

Modelica Standard Library (MSL) の SingleGasNasa 純物質理想気体のモデリングについて

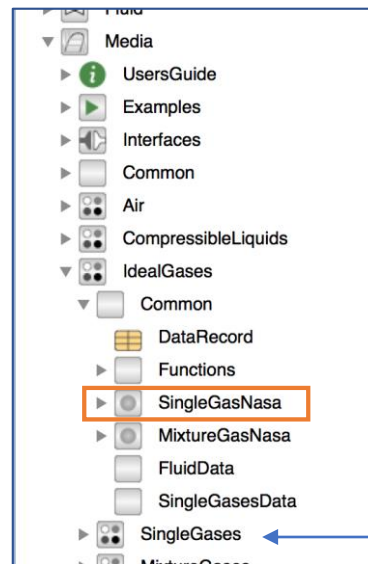
An Introduction of Modelica.Media.IdealGases.Common.SingleGasNasa

2018/09/29 第4回 Modelica ライブラリ勉強会
finback

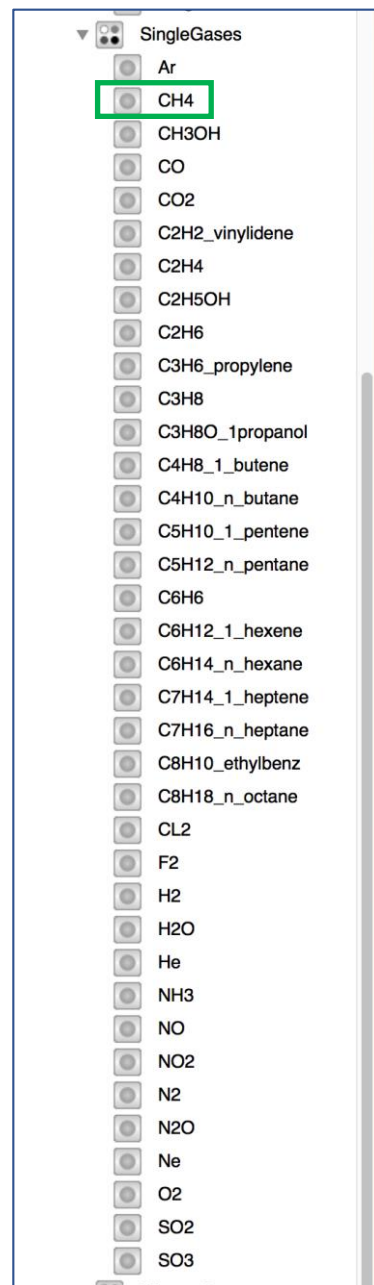
2018/10/07 加筆修正
2018/11/19 加筆修正

SingleGasNasa (純物質理想気体)

- [package の基本構成、継承関係とクラス](#)
 - [Types](#)
 - [物性値のタイプ](#)
 - [FluidConstants レコード](#)
- [定数](#)
 - [定数の継承と導入](#)
 - [熱物性データ](#)
- [ThermodynamicState](#)
- [BaseProperties](#)
 - [State Selection \(状態変数選択\)](#)
 - [BaseProperties の方程式](#)
 - [比エンタルピの計算方法](#)
 - [比エンタルピを関数にする理由](#)
- [setState XXX](#)
 - [setStte XXX 関数](#)
 - [温度を求める関数](#)
- [物性値関数](#)
 - [熱力学的状態変数を返す関数](#)
 - [粘性率](#)
 - [熱伝導率](#)
 - [密度の偏微分](#)
 - [等エントロピ過程のエンタルピ](#)
 - [その他の物性値](#)
 - [setSmoothState](#)
- [DryAirNasa 乾燥空気](#)
- [まとめ](#)



純物質理想気体の
ベースモデル



純物質理想気体

Modelica.Media の基本構成

3~6.
交換可能な record,
model, partial function, ...

全ての物質の
ベースパッケージ

継承

```
<<package>> Interfaces.Types
<<type>> AbsolutePressure = ...
<<type>> Density = ...
<<type>> DynamicViscosity = ...
...
```

1. Types

```
<<replaceable>>
<<record>> FluidConstants
<<record>> ThermodynamicState
<<partial model>> BaseProperties
<<partial function>> setState_pTX
...
<<partial function>> dynamicViscosity
...
```

Medium (媒体物質モデル)
の構成要素

1. Types 物性値の型や
定数レコードの型

2. 定数

物質名、成分名、参照状態
デフォルト状態などを示す定数

3. ThermodynamicState

熱力学的状態を表すレコード

4. BaseProperties

基本物性モデル

5. setStateXXX

熱力学的状態
(ThermodynamicState)
を返す関数

6. 粘性率、熱伝導率などの
物性値関数

```
<<partial package>> Interfaces.PartialMedium
<<constant>> ThermodynamicState
<<constant>> mediumName
<<constant>> singleState
...
```

2. 定数

(多段階) 継承

個々の物質の
パッケージ

```
<<redeclare>>
ThermodynamicState→...
BaseProperties→...
setState_pTX→...
...
dynamicViscosity→...
...
```

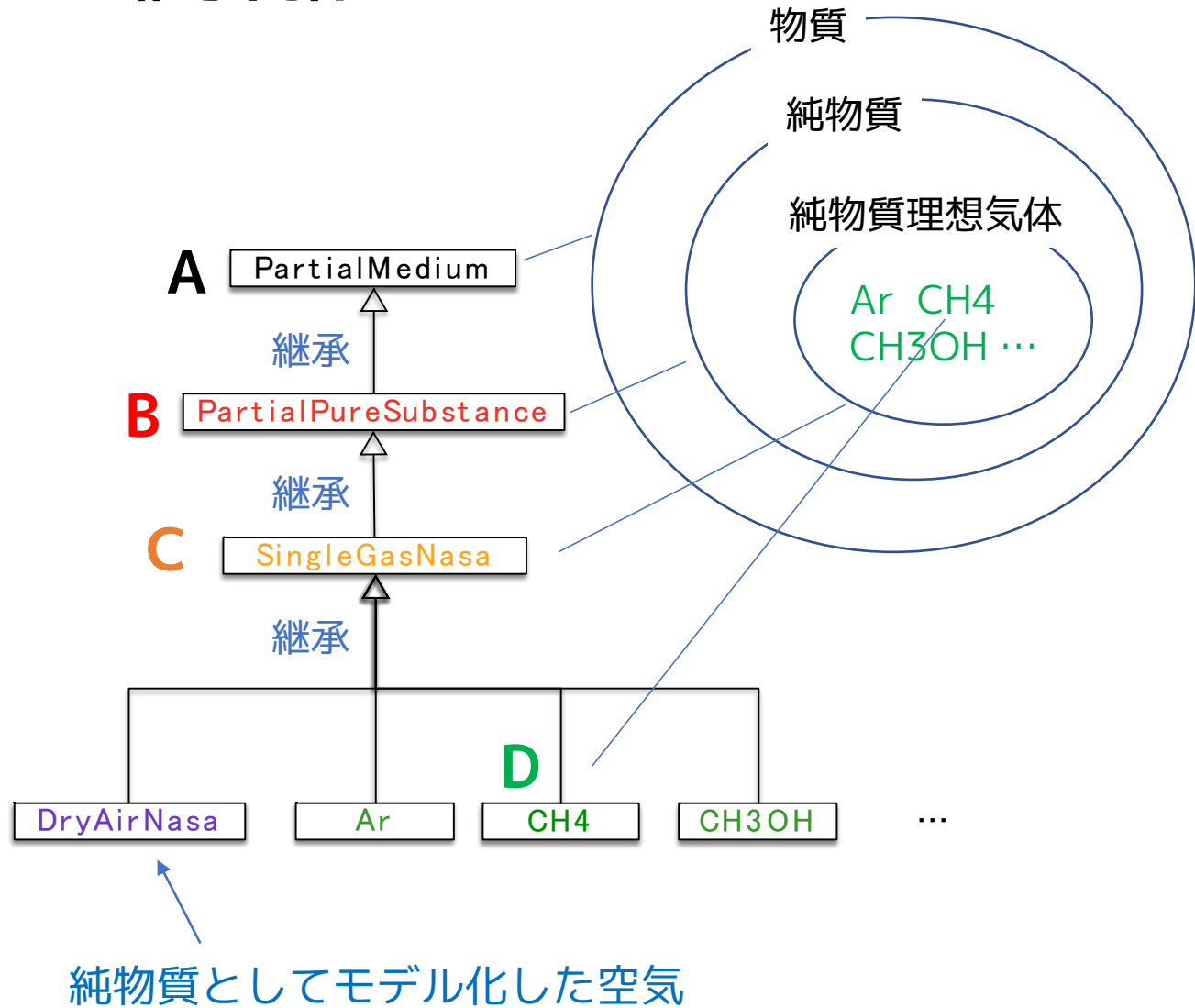
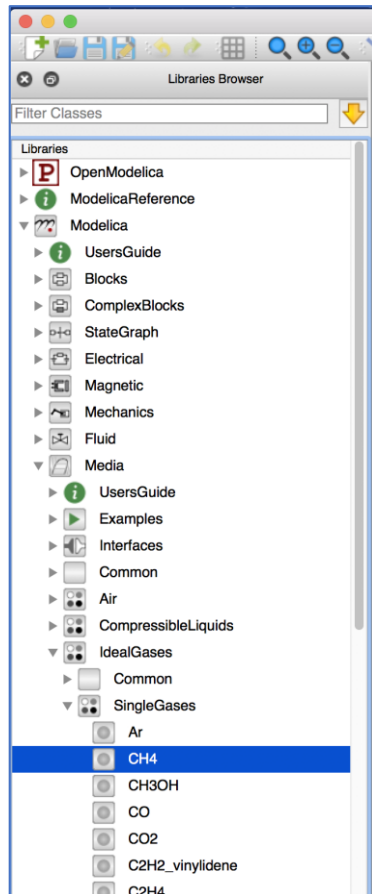
1 のパラメータ
変更

3~6 の宣言

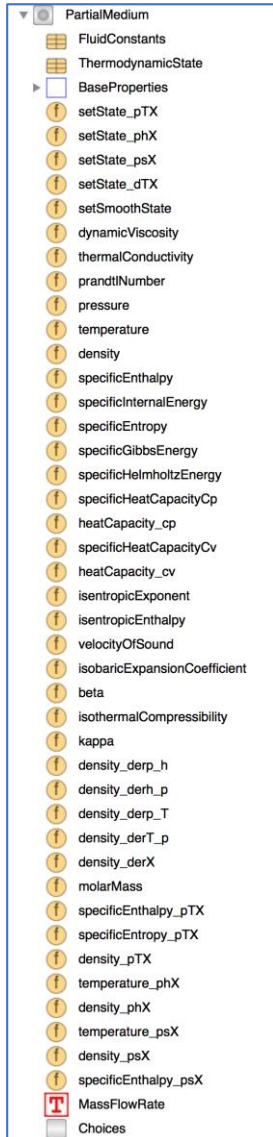
```
<<package>> XXX
<<constant>> ThermodynamicState=...
<<constant>> mediumName=...
<<constant>> singleState=...
...
```

2 の再定義
と追加

SingleGasNasa の継承関係



A. PartialMedium のクラス



- [FluidConstants](#)
- [ThermodynamicStates](#)
- [BaseProperties](#)
- [setState_pTX](#)
- [setState_phX](#)
- [setState_psX](#)
- [setState_dTX](#)
- [setSmoothState](#)
- [dynamicViscosity](#)
- [thermalConductivity](#)
- [prandtlNumber](#)
- [pressure](#)
- [temperature](#)
- [density](#)
- [specificEnthalpy](#)
- [specificInternalEnergy](#)
- [specificEntropy](#)
- [specificGibbsEnergy](#)
- [specificHelmholtzEnergy](#)
- [specificHeatCapacityCp](#)
- [heatCapacityCp](#)
- [specificHeatCapacityCv](#)
- [heatCapacityCv](#)
- [isentropicExponent](#)
- [isentropicEnthalpy](#)
- [velocityOfSound](#)
- [isobaricExpansionCoefficient](#)
- [beta](#)
- [isothermalCompressibility](#)
- [kappa](#)
- [density_derp_h](#)
- [density_derh_p](#)
- [density_derp_T](#)
- [density_derT_p](#)
- [density_derX](#)
- [molarMass](#)
- [specificEnthalpy_pTX](#)
- [specificEntropy_pTX](#)
- [density_pTX](#)
- [temperature_phX](#)
- [density_phX](#)
- [temperature_psX](#)
- [density_psX](#)
- [specificEnthalpy_psX](#)
- [MassFlowRate](#)
- [Choices](#)

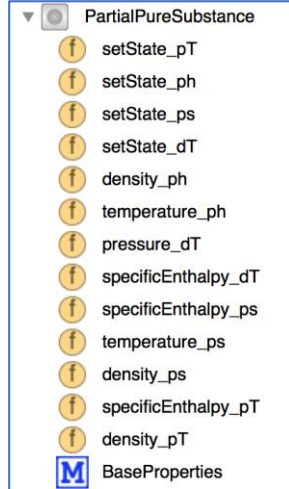
オレンジ色は

C. SingleGasNasa で再宣言(redeclare)

- [isentropicExponent](#)
- [isentropicEnthalpy](#)
- [velocityOfSound](#)
- [isobaricExpansionCoefficient](#)
- [beta](#)
- [isothermalCompressibility](#)
- [kappa](#)
- [density_derp_h](#)
- [density_derh_p](#)
- [density_derp_T](#)
- [density_derT_p](#)
- [density_derX](#)
- [molarMass](#)
- [specificEnthalpy_pTX](#)
- [specificEntropy_pTX](#)
- [density_pTX](#)
- [temperature_phX](#)
- [density_phX](#)
- [temperature_psX](#)
- [density_psX](#)
- [specificEnthalpy_psX](#)
- [MassFlowRate](#)
- [Choices](#)

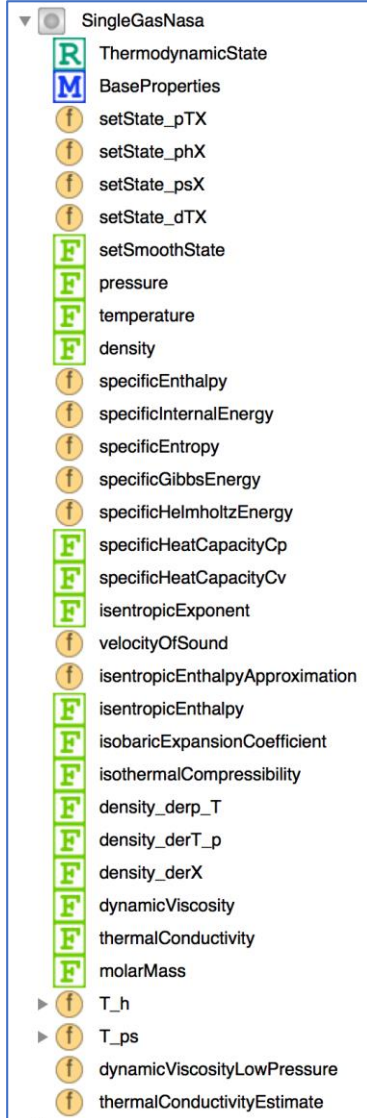
SingleGasNasa
では未使用

B. PartialPureSubstance のクラス



- [setState_pT](#)
- [setState_ph](#)
- [setState_ps](#)
- [setState_dT](#)
- [density_ph](#)
- [temperature_ph](#)
- [pressure_dT](#)
- [specificEnthalpy_dT](#)
- [specificEnthalpy_ps](#)
- [temperature_ps](#)
- [density_ps](#)
- [specificEnthalpy_pT](#)
- [density_pT](#)
- [BaseProperties](#)

