume = V

OpenTank 出入口のあるシンプルなタンク



形状パラメータ

- $height$ [m]: タンクの高さ
- $crossArea$ [m2]: タンクの断面積
- $level\_start = 0.5 * height$ [m]: 液位の初期値
- $p_{ambient} = system.p_{ambient}$ [Pa]:雰囲気圧力
- $T_{ambient} = system.T_{ambient}$ [K]:　雰囲気温度

fluidVolume = V
V = crossArea*level

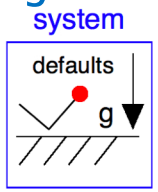
- 液面は固定圧力$p_{ambient}$の雰囲気に解放されている。
- 上面から雰囲気への流入出は無く、液位がタンク高さを超えるとシミュレーションが停止する。

HotRoom1

部屋の冷たい空気を温かい空気に入れ替える

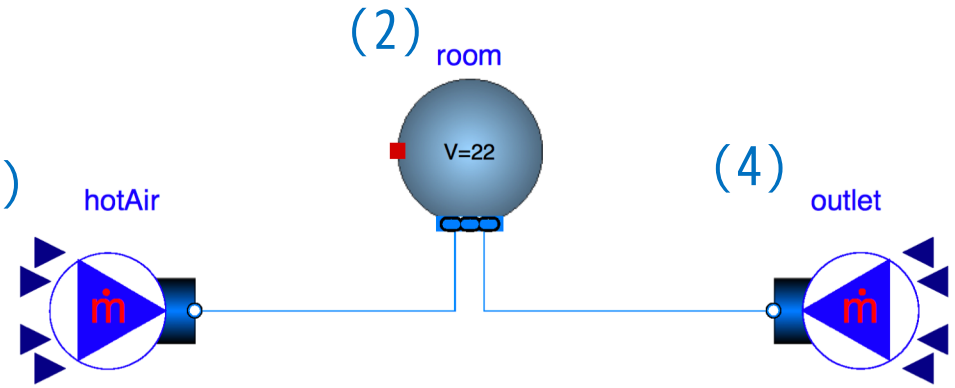
固定容積の部屋が冷たい空気(6.85 ℃)で満たされている。上流側から温風(40 ℃, 0.1 kg/s)を吹き出す。部屋は熱の移流によって温度が徐々に上昇する。下流側では 0.103 kg/s で空気を吸い込む。

部屋 22 m3 (4畳半ぐらい)
初期温度 280 K (6.85℃)



(1)

温風
323.15 K
(40.0℃)
0.1 kg/s

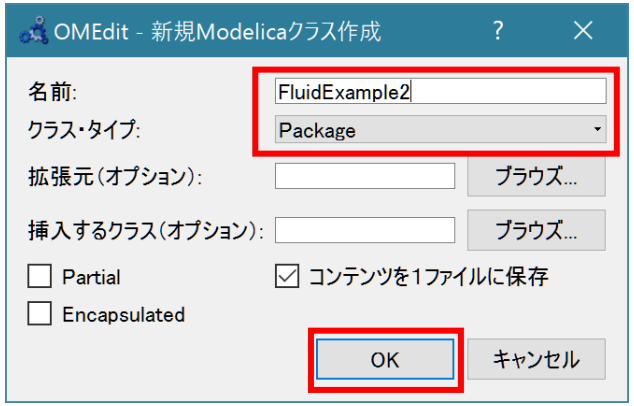


下流の出口
-0.103 kg/s

室内の空気の熱力学的状態の初期値を規定するため、energyDynamicsとmassDynamicsをFixedInitialにし、T_start=280 [K]とP_start=101325 [Pa] 設定する。
このモデルは非圧縮性流体では使えない。

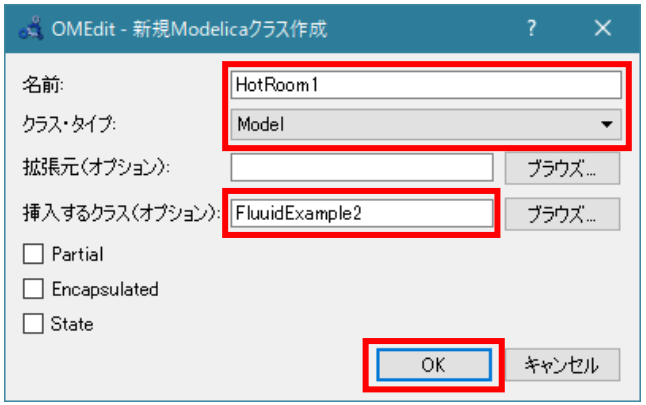
HotRoom1

① ファイル >Modelicaクラス新規作成で package を作る。



名前: FluidExample2
クラス・タイプ: Package

②ライブラリブラウザの FluidExample2を右クリックして「Modelicaクラス新規作成」を選び、model を作る。



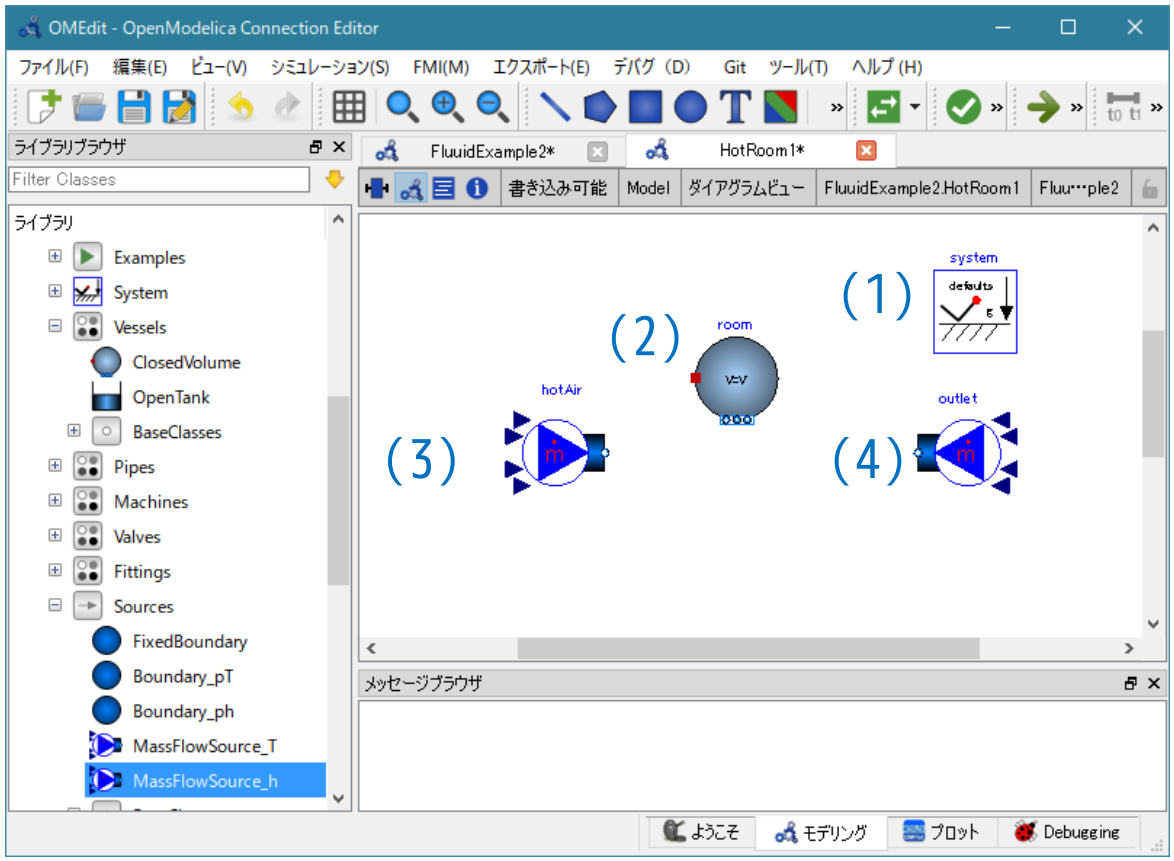
名前: HotRoom1
クラス・タイプ: Model
挿入するクラス: FluidExample2

HotRoom1

③ FluieExample2 のテキストビューに import 文を加える。

```
package FluidExample2
  import Modelica.Media;
```

④ VolumeTest1のダイアグラムビューにコンポーネントを配置する。

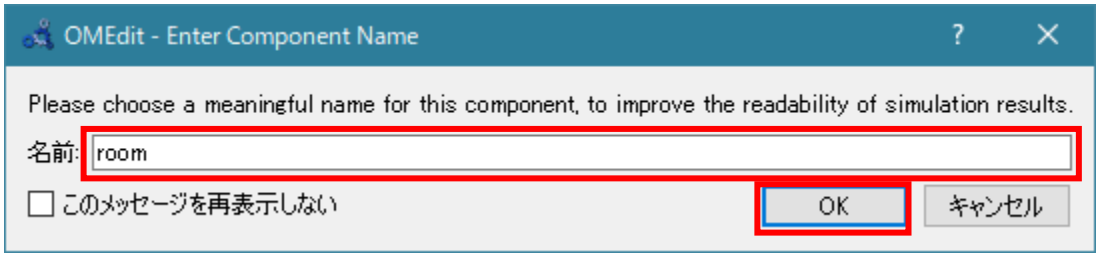


HotRoom1

配置するコンポーネントの名前とクラス

(1)system:　　Fluid.System
(2)room:　　Fluid.Vessels.ClosedVolume
(3)hotAir:　　Fluid.Sources.MassFlowSource_T
(4)outlet:　　Fluid.Sources.MassFlowSource_T

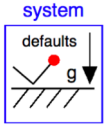
コンポーネントをドロップするときに現れるダイアログで名前をつける。



HotRoom1

⑤コンポーネントを右クリックして[パラメータ]を選択して入力する。

(1) Fluid.System

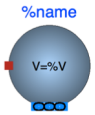


流体システムモデルで
必ず配置する。

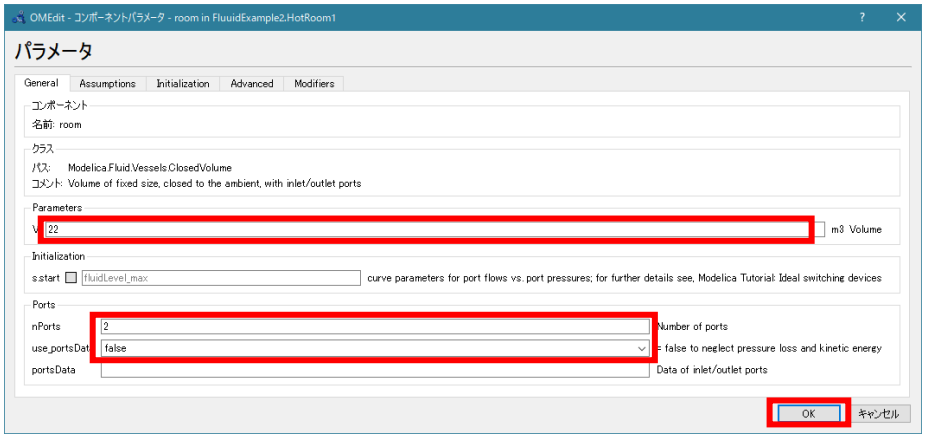
(1) system

- デフォルトのまま

(2) Fluid.Vessels.ClosedVolume



固定体積の容器



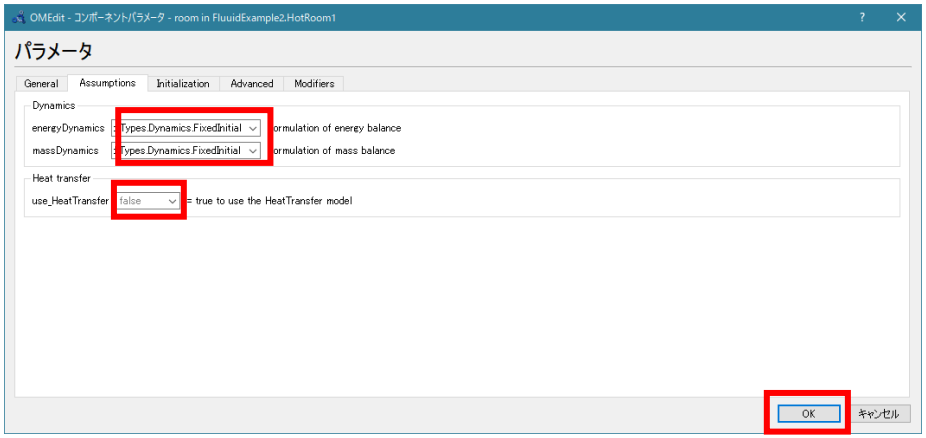
(2) room

[General]

- V = 22 [m3]
- nPorts = 2
- use_portsData = false

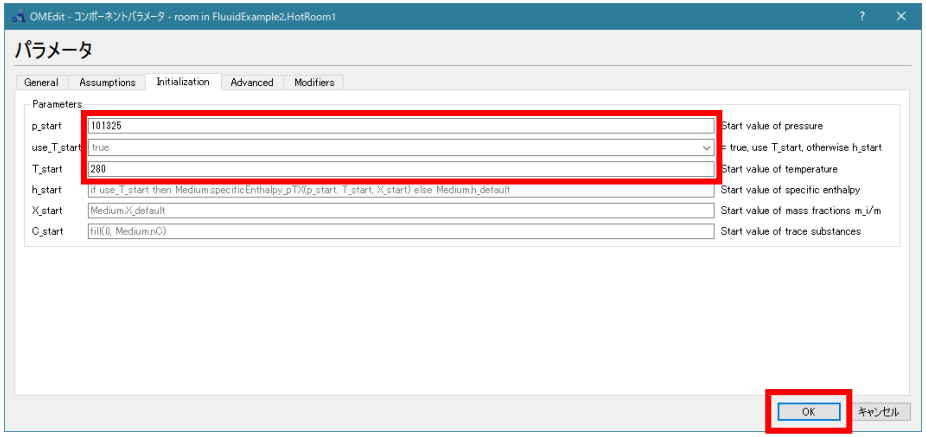
- 出入口を表す FluidPort を2個設けるため nPorts = 2 とする。
- use_portsData = false とする。これにより、容器内部の圧力とFluidPort の圧力が等しくなる。

HotRoom1



[Assumptions]

- energyDynamics = FixedInitial
- massDynamics = FixedInitial
- use_HeatTransfer=false

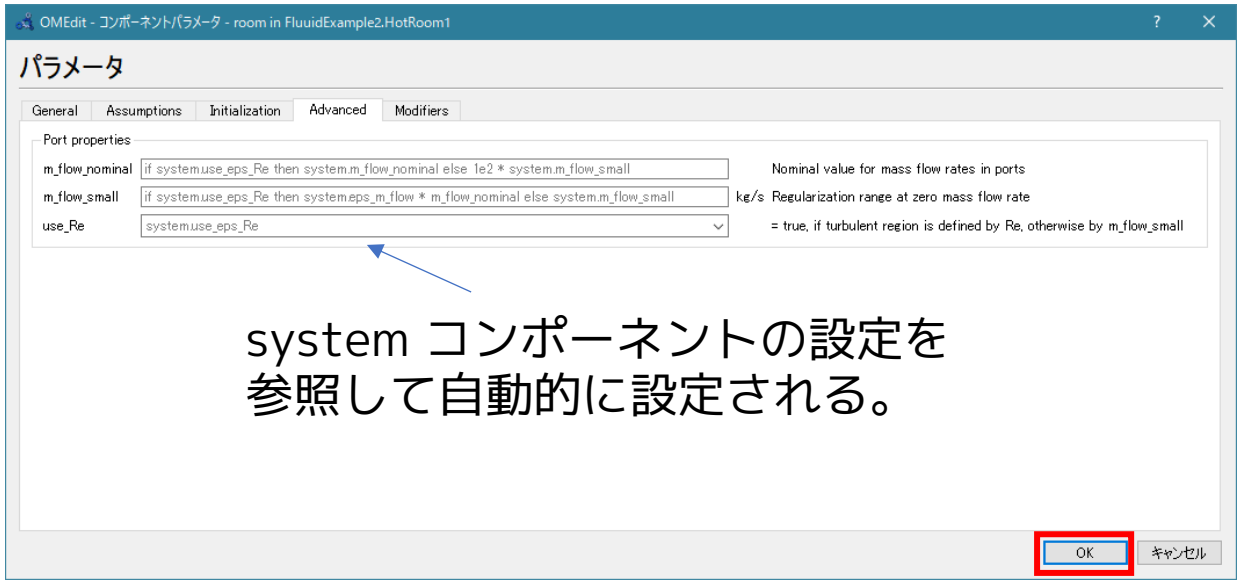


[Initialization]

- p_start = 101325 [Pa]
- use_T_start = true
- T_start = 280 [K]

energyDynamics と massDynamics を FixedInitial にして、室内の空気の熱力学的初期状態を圧力と温度またはエンタルピーで規定する。

HotRoom1



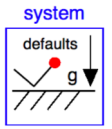
[Advanced]
- 設定しない

- Re(レイノルズ数)が小さく乱流領域から外れた場合は、**regularization (流れの適正化)** を行う。
- さらに流量が小さい範囲は **zero flow** として扱う。

このタブのパラメータは regularization を行う範囲や zero flow として扱う範囲を規定する。

HotRoom1

system コンポーネントの設定（デフォルト値）



多くのコンポーネントで Dynamics や Regularization などのパラメータのデフォルト値は、system コンポーネントを参照して設定されるようになっている。

Dynamics

$allowFlowReversal = true$
$energyDynamics = DynamicFreeInitial$
$massDynamics = energyDynamics$
$substanceDynamics = massDynamics$
$traceDynamics = massDynamics$
$momentumDynamics = SteadyState$
$m\_flow\_start = 0,$
$p_{start} = p_{ambient},$
$T_{start} = T_{ambient}$

General

$p_{ambient} = 101325$
$T_{ambient} = 293.15$
$g = Modelida.Constants.g\_n$

Regularization

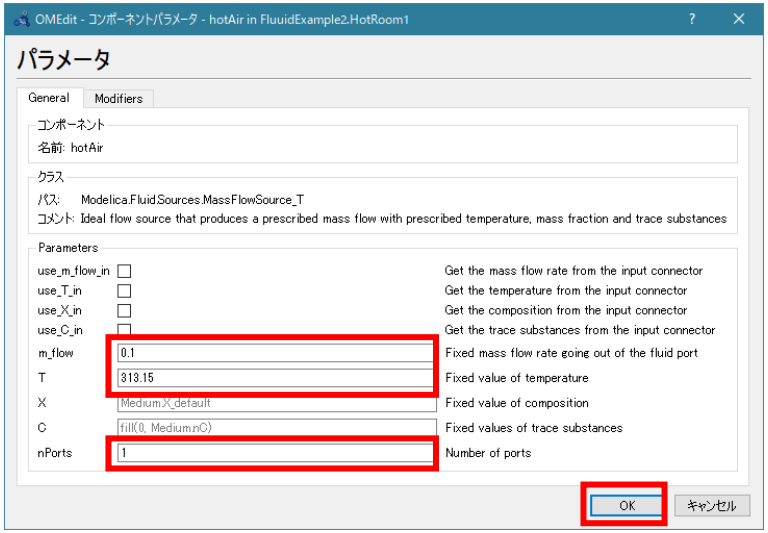
$use\_eps\_Re = false$

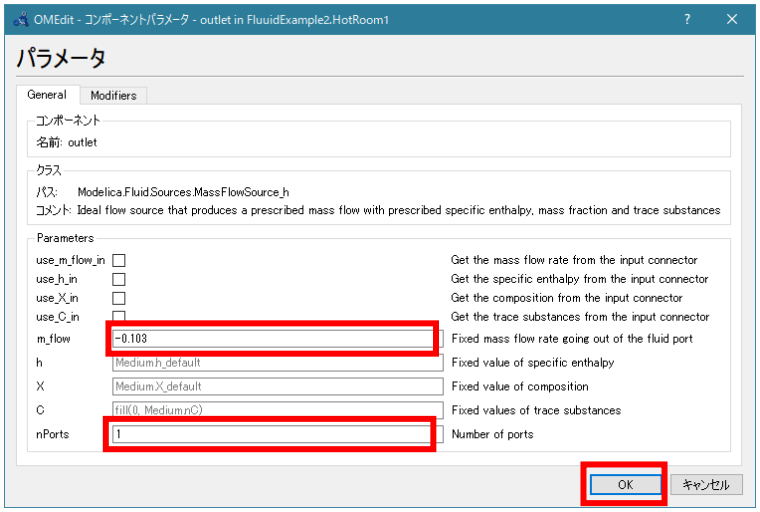
$$m\_flow\_nominal = \begin{cases} 1, & use\_eps\_Re = true \\ 10^2 \times m\_flow\_small, & use\_eps\_Re = false \end{cases}$$

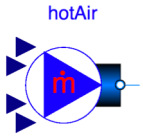
$dp\_small = 1$ [Pa]
$m\_flow\_small = 10^{-2}$ [kg/s]

HotRoom1

(3)(4) Fluid.Sources.MassFlowSource_T







流量と温度を規定する。

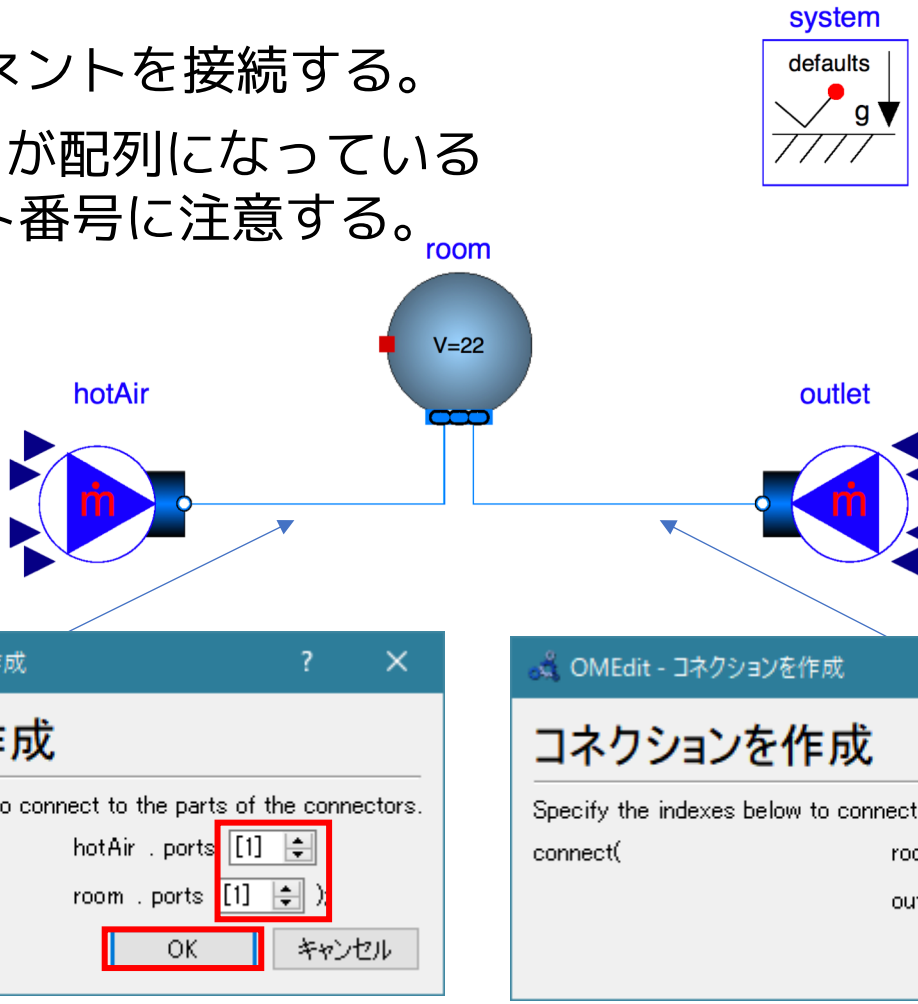
(3) hotAir
- m_flow = 0.1 [kg/s]
- T = 313.15 [K]
- nPorts = 1

(4) outlet
- m_flow = -0.103 [kg/s]
- nPorts = 1

HotRoom1

⑥コンポーネントを接続する。

FluidPort が配列になっているのでポート番号に注意する。




```
connect(hotAir.ports[1], room.ports[1]) annotation( ...);
connect(room.ports[2], outlet.ports[1]) annotation( ...);
```

HotRoom1

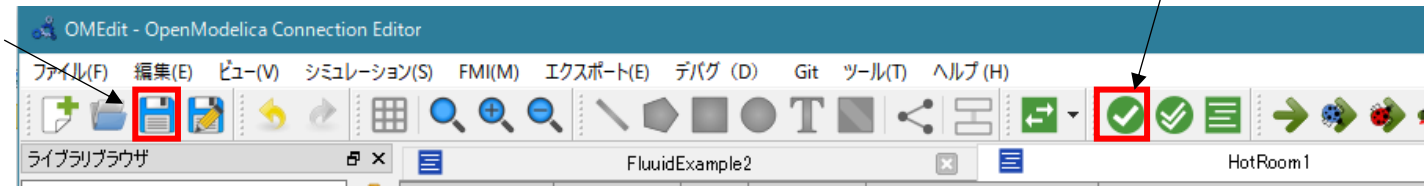
⑦テキストビューに切り替えて、ソースコードを編集する。

```
model HotRoom1
  replaceable package Medium = Media.Air.DryAirNasa;
  inner Modelica.Fluid.System system annotation( ...);                    (1)
  Modelica.Fluid.Vessels.ClosedVolume room(redeclare package Medium = Medium,  (2)
    T_start = 280, V = 22,
    energyDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial,
    massDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial,
    nPorts = 2, p_start = 101325, use_portsData = false) annotation( ...);
  Modelica.Fluid.Sources.MassFlowSource_T hotAir(redeclare package Medium = Medium,  (3)
    T = 313.15, m_flow = 0.1, nPorts = 1) annotation( ...);
  Modelica.Fluid.Sources.MassFlowSource_T outlet(redeclare package Medium = Medium,  (4)
    m_flow = -0.103, nPorts = 1) annotation( ...);
equation
  connect(hotAir.ports[1], room.ports[1]) annotation( ...);
  connect(room.ports[2], outlet.ports[1]) annotation( ...);
  annotation( ...);
end HotRoom1;
```