C. SingleGasNasa のクラス



- [ThermodynamicStates](#)
- [BaseProperties](#)
- [setState_pTX](#)
- [setState_phX](#)
- [setState_psX](#)
- [setState_dTX](#)
- [setSmoothState](#)
- [pressure](#)
- [temperature](#)
- [density](#)
- [specificEnthalpy](#)
- [specificInternalEnergy](#)
- [specificEntropy](#)
- [specificGibbsEnergy](#)
- [specificHelmholtzEnergy](#)
- [specificHeatCapacityCp](#)
- [specificHeatCapacityCv](#)
- [isentropicExponent](#)
- [velocityOfSound](#)
- [isentropicEnthalpyApproximation](#)
- [isentropicEnthalpy](#)
- [isobaricExpansionCoefficient](#)
- [isothermalCompressibility](#)
- [density_derp_T](#)
- [density_derT_p](#)
- [density_derX](#)
- [dynamicViscosity](#)
- [thermalConductivity](#)
- [molarMass](#)
- [T_h](#)
- [T_ps](#)
- [dynamicViscosityLowPressure](#)
- [thermalConductivityEstimate](#)

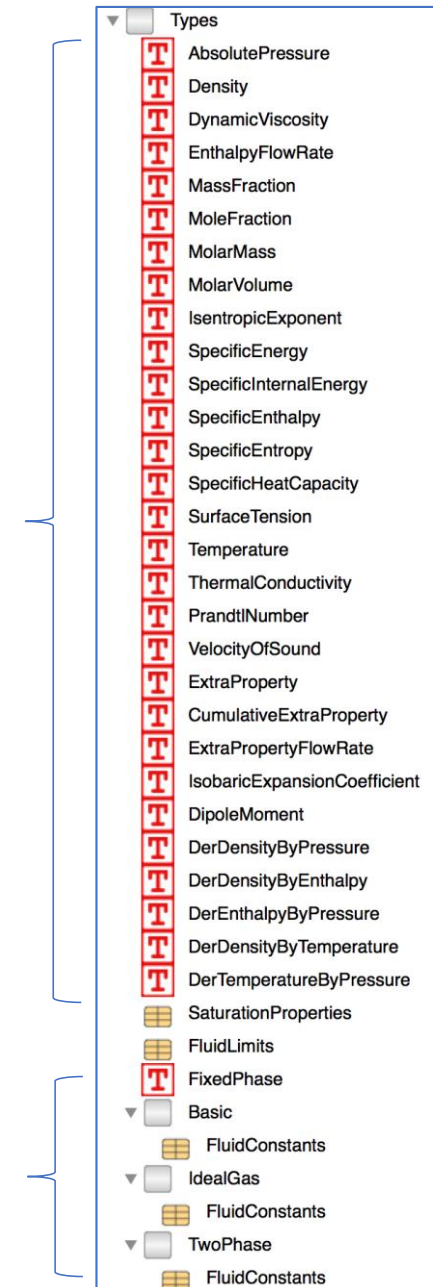
1 Types

Modelica.Media.Interfaces.Types

Modelica.Mediaで扱う流体の物性値の type や
定数の型を表す record が宣言されている

I. 物性値のタイプ

II. FluidConstantsレコード



I. 物性値のタイプ

Modelica.Media の冒頭部分で
Modelica.SIunits が import されている。

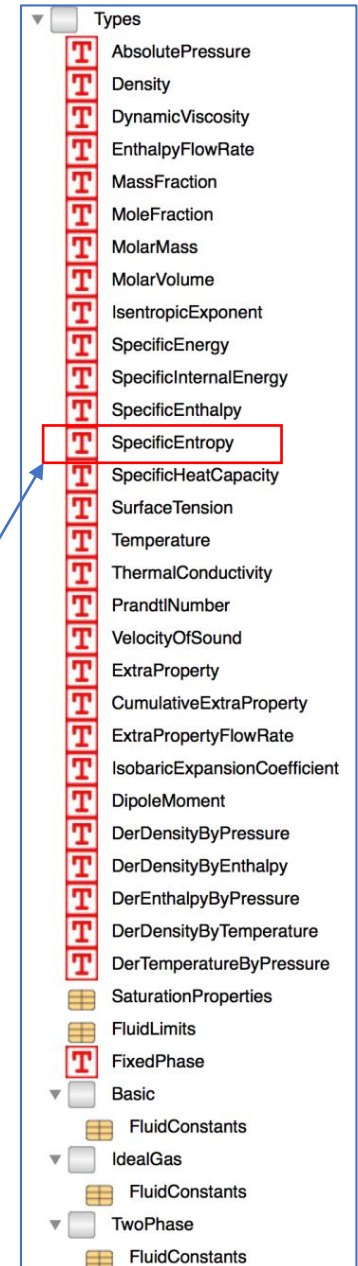
```
within Modelica;  
package Media  
  "Library of media property models" extends Modelica.Icons.Package;  
  import SI = Modelica.SIunits;  
  import Cv = Modelica.SIunits.Conversions;  
  ...
```

多数のタイプが Modelica.SIunits から生成されている。

```
type SpecificEnthalpy = SI.SpecificEnthalpy (  
  min=-1.0e10,  
  max=1.e10,  
  nominal=1.e6)  
  "Type for specific enthalpy with medium specific attributes";
```

SIunits を使用しない流体物性固有のタイプもある。

MassFraction, MoleFraction, MolarMass, MolarVolume,
ExtraProperty, IsobaricExpansionCoefficient,
DipoleMoment, ...



Modelica.Media.Interfaces.Types は全ての Medium パッケージで継承され、継承先で一部修正される。

A. PartialMedium の継承部分(extends)

```
partial package PartialMedium
  "Partial medium properties (base package of all media packages)"
  extends Modelica.Media.Interfaces.Types;
  extends Modelica.Icons.MaterialPropertiesPackage;
```

継承

A. PartialMedium で宣言されるタイプ

```
type MassFlowRate = SI.MassFlowRate (
  quantity="MassFlowRate." + mediumName,
  min=-1.0e5,
  max=1.e5) "Type for mass flow rate with medium specific attributes";
```

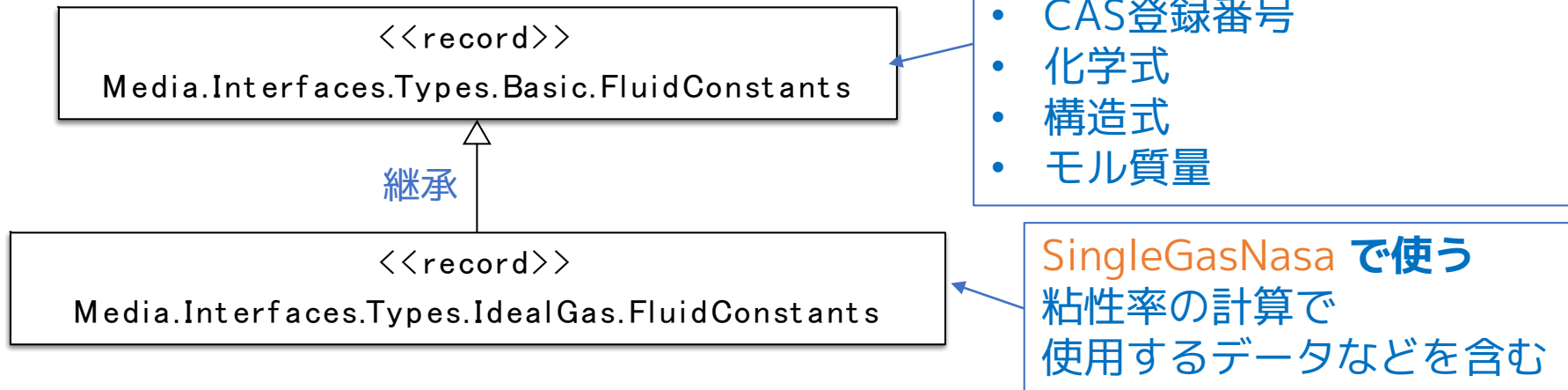
C. SingleGasNasaの継承部分 (extends)の抜粋

```
partial package SingleGasNasa
  "Medium model of an ideal gas based on NASA source" extends
  Interfaces.PartialPureSubstance(
    ...
    redeclare final record FluidConstants =
      Modelica.Media.Interfaces.Types.IdealGas.FluidConstants,
    ...
    Temperature(min=200, max=6000, start=500, nominal=500),
    SpecificEnthalpy(start=
      if Functions.referenceChoice==ReferenceEnthalpy.ZeroAt0K then data.H0
      else if Functions.referenceChoice==ReferenceEnthalpy.UserDefined then
        Functions.h_offset
      else 0, nominal=1.0e5),
    Density(start=10, nominal=10),
    AbsolutePressure(start=10e5, nominal=10e5));
```

FluidConstants レコードの宣言

タイプの修正

II. FluidConstantsレコード




Media.IdealGases.Common.FluidData.CH4では緑字部分のみが設定され、他はデフォルト値を使用する。SingleGasNasa ではすべての情報は必要としていない。

Modelica.Media.Interfaces.Types.Basic.FluidConstants

定数名	型	デフォルト値	
iupacName	String		IUPAC命名法に基づく物質名
casRegistryNumber	String		CAS登録番号
chemicalFormula	String		化学式
structureFormula	String		構造式
molarMass	MolarMass		モル質量

Modelica.Media.Interfaces.Types.IdealGas.FluidConstants

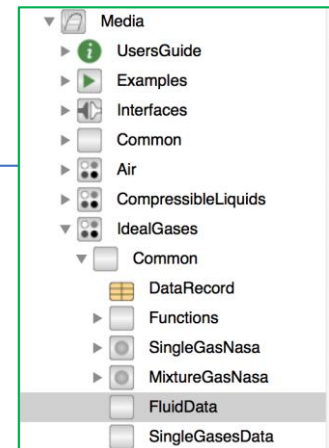
定数名	型	デフォルト値	
criticalTemperature	Temperature		Critical temperature (臨界温度)
criticalPressure	AbsolutePressure		Critical pressure (臨界圧力)
criticalMolarVolume	MolarVolume		Critical molar volume (臨界体積)
acentricFactor	Real		Pitzer acentric factor (偏心因子)
meltingPoint	Temperature		Melting Point at 101325 Pa (融点)
normalBoilingPoint	Temperature		Normal boiling point (at 101325 Pa) (沸点)
dipoleMoment	DipoeMoment		Dipole moment (双極子モーメント)
hasIdealGasHeatCapacity	Boolean	false	 データの有無を表す パラメータ
hasCriticalData	Boolean	false	
hasDipoleMoment	Boolean	false	
hasFundamentalEquation	Boolean	false	
hasLiquidHeatCapacity	Boolean	false	
hasSolidHeatCapacity	Boolean	false	
hasAccurateViscosityData	Boolean	false	
hasAccurateConductivityData	Boolean	false	
hasVapourPressureCurve	Boolean	false	
hasAcentricFactor	Boolean	false	
HCRIT0	SpecificEnthalpy	0.0	Critical specific enthalpy
SCRIT0	SpecificEntropy	0.0	Critical specific entropy
deltah	SpecificEnthalpy	0.0	
deltas	SpecificEntropy	0.0	

FluidConstants 型のデータの例

Modelica.Media.IdealGases.Common.FluidData.CH4

```
constant Modelica.Media.Interfaces.Types.IdealGas.FluidConstants CH4(  
    chemicalFormula = "CH4",  
    iupacName = "unknown",  
    structureFormula = "unknown",  
    casRegistryNumber = "74-82-8",  
    meltingPoint = 90.69,  
    normalBoilingPoint = 111.66,  
    criticalTemperature = 190.56,  
    criticalPressure = 45.99e5,  
    criticalMolarVolume = 98.60e-6,  
    acentricFactor = 0.011,  
    dipoleMoment = 0.0,  
    molarMass = SingleGasesData.CH4.MM,  
    hasDipoleMoment = true,  
    hasIdealGasHeatCapacity = true,  
    hasCriticalData = true,  
    hasAcentricFactor = true);
```

Modelica.Media.IdealGases.Common.FluidData
には 37 個の気体の FluidConstants 型定数が含まれている。



2. 定数

1. 定数の継承と導入

A. PartialMedium の定数

- 物質名と熱力学的変数特性
- 成分物質の情報
- 微小物質など付加的物質の情報
- reference state
- default state

定数名	型	デフォルト値	
mediumName	String	"unusablePartialMedium"	物質名
ThermoStates	IndependentVariables		T, pT, ph, phX, pTX, dTXのいずれか
singleState	Boolean		trueなら物性値が圧力に依存しない
substanceNames[:]	String	{mediumName}	成分名の配列
reducedX	Boolean	true	trueなら質量分率の和が1
fixedX	Boolean	false	trueならX=reference_X
nS	Integer	size(substanceNames, 1)	混合物の成分数
nX	Integer	nS	質量分率の要素数
nXi	Integer	if fixedX then 0 else if reducedX nS -1 else nS	独立な質量分率の要素数
extraPropertiesNames[:]	String	fill("",0)	付加的物質名(微小物質)
nC	Integer	size(extraPropertiesNames, 1)	付加的物質の成分数
C_nominal[nC]	Real	1.0e-6*ones(nC)	付加的物質の濃度
reference_p	AbsolutePressure	101325 [Pa]	圧力の参照値（基準値）
reference_T	Temperature	298.15 [K]	温度の参照値（基準値）
reference_X[nX]	MassFraction	fill(1/nX, nX)	質量分率の参照値
p_defalut	AbsolutePressure	101325 [Pa]	圧力のデフォルト値
T_default	Temperature	Conversions.from_degC(20)	温度のデフォルト値
h_default	SpecificEnthalpy	specifixEnthalpy_pTX(p_default, T_default, X_default)	比エンタルピのデフォルト値
X_default[nX]	MassFraction	reference_X	質量分率のデフォルト値

B. PartialPureSubstance の継承部分(extends)

```
Partial package PartialPureSubstance "Base class for pure substances of one chemical
substance" extends PartialMedium(final reducedX=true, final fixedX=true);
...
end PartialPureSubstance;
```

$n_{Xi} = 0$ となる。

C. SingleGasNasa の継承部分(extends) + 新しい定数

```
partial package SingleGasNasa
  "Medium model of an ideal gas based on NASA source" extends
  Interfaces.PartialPureSubstance(
    ThermoStates=Modelica.Media.Interfaces.Choices.IndependentVariables.pT,
    ...
    mediumName=data.name,
    substanceNames={data.name},
    singleState=false,
    ... ));
  ...
  constant IdealGases.Common.DataRecord data "Data record of ideal gas substance";
  constant FluidConstants[nS] fluidConstants "Constant data for the fluid";
  ...
end SingleGasNasa;
```

substanceNamesの要素数が1なので
 $n_S = n_X = 1$, $reference_X[1]=1$

新しい record 定数の導入

D. Modelica.Media.IdealGases.SingleGases.CH4

```
package CH4 "Ideal gas ¥"CH4¥" from NASA Glenn coefficients"
  extends Common.SingleGasNasa(
    mediumName="Methane",
    data=Common.SingleGasesData.CH4,
    fluidConstants={Common.FluidData.CH4});
  annotation ( ... );
end CH4;
```