⑧保存し、モデルチェックを行う。

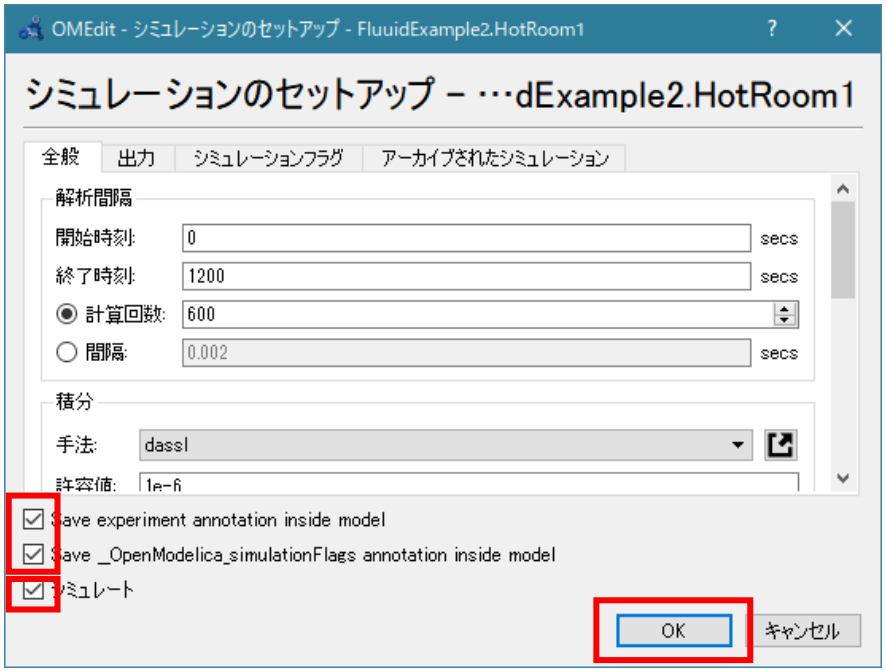
保存

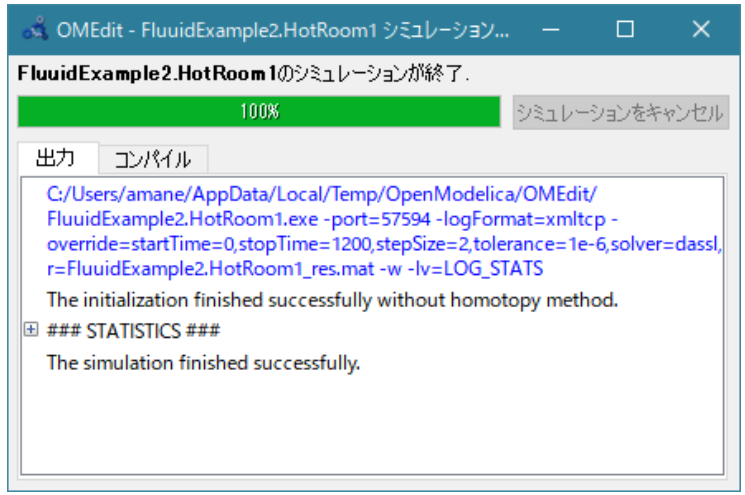
モデルチェック



HotRoom1

⑦シミュレーションのセットアップを行いシミュレーションを実行する。





うまくいけば、
この画面が表示される。

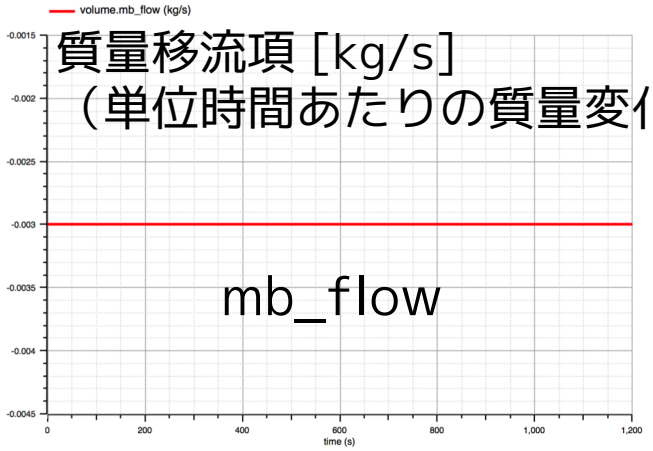
- 開始時間 0 [sec]
- 終了時間 1200 [sec]
- 計算回数 600

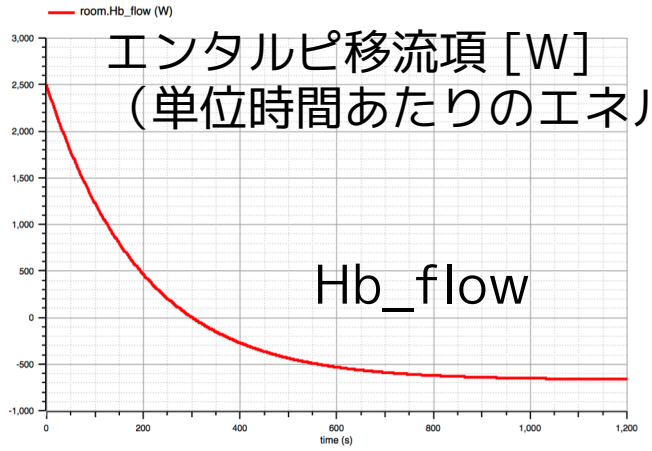
[シミュレート]をチェックして[OK]をクリックするとシミュレーションが実行される。

HotRoom1

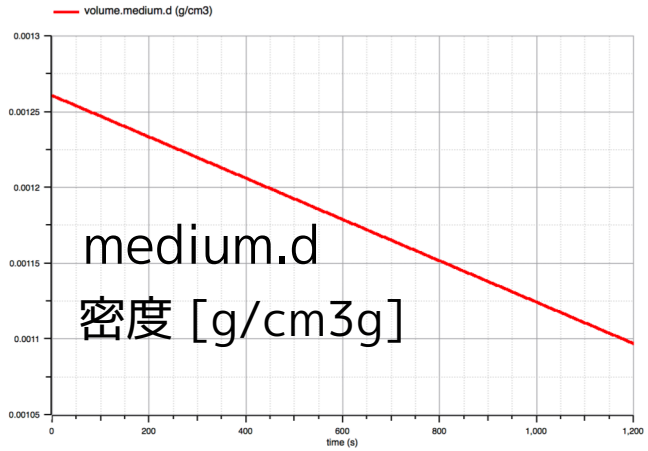
シミュレーション結果

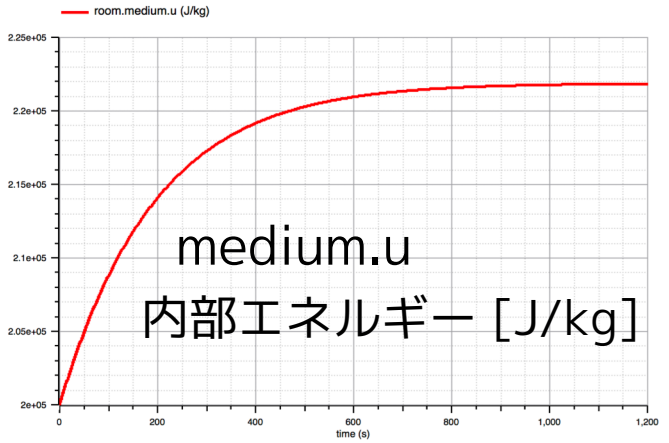
FluidPort から質量やエンタルピ（熱）が出入りする。





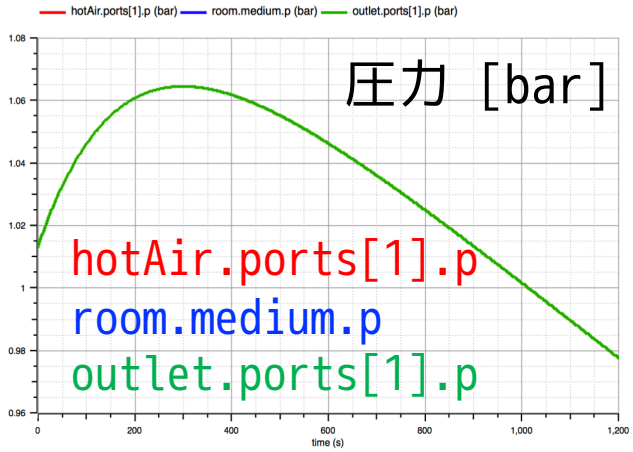
室内の空気の密度や内部エネルギーが変化する。

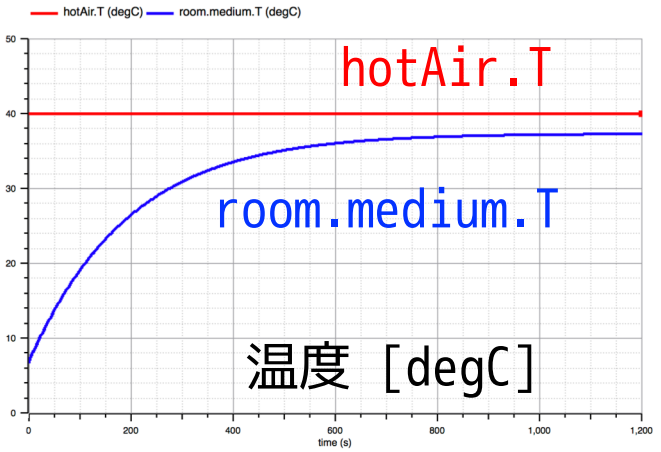




HotRoom1

熱力学的関係式より、室内の圧力や温度、比エンタルピも変化する。





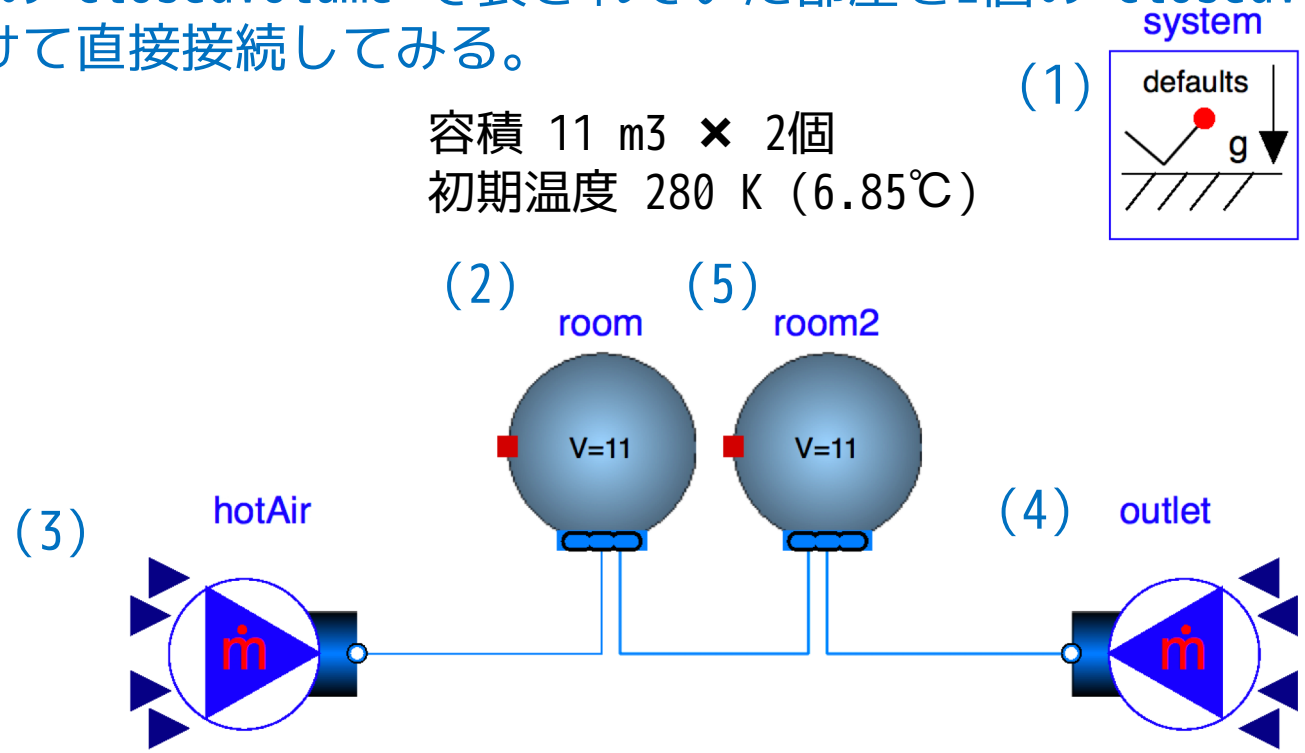
- volume モデルは、容器内の質量(または密度)と内部エネルギーの時間発展を計算する微分方程式モデルとなっている。
- Baseproperties モデルによって熱力学的変数の関係式が与えられるので、他の熱力学的変数の組み合わせ、例えば、圧力や温度に関する微分代数方程式モデルとみなすこともできる。

HotRoom2

部屋を2つに分割する

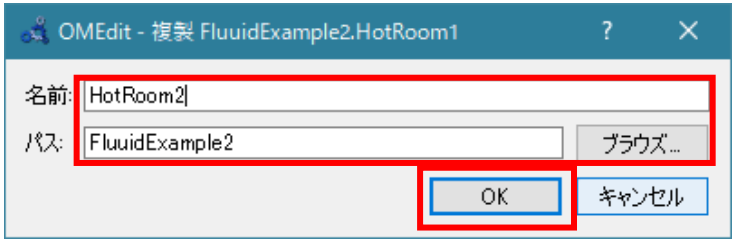
1個の ClosedVolume で表されていた部屋を2個の ClosedVolume に分けて直接接続してみる。

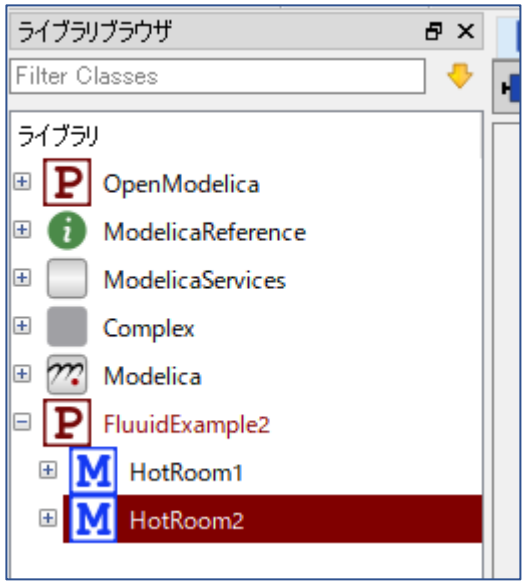
容積 11 m3 × 2個
初期温度 280 K (6.85℃)



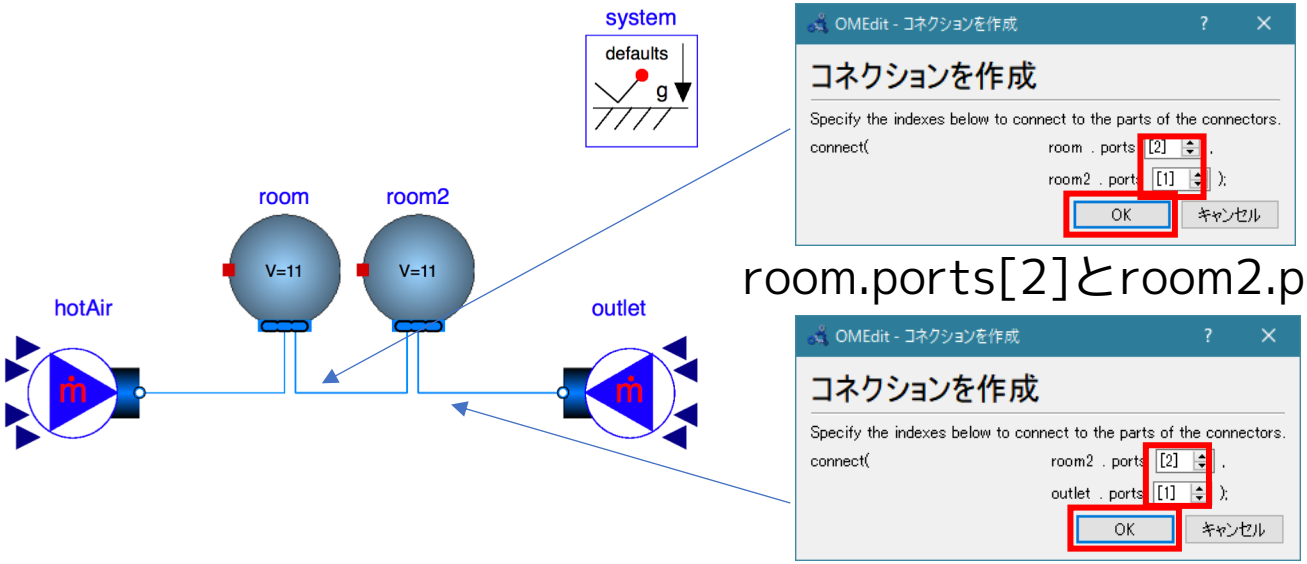
HotRoom2

②ライブラリブラウザの HotRoom1を
右クリックして「複製」を選んで、パス
FluidExample2 に HotRoom2 を作る。





③ダイアグラムビューに切り替えて、room を右クリックし、複製
を選択してコピー room2 を作成する。次のように接続する。

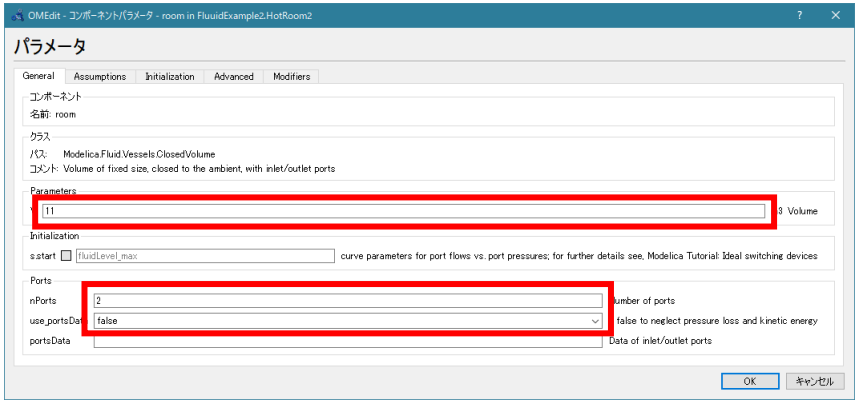


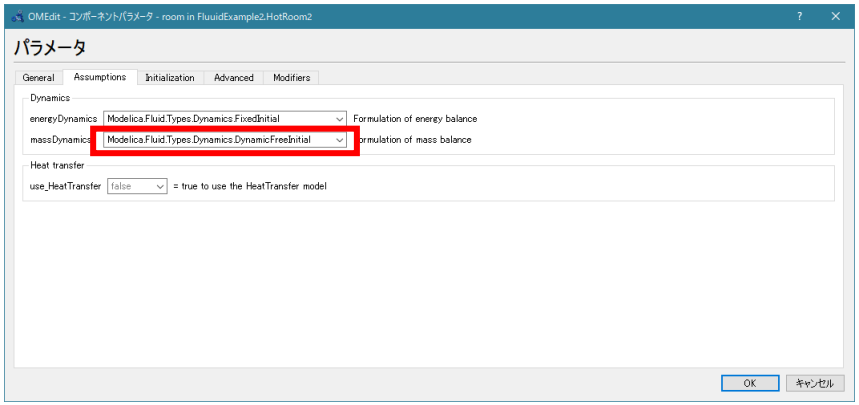
room.ports[2]とroom2.ports[1]

room2.ports[2]とoutlet.ports[1]

HotRoom2

③ room, room2 の容積を半分にする。room の massDynamics を FreeInitialにする。
Fluid.Vessels.ClosedVolume





(2) room, (5) room2
[General]
- V = 11 [m3]
- nPorts = 2
- use_portsData = false

(2) room
[Assumption]
massDynamics
= DynamicFreeInitial

直接接続したことにより、room と room2 の圧力は等しくなる。massDynamics を両方とも FixedInitial にして圧力の初期値を固定すると過剰設定となるので片方はFreeInitial にする。

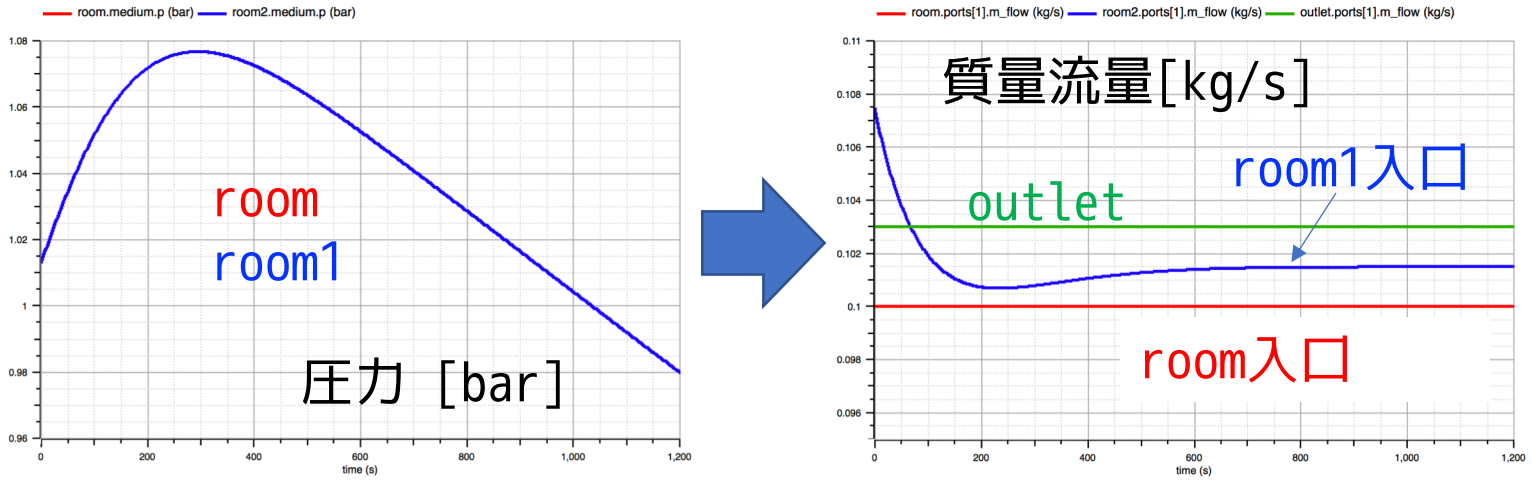
HotRoom2

④ テキストビューで、ソースコードの room2 の部分を編集する。

```
Modelica.Fluid.Vessels.ClosedVolume room2(redeclare package Medium = Medium,
  T_start = 280, V = 11,
  energyDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial,
  massDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial,
  nPorts = 2, p_start = 101325, use_portsData = false) annotation( ...);
```

HotRoom2

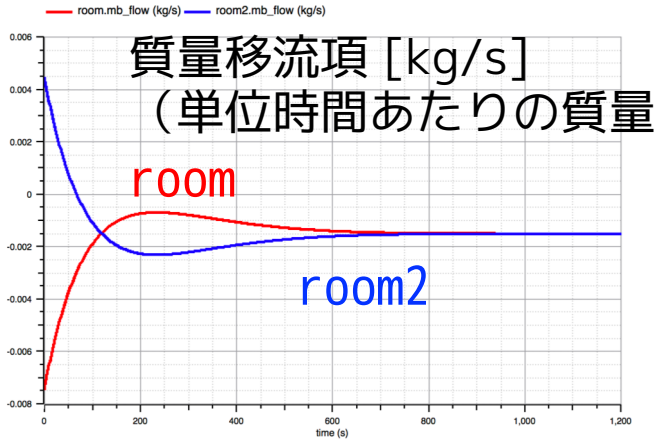
シミュレーション結果

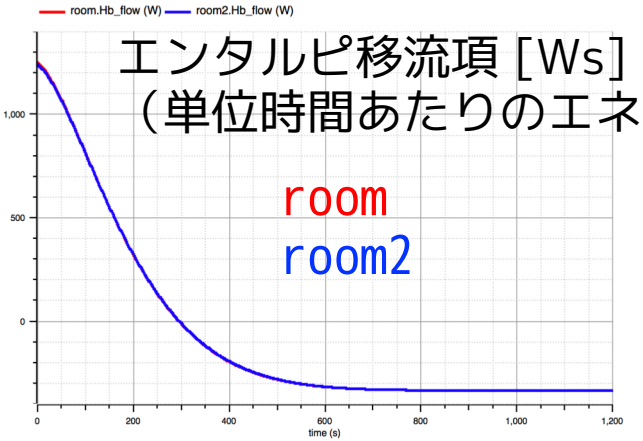


- volume モデルどうしを直接接続すると、２つのvolumeの状態変数が独立に変化するのではなく、圧力が一致する。
- 方程式は圧力に関して高インデックスの微分代数方程式となる。
- 圧力が一致するという制約条件の下で volume モデル間の質量流量が決定される。