SingleGasNasa をベースとする package の多くは
mediumName, data, fluidConstants を定義するだけで作成できる。

Modelica.Media.IdealGases.SingleGases.CH4の定数(まとめ)

定数名	型	設定値	
mediumName	String	"CH4"	物質名
ThermoStates	IndependentVariables	pT	T, pT, ph, phX, pTX, dTXのいずれか
singleState	Boolean	false	trueなら物性値が圧力に依存しない
substanceNames[:]	String	{"CH4"}	成分名の配列
reducedX	Boolean	true	trueなら質量分率の和が1
fixedX	Boolean	true	trueならX=reference_X
nS	Integer	1	混合物の成分数
nX	Integer	1	質量分率の要素数
nXi	Integer	0	独立な質量分率の要素数
extraPropertiesNames[:]	String	fill("",0)	付加的物質名(微小物質)
nC	Integer	size(extraPropertiesNames, 1)	付加的物質の成分数
C_nominal[nC]	Real	1.0e-6*ones(nC)	付加的物質の濃度
reference_p	AbsolutePressure	101325 [Pa]	圧力の参照値 (基準値)
reference_T	Temperature	298.15 [K]	温度の参照値 (基準値)
reference_X[nX]	MassFraction	{1}	質量分率の参照値
p_defalut	AbsolutePressure	101325 [Pa]	圧力のデフォルト値
T_default	Temperature	Conversions.from_degC(20)	温度のデフォルト値
h_default	SpecificEnthalpy	specifixEnthalpy_pTX(p_default, T_default, X_default)	比エンタルピのデフォルト値
X_default[nX]	MassFraction	reference_X = {1}	質量分率のデフォルト値
data	DataRecord	Common.SingleGasesData.CH4	理想気体熱物性データ
fluidConstants[nS]	FluidConstants	{Common.FluidData.CH4}	その他の物質固有データ

II. 熱物性データ

Modelica.Media.IdealGases.Common.DataRecord

DataRecord の Document より

比熱

$$C_p(T) = R \sum_{i=1}^7 a_i T^{i-3}$$

適用範囲

$$200 \text{ K} \leq T \leq 6000 \text{ K}$$

比エンタルピー

$$h(T) = RT \left(-\frac{a_1}{T^2} + a_2 \frac{\log T}{T} + \sum_{i=3}^7 a_i \frac{T^{i-3}}{i-2} + \frac{b_1}{T} \right)$$

比エントロピー

$$s_0(T) = R \left(-\frac{a_1}{2T^2} - \frac{a_2}{T} + a_3 \log T + \sum_{i=4}^7 a_i \frac{T^{i-3}}{i-3} + b_2 \right)$$

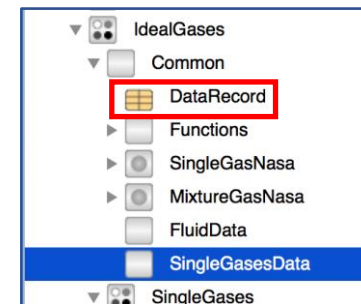
$$s(T, p) = s_0(T) - R \ln \frac{p}{p_0}$$

係数

$$a_i = \begin{cases} a_{low}[i] & T < T_{limit} \\ a_{high}[i] & T \geq T_{limit} \end{cases}$$

積分定数

$$b_i = \begin{cases} b_{low}[i] & T < T_{limit} \\ b_{high}[i] & T \geq T_{limit} \end{cases}$$



$h(T), s_0(T)$ は、

$$\left(\frac{\partial h}{\partial T} \right)_p = C_p$$

$$\left(\frac{\partial s}{\partial T} \right)_p = \frac{C_p}{T}$$

を積分して
得られる。

<https://www.grc.nasa.gov/WWW/CEAWeb/TP-2002-211556.pdf>

McBride B.J., Zehe M.J., and Gordon S. (2002):
**NASA Glenn Coefficients for Calculating
Thermodynamic Properties of Individual Species.**
NASA report TP-2002-211556

Modelica.Media.IdealGases.Common.DataRecord

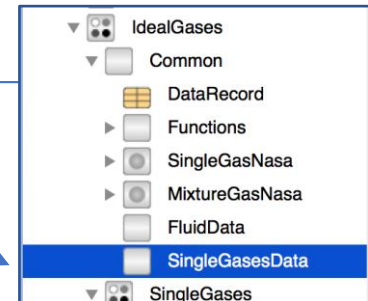
変数名	型	
name	String	物質名
MM	MolarMass	モル質量 [kg/mol]
Hf	SpecificEnthalpy	標準生成エンタルピ [J/kg]
H0	SpecificEnthalpy	0 K を基準とした場合の 25 °C の比エンタルピ [J/kg]
Tlimit	Temperature	alow, ahigh など切り替える分岐温度 [K]
alow, blow, ahigh, bhigh	Real[]	係数と積分定数
R	SpecificHeatCapacity	ガス定数 [J/kg.K]

Modelica.Media.IdealGases.Common.SingleGasesData.CH4

```

constant IdealGases.Common.DataRecord CH4(
  name="CH4",
  MM=0.01604246,
  Hf=-4650159.63885838,
  H0=624355.7409524474,
  Tlimit=1000,
  alow={-176685.0998, 2786.18102, -12.0257785, 0.0391761929, -3.61905443e-005,
        2.026853043e-008, -4.976705489999999e-012},
  blow={-23313.1436, 89.0432275},
  ahigh={3730042.76, -13835.01485, 20.49107091, -0.001961974759, 4.72731304e-007,
        -3.72881469e-011, 1.623737207e-015},
  bhigh={75320.6691, -121.9124889},
  R=518.2791167938085);
  
```

1241種の
データ



3. ThermodynamicState 独立な熱力学的状態変数

物質の熱力学的状態を決定するのに必要な変数の組を表す record

物質の熱力学的状態は、

2つの熱力学的状態変数と成分物質の質量分率で決定できる。

- 圧力 p 、温度 T 、密度 d 、比エンタルピー h 、内部エネルギー u のうち2つ
- 混合物質の成分の質量分率ベクトル $X[nX]$

C. Media.IdealGases.Common.SingleGasNasa.ThermodynamicState

```
record extends ThermodynamicState
  "Thermodynamic state variables for ideal gases"
  AbsolutePressure p "Absolute pressure of medium";
  Temperature T "Temperature of medium";
end ThermodynamicState;
```

SingleGasNasa では独立な熱力学的変数
として **p** と **T** を選択する。
(純物質なので質量分率はいらない)

4. BaseProperties

熱力学的状態変数、ガス定数、モル質量などの関係を表す model

A. PartialMedium.BaseProperties

<<replaceable partial model>> PartialMedium.BaseProperties	
<<connector>> p: InputAbsolutePressure	}
<<connector>> h: InputSpecificEnthalpy	
<<connector>> Xi[nXi]: InputMassFraction	
<<variable>> d: Density	
<<variable>> T: Temperature	
<<variable>> X[nX]: MassFraction	
<<variable>> u: SpecificInternalEnergy	
<<variable>> R: SpecificHeatCapacity	
<<variable>> MM: MolarMass	
<<variable>> state: ThermodynamicState	
<<variable>> T_degC = to_degC(T)	}
<<variable>> p_bar = to_bar(p)	
<<parameter>> preferredMediumStates: Boolean	}
<<parameter>> standardOrderComponents: Boolean	
connector inputAbsolutePressure	
connector inputMassFraction	
connector inputSpecificEnthalpy	
equation (X,Xi)	

変数

p: 圧力[Pa]
h: 比エンタルピー[J/kg]
Xi[nXi]: 独立な成分の質量分率
d: 密度[kg/m³] ← 数式では ρ で表す
T: 温度[K]
X[nX]: 成分の質量分率
u: 内部エネルギー[J/kg]
R: ガス定数[J/kg.K]
MM: モル質量[kg/mol]
state: ThermodynamicState

パラメータ

preferredMediumStates = false
standardOrderComponents = false

XとXiの方程式が記述されている。

I. State Selection (状態変数選択)

微分代数方程式の最も一般的な形

陰的微分方程式

$$F(t, y, y') = 0$$

物理学的、工学的現象の
数学的モデル



ツールによる translation

半陽的微分代数方程式

制約付きの常微分方程式

数値解析に適した
モデル

$$\begin{cases} x' = f(t, x, z) & \text{微分方程式} \\ 0 = g(t, x, z) & \text{代数方程式 (制約条件)} \end{cases}$$



Modelica のモデル

t : 時間
 p : パラメータ
 c : 定数
 x : 状態変数
 z : 他の変数

$x' z$ ↓ ↑ $t x z$

ツール

ツールがどの変数を状態変数(state variables)として微分方程式を構成するかを選択する

ツールによる simulation

- Static State Selection (静的状態変数選択)

ツールが translation の間に状態変数を決定する

- Dynamic State Selection (動的状態変数選択)

ツールが simulation の間に状態変数を決定する

State Selection のためのパラメータ preferredMediumStates

保存則を満たす体積要素のモデル (valance volume) などで、
medium モデルの State Selection をツールに指示するパラメータ

Modelica.Fluid.Interfaces.PartialLumpedVolume での設定をみると...

```
partial model PartialLumpedVolume
...
  Medium.BaseProperties medium(
    preferredMediumStates=true,
    p(start=p_start),
    h(start=h_start),
    T(start=T_start),
    Xi(start=X_start[1:Medium.nXi]));
...
```

preferredMediumStates の設定を変えると、BaseProperties モデル
の変数 p, T の stateSelect パラメータが切り替わる。

C. SingleGasNasa.BasePropertiesの継承部分

```
model extends BaseProperties(
  T(stateSelect=if preferredMediumStates then StateSelect.prefer else StateSelect.default),
  p(stateSelect=if preferredMediumStates then StateSelect.prefer else StateSelect.default))
```


stateSelect パラメータ とは...

stateSelect は、Modelica 言語の仕様として規定されているReal 変数に設定可能なパラメータである。

Predefined Type Real

仕様書

<https://www.modelica.org/documents/ModelicaSpec34.pdf> 4.8.1 Real Type, p.52

```
type Real // Note: Defined with Modelica syntax although predefined
  RealType value;
  parameter StringType quantity
  parameter StringType unit
  parameter StringType displayUnit = "" "Default display unit";
  parameter RealType min=-Inf, max=+Inf; // Inf denotes a large value
  parameter RealType start = 0; // Initial value
  parameter BooleanType fixed = true, // default for parameter/constant;
                                = false; // default for other variables
  parameter RealType nominal; // Nominal value
  parameter BooleanType unbounded=false; // For error control
  parameter StateSelect stateSelect = StateSelect.default;
equation
  assert(value >= min and value <= max, "Variable value out of limit");
end Real;
```

stateSelect に設定可能な値は...

<https://www.modelica.org/documents/ModelicaSpec34.pdf> 4.8.8.1, p.56

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

はじめてのModelicaプログラミング, 広野友英, TechShare, 2017, p.34

never: 独立変数として使用しない
avoid: 独立変数として使用することをできるだけ避ける
default: 適切と判断されれば独立変数として使用する
 (ただし微分されている場合のみ)
prefer: 独立変数としてできるだけ使用する
always: 必ず独立変数として使用する

Modelica.Media.UserGuide.MediumDefinition.StaticStateSelection より

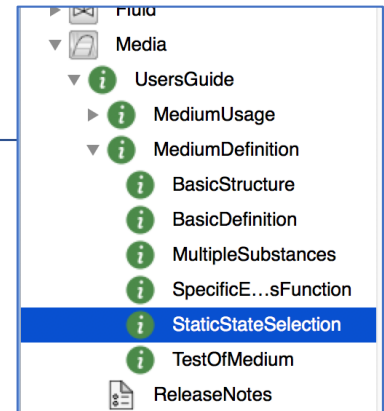
保存則の成り立つ体積要素 (valance volume) で medium が次のように定義されている場合を考える。

```
package Medium = Modelica.Media.Interfaces.PartialMedium;
  Medium.BaseProperties medium;
equation
  // mass balance
  der(M) = port_a.m_flow + port_b.m_flow;
  der(MX) = port_a_mX_flow + port_b_mX_flow;
  M = V*medium.d;
  MX = M*medium.X;
  // Energy balance
  U = M*medium.u;
  der(U) = port_a.H_flow+port_b.H_flow;
```

BaseProperties モデル

容器全体の質量保存則
成分物質の質量保存則

エネルギー保存則



単成分物質（純物質）の場合

- preferredMediumStates = true
- 状態変数 p , T , d , u , h のうち 2 つが stateSelect = StateSelect.prefer
- 残りの 3 つの変数が、この 2 つの状態変数の関数の形で記述される

のとき

- Static State Selection (静的状態変数選択)
- 2 つの状態変数の微分による線形方程式系

となる。

不要な非線形方程式の
繰り返し計算が
避けられる。

例として、以下のように設定されている場合を考える。

```
p(stateSelect = StateSelect.prefer)
T(stateSelect = StateSelect.prefer)
d = fd(p,T)
u = fu(p,T)
h = fh(p,T)
```

index reduction の結果、medium の微分方程式系は次のようになる

```
der(M) = V*der(d)
der(U) = der(M)*u + M*der(u)
der(d) = der(fd,p)*der(p) + der(fd,T)*der(T)
der(u) = der(fu,p)*der(p) + der(fu,T)*der(T)
```

der(x,y)

x を y で偏微分する。
Modelica に将来的に
導入されるらしい…

- 状態変数 p, T は ツールの積分器から与えられる。
- fd(p,T), der(fd,p) など他の関数は、p, T の関数として評価される。
- der(M), der(U), der(d), der(u)などを消去した全方程式系は der(p) とder(T)の線形方程式系となる。

うまくいかない場合の例 (Counter Example)

```
p(stateSelect = StateSelect.prefer)
```

```
T(stateSelect = StateSelect.prefer)
```

```
h = h(T)
```

```
u = h - R*T
```

```
p = d*R*T
```

← d が p, T の関数の形になっていない

- もし p と T が状態変数なら、 d を導出できなければならない。
- $d = p/(R*T)$ と導出できるが R または T がゼロなら division by zero になってしまう。ツールは R や T がゼロになれないことを知らない。
- そのため、ツールは p と T が、常に状態変数とはなれないことを仮定しなければならない。
- したがって、ツールは、動的状態変数選択を行うか他の静的状態変数を探すことになる。
- 他の静的状態変数の候補として d が考えられる。（ p は d と T の関数で表すことができる。）
- d を静的状態変数にするためには、ポテンシャル変数として微分の形（ $\text{der}(d) = \dots$ ）で与えられるか、`selectState` が `SelectState.prefer` か `SelectState.always` でなければならない。



結局、動的状態変数選択になる

State Selection のテスト

簡単な valance volume モデルを作成する。

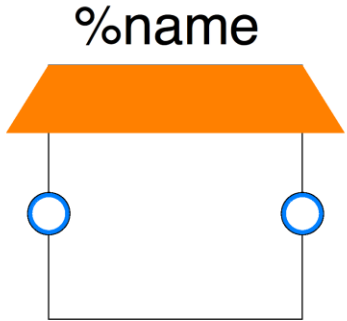
```
model Room
  replaceable package Medium = Modelica.Media.Air.DryAirNasa;
  parameter SI.Volume V = 22.0;
  Modelica.Fluid.Interfaces.FluidPort_b port_a(redeclare package Medium = Medium)
    annotation( ...);
  Modelica.Fluid.Interfaces.FluidPort_b port_b(redeclare package Medium = Medium)
    annotation( ...);
  Medium.BaseProperties medium;
  SI.Mass M;
  SI.Energy U;
equation
  M = medium.d * V;
  U = medium.u * M;
  der(M) = port_a.m_flow + port_b.m_flow;
  der(U) = actualStream(port_a.h_outflow) * port_a.m_flow +
    actualStream(port_b.h_outflow) * port_b.m_flow;
  port_a.p = medium.p;
  port_b.p = medium.p;
  port_a.h_outflow = medium.h;
  port_b.h_outflow = medium.h;
initial equation
  medium.T = 293.15;
  medium.p = 101325;
  annotation( ...);
end Room;
```

4 畳半ぐらいの大きさ

BaseProperties モデル

質量保存則
エネルギー保存則

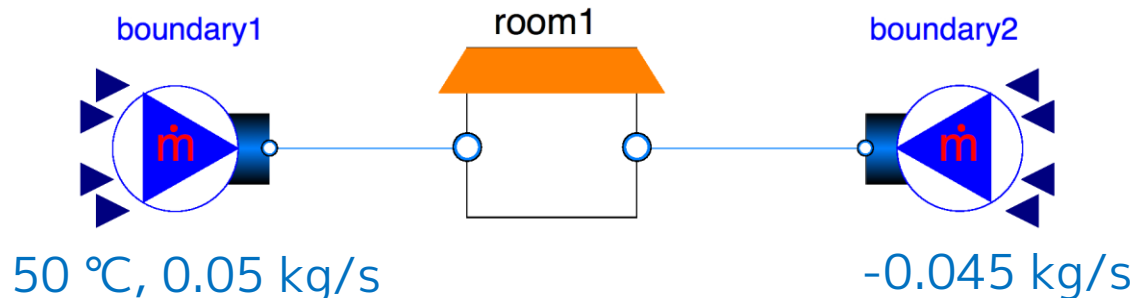
初期状態
20 °C, 1 atom



システムモデル① RoomTest_prefer

room1 の medium の preferredMediumStatus を true に設定

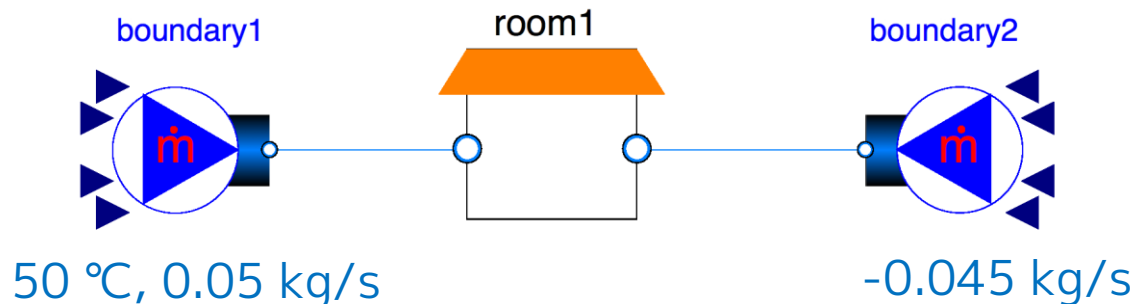
```
model RoomTest_prefer
  replaceable package Medium = Modelica.Media.Air.DryAirNasa(
    AbsolutePressure(nominal = 100000.0));
  Room room1(redeclare package Medium = Medium,
    medium(preferredMediumStates = true)) annotation( ...);
  Modelica.Fluid.Sources.MassFlowSource_T boundary1(
    redeclare package Medium = Medium, T = 323.15, m_flow = 0.05,
    nPorts = 1) annotation( ...);
  Modelica.Fluid.Sources.MassFlowSource_T boundary2(
    redeclare package Medium = Medium, m_flow = -0.045, nPorts = 1)
    annotation( ...);
equation
  connect(boundary2.ports[1], room1.port_b) annotation( ...);
  connect(boundary1.ports[1], room1.port_a) annotation( ...);
annotation( ...);
end RoomTest_prefer;
```



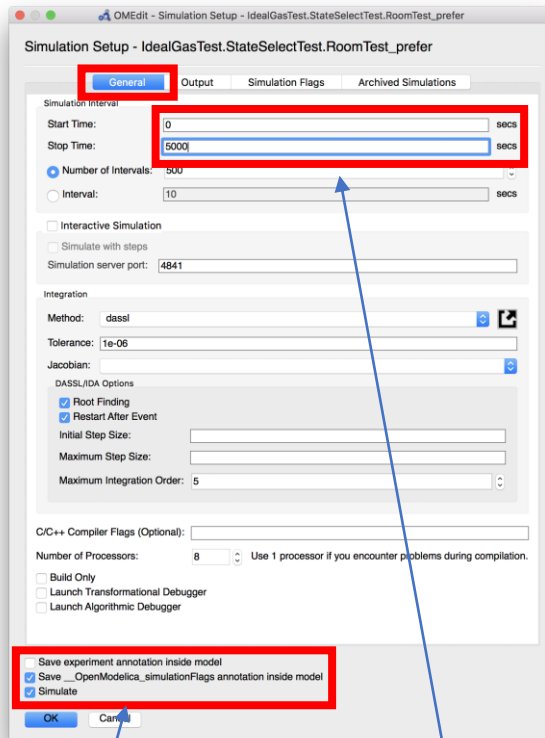
システムモデル② RoomTest_default

room1 の medium の preferredMediumStatus を false に設定

```
model RoomTest_default
  replaceable package Medium = Modelica.Media.Air.DryAirNasa(
    AbsolutePressure(nominal = 100000.0));
  Room room1(redeclare package Medium = Medium,
    medium(preferredMediumStates = false)) annotation( ...);
  Modelica.Fluid.Sources.MassFlowSource_T boundary1(
    redeclare package Medium = Medium, T = 323.15, m_flow = 0.05,
    nPorts = 1) annotation( ...);
  Modelica.Fluid.Sources.MassFlowSource_T boundary2(
    redeclare package Medium = Medium, m_flow = -0.045, nPorts = 1)
    annotation( ...);
equation
  connect(boundary2.ports[1], room1.port_b) annotation( ...);
  connect(boundary1.ports[1], room1.port_a) annotation( ...);
annotation( ...);
end RoomTest_default;
```



Simulation > Simulation Setup の設定

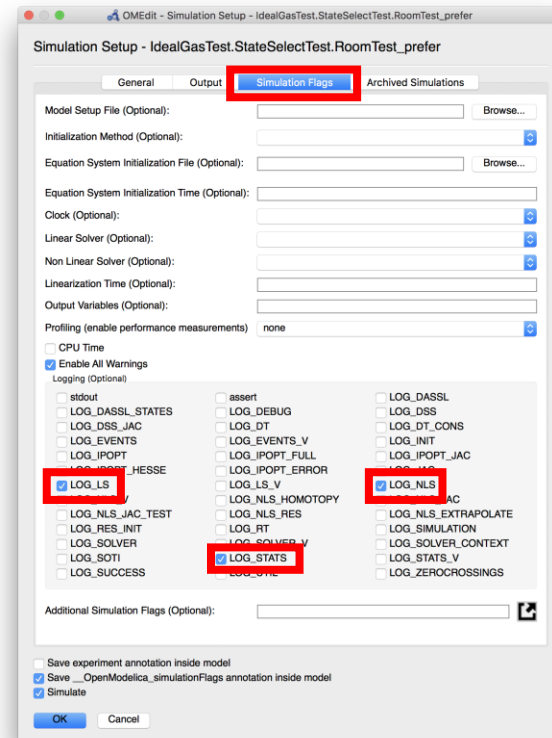


General

- Start Time = 0 [secs]
- End Time = 5000 [secs]

チェック

- Save __OpenModelica_SimulationFlags annotation inside model
- Simulate

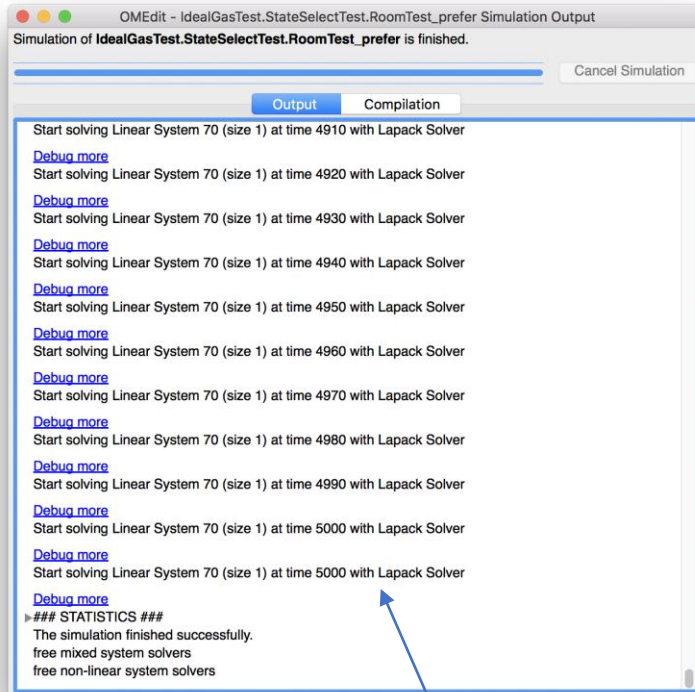


Simulation Flags のチェック

- LOG_LS
- LOG_NLS
- LOG_STATS

実行時の出力ログ

① RoomTest_prefer



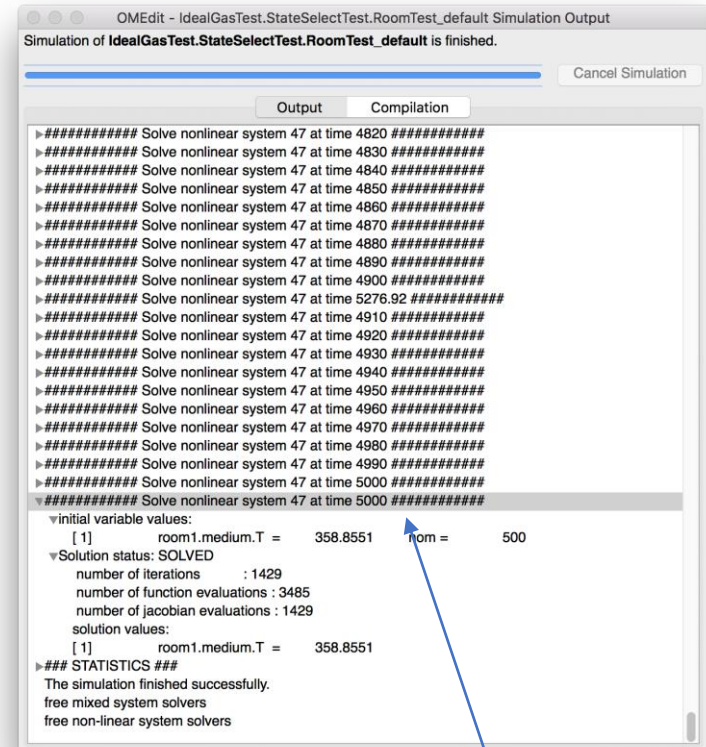
```
OMEdit - IdealGasTest.StateSelectTest.RoomTest_prefer Simulation Output
Simulation of IdealGasTest.StateSelectTest.RoomTest_prefer is finished.

Output  Compilation

Start solving Linear System 70 (size 1) at time 4910 with Lapack Solver
Debug more
Start solving Linear System 70 (size 1) at time 4920 with Lapack Solver
Debug more
Start solving Linear System 70 (size 1) at time 4930 with Lapack Solver
Debug more
Start solving Linear System 70 (size 1) at time 4940 with Lapack Solver
Debug more
Start solving Linear System 70 (size 1) at time 4950 with Lapack Solver
Debug more
Start solving Linear System 70 (size 1) at time 4960 with Lapack Solver
Debug more
Start solving Linear System 70 (size 1) at time 4970 with Lapack Solver
Debug more
Start solving Linear System 70 (size 1) at time 4980 with Lapack Solver
Debug more
Start solving Linear System 70 (size 1) at time 4990 with Lapack Solver
Debug more
Start solving Linear System 70 (size 1) at time 5000 with Lapack Solver
Debug more
Start solving Linear System 70 (size 1) at time 5000 with Lapack Solver
Debug more
>### STATISTICS ###
The simulation finished successfully.
free mixed system solvers
free non-linear system solvers
```

線形方程式系を解いている

② RoomTest_default



```
OMEdit - IdealGasTest.StateSelectTest.RoomTest_default Simulation Output
Simulation of IdealGasTest.StateSelectTest.RoomTest_default is finished.

Output  Compilation

>##### Solve nonlinear system 47 at time 4820 #####
>##### Solve nonlinear system 47 at time 4830 #####
>##### Solve nonlinear system 47 at time 4840 #####
>##### Solve nonlinear system 47 at time 4850 #####
>##### Solve nonlinear system 47 at time 4860 #####
>##### Solve nonlinear system 47 at time 4870 #####
>##### Solve nonlinear system 47 at time 4880 #####
>##### Solve nonlinear system 47 at time 4890 #####
>##### Solve nonlinear system 47 at time 4900 #####
>##### Solve nonlinear system 47 at time 5276.92 #####
>##### Solve nonlinear system 47 at time 4910 #####
>##### Solve nonlinear system 47 at time 4920 #####
>##### Solve nonlinear system 47 at time 4930 #####
>##### Solve nonlinear system 47 at time 4940 #####
>##### Solve nonlinear system 47 at time 4950 #####
>##### Solve nonlinear system 47 at time 4960 #####
>##### Solve nonlinear system 47 at time 4970 #####
>##### Solve nonlinear system 47 at time 4980 #####
>##### Solve nonlinear system 47 at time 4990 #####
>##### Solve nonlinear system 47 at time 5000 #####
>##### Solve nonlinear system 47 at time 5000 #####
▽initial variable values:
[ 1]      room1.medium.T =   358.8551  nom =      500
▽Solution status: SOLVED
  number of iterations      : 1429
  number of function evaluations : 3485
  number of jacobian evaluations : 1429
  solution values:
[ 1]      room1.medium.T =   358.8551
>### STATISTICS ###
The simulation finished successfully.
free mixed system solvers
free non-linear system solvers
```

非線形方程式系を解いている

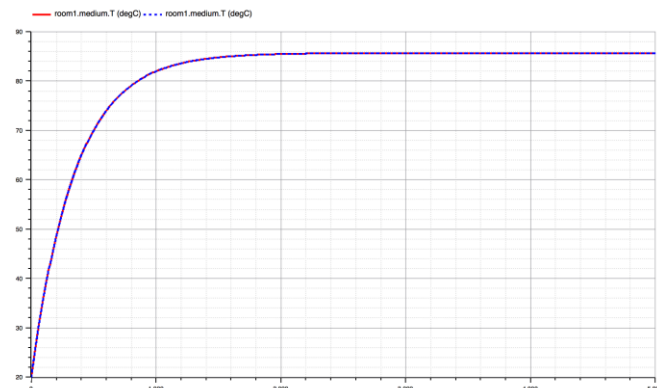
シミュレーション結果

結果は変わらない!!