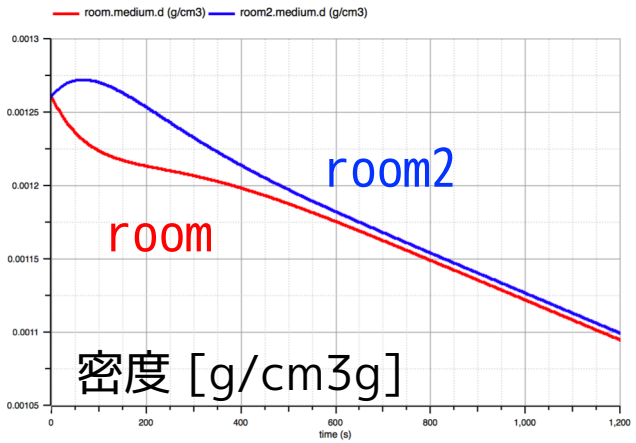
HotRoom2

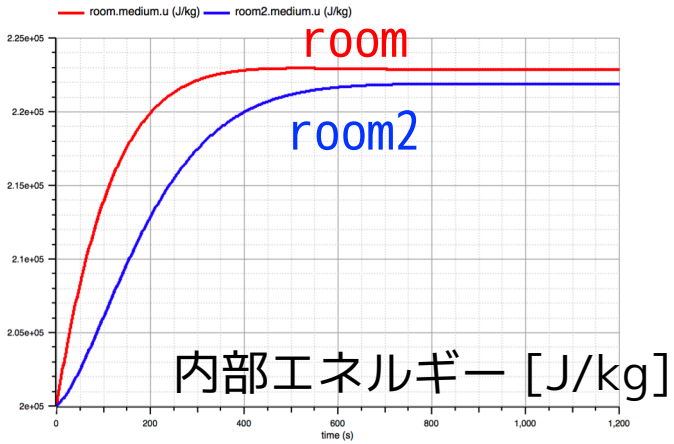
FluidPort から質量やエンタルピ（熱）が出入りする。





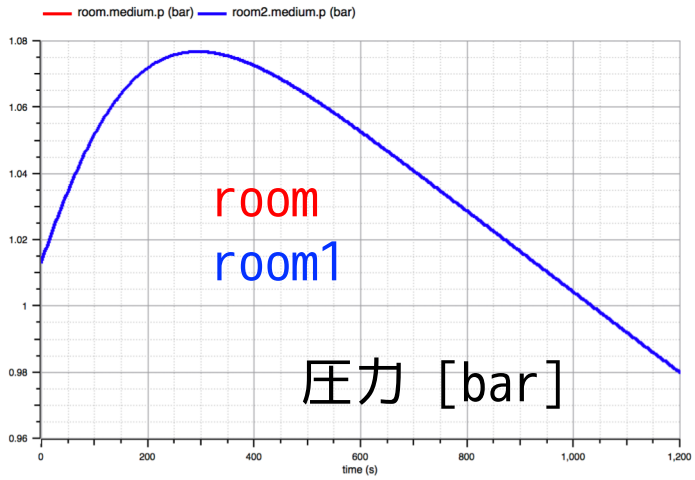
容器内の密度や内部エネルギーが変化する。

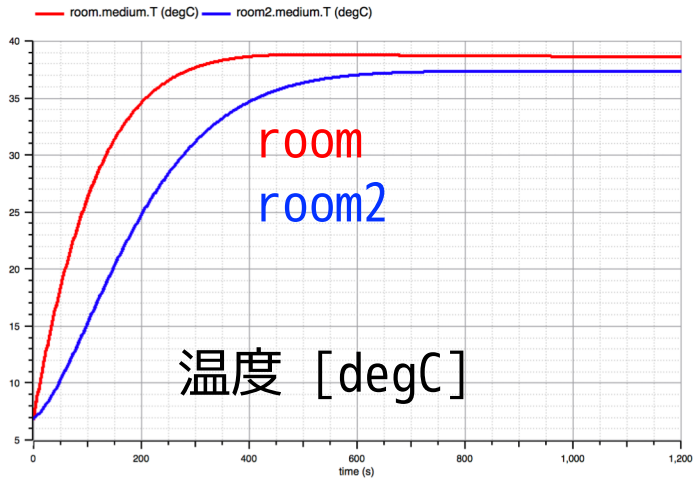




HotRoom2

熱力学的関係式より圧力や温度も変化する。

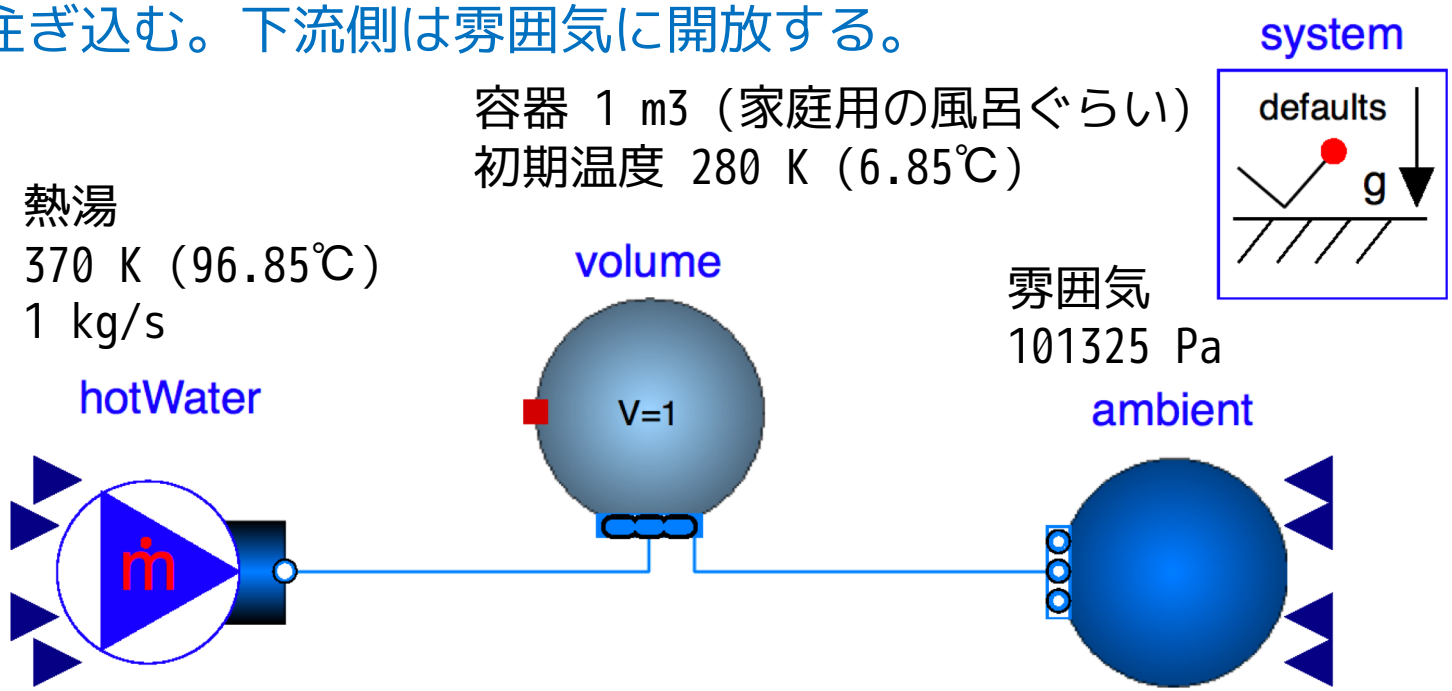




WaterVolume1

容器の冷水を熱湯と入れ替える

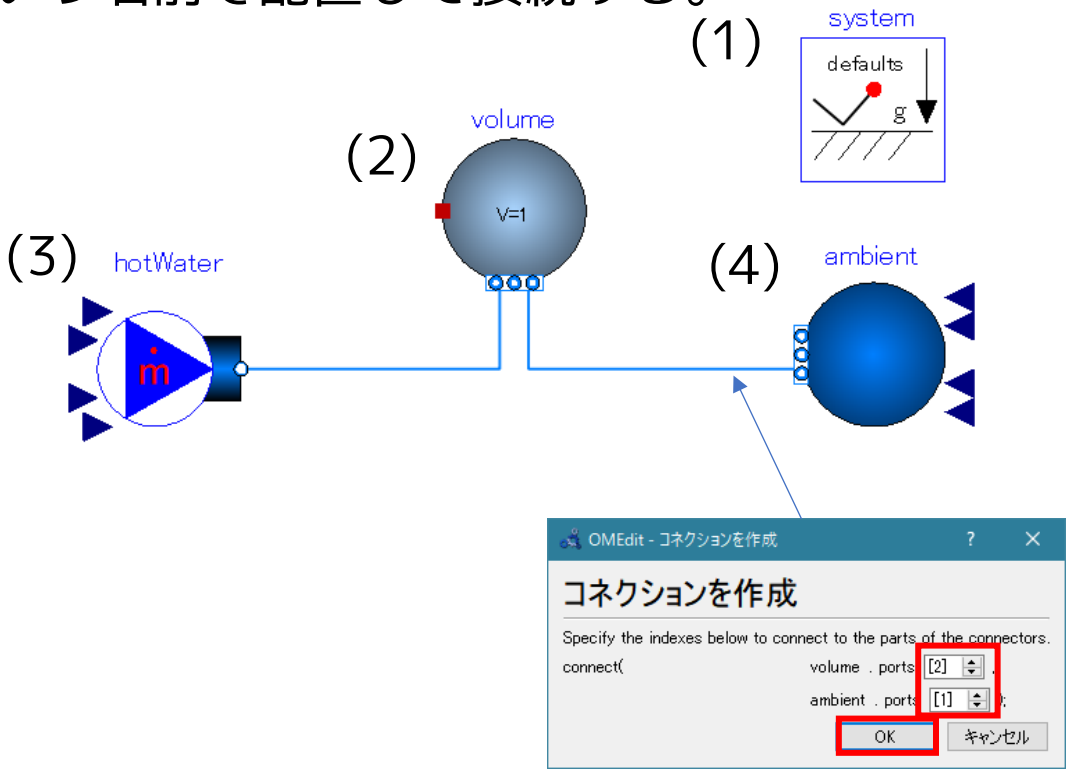
容積 1 m3 の容器に冷水が入っている。上流側から一定流量で熱湯を注ぎ込む。下流側は雰囲気に開放する。



この場合、容器内の水の圧力は雰囲気圧力によって規定される。
非圧縮性流体も使用可能。

WaterVolume1

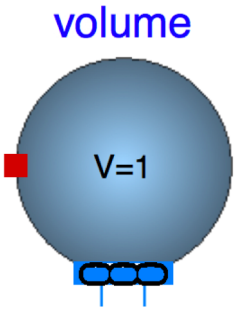
① HotRoom1 を右クリックして複製を選んで、パスFluidExample2 に WaterVolume1 という名前でコピーを作成する。
② hotAir を hotWater に、room を volume にリネームする。
③ outlet を削除し、代わりに Fluid.Sources.Boundary_pT を、ambientという名前で配置して接続する。



WaterVolume1

④ パラメータを設定する。

Fluid.Vessels.ClosedVolume



(2) volume
[General]
- V = 1 [m3]
- nPorts = 2
- use_portsData = false

[Assumptions]
- energyDynamics = FixedInitial
- MassDynamics = FreeInitial
- use_HeatTransfer = false

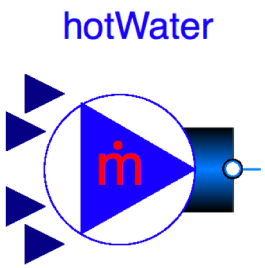
[Initialization]
- use_T_start = true
- T_start = 280

WaterVolume1

Fluid.Sources.MassFlowSource_T

(3) hotWater

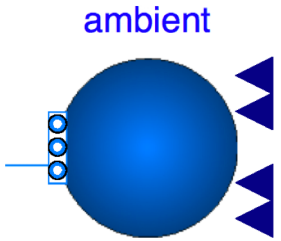
- m_flow = 1 [kg/s]
- T = 370 [K]
- nPorts = 1



Fluid.Sources.Boundary_pT

(4) ambient

- p = 101325 [Pa]
- nPorts = 1



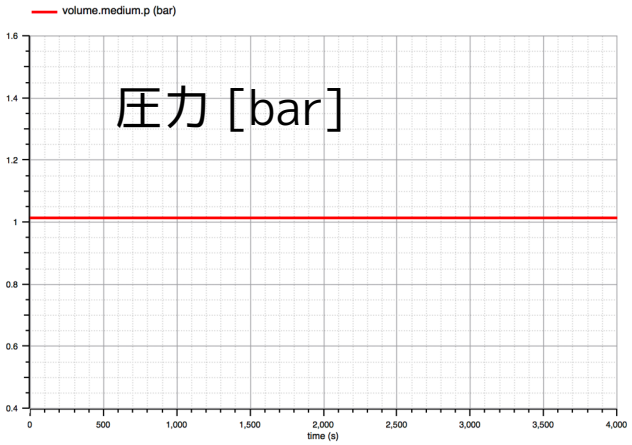
WaterVolume1

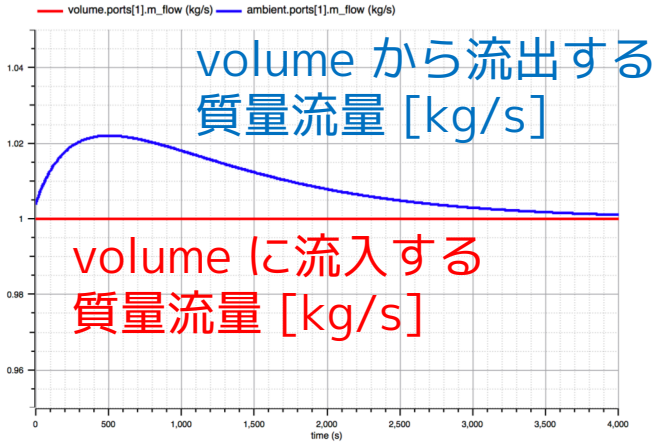
⑤ テキストビューでソースコードを編集する。

```
model WaterVolume1
  replaceable package Medium = Media.Water.StandardWater;
  //replaceable package Medium = Media.Water.ConstantPropertyLiquidWater;
  inner Modelica.Fluid.System system annotation( ...);
  Modelica.Fluid.Vessels.ClosedVolume volume(redeclare package Medium = Medium,
    T_start = 280, V = 1,
    energyDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial,
    massDynamics = Modelica.Fluid.Types.Dynamics.DynamicFreeInitial,
    nPorts = 2, p_start = 101325, use_portsData = false) annotation( ...);
  Modelica.Fluid.Sources.MassFlowSource_T hotWater(redeclare package Medium = Medium,
    T = 370, m_flow = 1, nPorts = 1, use_T_in = false) annotation( ...);
  Modelica.Fluid.Sources.Boundary_pT ambient(redeclare package Medium = Medium,
    nPorts = 1, p = 101325) annotation( ...);
equation
  connect(hotWater.ports[1], volume.ports[1]) annotation( ...);
  connect(volume.ports[2], ambient.ports[1]) annotation( ...);
  annotation( ...);
end WaterVolume1;
```

WaterVolume1

シミュレーション結果



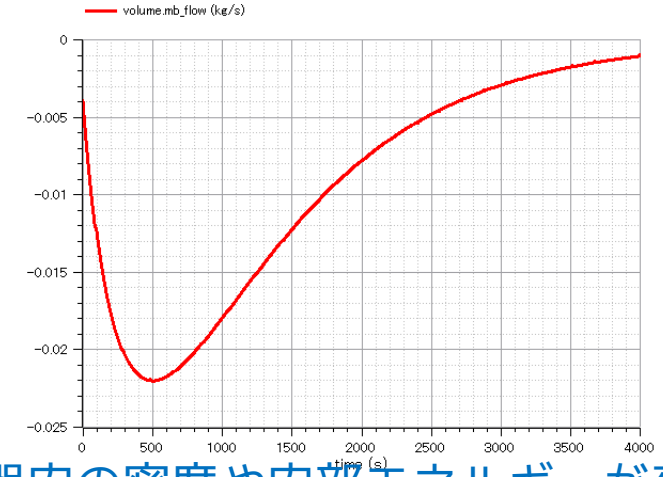


volume の圧力が ambient の圧力と一致するという制約条件の下で volume から流出する質量流量が決定される

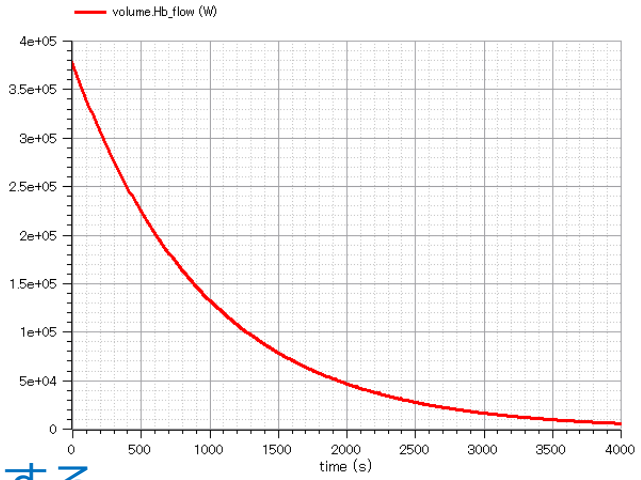
WaterVolume1

FluidPort から質量やエンタルピ（熱）が出入りする。

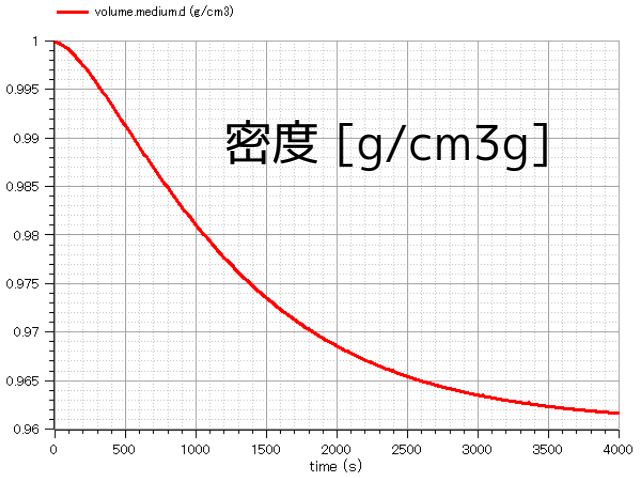
質量移流項 [kg/s]
（単位時間あたりの質量変化）

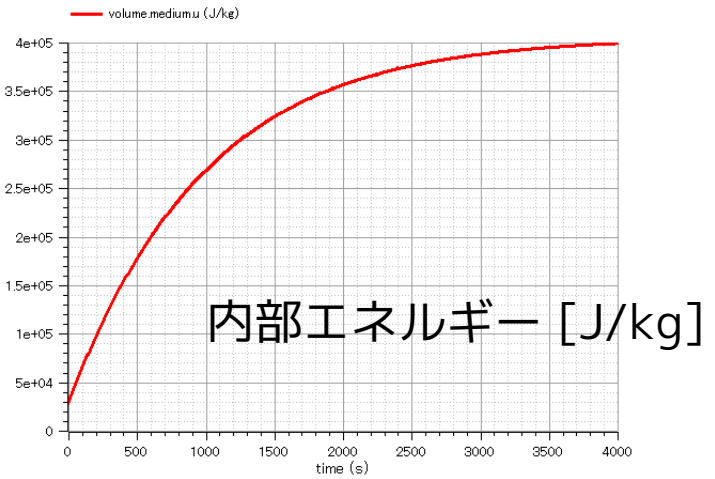


エンタルピ移流項 [Ws]
（単位時間あたりのエネルギー変化）



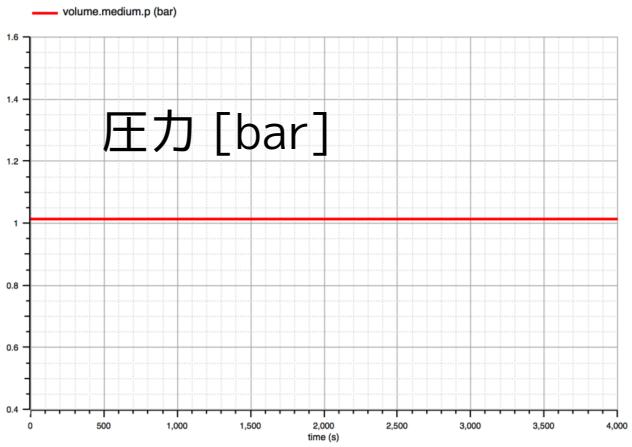
容器内の密度や内部エネルギーが変化する。

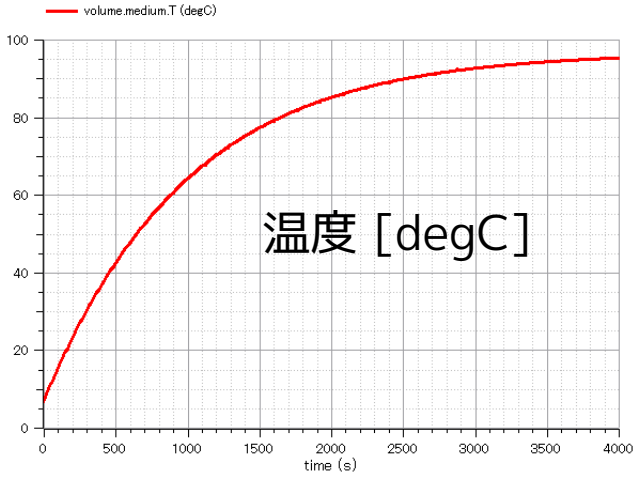




WaterVolume1

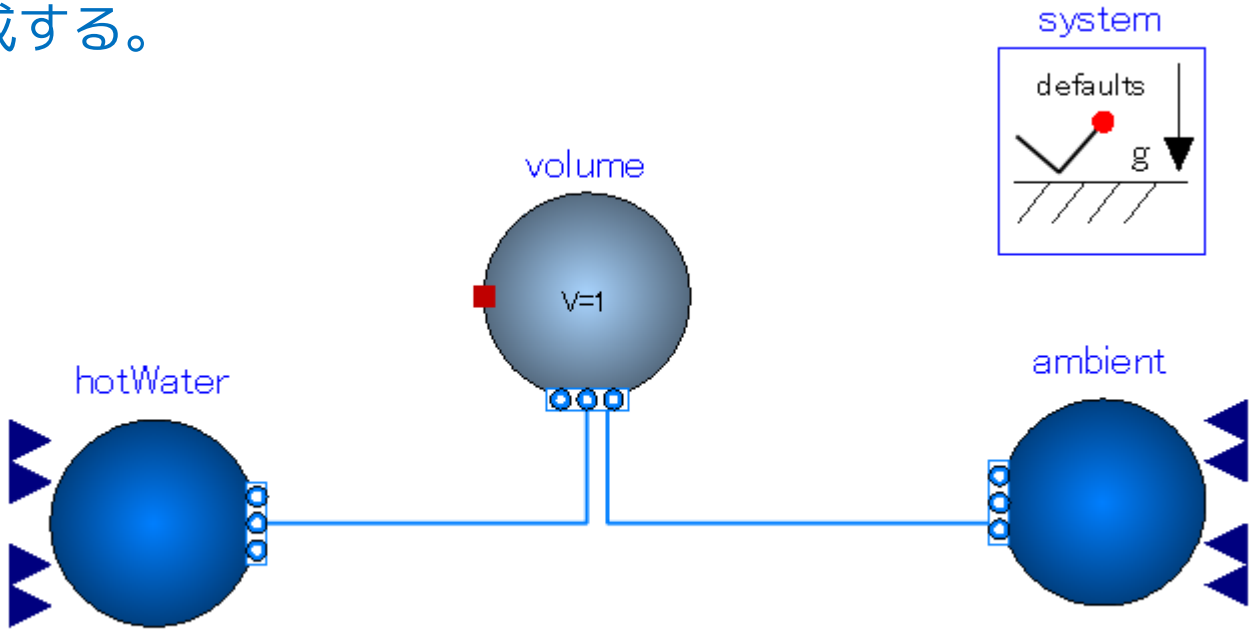
熱力学的関係式より、室内の圧力や温度も変化する。





WaterVolume2
容器出入り口の圧力損失を考慮する。

容器の上流と下流の圧力を規定し、容器入口の流れの急拡大による圧力損失と容器出口の流れの急縮小による圧力損失を考慮したモデルを作成する。

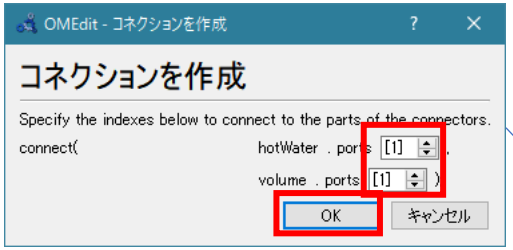


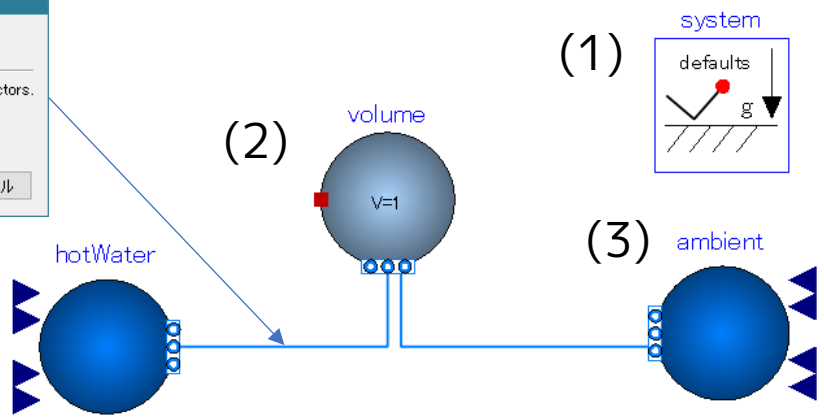
WaterValume2

① WaterVolume1 を右クリックして複製を選んで、FluidExample2に WaterVolume2 という名前でコピーを作成する。
② テキストビューに切り替え、WaterVolume2の最初の方に次の import 文を付け加える。

```
model WaterVolume2
  import Modelica.Fluid.Vessels.BaseClasses.VesselPortsData;
  replaceable package Medium = Media.Water.StandardWater;
```