温度

RoomTest_prefer

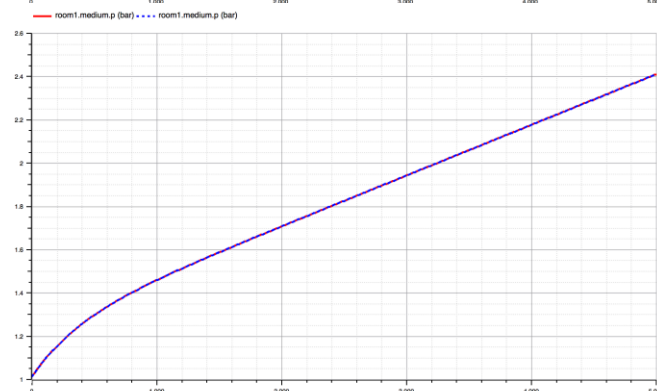
RoomTest_default



圧力

RoomTest_prefer

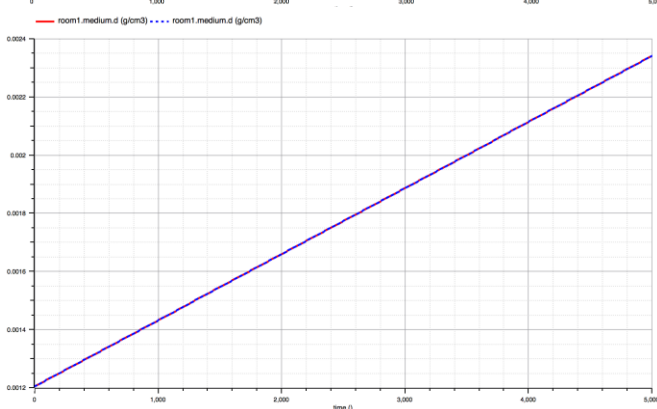
RoomTest_default



密度

RoomTest_prefer

RoomTest_default



II. BaseProperties の方程式

A. PartialMedium.BaseProperties の方程式（抜粋）

```
equation
  if standardOrderComponents then
    Xi = X[1:nXi];
    if fixedX then
      X = reference_X;
    end if;
    if reducedX and not fixedX then
      X[nX] = 1 - sum(Xi);
    end if;
    ...
  end if;
  ...
end BaseProperties;
```

X と Xi に関する方程式

standardOrderComponents = true

$nXi = nS$

$Xi = X[1:nXi]$

B. PartialPureSubstance.BaseProperties の継承部分

```
partial model extends BaseProperties(
  final standardOrderComponents=true)
end BaseProperties;
```

C. SingleGasNasa.BaseProperties の方程式

p, d, T, h, u, R, MM に関する方程式

equation

```
assert(T >= 200 and T <= 6000, "Temperature T (= " + String(T) + " K) is not  
in the allowed range 200 K <= T <= 6000 K required from medium model ¥" +  
mediumName + "¥".");
```

```
MM = data.MM;
```

```
R = data.R;
```

```
h = Modelica.Media.IdealGases.Common.Functions.h T(  
    data, T,  
    Modelica.Media.IdealGases.Common.Functions.excludeEnthalpyOfFormation,  
    Modelica.Media.IdealGases.Common.Functions.referenceChoice,  
    Modelica.Media.IdealGases.Common.Functions.h_offset);
```

```
u = h - R*T;
```

```
// Has to be written in the form d=f(p,T) in order that static  
// state selection for p and T is possible
```

```
d = p/(R*T);
```

```
// connect state with BaseProperties
```

```
state.T = T;
```

```
state.p = p;
```

```
end BaseProperties;
```

温度から比エンタルピを求める関数

デフォルト値

true zeroAt0K 0.0

エンタルピ

$$h \equiv u + pv$$
$$= u + RT$$

理想気体
状態方程式

$$pv = RT$$

内部エネルギー

$$u = h - RT$$

密度

$$\rho \equiv \frac{1}{v} = \frac{p}{RT}$$

stateレコード
との関係の式

III. 比エンタルピの計算方法と標準状態

Media.IdealGases.Common.Functions.h_T

温度から比エンタルピを求める関数

← DataRecord 型：係数 a_i, b_i を含む

← ソースコードの形式

$$h_T(\mathbf{data}, T, \text{exclEnthForm}, \text{refChoice}, h_{\text{off}}) \\ = R \frac{-a_1 + T((b_1 + a_2 \ln T) + T a_3 + T(a_4/2 + T(a_5/3 + T(a_6/4 + a_7/5 T))))}{T} \\ + h_1 + h_2 \\ = RT \left(-\frac{a_1}{T^2} + a_2 \frac{\log T}{T} + \sum_{i=3}^7 a_i \frac{T^{i-3}}{i-2} + \frac{b_1}{T} \right) + h_1 + h_2$$

← DataRecord の Document の式

$$a_i = \begin{cases} a_{\text{low}}[i] & T < T_{\text{limit}} \\ a_{\text{high}}[i] & T \geq T_{\text{limit}} \end{cases} \quad b_1 = \begin{cases} b_{\text{low}}[1] & T < T_{\text{limit}} \\ b_{\text{high}}[1] & T \geq T_{\text{limit}} \end{cases}$$

$$h_1 = \begin{cases} -h_f, & \text{exclEnthForm} = \text{true} \\ 0, & \text{exclEnthForm} = \text{false} \end{cases} \quad h_2 = \begin{cases} 0, & \text{refChoice} = \text{ZeroAt25C} \\ h_0, & \text{refChoice} = \text{ZeroAt0K} \\ h_{\text{off}}, & \text{refChoice} = \text{UserDefined} \end{cases}$$

デフォルト

- `exclEnthForm = Functions.excludeEnthalpyOfFormation = true`
- `refChoice = Functions.referenceChoice = ZeroAt0K`
- `h_off = Functions.h_offset = 0.0`

CH₄の標準状態 (25℃) の比エンタルピ (h_T の検算)

(1) 生成エンタルピを排除しない場合、

標準状態のエンタルピが標準生成エンタルピとなる

exclEnthForm = *false*, refChoice = *ZeroAt25C*

$$h_T(T) = RT \left(-\frac{a_1}{T^2} + a_2 \frac{\log T}{T} + \sum_{i=3}^7 a_i \frac{T^{i-3}}{i-2} + \frac{b_1}{T} \right)$$

生成エンタルピが含まれている

計算結果 $h_T(298.14) = -4.65016 \times 10^6$ [J/kg]
 $h_T(298.15) = -4.65014 \times 10^6$ [J/kg]

標準生成エンタルピ[J/kg]
data.Hf=-4650159.63885838

(2) 生成エンタルピを排除した場合、

標準状態のエンタルピが0 [J/kg] となる

exclEnthForm = *true*, refChoice = *ZeroAt25C*

$$h_T(T) = RT \left(-\frac{a_1}{T^2} + a_2 \frac{\log T}{T} + \sum_{i=3}^7 a_i \frac{T^{i-3}}{i-2} + \frac{b_1}{T} \right) - h_f$$

計算結果 $h_T(298.14) = -0.994223$ [J/kg]
 $h_T(298.15) = 21.2536$ [J/kg]

計算誤差のため、
298.14~298.15℃の間に
0 [J/kg] になる温度がある

CH4の標準状態 (25℃) の比エンタルピ (h_T の検算)

(3) 生成エンタルピを排除して、ZeroAt0K を選択した場合、標準状態のエンタルピが H0 [J/kg] となる。

デフォルト設定

exclEnthForm = *true*, refChoice = *ZeroAt0K*

$$h_T(T) = RT \left(-\frac{a_1}{T^2} + a_2 \frac{\log T}{T} + \sum_{i=3}^7 a_i \frac{T^{i-3}}{i-2} + \frac{b_1}{T} \right) - h_f + h_0$$

計算結果 $h_T(298.14) = 624355 \text{ [J/kg]}$
 $h_T(298.15) = 624377 \text{ [J/kg]}$ data.H0=624355.7409524474

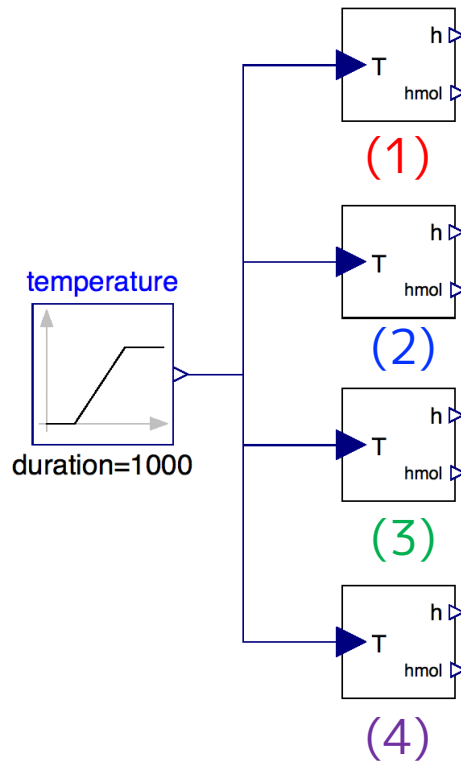
(4) 生成エンタルピを排除して、UserDefined を選択した場合、標準状態のエンタルピが h_off [J/kg] となる。

exclEnthForm = *true*, refChoice = *UserDefined*

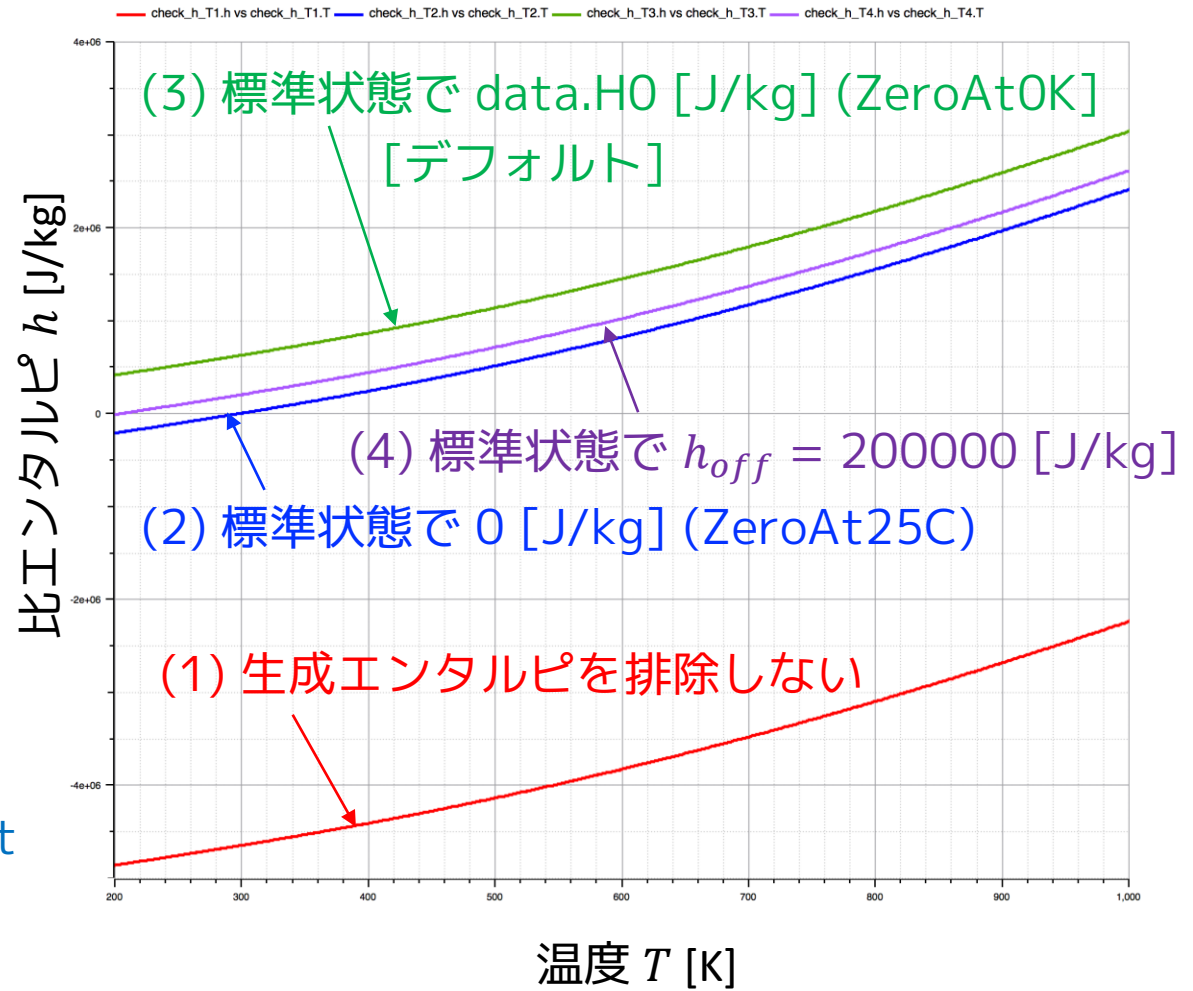
$$h_T(T) = RT \left(-\frac{a_1}{T^2} + a_2 \frac{\log T}{T} + \sum_{i=3}^7 a_i \frac{T^{i-3}}{i-2} + \frac{b_1}{T} \right) - h_f + h_{off}$$

計算結果 $h_{off} = 20000 \text{ [J/kg]}$ とした場合
 $h_T(298.14) = 19999$
 $h_T(298.15) = 20021.3$

CH4 の比エンタルピーのパラメータ設定による違い

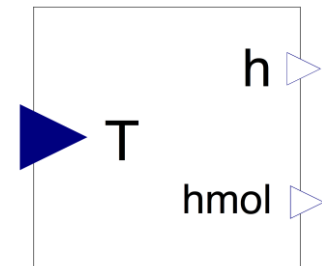


Check_h_T_Test



Check_h_T

```
model Check_h_T
  replaceable package Medium = Modelica.Media.IdealGases.SingleGases.CO2;
  parameter Boolean exclEnthForm=false;
  parameter ReferenceEnthalpy refChoice=ReferenceEnthalpy.ZeroAt25C;
  parameter Modelica.SIunits.SpecificEnthalpy h_off=0.0;
  Modelica.Blocks.Interfaces.RealInput T annotation( ...);
  Modelica.Blocks.Interfaces.RealOutput h annotation( ...);
  Modelica.Blocks.Interfaces.RealOutput hmol annotation( ...);
equation
  h = Modelica.Media.IdealGases.Common.Functions.h_T(
    Medium.data, T, exclEnthForm, refChoice, h_off);
  hmol = h*Medium.data.MM;
  annotation( ...);
end Check_h_T;
```



Check_h_T_Test

```
model Check_h_T_Test
  replaceable package Medium = Modelica.Media.IdealGases.SingleGases.CH4;
  Check_h_T check_h_T1(redeclare package Medium = Medium,
    exclEnthForm = false, refChoice = ReferenceEnthalpy.ZeroAt25C) annotation( ...); (1)
  Check_h_T check_h_T2(redeclare package Medium = Medium,
    exclEnthForm = true, refChoice = ReferenceEnthalpy.ZeroAt25C) annotation( ...); (2)
  Check_h_T check_h_T3(redeclare package Medium = Medium,
    exclEnthForm = true, refChoice = ReferenceEnthalpy.ZeroAt0K) annotation( ...); (3)
  Check_h_T check_h_T4(redeclare package Medium = Medium,
    exclEnthForm = true, h_off = 200000, refChoice = ReferenceEnthalpy.UserDefined) (4)
    annotation( ...);
  Modelica.Blocks.Sources.Ramp temperature(
    duration = 1000, height = 800, offset = 200, startTime = 0) annotation( ...);
equation
  connect(temperature.y, check_h_T4.T) annotation( ...);
  connect(temperature.y, check_h_T3.T) annotation( ...);
  connect(temperature.y, check_h_T2.T) annotation( ...);
  connect(temperature.y, check_h_T1.T) annotation( ...);
annotation( ...
  __OpenModelica_simulationFlags(lv = "LOG_STATS", outputFormat = "mat", s = "dassl"));
end Check_h_T_Test;
```

IV. 比エンタルピを Modelica 関数にする理由

[Modelica.Media.UsersGuide.MediumDefinition.SpecificEnthalpyAsFunction](#) より

もし medium の独立変数が(p, h)でない場合、比エンタルピは、その独立変数のModelica 関数として計算すべきである。

BaseProperties モデルの中の式

$$h = \left(-\frac{a_1}{T^2} + a_2 \frac{\log T}{T} + \sum_{i=3}^7 a_i \frac{T^{i-3}}{i-2} + \frac{b_1}{T} \right) + h_1 + h_2$$

状態変数 p, T
の方程式



$$h = h_T(T)$$

状態変数 p, T を引数とする Modelica 関数

```
function h_T;  
  input Temperature T;  
  ...  
  output specificEnthalpy h;  
algorithm  
   $h := RT \left( -\frac{a_1}{T^2} + a_2 \frac{\log T}{T} + \sum_{i=3}^7 a_i \frac{T^{i-3}}{i-2} + \frac{b_1}{T} \right) + h_1 + h_2 ;$   
end h_T
```

実際のソースコードは
もっと式を変形してます。

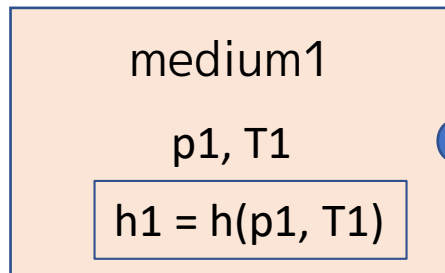


FluidPort が次のように定義されている場合を考える

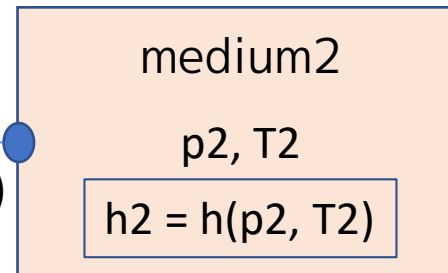
```
connector FluidPort
  replaceable package Medium = Modelica.Media.Interfaces.PartialMedium;
  Medium.AbsolutePressure p;
  flow Medium.MassFlowRate m_flow;
  Medium.SpecificEnthalpy h;
  flow Medium.EnthalpyFlowRate H_flow;
  Medium.MassFraction Xi[Medium.nXi];
  flow Medium.MassFlowRate mX_flow[Medium.nXi];
end FluidPort
```

FluidPort の方程式 (純物質の場合)

$$\begin{aligned} p_1 &= p_2 \\ h_1 &= h_2 \\ 0 &= m_flow_1 + m_flow_2 \\ 0 &= H_flow_1 + H_flow_2 \end{aligned}$$



BaseProperties の式



BaseProperties の式

(p1,h1) (p2,h2)

(p1, T1)が与えられたときに(p2, T2)を求める場合を考える。

ポテンシャル変数に関する方程式

```
h1 = h(p1, T1)
h2 = h(p2, T2)
p1 = p2
h1 = h2
```

h(p, T)が方程式として
与えられる場合

```
h1 := h(p1, T1)
p2 := p1
h2 := h1
0 := h2 - h(p2, T2)
```

すべてのFluidPortで
非線形方程式を解くこと
によって T2 を求めることになる。

h(p, T)が Modelica 関数として
与えられる場合

```
h1 := h(p1, T1)
h1 := h(p1, T2)
```

Modelica のトランスレータが
左辺と第1引数が一致するので
第2引数も一致すると解釈する。

```
T2 := T1
```

非線形方程式を解く必要はない!!

現状のFluidPortでは...

Modelica.Fluid.Interfaces.FluidPort

```
connector FluidPort
  "Interface for quasi one-dimensional fluid flow in a piping network (incompressible or
  compressible, one or more phases, one or more substances)"

  replaceable package Medium = Modelica.Media.Interfaces.PartialMedium
    "Medium model" annotation (choicesAllMatching=true);

  flow Medium.MassFlowRate m_flow
    "Mass flow rate from the connection point into the component";
  Medium.AbsolutePressure p "Thermodynamic pressure in the connection point";
  stream Medium.SpecificEnthalpy h_outflow
    "Specific thermodynamic enthalpy close to the connection point if m_flow < 0";
  stream Medium.MassFraction Xi_outflow[Medium.nXi]
    "Independent mixture mass fractions m_i/m close to the connection point if m_flow < 0";
  stream Medium.ExtraProperty C_outflow[Medium.nC]
    "Properties c_i/m close to the connection point if m_flow < 0";
end FluidPort;
```

FluidPortの比エンタルピは stream 変数として定義されている。
流れが medium1 から medium 2 に向かう場合、方程式は以下のようなになる。

```
h_outflow1 = h(p1,T1)
inStream(h_outflow2) = h(p2,T2)
p1 = p2
h_outflow1 = inStream(h_outflow2)
```



```
h_outflow1 = h(p1,T1)
h_outflow1 = h(p2,T2)
```

5. setState_XXX

I. setState_XXX 関数

異なる熱力学的状態変数の組み合わせから **ThermodynamicState** を返す関数

A. PartialMedium では partial function (algorithm が未定義)

- *setState_pTX(p, T, X)*
- *setState_phX(p, h, X)*
- *setState_psX(p, s, X)*
- *setState_dTX(d, T, X)*

p: 圧力

T: 温度

h: 比エンタルピ

s: 比エントロピ

d: 密度

X[nX]: 質量分率

B. PartialPureSubstances

質量分率 X なしで熱力学的状態を返す関数が導入されている。

- *setState_pT(p, T) = setState_pTX(p, T, fill(0,0))*
- *setState_ph(p, h) = setState_phX(p, h, fill(0,0))*
- *setState_ps(p, s) = setState_psX(p, s, fill(0,0))*
- *setState_dT(d, T) = setState_dTX(d, T, fill(0,0))*

partial 関数を使って宣言してるが partial 関数ではない!

B. PartiaPureSubstances の setState_XX のアルゴリズム

setState_dT(d,T)のアルゴリズム

```
algorithm
  state := setState_dTX(
    d,
    T,
    fill(0, 0));
end setState_dT;
```

setState_ph(p,h) のアルゴリズム

```
algorithm
  state := setState_phX(
    p,
    h,
    fill(0, 0));
end setState_ph;
```

setState_ps(p,s) のアルゴリズム

```
algorithm
  state := setState_psX(
    p,
    s,
    fill(0, 0));
end setState_ps;
```

setState_dT(d,T) のアルゴリズム

```
algorithm
  state := setState_dTX(
    d,
    T,
    fill(0, 0));
end setState_dT;
```

C. SingleGasNASA の setState_XXX

setState_pTX(p, T, X) のアルゴリズム

```
algorithm
  state := ThermodynamicState(p=p,T=T);
  annotation(Inline=true,smoothOrder=2);
end setState_pTX;
```

setState_dTX(d, T, X) のアルゴリズム

```
algorithm
  state := ThermodynamicState(p=d*data.R*T,T=T);
  annotation(Inline=true,smoothOrder=2);
end setState_dTX;
```

理想気体の状態方程式

$$p = \rho RT = \frac{1}{v} RT$$

setState_phX(p, h, X) のアルゴリズム

```
algorithm
  state := ThermodynamicState(p=p,T=T_h(h));
  annotation(Inline=true,smoothOrder=2);
end setState_phX;
```

エンタルピから
温度を求める関数
(非線形方程式)

setState_psX(p, s, X) のアルゴリズム

```
algorithm
  state := ThermodynamicState(p=p,T=T_ps(p,s));
  annotation(Inline=true,smoothOrder=2);
end setState_psX;
```

圧力とエントロピから
温度を求める関数
(非線形方程式)

II. 比エンタルピ、比エントロピから温度を求める関数

Modelica.Media.IdealGases.Common.SingleGasNasa.T_h

```
function T_h "Compute temperature from specific enthalpy"  
  extends Modelica.Icons.Function;  
  input SpecificEnthalpy h "Specific enthalpy";  
  output Temperature T "Temperature";
```

Protected

package **Internal** ①

"Solve $h(\text{data}, T)$ for T with given h (use only indirectly via `temperature_phX`)"

extends Modelica.Media.Common.**OneNonLinearEquation**;

redeclare record extends **f_nonlinear_Data** ③

"Data to be passed to non-linear function"

extends Modelica.Media.IdealGases.Common.DataRecord;

end f_nonlinear_Data;

redeclare function extends **f_nonlinear** ②

algorithm

y := Modelica.Media.IdealGases.Common.Functions.h T(
 f_nonlinear_data, x);

end f_nonlinear;

// Dummy definition has to be added for current Dymola

redeclare function extends **solve** ④

end solve;

end Internal;

algorithm

T := Internal.solve(h, 200, 6000, 1.0e5, {1}, data);

end T_h;

エンタルピから温度
を求める非線形方程式

$$y = h_T(x)$$

x : 温度

y : エンタルピ

Modelica.Media.IdealGases.Common.SingleGasNasa.T_ps

```
function T_ps "Compute temperature from pressure and specific entropy"
  extends Modelica.Icons.Function;
  input AbsolutePressure p "Pressure";
  input SpecificEntropy s "Specific entropy";
  output Temperature T "Temperature";
Protected
  package Internal ①
    "Solve h(data,T) for T with given h (use only indirectly via temperature_phX)"
    extends Modelica.Media.Common.OneNonLinearEquation;

    redeclare record extends f_nonlinear_Data ③
      "Data to be passed to non-linear function"
      extends Modelica.Media.IdealGases.Common.DataRecord;
    end f_nonlinear_Data;

    redeclare function extends f_nonlinear ②
    algorithm
      y := Modelica.Media.IdealGases.Common.Functions.s0 T(
        f_nonlinear_data,x)- data.R*Modelica.Math.log(p/reference_p);
    end f_nonlinear;

    // Dummy definition has to be added for current Dymola
    redeclare function extends solve ④
    end solve;
  end Internal
algorithm
  T := Internal.solve(s, 200, 6000, p, {1}, data);
end T_ps;
```

圧力とエントロピから温度
を求める非線形方程式

$$y = s(x, p) = s_0 T(x) - R \ln \frac{p}{p_0}$$

$p_0 = \text{reference_p}$

x: 温度
y: エントロピ

Media.IdealGases.Common.Functions.s0_T

比エントロピーの温度依存性

ソースコードの式

$$s0_T(T) = R(b_2 - \frac{1}{2} \frac{a_1}{T^2} - \frac{a_2}{T} + a_3 \log T + T(a_4 + T(\frac{1}{2} a_5 + T(\frac{1}{3} a_6 + \frac{1}{4} a_7 T))))$$

$$= R \left(-\frac{a_1}{2T^2} - \frac{a_2}{T} + a_3 \log T + \sum_{i=4}^7 a_i \frac{T^{i-3}}{i-3} + b_2 \right)$$

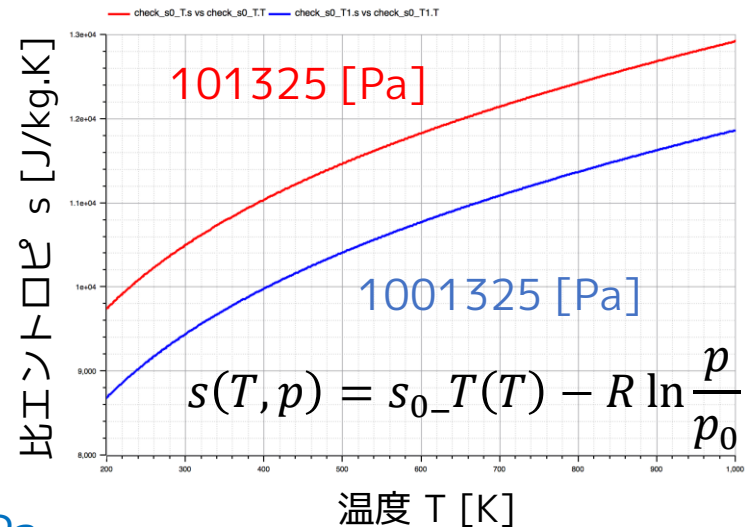
DataRecord の Document の式

$$a_i = \begin{cases} a_{low}[i] & T < T_{limit} \\ a_{high}[i] & T \geq T_{limit} \end{cases} \quad b_2 = \begin{cases} b_{low}[2] & T < T_{limit} \\ b_{high}[2] & T \geq T_{limit} \end{cases}$$

標準エントロピーとの比較 [J/mol.K]

s0_T(298.15) 化学便覧

	s0_T(298.15)	化学便覧
CH4	186.37	186.38
C2H6	229.22	229.60
CO2	213.786	213.74
H2	130.68	130.68
H2O	188.828	188.83



化学便覧は標準圧力を 0.1 MPa = 100000 Pa

本モデルは $p_0 = \text{reference_p} = 101325 \text{ Pa}$ とするため違いが出る可能性あり。

関数 T_h(h) と T_ps(p,s) の構成

① 関数内に非線形方程式を解く機能を追加するため、OneNonLinearEquationを継承する ローカルパッケージ **Internal** を宣言する。

<<function>> T_h
<<input variable>> h
<<output variable>> T
package Internal
algorithm T = Internal.solve(h,200,6000,1.0e5, { 1 },data)

<<function>> T_ps
<<input variable>> p : AbsolutePressure
<<input variable>> s : Specific tropy
<<output variable>> T : Temperature
package Internal
algorithm T = Internal.solve(s,200,6000,p,{ 1 },data)

1変数の非線形方程式ソルバ

$$y_0 = f_{\text{nonlinear}}(x, p, X, f_{\text{nonlinear_data}})$$

<<replaceable>>
partial record f_nonlinear_Data
partial function f_nonlinear
function solve

<<package>>
OneNonLinearEquation

- 非線形方程式が使うデータの型 (レコード)
- 非線形方程式
- 解を返す関数

<<redeclare>> T_h.Internal.f_nonlinear_Data
<<redeclare>> T_h.Internal.f_nonliner
<<redeclare>> T_h.Internal.solve

<<redeclare>> T_ps.Internal.f_nonlinear_Data
<<redeclare>> T_ps.Internal.f_nonliner
<<redeclare>> T_ps.Internal.solve