③ ダイアグラムに切り替え、hotWater を削除し、代わりに Fluid.Sources.Boundary_pT を hotWaterという名前で配置して接続する。





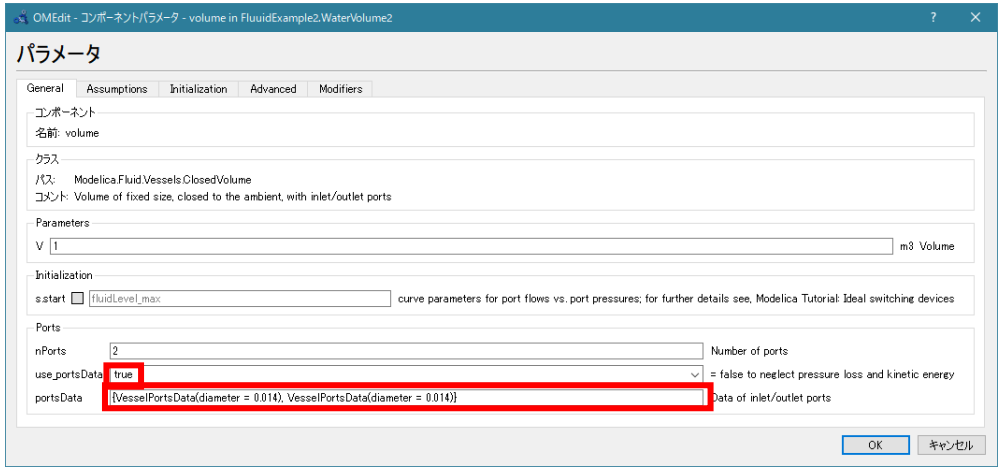
Fluid.Sources.Boundary_pT

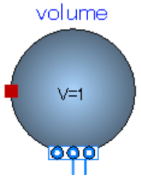
(4) hotWater
- p = 131325 [Pa]
- T = 370 [K]
- nPorts = 1

WaterVolume2

④ コンポーネントのパラメータを設定する。

(2) Fluid.Vessels.ClosedVolume





圧力損失を計算するために
portsData を使用し、
出入り口の直径を設定する。

[General]

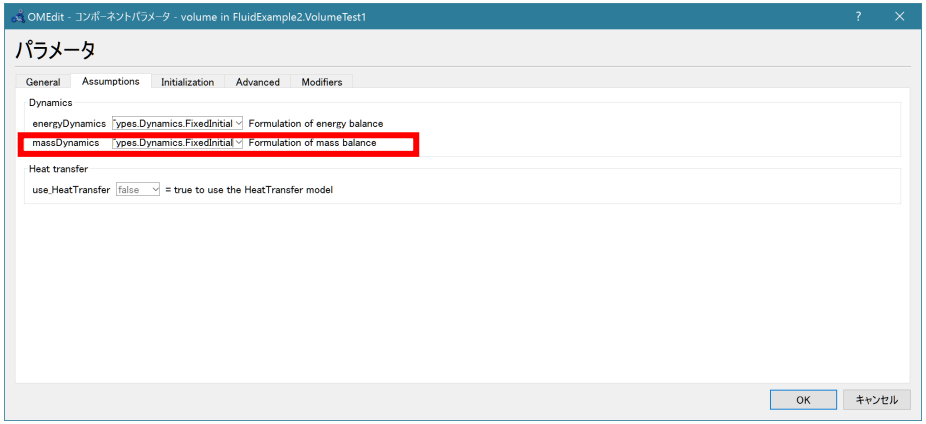
- V=1
- nPorts = 2
- **use_portsData = true**
- **portsData**
  = {VesselPortsData(diameter = 0.014), VesselPortsData(diameter = 0.014)}

出入り口の径 1.4 cm
（水道の給水管ぐらい）

‘=‘は入力しない。

portsData のテキストボックスは挙動がおかしく、設定が無効になることがあるので、メモ帳などにコピペして編集内容をキープしておくと便利です。

WaterVolume2



[Assumptions]

- energyDynamics = Dynamics.FixedInitial
- massDynamics = **Dynamics.SteadyStateInitial**
- use_HeatTransfer = false

WaterVolume2

⑤ テキストビューに切り替えてソースコードを編集する。

volume

```
Modelica.Fluid.Vessels.ClosedVolume volume(redeclare package Medium = Medium,
   T_start = 280, V = 1, fluidLevel = 1.0,
   energyDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial,
   massDynamics = Mod elica.Fluid.Types.Dynamics.DynamicFreeInitial, nPorts = 2,
   portsData = {VesselPortsData(diameter = 0.014), VesselPortsData(diameter = 0.014)},
   use_portsData = true) annotation( ...);
```

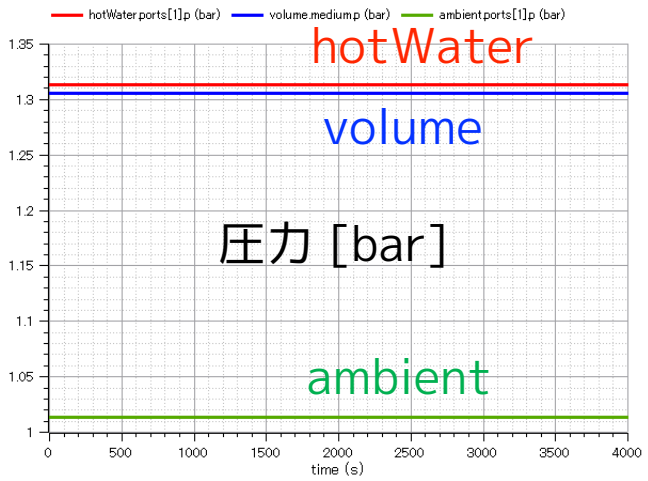
PartialLumpedVessel は、fluidLevel に
fluidLevel ＞ 0.2*diamter
となるような値を設定しないと ports_penetration [i] < 1
となり、圧力損失係数が正しく評価されないようです。

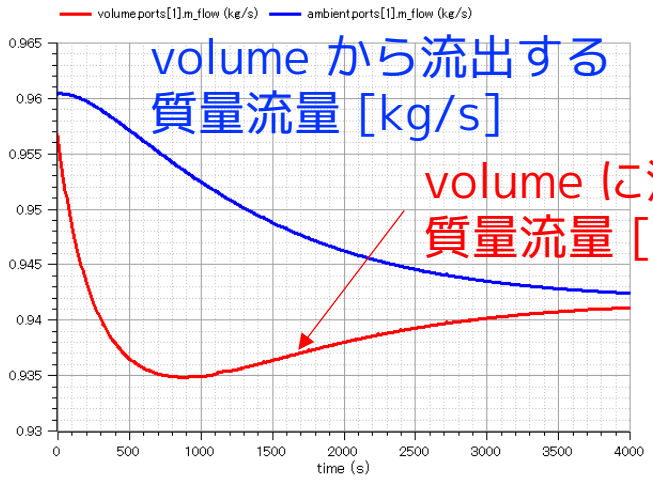
hotWater

```
Modelica.Fluid.Sources.Boundary_pT hotWater(redeclare package Medium = Medium,
   T = 370, nPorts = 1, p = 131325) annotation( ...);
```

⑥ 保存、モデルチェックなどを行い、シミュレーションを実行する。

WaterVolume2



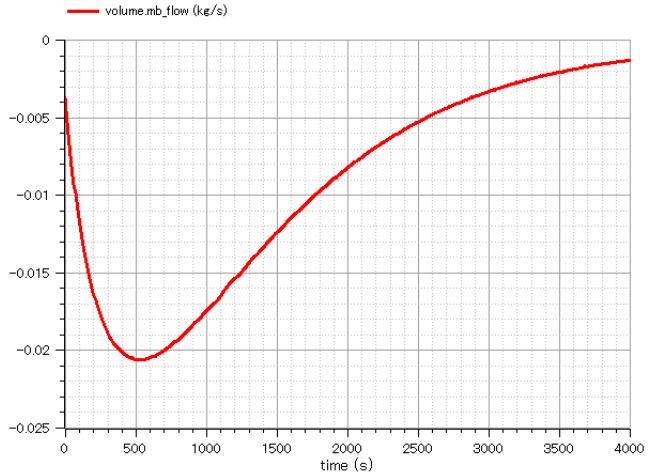


- 容器出入口の圧力損失は流量に対する非線形方程式として表される。（付録２参照)
- 容器の圧力や質量流量は、この非線形方程式を連立して得られる。

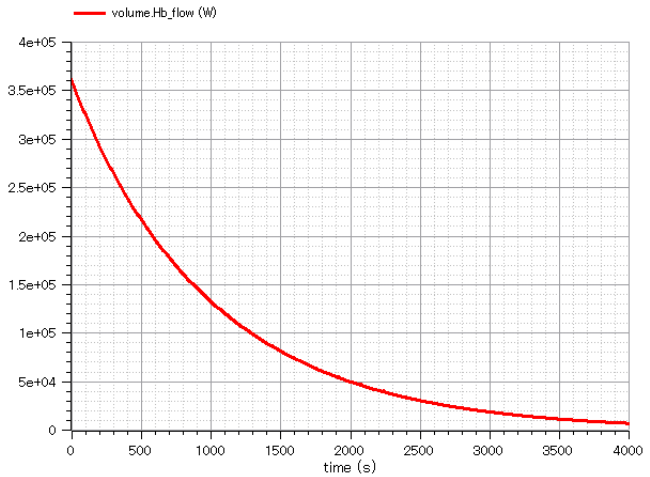
WaterVolume2

FluidPort から質量やエンタルピ（熱）が出入りする。

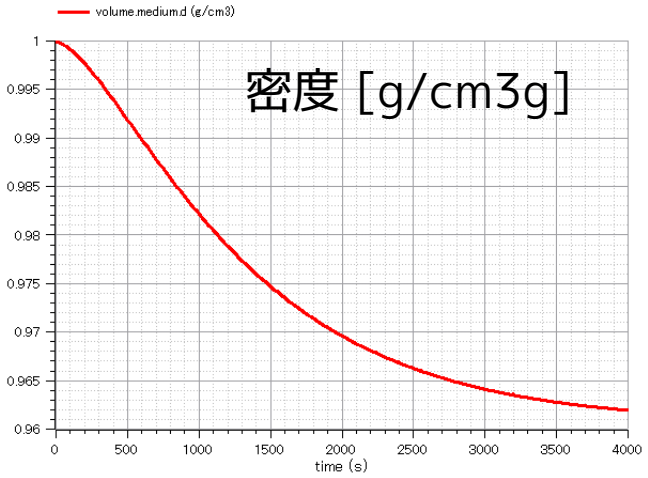
質量移流項 [kg/s]
（単位時間あたりの質量変化）

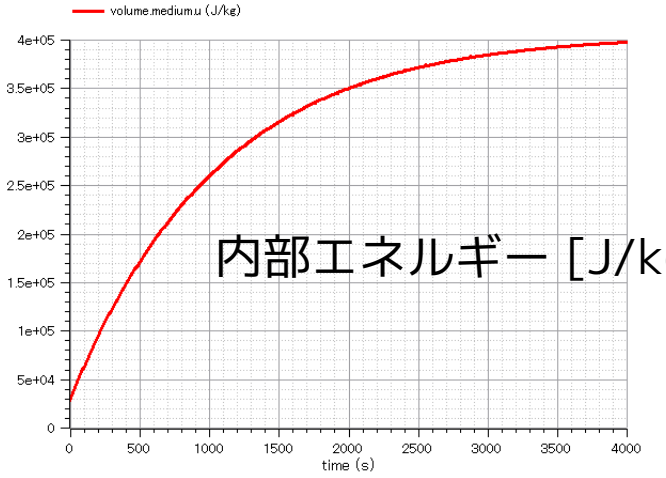


エンタルピ移流項 [Ws]
（単位時間あたりのエネルギー変化）



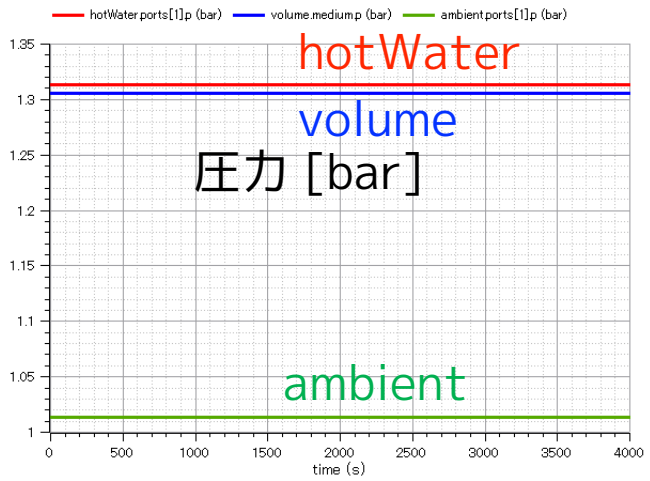
容器内の密度や内部エネルギーが変化する。

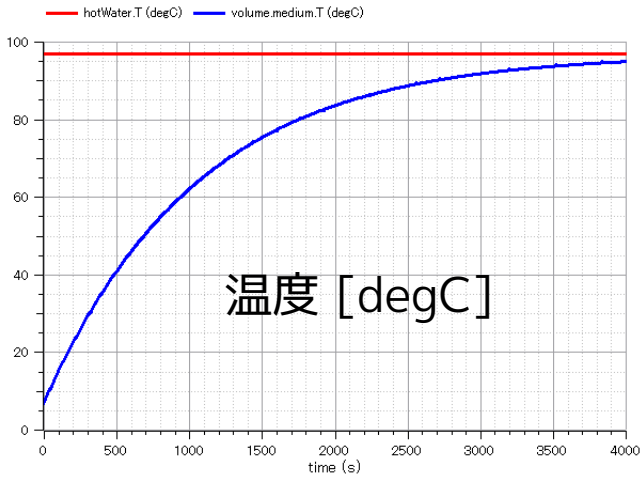




WaterVolume2

圧力や温度も変化する。

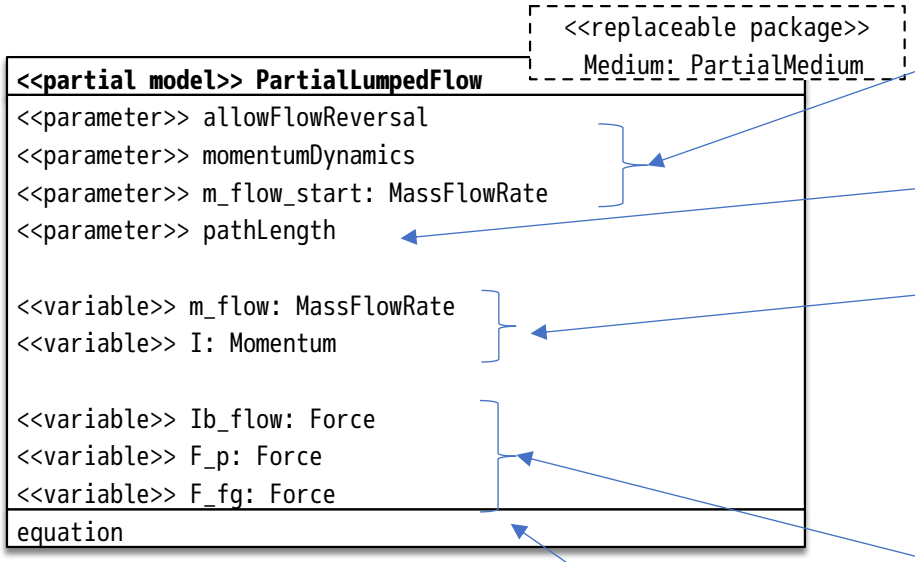




fluidExample3

flow モデル 運動量方程式を実装するモデル

PartialLumpedFlow 1個のflowモデルのベースモデル



定常/非定常、初期条件を設定するパラメータ

流路長さ

状態変数　m_{flow}, I

継承先のモデルで方程式を定義する変数

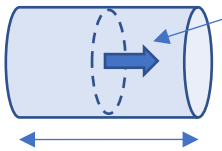
Ib_{flow} [N]: 運動量移流項

F_p [N]: 圧力による力

F_{fg} [N]: 摩擦力と重力

運動量保存式

$I = m_{flow} \cdot pathLength$ [kg m/s]: 運動量



m_{flow} [kg/s]
:質量流量

$pathLength$ [m]:長さ

flowモデル

PartialLumpedFlowの方程式

```
equation
  // Total quantities
  I = m_flow*pathLength;

  // Momentum balances
  if momentumDynamics == Types.Dynamics.SteadyState then
    0 = Ib_flow - F_p - F_fg;
  else
    der(I) = Ib_flow - F_p - F_fg;
  end if;

initial equation
  if momentumDynamics == Types.Dynamics.FixedInitial then
    m_flow = m_flow_start;
  elseif momentumDynamics == Types.Dynamics.SteadyStateInitial then
    der(m_flow) = 0;
  end if;

  annotation ( ...);
end PartialLumpedFlow;
```

運動量と質量流量の関係

$$I = m_{flow} \cdot pathLength$$

運動量保存式 (momentum balance)

$$0 = Ib_{flow} - F_p - F_{fg} \quad \text{定常}$$

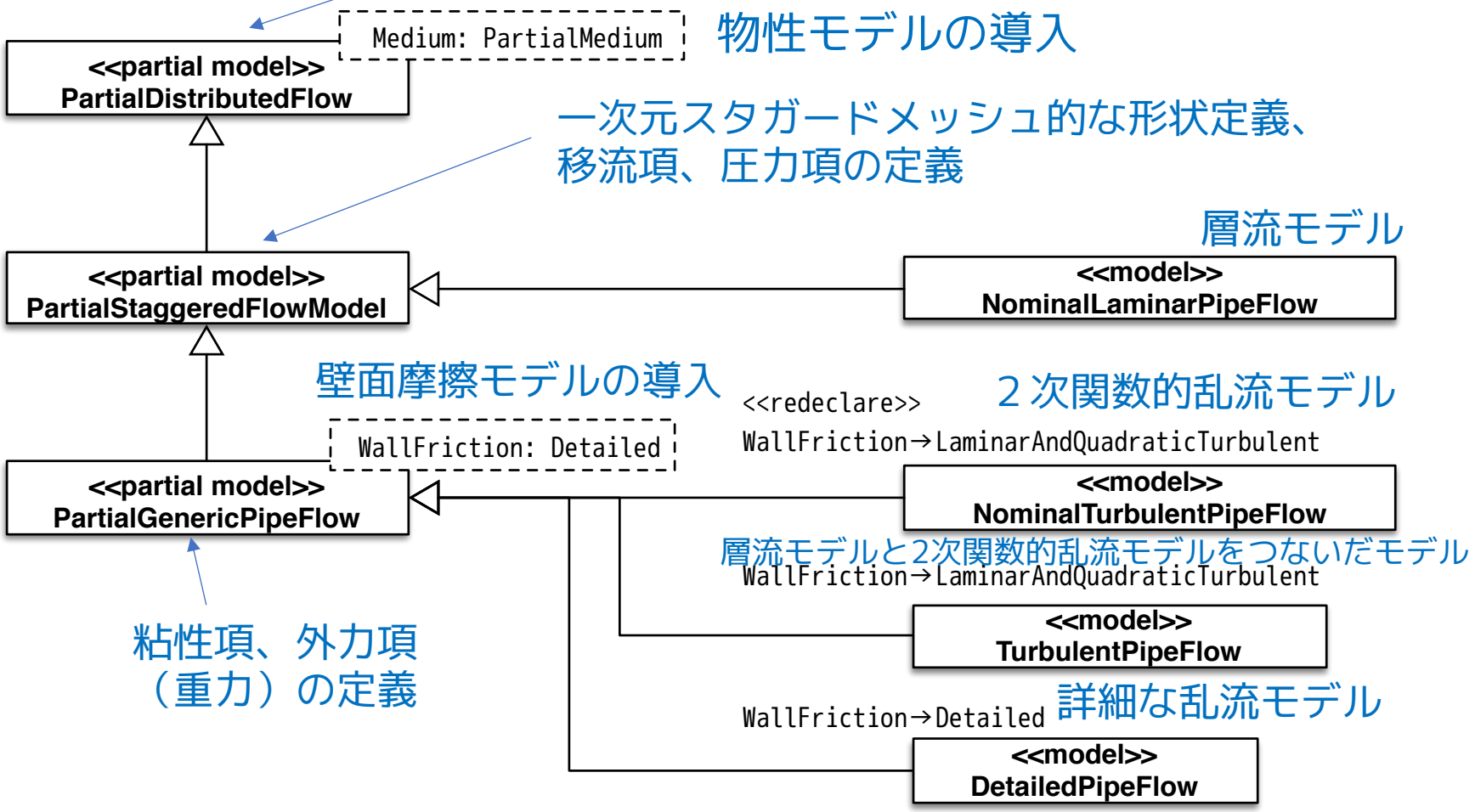
$$\frac{dI}{dt} = Ib_{flow} - F_p - F_{fg} \quad \text{非定常}$$

初期条件

m_{flow}=m_fow_start or $\frac{dm_{flow}}{dt} = 0$

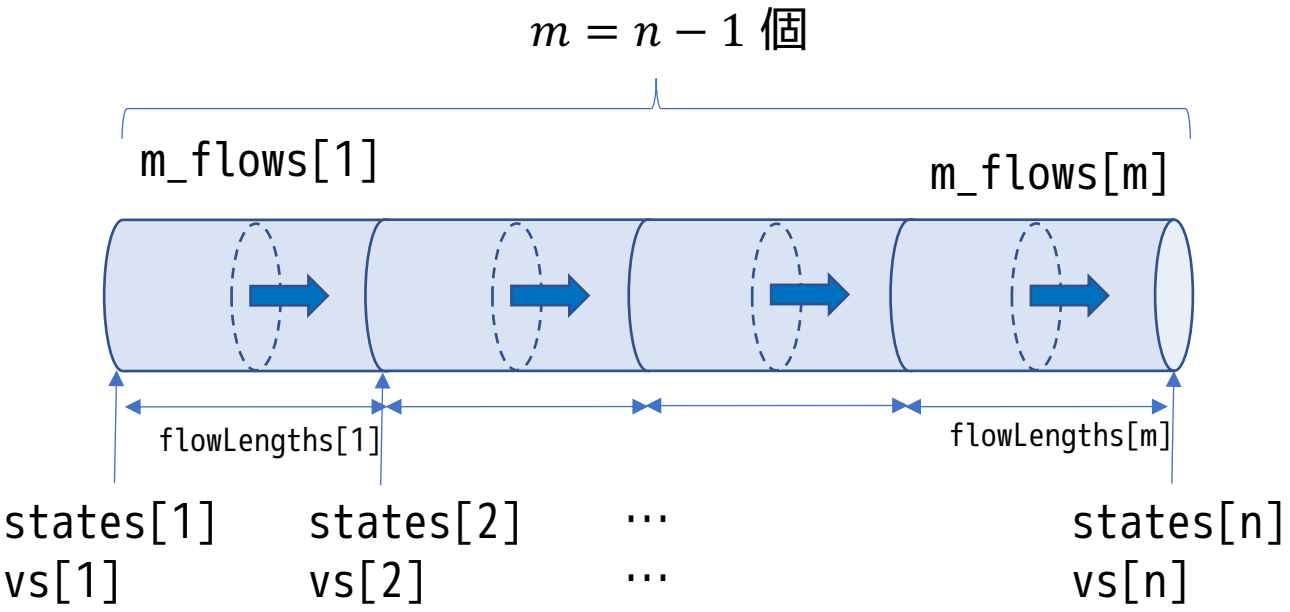
flowモデル

flow モデルの継承関係



flowモデル

PartialStageredFlowModel



境界条件

states[i]: 熱力学的状態変数(ThermodynamicState レコード)