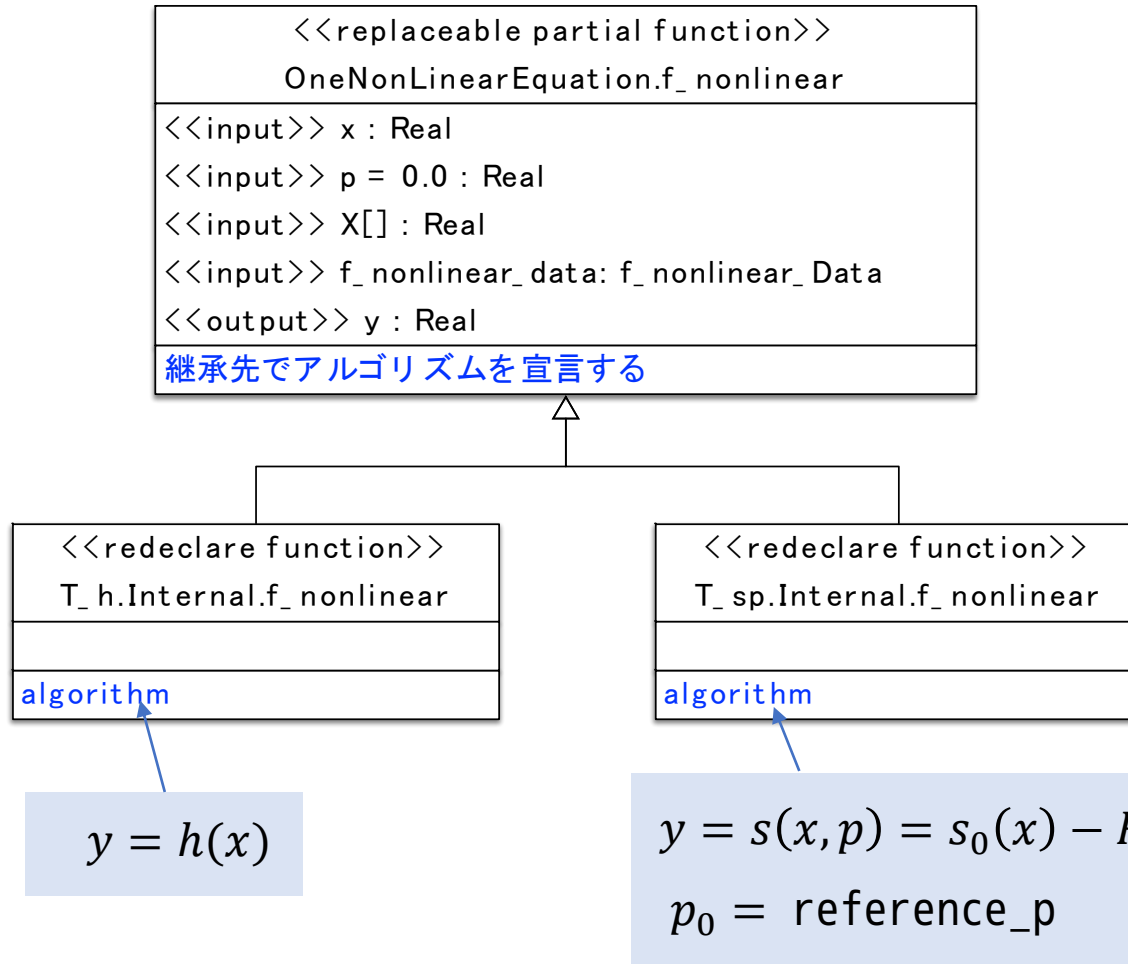
<<package>> .T_h.Internal

<<package>> T_ps.Internal

関数 T_h(h) と T_ps(p,s) の構成

② 関数 **f_nonlinear** の algorithm文に非線形方程式を宣言する。

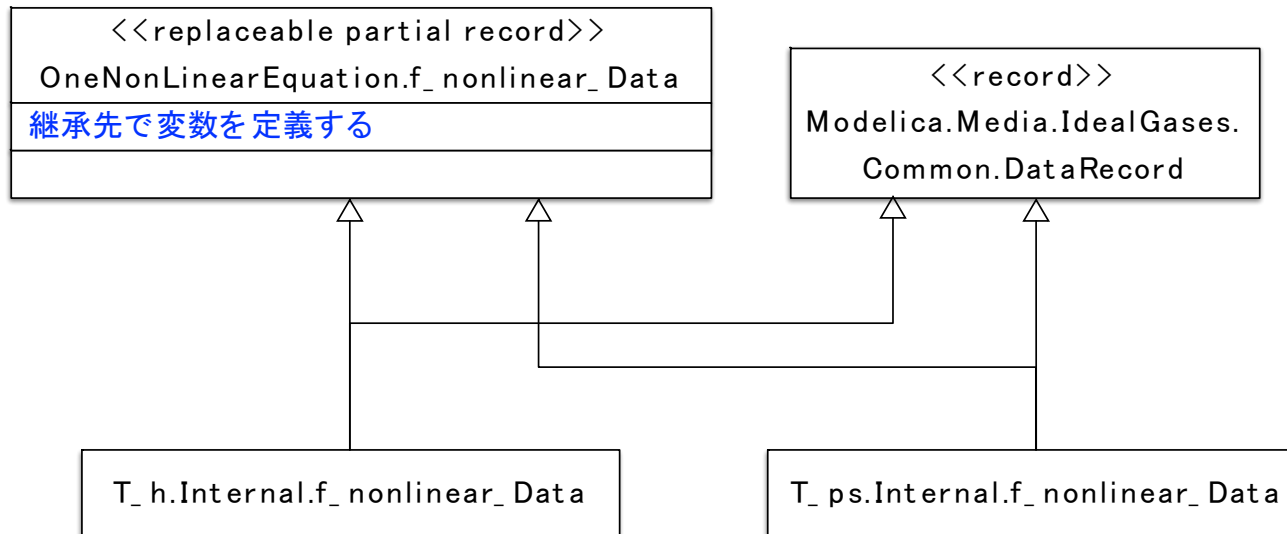
非線形方程式 $y = f_{\text{nonlinear}}(x, p, X, f_{\text{nonlinear_data}})$



関数 $T_h(h)$ と $T_{ps}(p,s)$ の構成

③ レコード `f_nonlinear_Data` に 非線形方程式が使用するデータの型を宣言する。

$f_{nonlinear}$ を作るのに必要なデータの型（レコード）

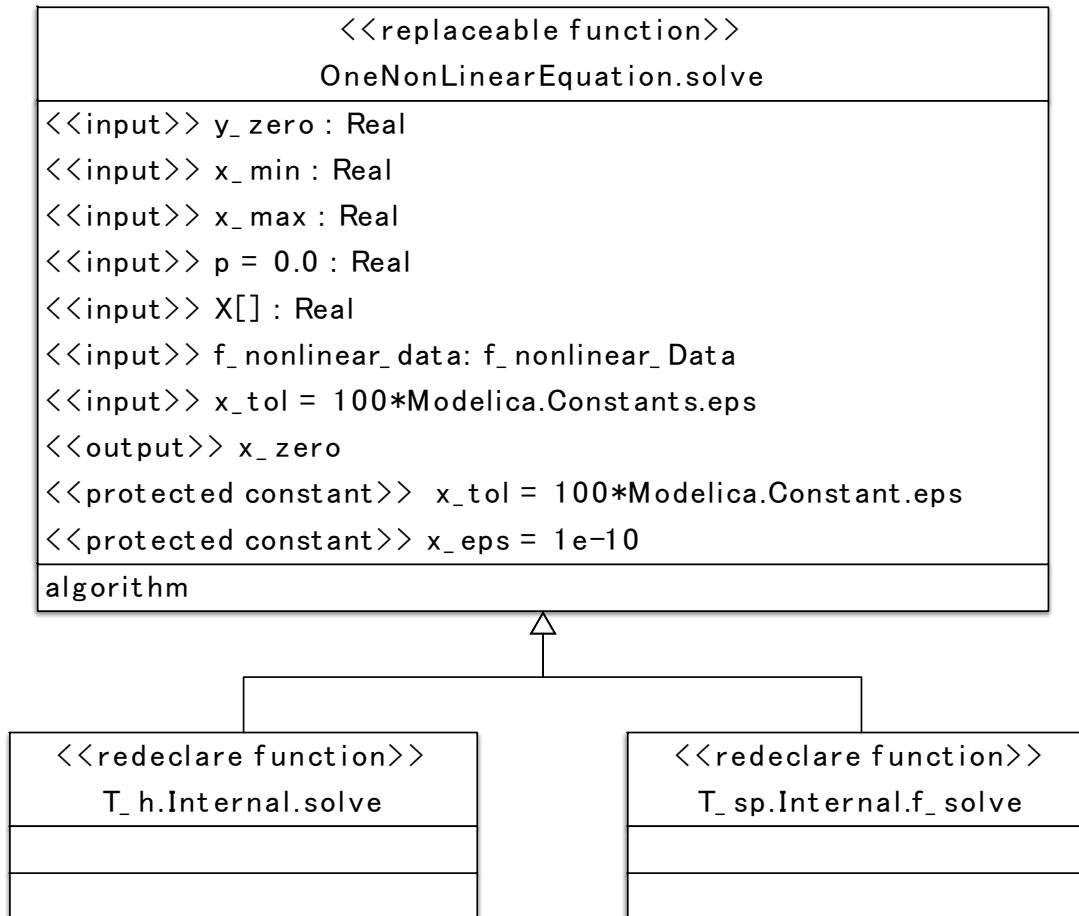


変数を新たに定義する代わりに
DataRecordを継承している

関数 T_h(h) と T_ps(p,s) の構成

- ④ 解を返す関数 **solve** を継承して再宣言する。
(現状の Dymola ではダミーの再宣言が必要)

$x_0 = \text{solve}(y_0, x_{\min}, x_{\max}, p, X, f_{\text{nonlinear_data}}, x_{\text{tol}})$

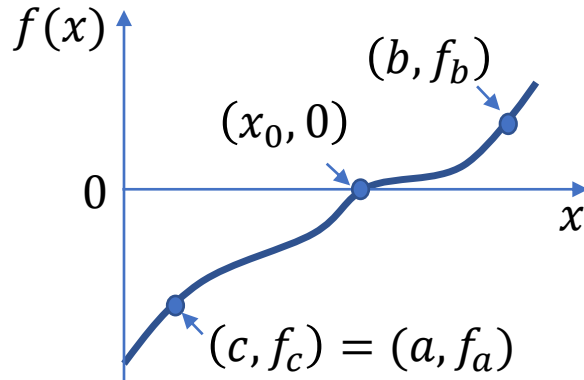


Modelica.Media.Common.OneNonLinearEquation.solve のアルゴリズム概略

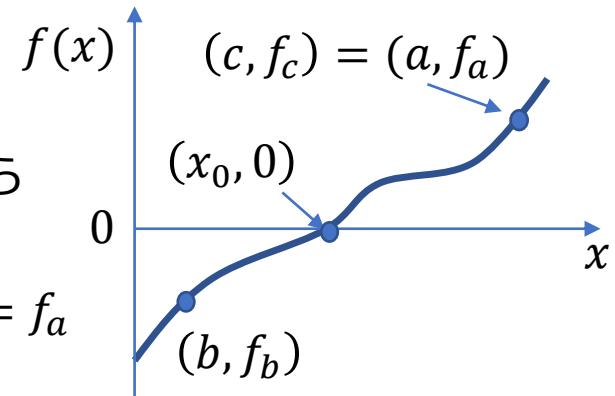
Brent R.P.: **Algorithms for Minimization without derivatives**. Prentice Hall, 1973, pp. 58-59.

非線形方程式 $f(x) \equiv f_{\text{nonlinear}}(x) - y_0 = 0$, $f_a = f(a), f_b = f(b), f_c = f(c)$
 $(a, f_a), (b, f_b), (c, f_c)$ から $(x_0, 0)$ を推定するプロセスを繰り返す。

初期状態 $a = x_{\min 2} = x_{\min} - x_\varepsilon$
 $b = x_{\max 2} = x_{\max} + x_\varepsilon$
 $c = a$



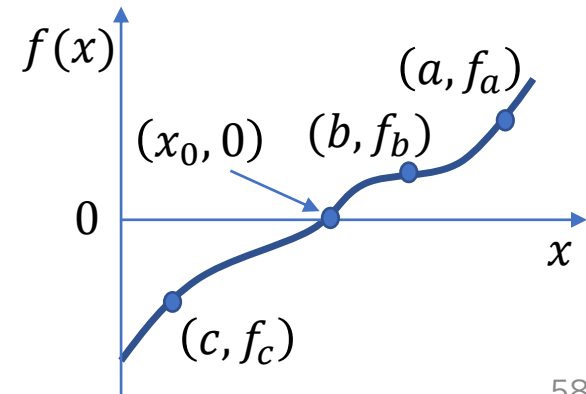
もし $|f_c| < |f_b|$ なら
 $a = b, b = c, c = a$
 $f_a = f_b, f_b = f_c, f_c = f_a$



収束判定前の状態

a, b, c の入れ替えを行って、次のようになるようにする。

- 解 x_0 は b と c の間にある。
- $|f_b| \leq |f_c|$
- a は、 b と c の外にあるか $a = c$ となる



収束判定

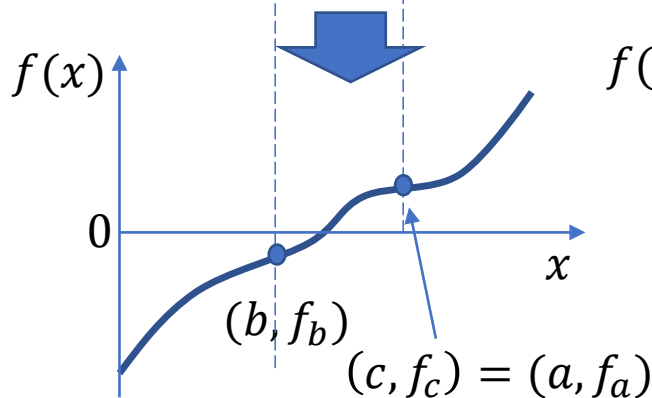
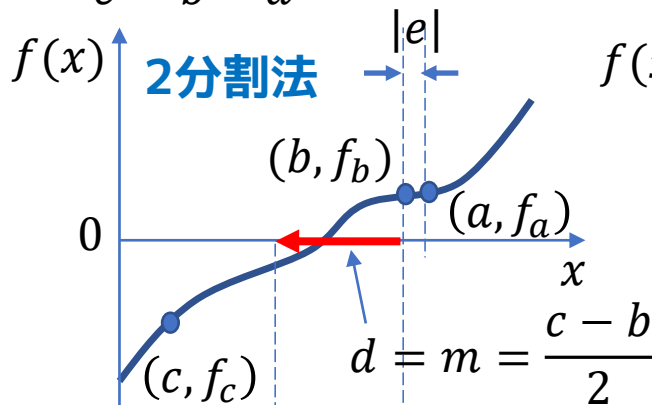
$$m = \frac{c - b}{2}, \quad tol = 2\varepsilon|b| + x_{tol}$$