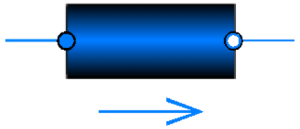
vs[i]: 流速

状態変数

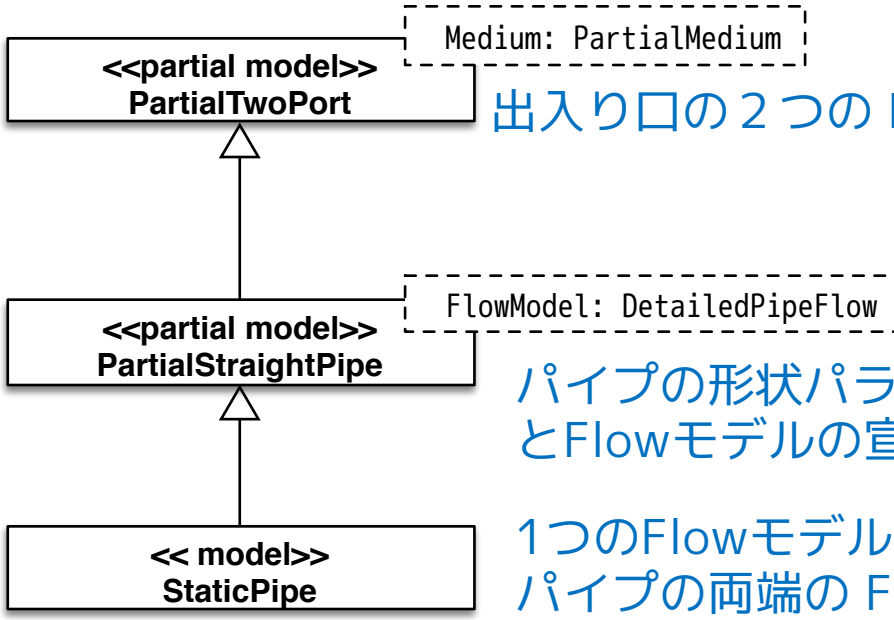
ls[i] = m_flows[i]*flowLengths[i]

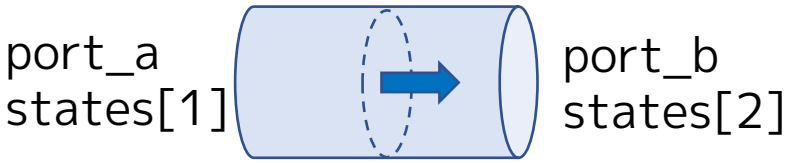
flowモデル

StaticPipe



- 定常状態を表すため massDynamics は steadyStateに固定されている。
- 内部に流体を蓄える機能は持たない。



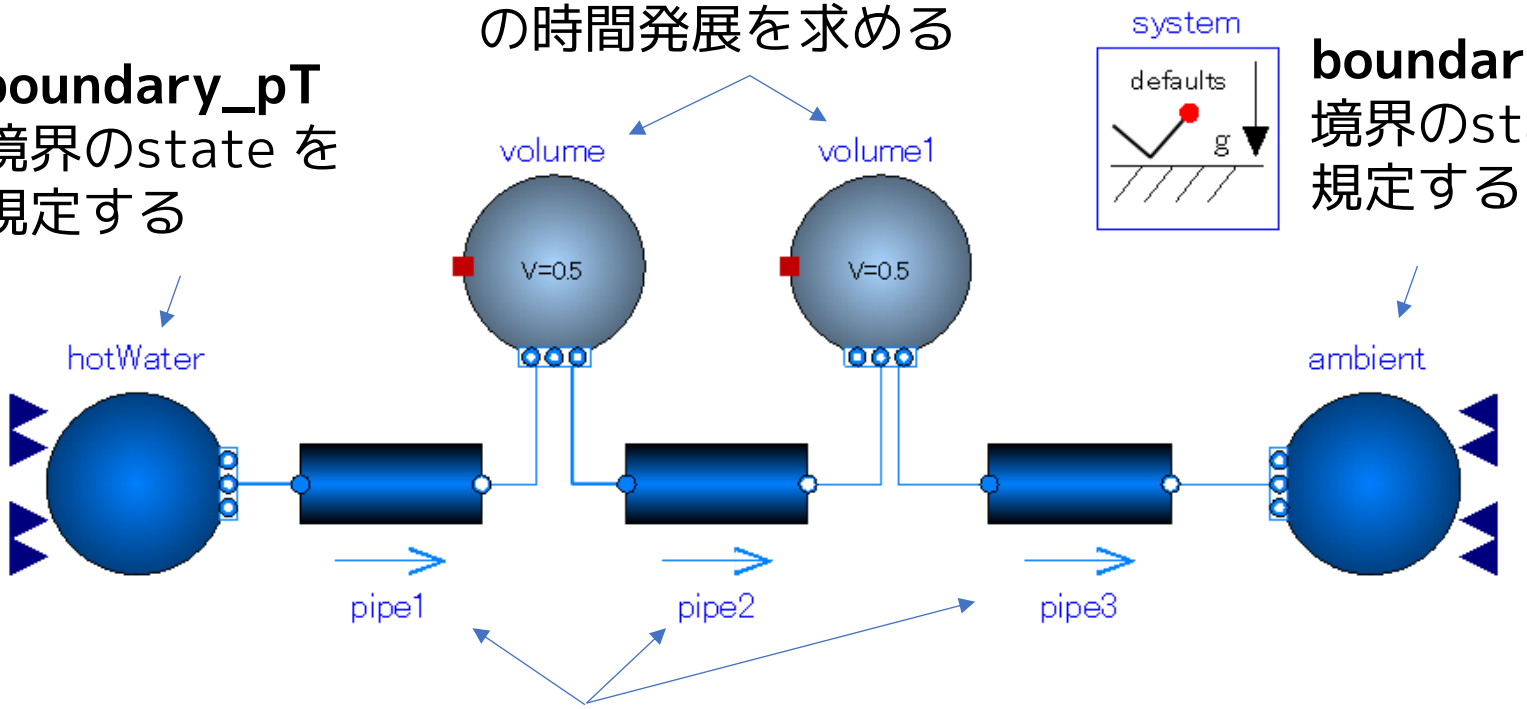


2つの熱力学的状態
states[1], states[2] の間の
定常流を求めるモデル

StaticPipeTest1
flowモデルとvolumeモデルを配置する。

volumeモデル
state (熱力学的状態)
の時間発展を求める

boundary_pT
境界のstate を
規定する

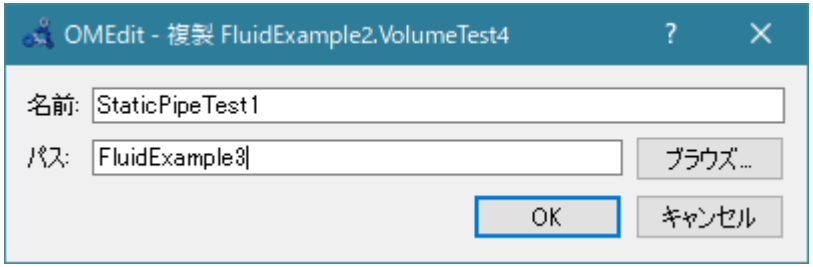


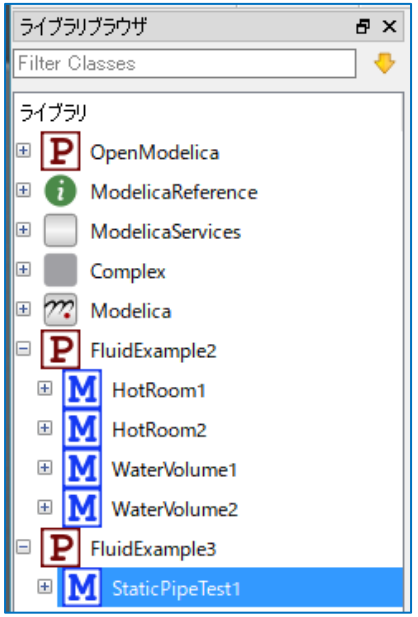
boundary_pT
境界のstate を
規定する

flowモデル
２つのstate間の質量流量を求める。

StaticPipeTest1

① FluidExample3 という名前の package を作成する。
② FluidExample2 の WaterVolume2 を右クリックして複製を選択し、FluidExample3にStaticPipeTest1という名前でコピーする。

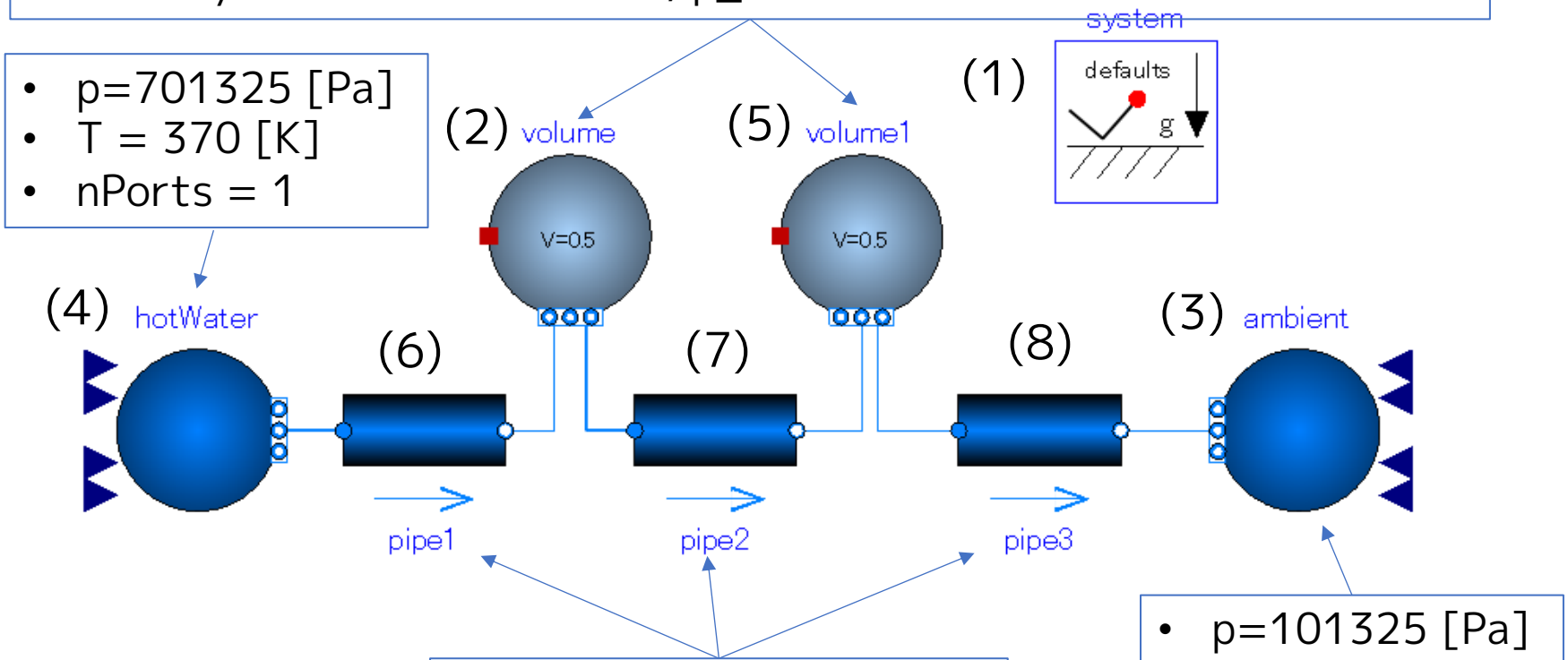




③ volume を複製して volume1 を作成する。
④ 次のスライドのように、Fluid.Pipes.StaticPIpe を 3 個配置して接続する。

⑤ パラメータを設定する。

- V=0.5 [m3]
- nPorts = 2
- use_portsData = false
- portsDataなし
- energyDynamics = FixedInitial, use_T_start = true, T_start = 280
- massDynamics = FixedInitial, p_start = 101325



- p=701325 [Pa]
- T = 370 [K]
- nPorts = 1

- length = 5 [m]
- diameter = 0.014 [m]

- p=101325 [Pa]
- nPorts = 1

StaticPipeTest1

⑥ テキストビューに切り替え、FluidExample3 の最初に import 文を入れる。新しく配置した volume1 とpipe1～pipe3の物性パッケージを設定する。

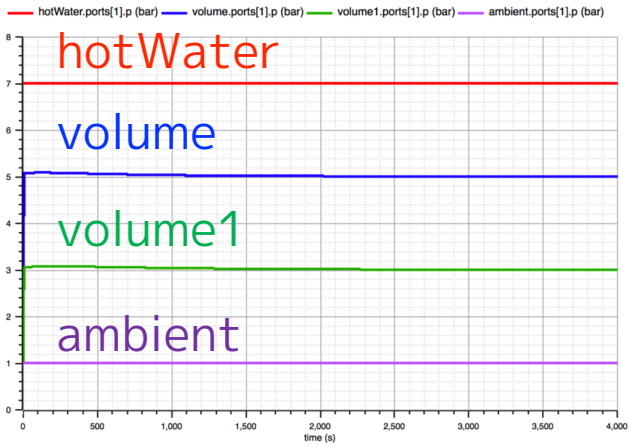
```
package FluidExample3
  import Modelica.Media;
  model StaticPipeTest1
```

```
 Modelica.Fluid.Vessels.ClosedVolume volume1(redeclare package Medium = Medium,
   T_start = 280, V = 0.5,
   energyDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial, fluidLevel = 1.0,
   massDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial,
   nPorts = 2, p_start = 101325, use_portsData = false) annotation( ...);
```

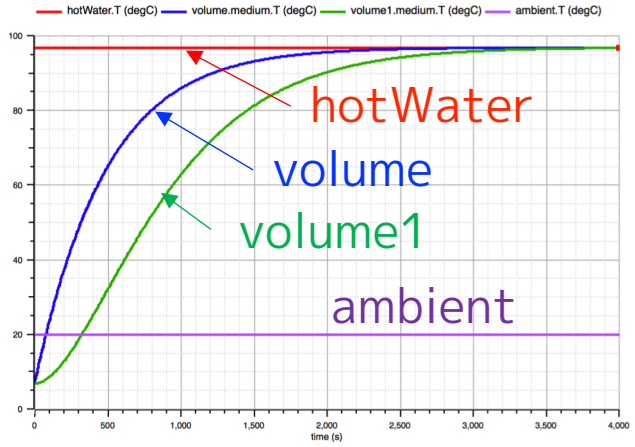
```
 Modelica.Fluid.Pipes.StaticPipe pipe1(redeclare package Medium = Medium,
   diameter = 0.014, length = 5)  annotation( ...);
 Modelica.Fluid.Pipes.StaticPipe pipe2(redeclare package Medium = Medium,
   diameter = 0.014, length = 5)  annotation( ...);
 Modelica.Fluid.Pipes.StaticPipe pipe3(redeclare package Medium = Medium,
   diameter = 0.014, length = 5)  annotation( ...);
```

⑦ 保存、モデルチェックなどを行なってシミュレーションを実行する。

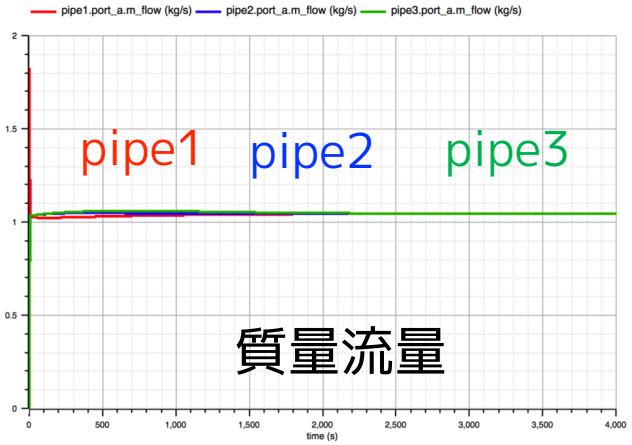
StaticPipeTest1

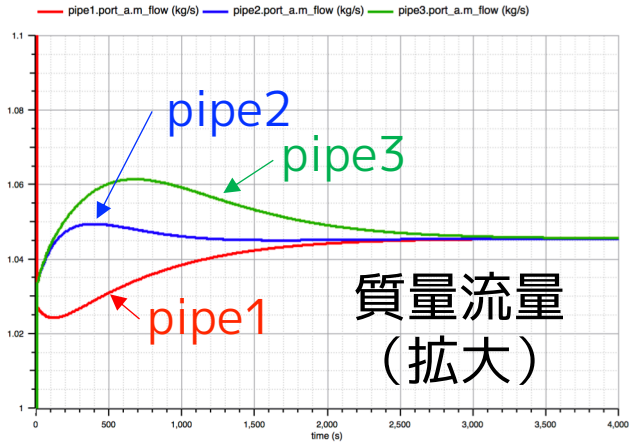


圧力



温度

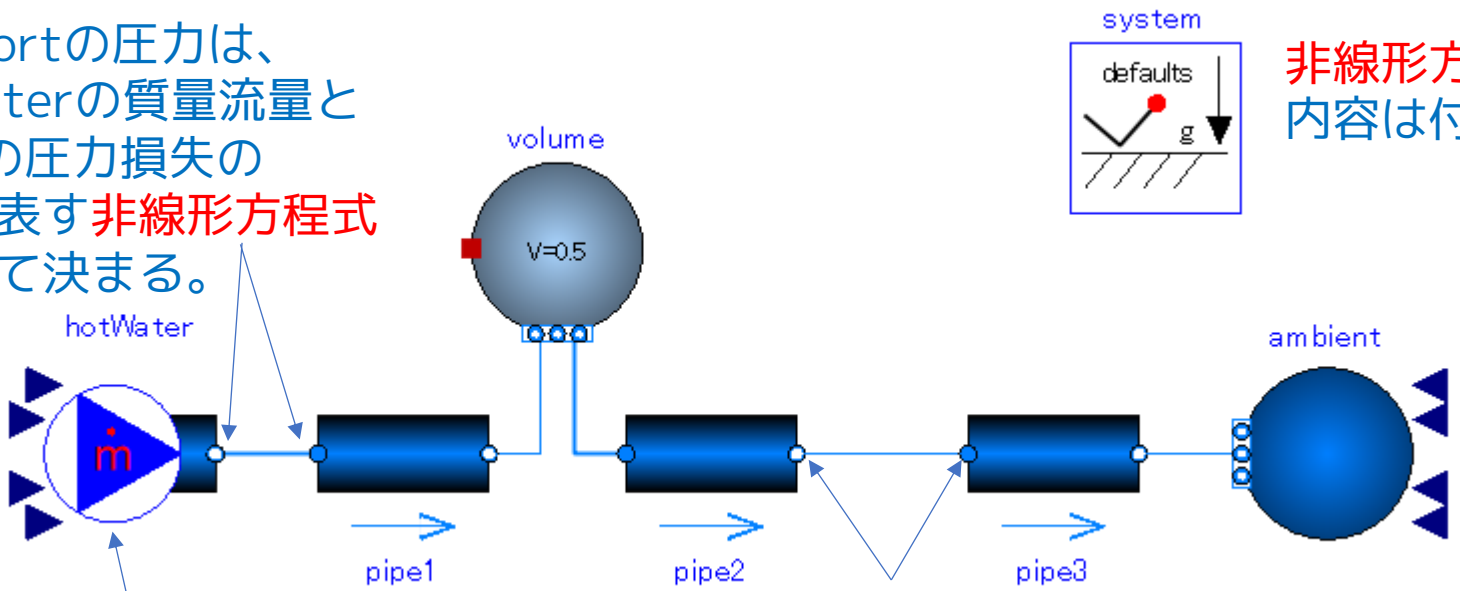




StaticPipeTest2
flow モデルを直接接続する

- volume1 を削除して pipe2 と pipe3 を直接接続する。
- hotWater を Boundary_pT から MassFlowSource_T に置き換える。

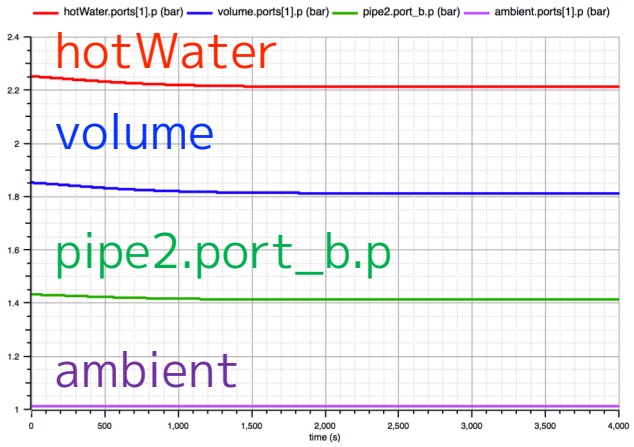
FluidPortの圧力は、
hotWaterの質量流量と
pipe1の圧力損失の
関係を表す非線形方程式
によって決まる。



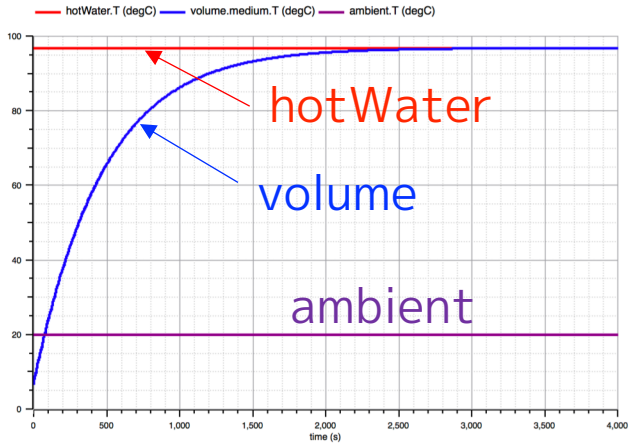
m_flow = 1.045 [kg/s]
T = 370 [K]
nPorts = 1

FluidPortの圧力と流量は、
pipe1とpipe2の
質量流量と圧力損失の関係を表す
連立非線形方程式によって決まる。

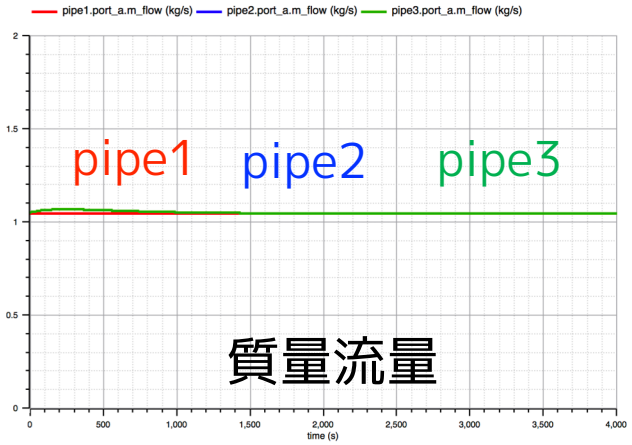
StaticPipeTest2

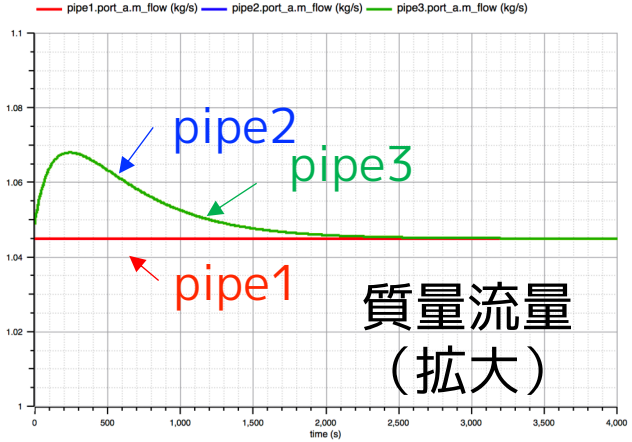


圧力

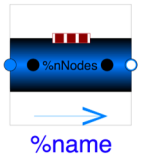


温度





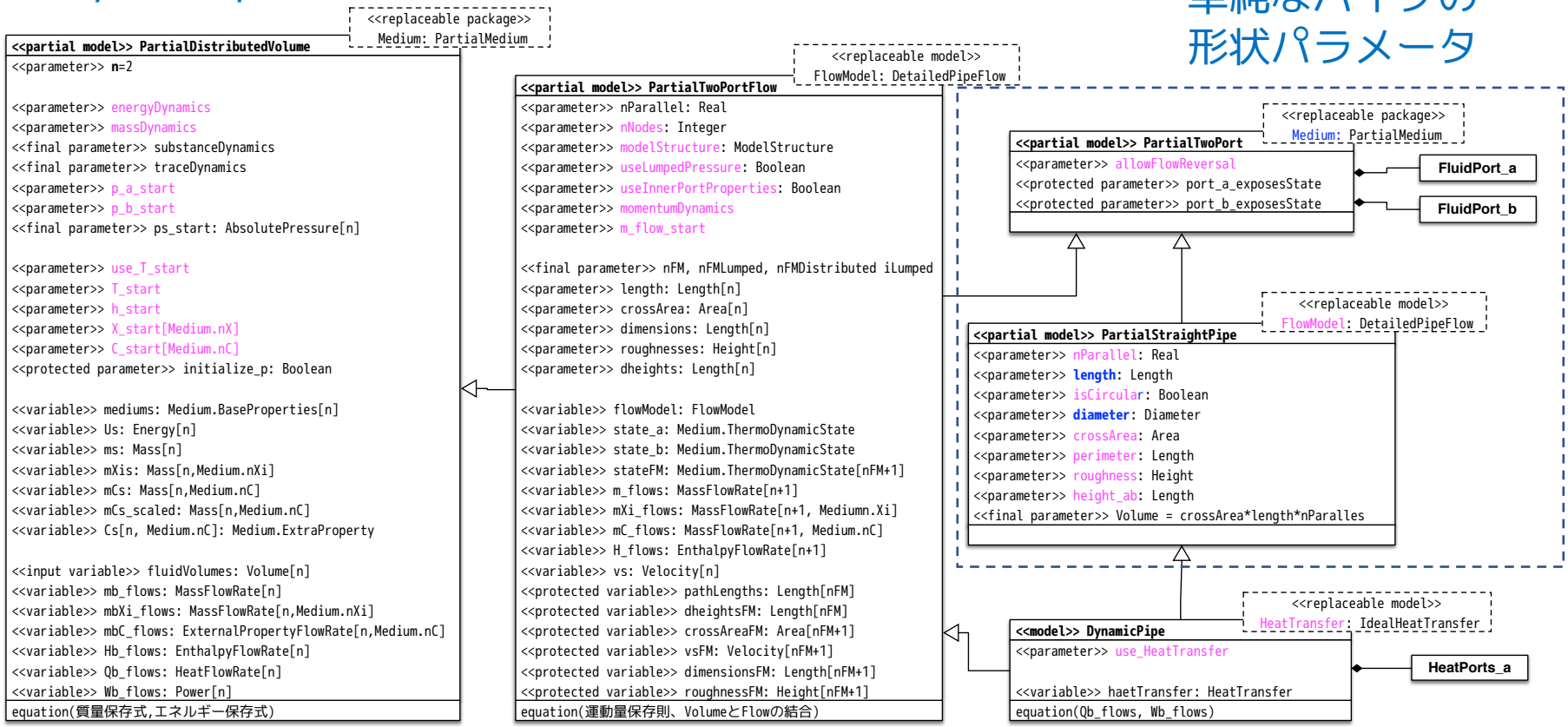
FluidExample4
DinamicPipe



必須入力パラメータ
選択入力パラメータ

DynaicPipeの継承関係

単純なパイプの
形状パラメータ



複数のVolumeモデル
（質量保存式、
エネルギー保存式）

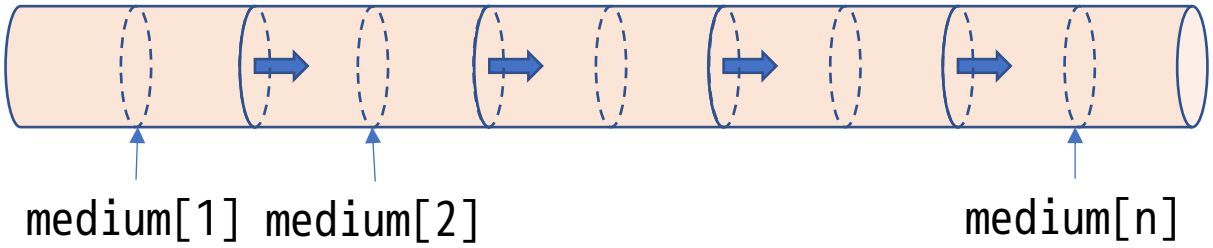
VolumeモデルとFlowモデルの
モデル構造
（運動量、質量保存式の実装）

パイプの形状パラメータと
モデルパラメータの対応付け
伝熱モデルの追加

DynamicPipe

PartialDistributedVolume

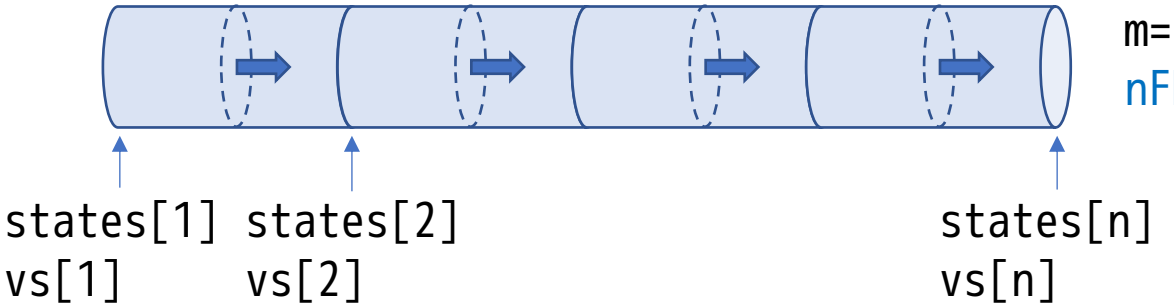
質量保存則、エネルギー保存則を満たす Volume モデル



PartialStaggeredFlowModel

運動量保存則を実装した Flow モデル

partialTwoPortFlow
でのモデルの数

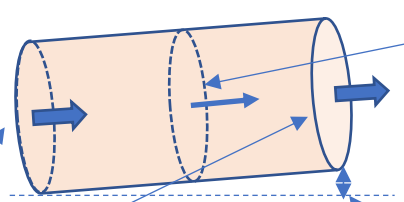


PartialTwoPortFlow
volumeモデルとflowモデルを互い違いに配置する。

DynamicPipe

volume モデル

状態変数
Us[n], ms[n], mXis[n,Xi], mCs[n,nC]



mediums[n]
vs[n]

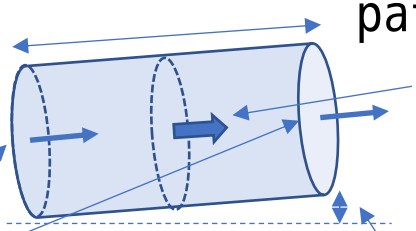
境界条件
m_flows[n+1]
mXi_flows[n+1,nXi]
mC_flows[n+1,nC]
H_flows[n+1]

パラメータ
dheights[n]: 高低差
lengths[n]:長さ
crossAreas[n]: 断面積
roughnesses[n]: 表面粗さ
dimensions[n]: 代表長さ

DynamicPipe

flowモデル

状態変数
Is[i] = m_flows[i]*pathLengths[i]



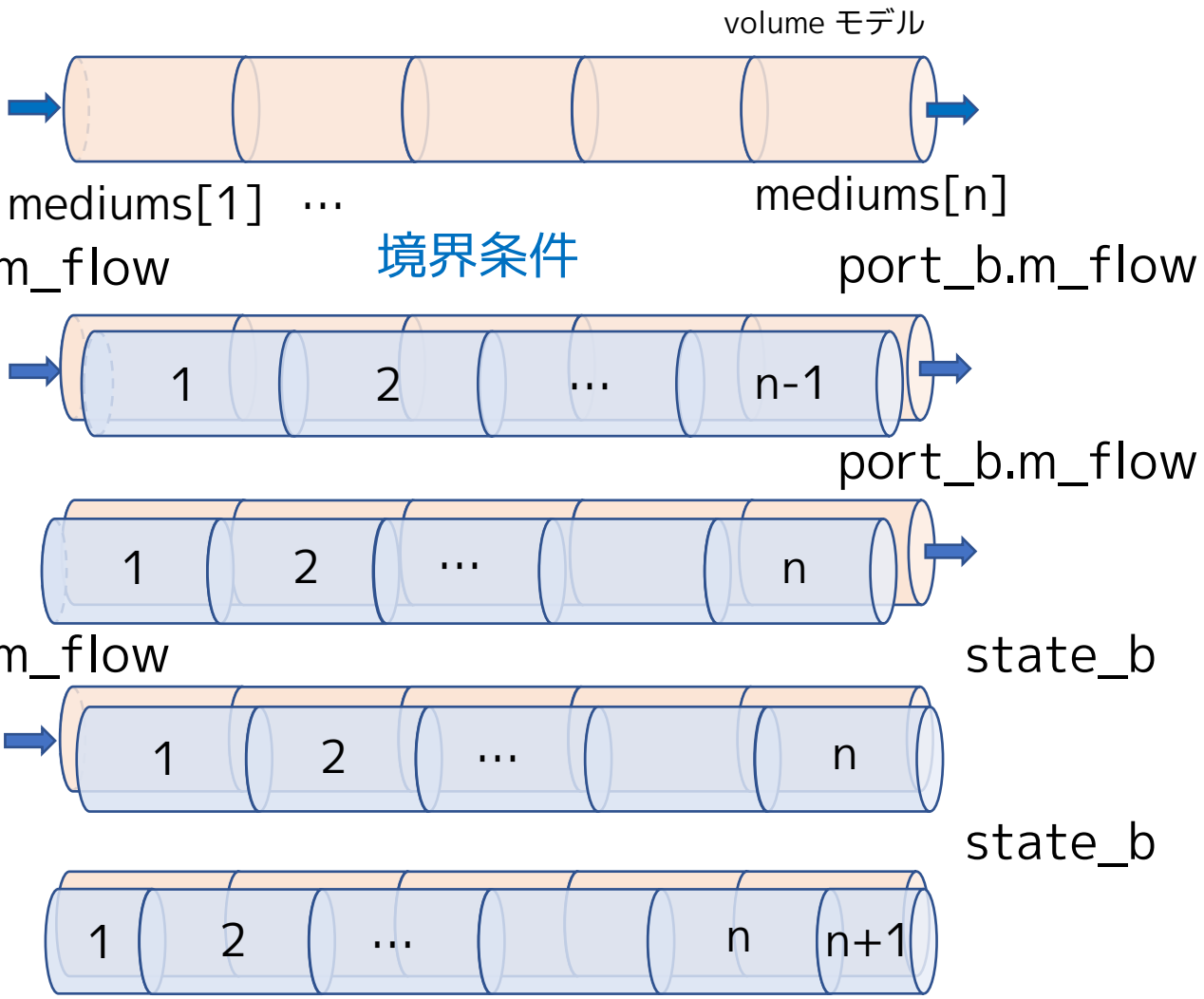
境界条件
statesFM[nFM+1]
vsFM[nFM+1]:流速

パラメータ
dheightsFM[nFM]: 高低差
crossAreasFM[nFM+1]:断面積
dimensionsFM[nFM+1]: 代表長さ
roughnessFM[nFM+1]: 表面粗さ

DynamicPipe

modelStructure

n = nNodes



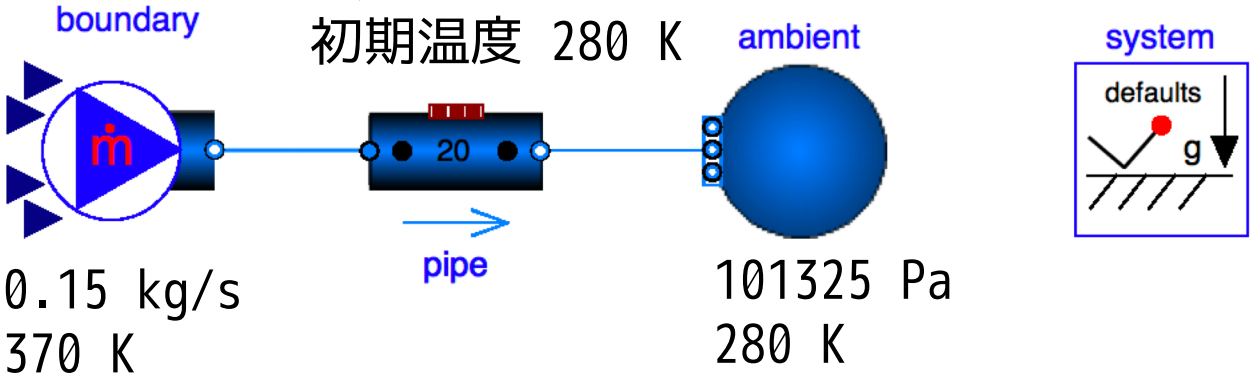
WaterExchange1～4
冷水の入ったパイプに熱湯を流し込む

内径 0.015 m, 長さ 20 m の冷水（初期温度 280 K (6.85℃))が入ったパイプに、熱湯(370 K (96.85℃))を流し込む。PipeをDaynamicPipeで表す。

パイプ内の流速が約 1.0 m/s 程度となるよう上流側の質量流量を 0.15 kg/s にする。

WaterExchange1 20 分割
WaterExchange2 100 分割

パイプ
内径 0.015 m
長さ 20 m
初期温度 280 K

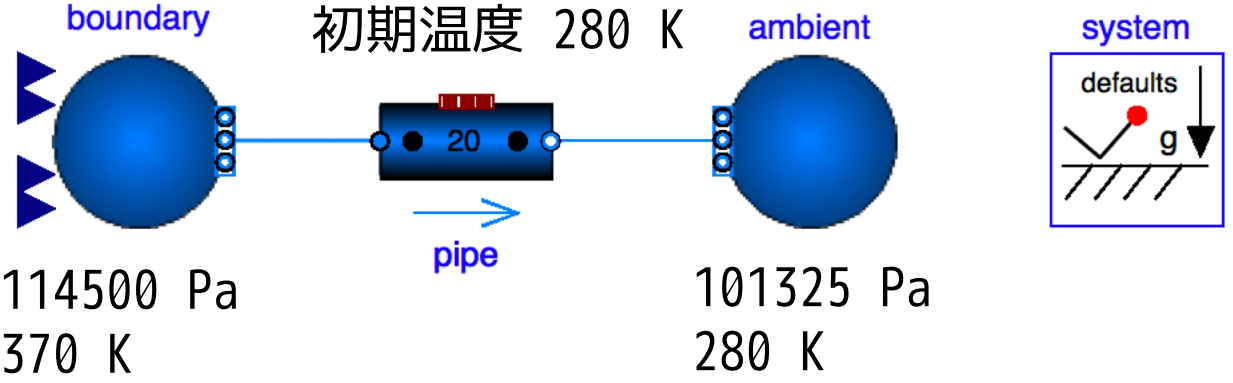


上流端は流量境界となるのでパイプ内部で volume モデル、下流端は熱力学的状態 state が確定しているのでパイプ内部で flow モデルににする。したがって、modelStructure は av_b を選択する。

WaterExchange1～4

パイプ内の流速が約 1.0 m/s 程度となるよう上流側の圧力を 114500 Pa に調整する。

WaterExchange3　20 分割　**パイプ**
WaterExchange4　100 分割　内径 0.015 m
長さ 20 m
初期温度 280 K



パイプの両端の温度と圧力が規定されており、熱力学的状態 state が確定している。両端の state を境界条件とする a_v_b を選択する。

WaterExchange1～4

```
model WaterExchange1
  replaceable package Medium = Media.Water.StandardWater;
  inner Modelica.Fluid.System system annotation( ...);
  Modelica.Fluid.Pipes.DynamicPipe pipe(redeclare package Medium = Medium,
    T_start = 280, diameter = 0.015,
    energyDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial,
    length = 20, m_flow_start = 0.15,
    massDynamics = Modelica.Fluid.Types.Dynamics.DynamicFreeInitial,
    modelStructure = Modelica.Fluid.Types.ModelStructure.av_b,
    momentumDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial,
    nNodes = 20, use_T_start = true) annotation( ...);
  Modelica.Fluid.Sources.FixedBoundary ambient(redeclare package Medium = Medium,
    T = 280, nPorts = 1, p = 101325) annotation( ...);
  Modelica.Fluid.Sources.MassFlowSource_T boundary(redeclare package Medium = Medium,
    T = 370, m_flow = 0.15, nPorts = 1) annotation( ...);
equation
  connect(boundary.ports[1], pipe.port_a) annotation( ...);
  connect(pipe.port_b, ambient.ports[1]) annotation( ...);
  annotation( ...);
end WaterExchange1;
```

- Start Time = 0
- Stop Time = 60
- Number of Intervals = 150
- Non Linear Solver = hybrid

1タイムステップに流れる距離がpathLengthの 1/2 以下になるようにする。

WaterExchange1～4

```
model WaterExchange2
  replaceable package Medium = Media.Water.StandardWater;
  inner Modelica.Fluid.System system annotation( ...);
  Modelica.Fluid.Pipes.DynamicPipe pipe(redeclare package Medium = Medium,
    T_start = 280, diameter = 0.015,
    energyDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial,
    length = 20, m_flow_start = 0.15,
    massDynamics = Modelica.Fluid.Types.Dynamics.DynamicFreeInitial,
    modelStructure = Modelica.Fluid.Types.ModelStructure.av_b,
    momentumDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial,
    nNodes = 100, use_T_start = true) annotation( ...);
  Modelica.Fluid.Sources.FixedBoundary ambient(redeclare package Medium = Medium,
    T = 280, nPorts = 1, p = 101325) annotation( ...);
  Modelica.Fluid.Sources.MassFlowSource_T boundary(redeclare package Medium = Medium,
    T = 370, m_flow = 0.15, nPorts = 1) annotation( ...);
equation
  connect(boundary.ports[1], pipe.port_a) annotation( ...);
  connect(pipe.port_b, ambient.ports[1]) annotation( ...);
  annotation( ...);
end WaterExchange2;
```

- Start Time = 0
- Stop Time = 60
- Number of Intervals = 750
- Non Linear Solver = hybrid

WaterExchange1～4

```
model WaterExchange3
  replaceable package Medium = Media.Water.StandardWater;
  inner Modelica.Fluid.System system annotation( ...);
  Modelica.Fluid.Sources.Boundary_pT boundary(redeclare package Medium = Medium,
    T = 370, nPorts = 1, p = 114500) annotation( ...);
  Modelica.Fluid.Pipes.DynamicPipe pipe(redeclare package Medium = Medium,
    T_start = 280, diameter = 0.015,
    energyDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial,
    length = 20, m_flow_start = 0.1,
    massDynamics = Modelica.Fluid.Types.Dynamics.DynamicFreeInitial,
    modelStructure = Modelica.Fluid.Types.ModelStructure.a_v_b,
    momentumDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial,
    nNodes = 20, use_T_start = true) annotation( ...);
  Modelica.Fluid.Sources.FixedBoundary ambient(redeclare package Medium = Medium,
    T = 280, nPorts = 1, p = 101325) annotation( ...);
equation
  connect(pipe.port_b, ambient.ports[1]) annotation( ...);
  connect(boundary.ports[1], pipe.port_a) annotation( ...);
  annotation( ...);
end WaterExchange3;
```

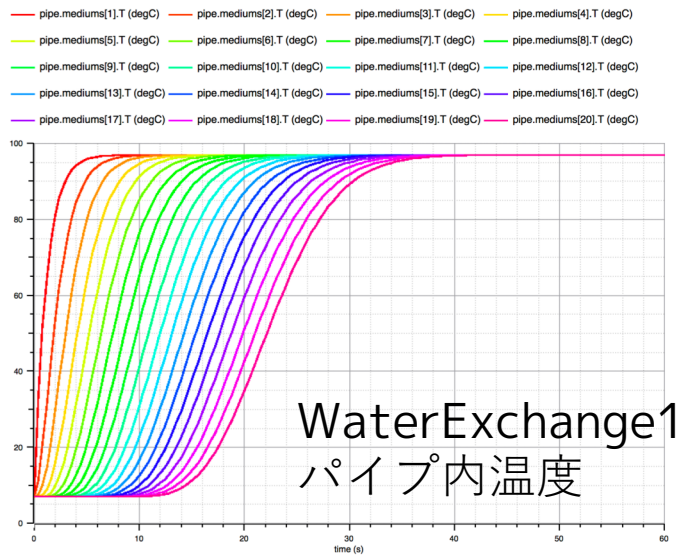
- Start Time = 0
- Stop Time = 60
- Number of Intervals = 150
- Non Linear Solver = hybrid

WaterExchange1～4

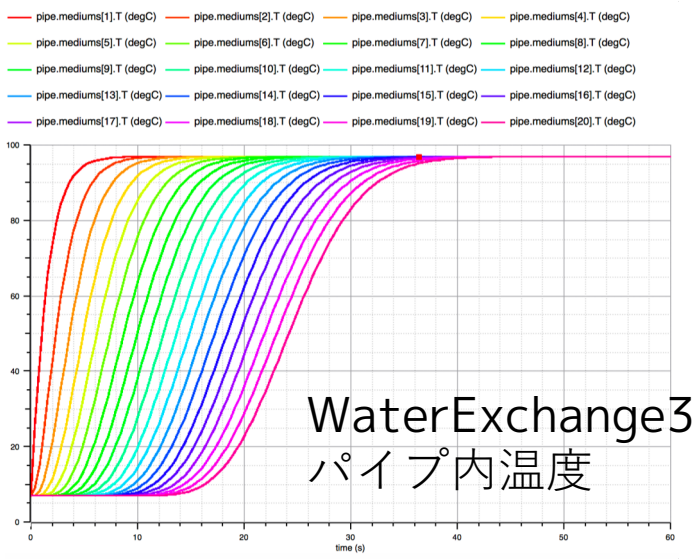
```
model WaterExchange4
  replaceable package Medium = Media.Water.StandardWater;
  inner Modelica.Fluid.System system annotation( ...);
  Modelica.Fluid.Sources.Boundary_pT boundary(redeclare package Medium = Medium,
    T = 370, nPorts = 1, p = 114500) annotation( ...);
  Modelica.Fluid.Pipes.DynamicPipe pipe(redeclare package Medium = Medium,
    T_start = 280, diameter = 0.015,
    energyDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial,
    length = 20, m_flow_start = 0.1,
    massDynamics = Modelica.Fluid.Types.Dynamics.DynamicFreeInitial,
    modelStructure = Modelica.Fluid.Types.ModelStructure.a_v_b,
    momentumDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial,
    nNodes = 100, use_T_start = true) annotation( ...);
  Modelica.Fluid.Sources.FixedBoundary ambient(redeclare package Medium = Medium,
    T = 280, nPorts = 1, p = 101325) annotation( ...);
equation
  connect(pipe.port_b, ambient.ports[1]) annotation( ...);
  connect(boundary.ports[1], pipe.port_a) annotation( ...);
  annotation( ...);
end WaterExchange4;
```

- Start Time = 0
- Stop Time = 60
- Number of Intervals = 750
- Non Linear Solver = hybrid

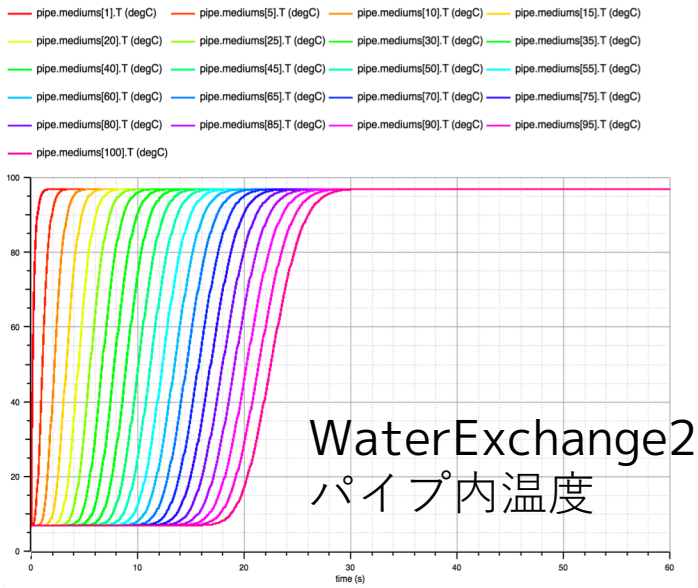
WaterExchange1～4



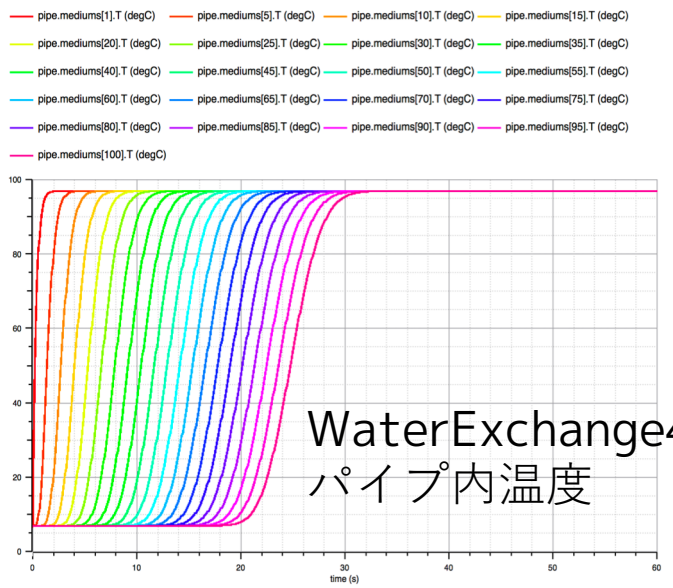
WaterExchange1
パイプ内温度



WaterExchange3
パイプ内温度



WaterExchange2
パイプ内温度

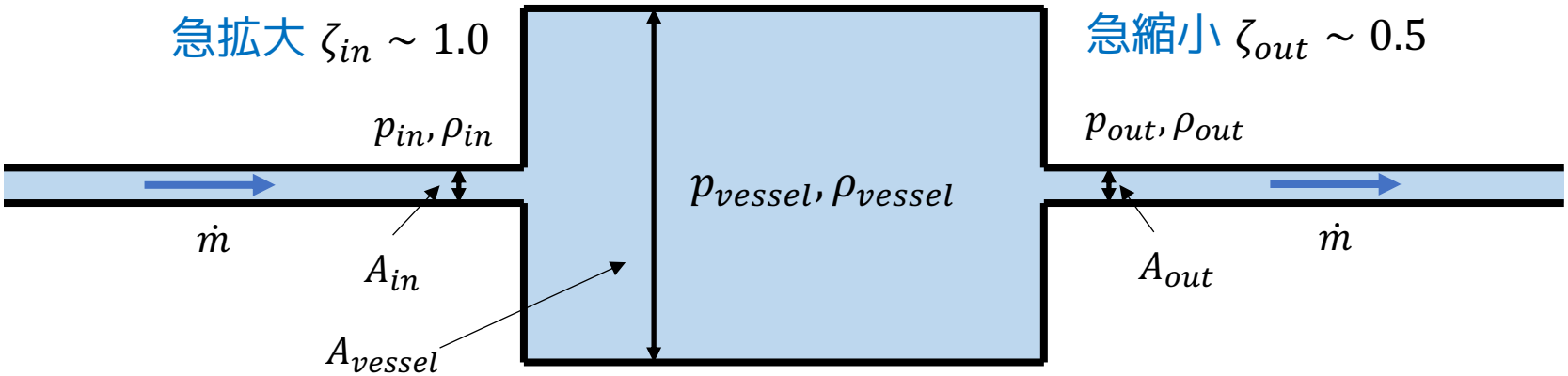


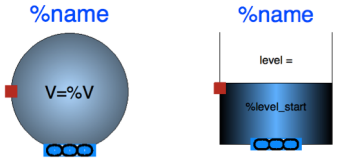
WaterExchange4
パイプ内温度

FluidExample5
VesselPortsData FluidPort の特性データ

<<record>> **Modelica.Fluid.Vessels.BaseClasses.VesselPortsData**
<<parameter>> **diameter**: SI.Diameter <<parameter>> **height**=0: SI.Height <<parameter>> **zeta_out**(min=0)=0.5: Real <<parameter>> **zeta_in**(min=0)=1.04: Real

$diameter$: 内径 [m]
$height$: 底面からの高さ [m]
ζ_{out} : 流出時の圧力損失係数
ζ_{in} : 流入時の圧力損失係数

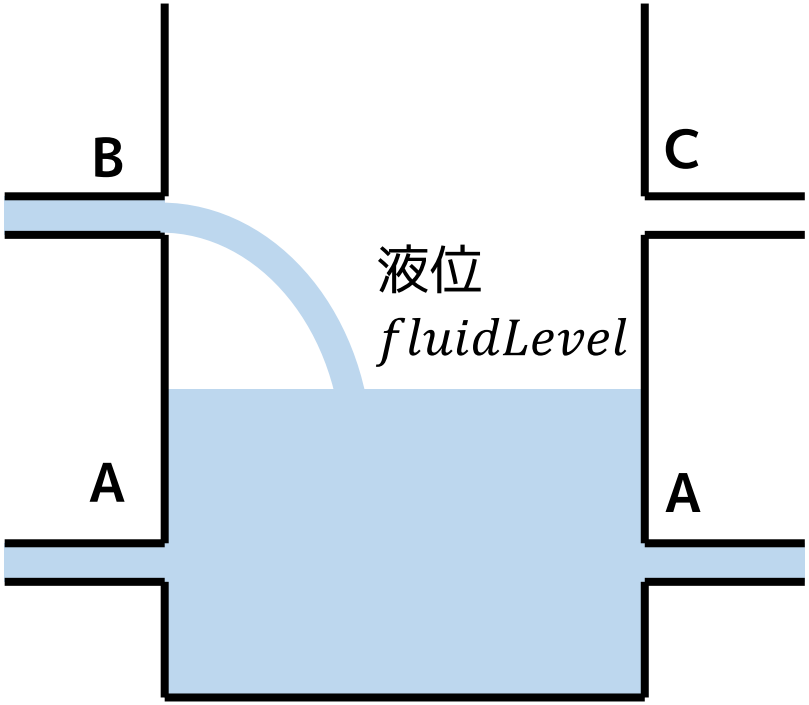




ClosedVolume　　OpenTank

- use_portsData = true で有効
- ClosedVolume では height は無効

VesselPortsData



A. regularFlow

条件 $fluidLevel \geq height_i$

処理 use_portsData = true なら

$p_i = p_{port}$ (付録２参照)

use_portsData=false なら

$p_i = p_{vessel_i}$

$s_i = fluidLevel - height_i$

s_i = s[i]: 流入を表す状態変数

use_portsData = false の場合、$height_i = 0$ となるため、regularFlow になる。

VesselPortsData

B. inFlow

条件 $not\ regularFlow\ and\ (s_i > 0\ \ or\ height_i \geq fluidLevel_{max})$

処理 $p_i = p_{vessel_i}$

$s_i = \dot{m}_i$

C. noFlow

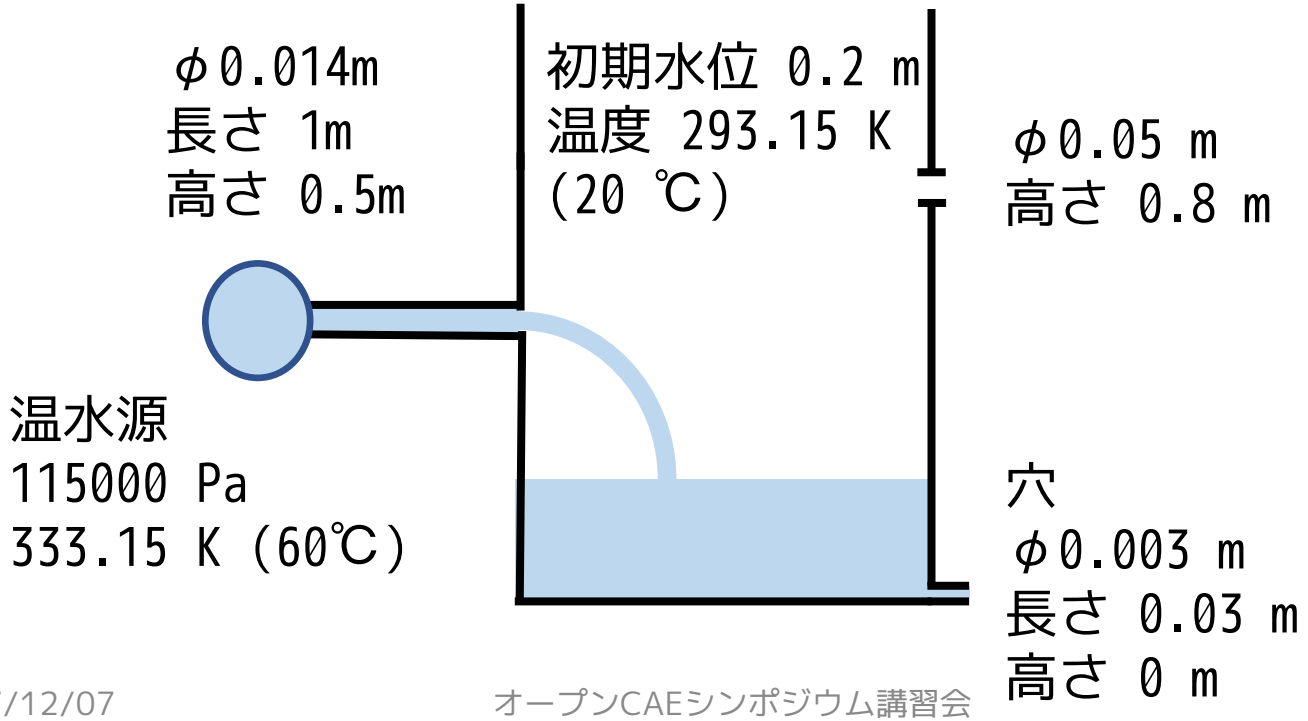
条件 その他の場合

処理 $\dot{m}_i = 0$

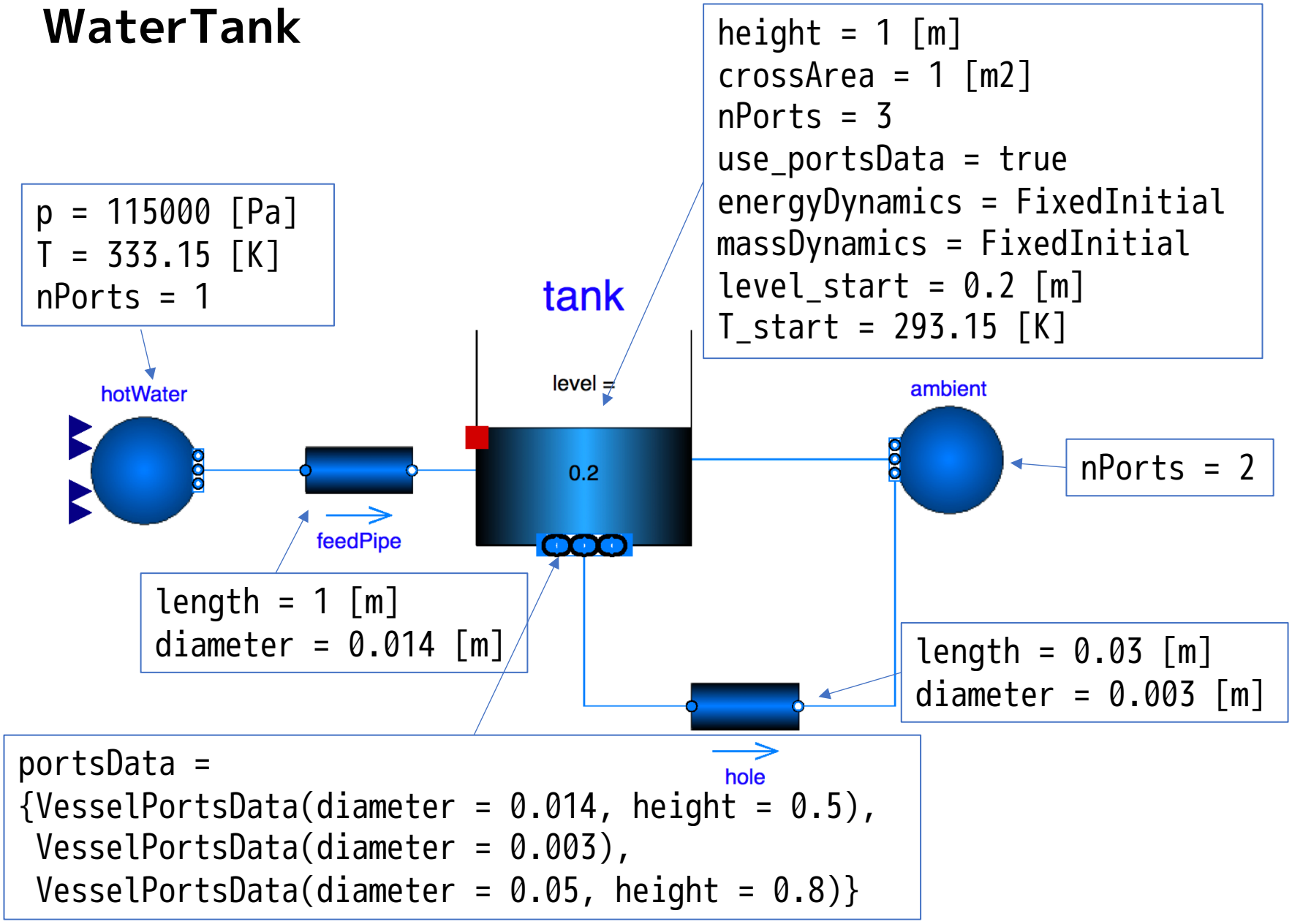
$$s_i = \frac{p_i - p_{vessel_i}}{p_{default}} \cdot (height_i - fluidLevel)$$

WaterTank
水の入ったタンクにお湯を注ぎ込む

- タンクに温度 20 ℃、水位 0.2 m の水が入っている。
- 高さ 50 cm の位置に給水口があり 60 ℃の温水が注がれている。
- タンクの底には径 3 mm、長さ 3 cm の穴が空いている。
- タンクの壁面には径 5 cm、高さ 0.8 mの排水口がある。
- 給水口は径 14 mm、長さ 1 m のパイプで、温度 60 ℃、圧力 115 kPa の温水源と接続されている。



WaterTank


height = 1 [m]
crossArea = 1 [m2]
nPorts = 3
use_portsData = true
energyDynamics = FixedInitial
massDynamics = FixedInitial
level_start = 0.2 [m]
T_start = 293.15 [K]
p = 115000 [Pa]
T = 333.15 [K]
nPorts = 1
tank
level
hotWater
ambient
0.2
nPorts = 2
feedPipe
length = 1 [m]
diameter = 0.014 [m]
length = 0.03 [m]
diameter = 0.003 [m]
hole
portsData =
{VesselPortsData(diameter = 0.014, height = 0.5),
VesselPortsData(diameter = 0.003),
VesselPortsData(diameter = 0.05, height = 0.8)}

WaterTank

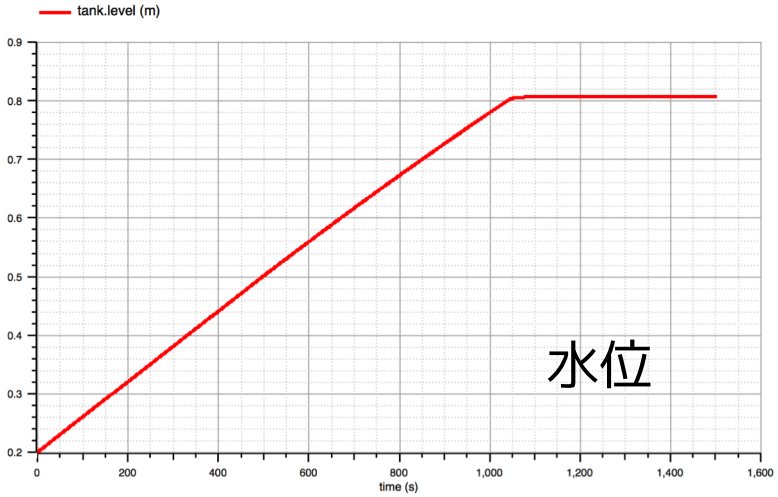
```
model WaterTank
  replaceable package Medium = Modelica.Media.Water.StandardWater;
  import Modelica.Fluid.Vessels.BaseClasses.VesselPortsData;
  Modelica.Fluid.Vessels.OpenTank tank(redeclare package Medium = Medium,
    T_start = 293.15,crossArea = 1,
    energyDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial,
    height = 1, level_start = 0.2,
    massDynamics = Modelica.Fluid.Types.Dynamics.FixedInitial, nPorts = 3,
    portsData
     = {VesselPortsData(diameter = 0.014, height = 0.5),
        VesselPortsData(diameter = 0.003),
        VesselPortsData(diameter = 0.05, height = 0.8)})
    annotation( ...);
  Modelica.Fluid.Sources.Boundary_pT hotWater(redeclare package Medium = Medium,
    T = 333.15, nPorts = 1, p = 115000)
    annotation( ...);
  Modelica.Fluid.Pipes.StaticPipe pipe(redeclare package Medium = Medium,
    diameter = 0.014, length = 1)  annotation( ...);
  Modelica.Fluid.Sources.FixedBoundary ambient(redeclare package Medium = Medium,
    nPorts = 2)  annotation( ...);
  Modelica.Fluid.Pipes.StaticPipe hole(redeclare package Medium = Medium,
    diameter = 0.003, length = 0.03)  annotation( ...);
  inner Modelica.Fluid.System. system annotation( ...);
```

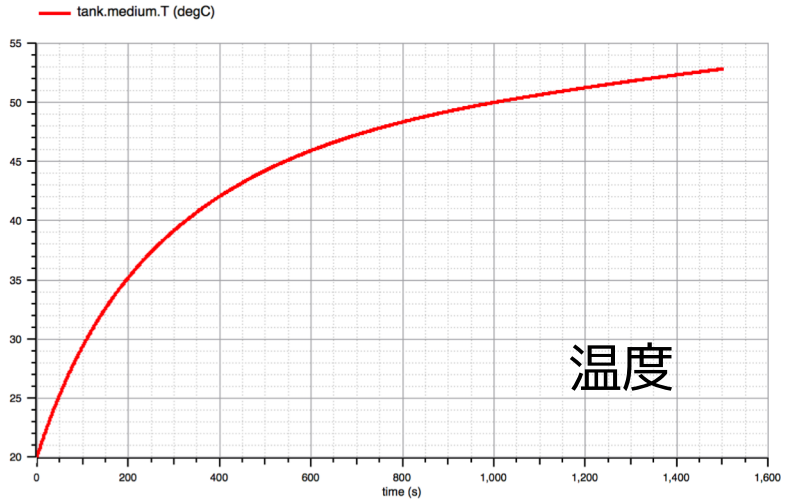
WaterTank

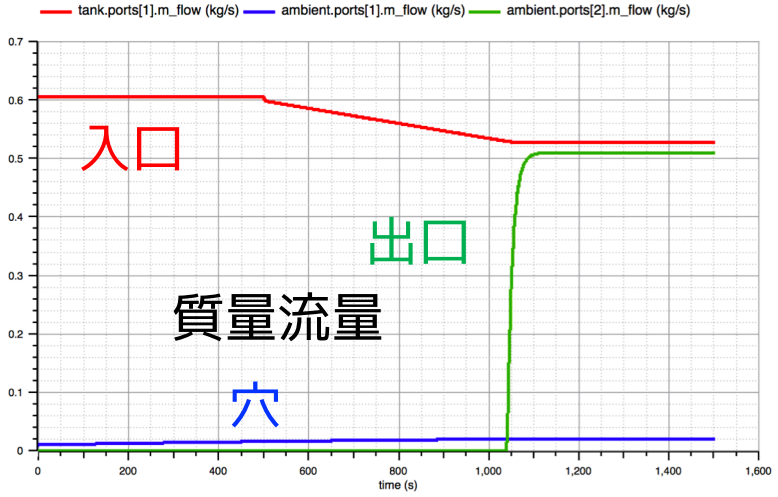
```
equation
  connect(tank.ports[3], ambient.ports[2]) annotation( ...);
  connect(hole.port_b, ambient.ports[1]) annotation( ...);
  connect(hole.port_a, tank.ports[2]) annotation( ...);
  connect(pipe.port_b, tank.ports[1]) annotation( ...);
  connect(pipe.port_a, hotWater.ports[1]) annotation( ...);
  annotation(
end WaterTank;
```

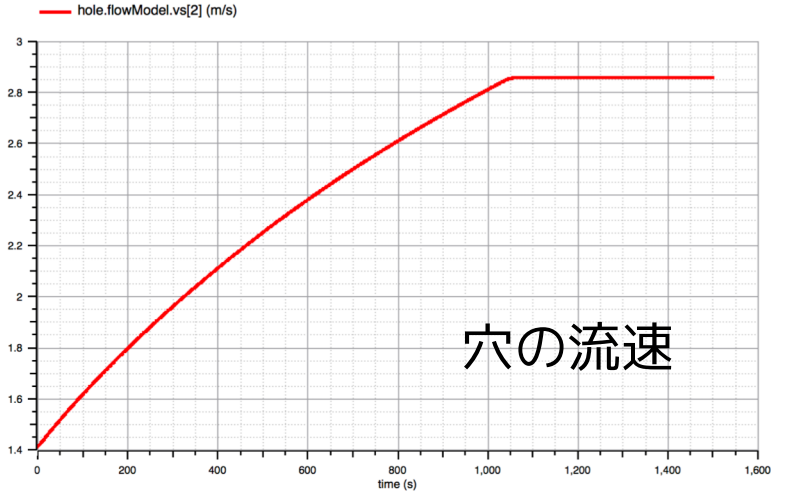
WaterTank

シミュレーション結果





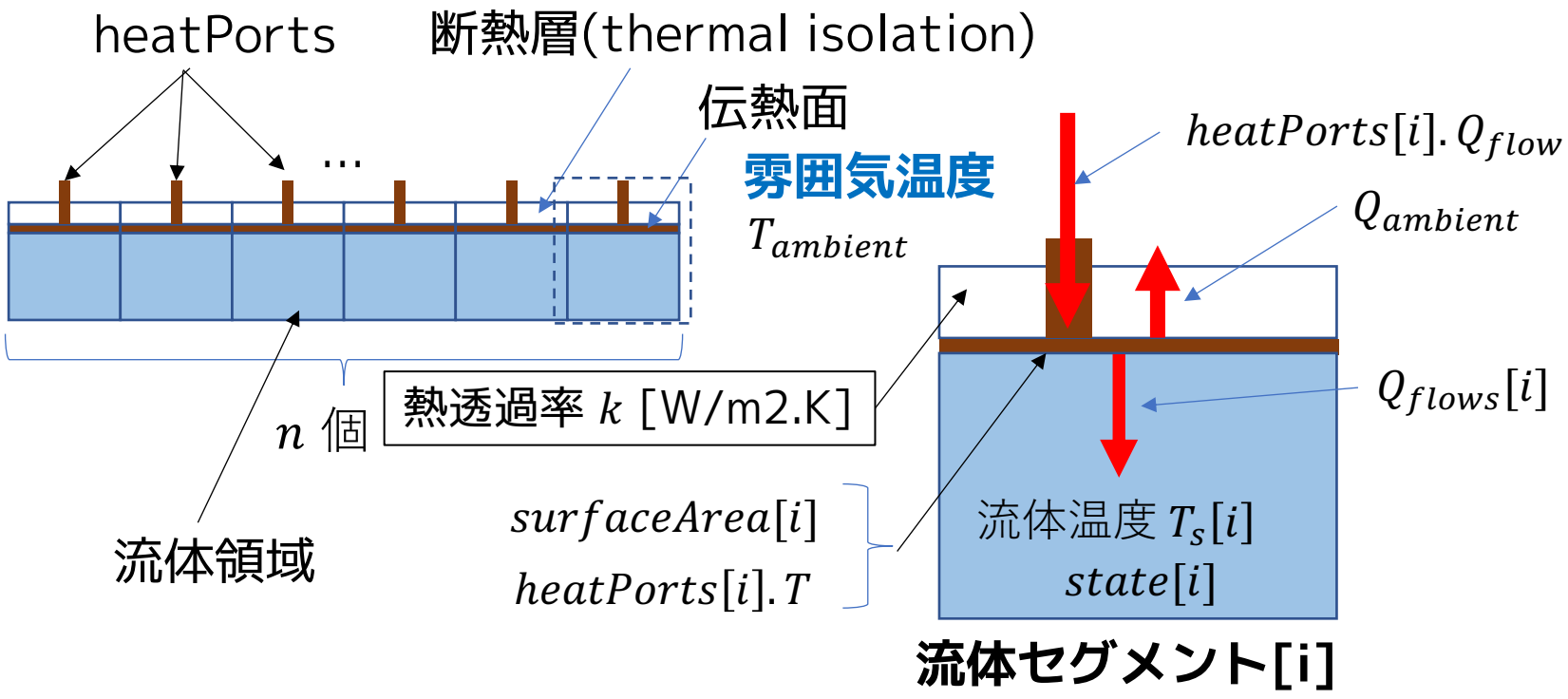




FluidExample6
HeatTransfer モデル

PartialHeatTransfer

- 伝熱のベースモデル
- 流体領域、伝熱面、断熱層を n 個のセグメントに分割し流体に供給される熱量を計算する。



HeatTransfer モデル

$use\_k = true$ の場合

$$Q_{flows}[i] = heatPorts[i].Q_{flow} - Q_{ambient}$$
$$= heatPorts[i].Q_{flow}$$
$$+ k \cdot surfaceAreas[i] \cdot (T_{ambient} - heatPorts[i].T)$$

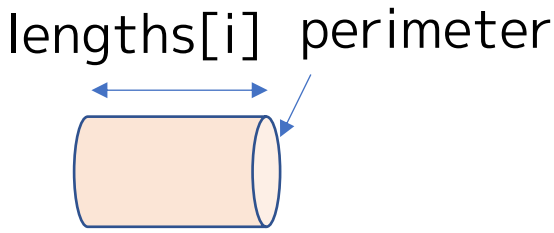
$use\_k = false$ の場合、雰囲気への熱流量は無し

$$Q_{flows}[i] = heatPorts[i].Q_{flow}$$

IdealHeatTransfer

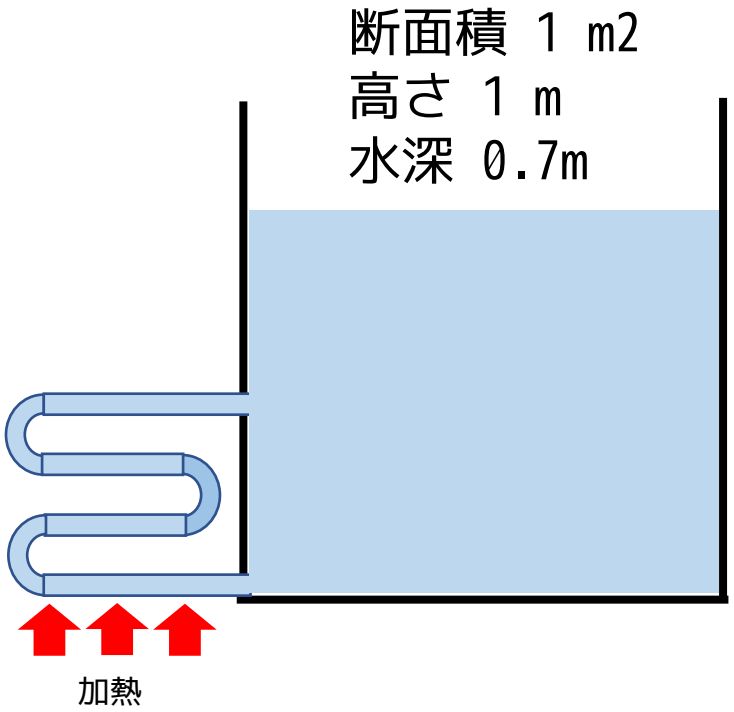
$$heatPorts[i].T = T_s[i]$$ **伝熱面温度 = 流体温度**

DynamicPipe

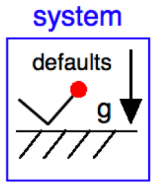


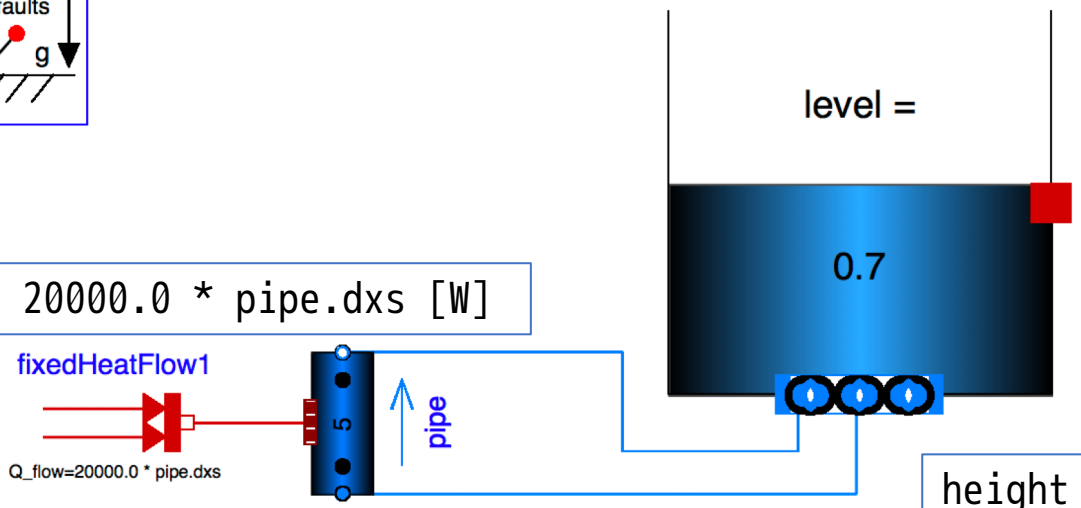

```
surfaceAreas=perimeter*lengths
```

WaterHeating
パイプを加熱してタンク内の水を温める



WaterHeating






```
Q_flow = 20000.0 * pipe.dxs [W]
```

```
length = 5 [m]
diameter = 0.03 [m]
height_ab = 0.3
energyDynamics = FixedInitial
massDynamics = FixedInitial
momentumDynamics = FixedInitial
use_HeatTransfer = true
T_start = 293.15 [K]
m_flow_start = 0.02 [kg/s]
nNodes = 5
modelStructure = a_v_b
```

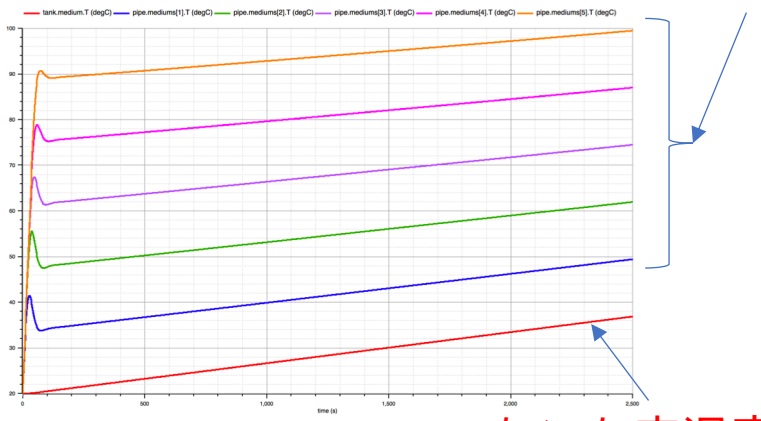
```
height = 1 [m]
crossArea = 1 [m]
nPorts = 2
use_portsData = true
energyDynamics = FixedInitial
massDynamics = FixedInitial
level_start = 0.7 [m]
use_T_start = true
T_start = 293.15 [K]
```

```
portsData
={VesselPortsData(diameter = 0.03),
VesselPortsData(diameter = 0.03, height = 0.3)}
```

WaterHeating

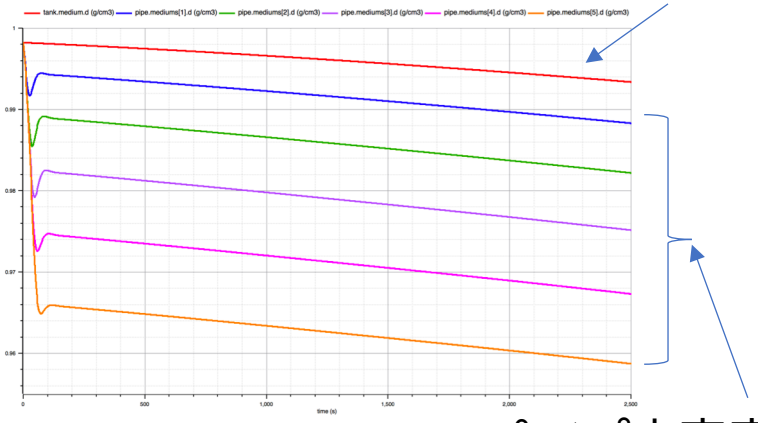
シミュレーション結果

パイプ内温度

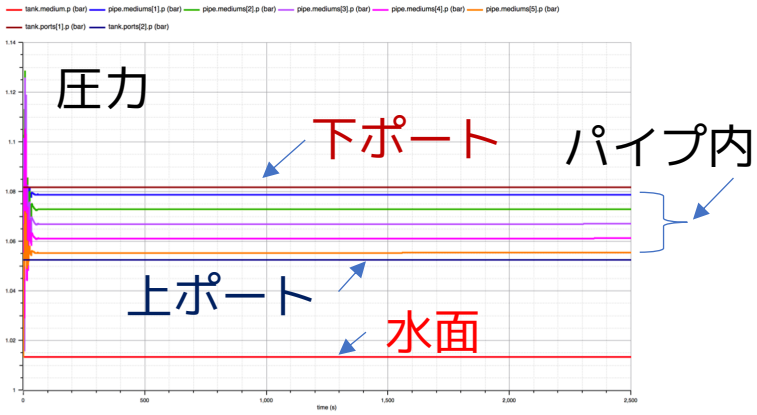


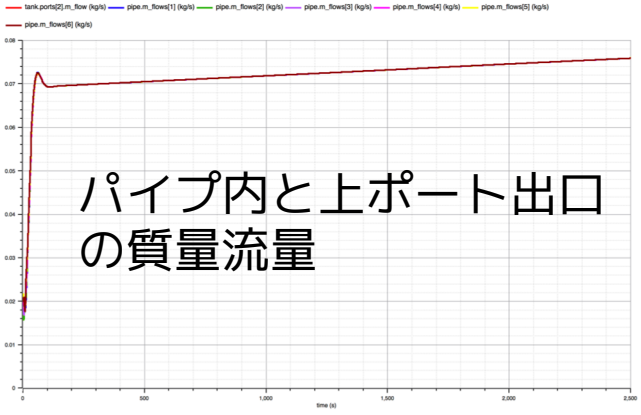
タンク内温度

タンク内密度



パイプ内密度





IncompressibleFluidNetwork
Modelica.Fluid.Examples のモデルをコピーする

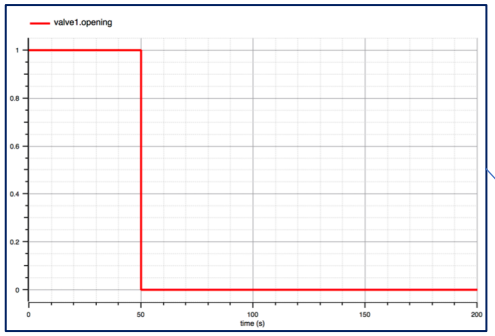
Modelica.Fluid.Examples.IncompressibleFluidNetwork をユーザが作った package の中にコピーする。

スコープの問題でシミュレーションできない場合は、import文を記述して使用するコンポーネントが参照できるようにする。

```
package MyPackage
  import Modelica.Fluid.Types;
  import Modelica.Fluid.Pipes;
  import Modelica.Fluid.Valves;
  import Modelica.Fluid.Sources;
  import Modelica.Thermal;

  model IncompressibleFluidNetwork "Multi-way connections of pipes and incompressible
medium model"
...
```

IncompressibleFluidNetwork

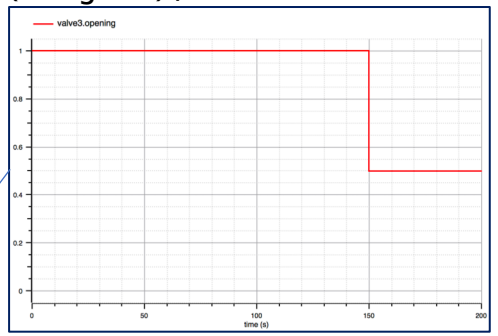
2個の圧力源と11本のパイプ、3個のバルブから構成される非圧縮性流体のネットワークモデル



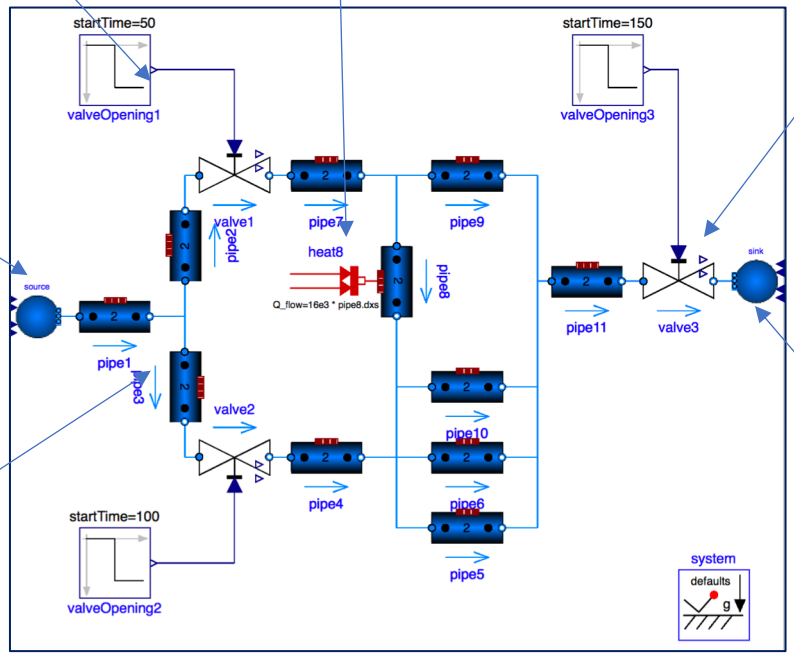
heat8
- 16e3*pipe8.dxs [W]

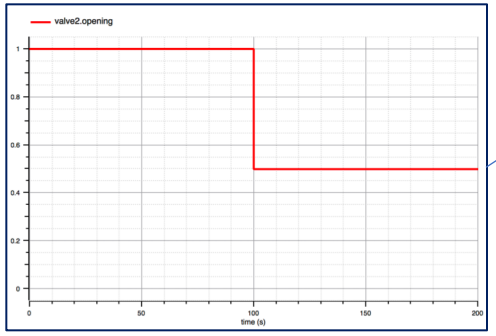
作動流体
- Glycol47

```
final parameter Real[n] dxs = lengths/sum(lengths);
```



source
- 5.0e5 [Pa]
- 300 [K]

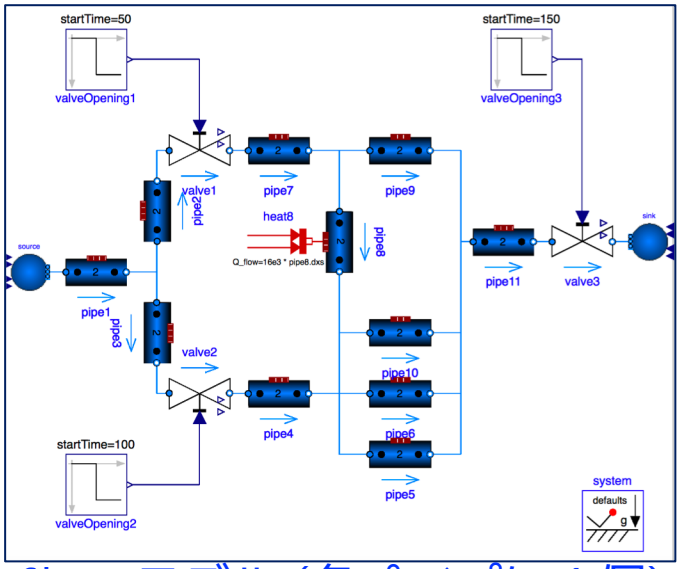




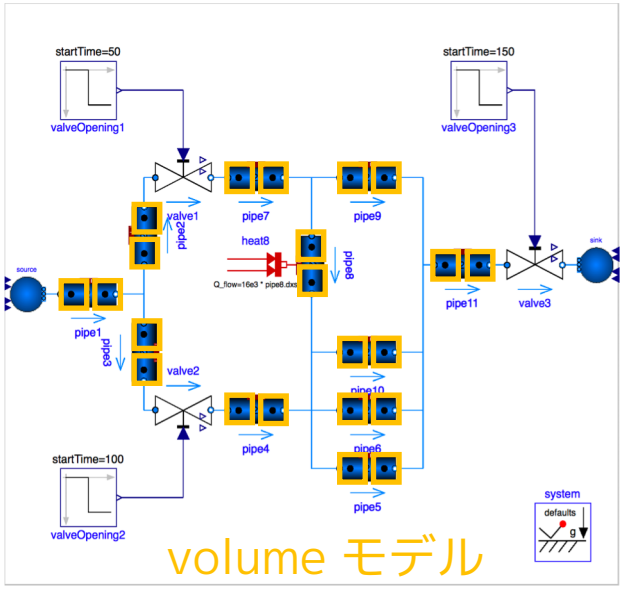
sink
- 1.0e5 [Pa]
- 300 [K]

IncompressibleFluidNetwork

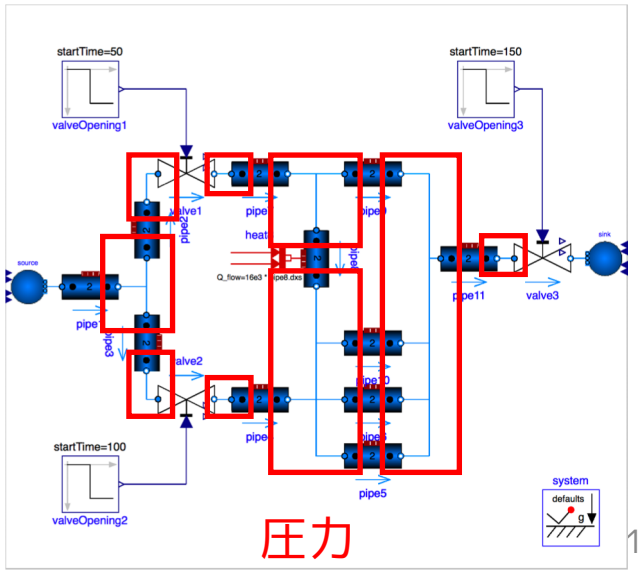
- 11本のDynamicPipeは nNodes = 2 で、合計22個の volume モデル（右図の黄色い四角）がある。
- DynamicPipe の ModelStructure は av_vb なので、パイプどうしの接続ではvolume モデルが直接接続されるので、圧力が等しくなり、９個の状態（右下の赤い四角）のみとなる。



flow モデル (各パイプに１個)

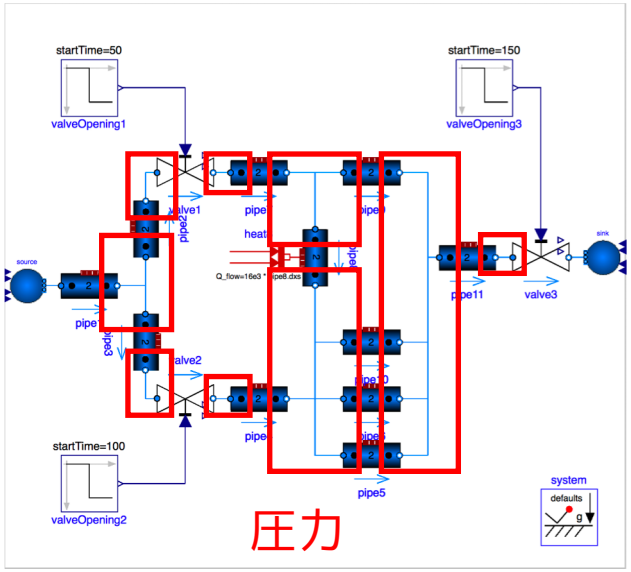


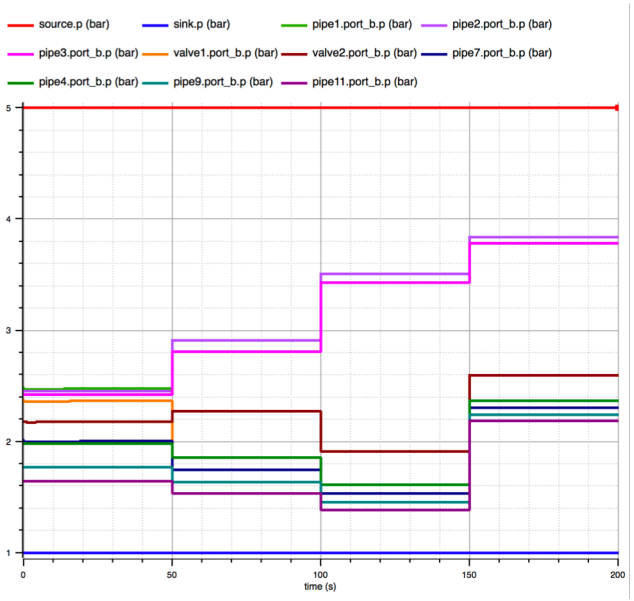
volume モデル

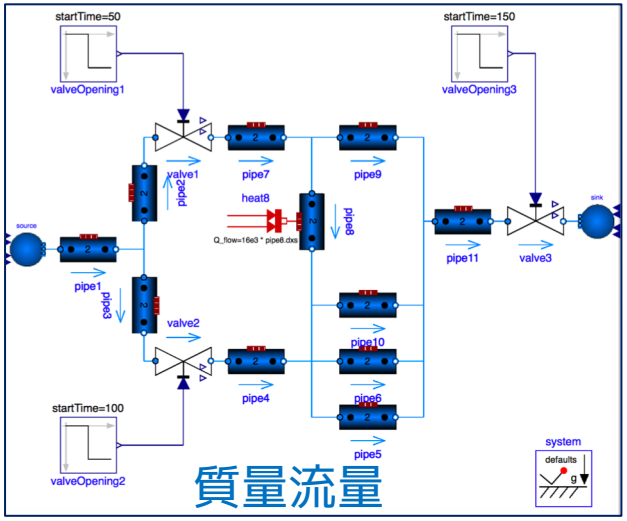


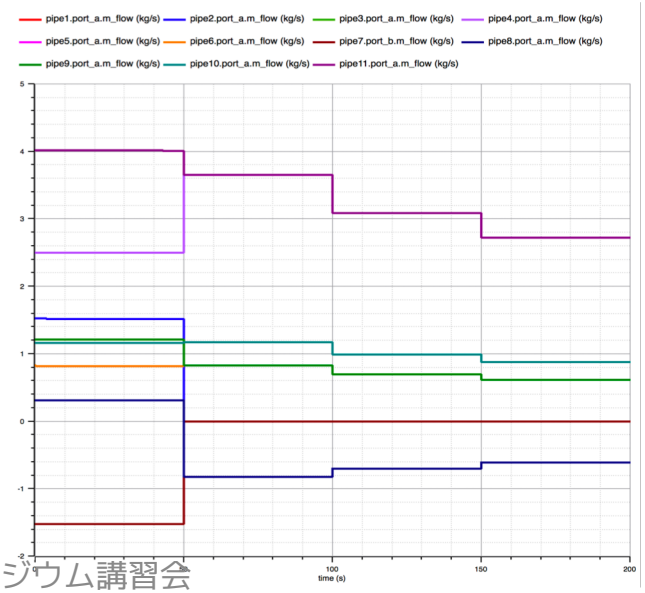
圧力

IncompressibleFluidNetwork

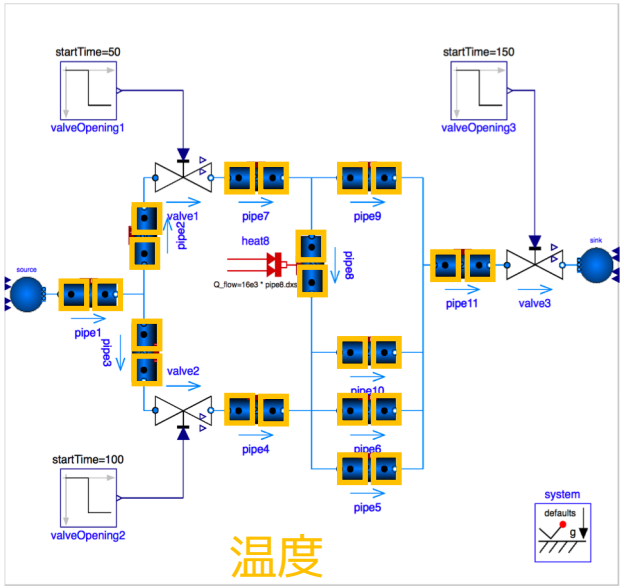


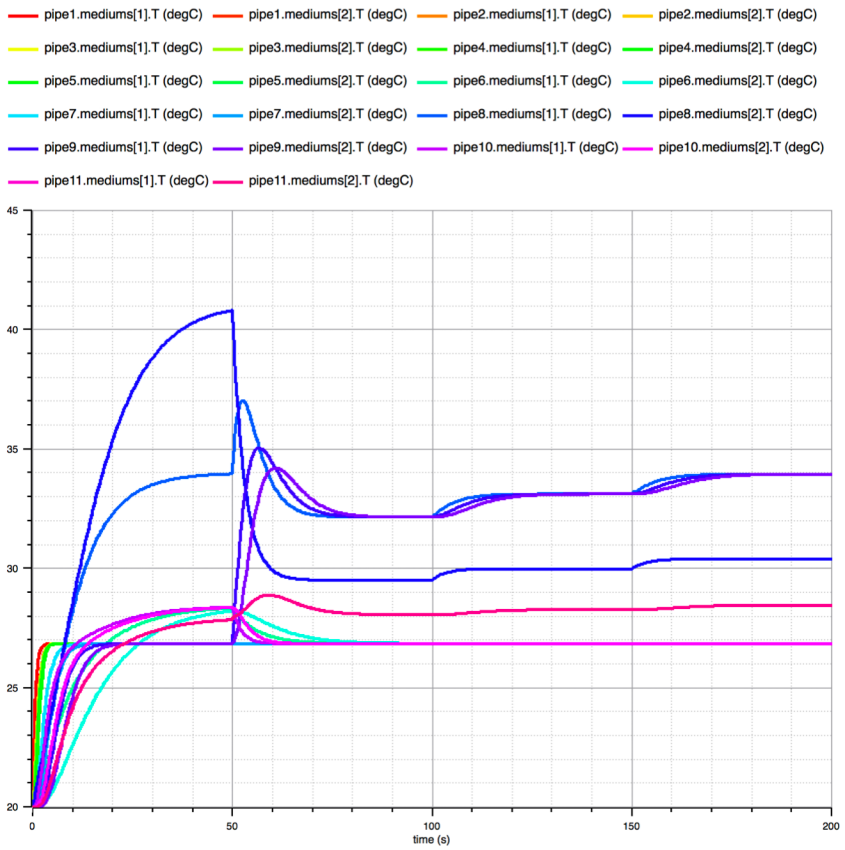






IncompressibleFluidNetwork





まとめ

Modelica.Fluid ライブラリに関する以下の内容について例題を用いて概念や使い方を述べた。

- FluidPort
- volume モデル
- flow モデル
- DynamicPipe
- VesselPortsData
- HeatTransfer モデル

- The purpose of this document is introducing Media and Fluid Libraries 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 and License of this document are written in the last page.

Modelica Standard Library License

https://github.com/modelica/ModelicaStandardLibrary/blob/master/LICENSE