$|m| < tol$ または $f_b = 0$ なら収束 $\implies x_0 = b$

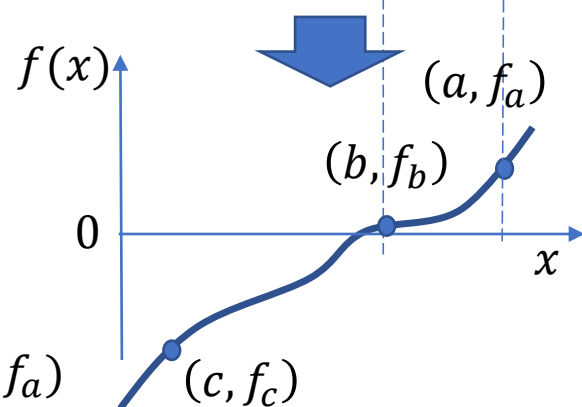
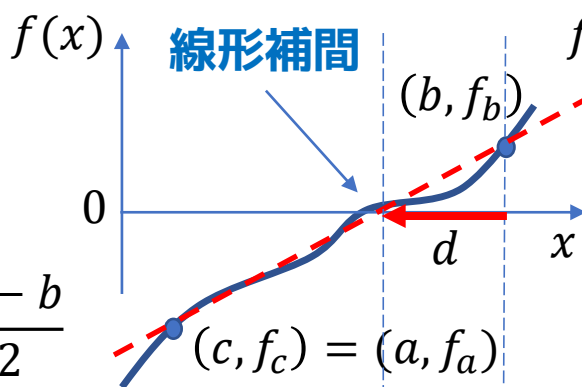
収束していない場合、以下のような方法で $(a, f_a), (b, f_b), (c, f_c)$ の移動を繰り返して、収束するまで解の範囲を狭めていく。

① $|e| < tol$ または $|f_a| < |f_b|$ の場合

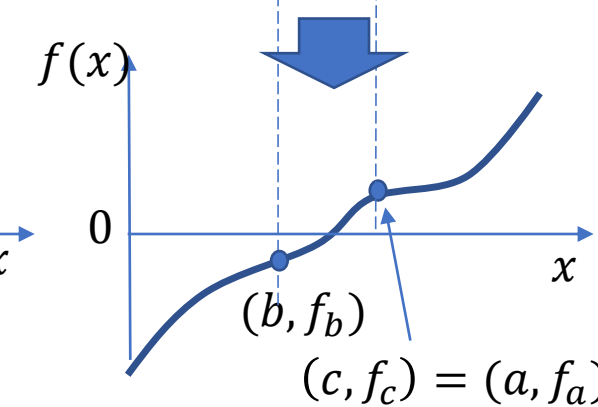
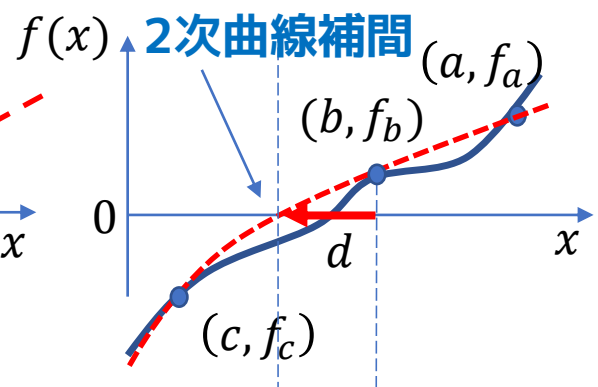
$$e = b - a$$



② $a = c$ の場合

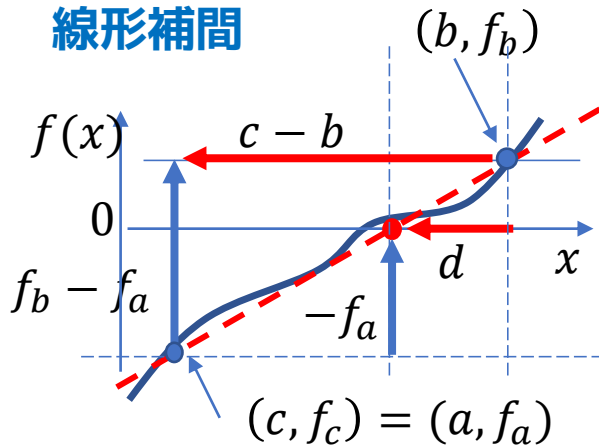


③ $a \neq c$ の場合



補間式の検算

線形補間



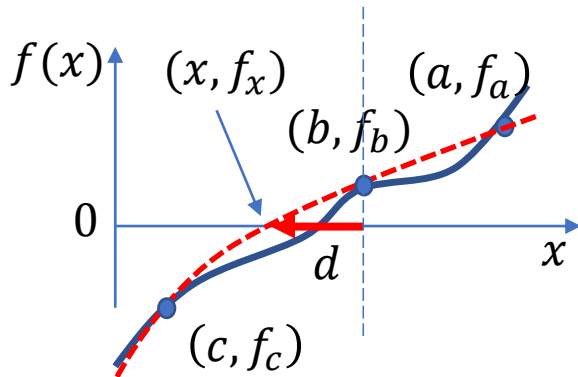
$$m = \frac{c-b}{2}, s = \frac{f_b}{f_a}$$

$$p = 2ms$$

$$q = 1-s$$

$$d = \frac{-p}{q} = (c-b) \frac{-f_a}{f_b - f_a}$$

2次曲線補間



$$m = \frac{c-b}{2}, s = \frac{f_b}{f_a},$$

$$q = \frac{f_a}{f_c}, r = \frac{f_b}{f_c}$$

$$p = s(2mq(q-r) - (b-a)(r-1))$$

$$q' = (q-1)(r-1)(s-1)$$

$$d = -\frac{p}{q'} = \frac{(a-b)f_b f_c}{(f_a - f_c)(f_a - f_b)} + \frac{(c-b)f_a f_b}{(f_c - f_b)(f_c - f_a)}$$

ラグランジュの補間公式で $f_x = 0$ とした場合と一致する

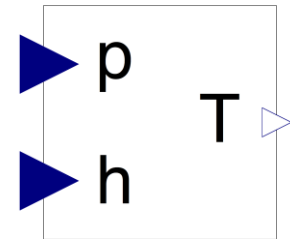
$$d = x - b = (a-b) \frac{(f_x - f_b)(f_x - f_c)}{(f_a - f_b)(f_a - f_c)} + (b-b) \frac{(f_x - f_a)(f_x - f_c)}{(f_b - f_a)(f_b - f_c)} + (c-b) \frac{(f_x - f_a)(f_x - f_b)}{(f_c - f_a)(f_c - f_b)}$$

```
// Determine if a bisection is needed
if abs(e) < tol or abs(fa) <= abs(fb) then
  e := m;
  d := e;
else
  s := fb/fa;
  if a == c then
    // linear interpolation
    p := 2*m*s;
    q := 1 - s;
  else
    // inverse quadratic interpolation
    q := fa/fc;
    r := fb/fc;
    p := s*(2*m*q*(q - r) - (b - a)*(r - 1));
    q := (q - 1)*(r - 1)*(s - 1);
  end if;
  if p > 0 then
    q := -q;
  else
    p := -p;
  end if;
  s := e;
  e := d;
  if 2*p < 3*m*q - abs(tol*q) and p < abs(0.5*s*q) then
    // interpolation successful
    d := p/q;
  else
    // use bi-section
    e := m;
    d := e;
  end if;
end if;
```

setState_ph(p,h) のテスト

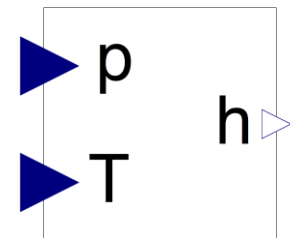
エンタルピから温度を求める

```
model T_ph_Check
  replaceable package Medium = Modelica.Media.IdealGases.SingleGases.CH4;
  Medium.ThermodynamicState state;
  Modelica.Blocks.Interfaces.RealInput p annotation( ...);
  Modelica.Blocks.Interfaces.RealInput h annotation( ...);
  Modelica.Blocks.Interfaces.RealOutput T annotation( ...);
equation
  state = Medium.setState_ph(p, h);
  T = state.T;
  annotation( ...);
end T_h_Check;
```



温度からエンタルピを求める

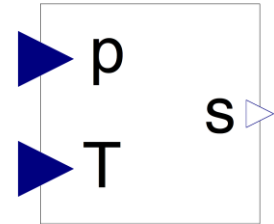
```
model H_pT_Check
  replaceable package Medium = Modelica.Media.IdealGases.SingleGases.CH4;
  Medium.ThermodynamicState state;
  Modelica.Blocks.Interfaces.RealInput p annotation( ...);
  Modelica.Blocks.Interfaces.RealInput T annotation( ...);
  Modelica.Blocks.Interfaces.RealOutput h annotation( ...);
equation
  state = Medium.setState_pT(p, T);
  h = Medium.specificEnthalpy(state);
  annotation( ...);
end H_T_Check;
```



setState_ps(p,s) のテスト

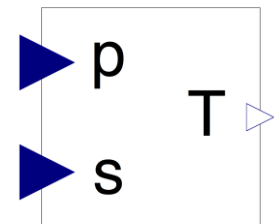
圧力と温度からエントロピを求める

```
model S_pT_Check
  replaceable package Medium = Modelica.Media.IdealGases.SingleGases.CH4;
  Medium.ThermodynamicState state;
  Modelica.Blocks.Interfaces.RealInput p annotation( ...);
  Modelica.Blocks.Interfaces.RealInput T annotation( ...);
  Modelica.Blocks.Interfaces.RealOutput s annotation( ...);
equation
  state = Medium.setState_pT(p, T);
  s = Medium.specificEntropy(state);
  annotation( ...);
end S_pT_Check;
```

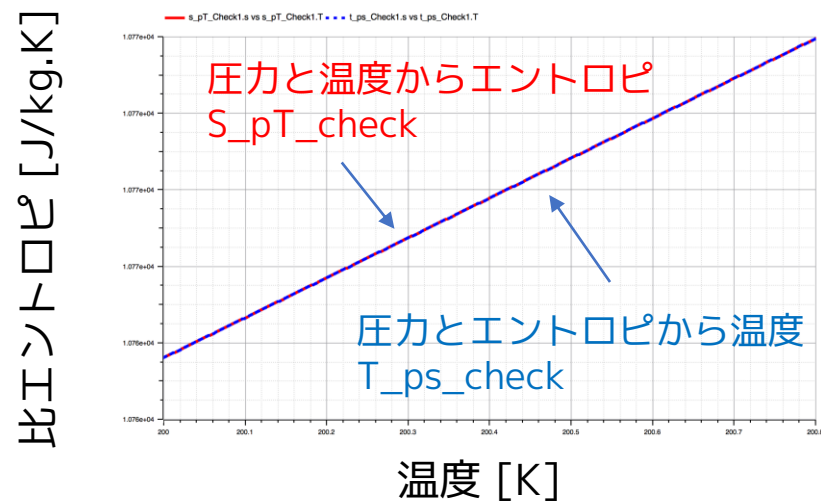
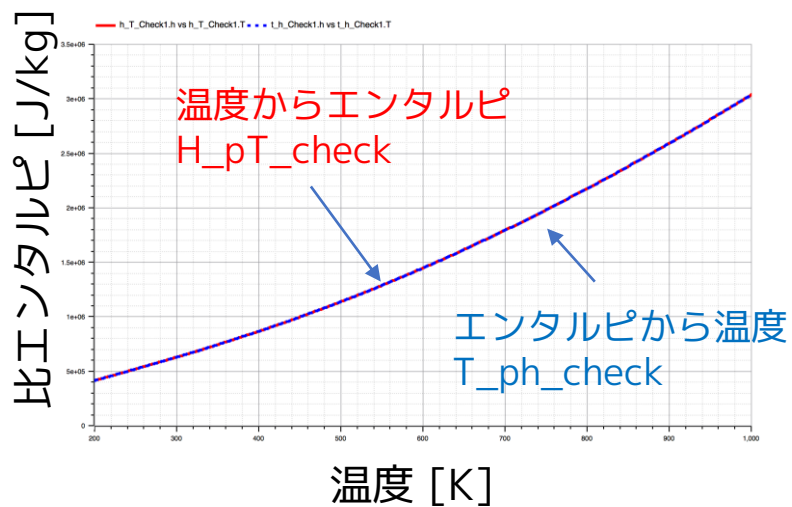
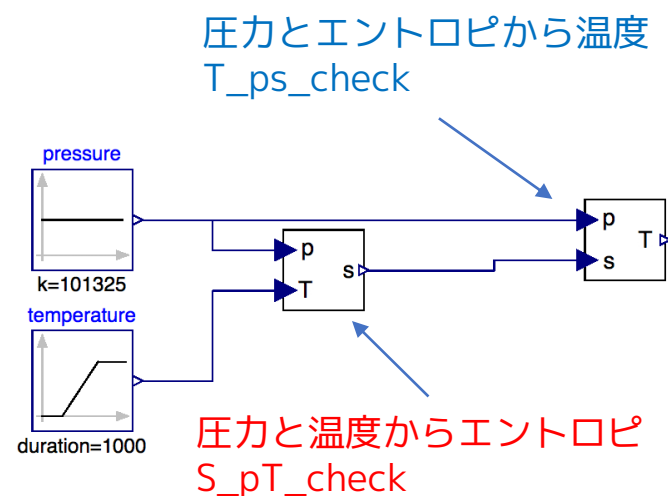
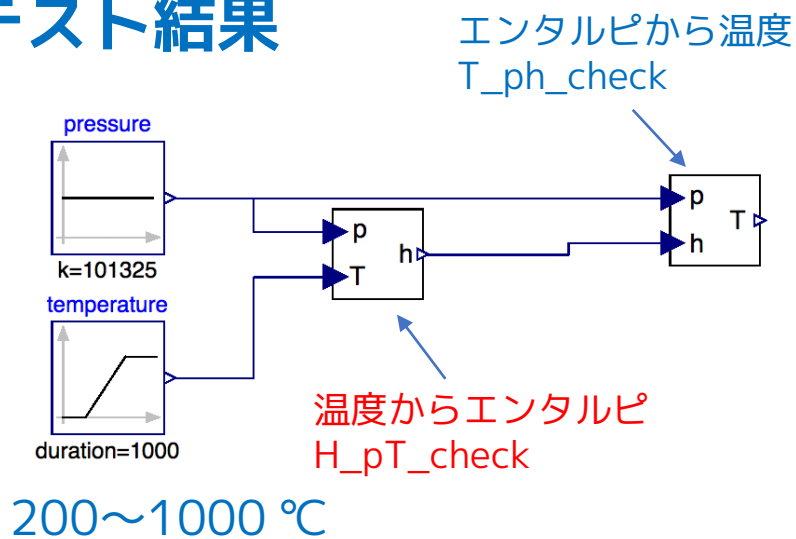


圧力とエントロピから温度を求める

```
model T_ps_Check
  replaceable package Medium = Modelica.Media.IdealGases.SingleGases.CH4;
  Medium.ThermodynamicState state;
  Modelica.Blocks.Interfaces.RealInput p annotation( ...);
  Modelica.Blocks.Interfaces.RealInput s annotation( ...);
  Modelica.Blocks.Interfaces.RealOutput T annotation( ...);
equation
  state = Medium.setState_ps(p, s);
  T = state.T;
  annotation( ...);
end T_ps_Check;
```



テスト結果



6. 物性値関数

I. 熱力学の状態変数を返す関数

A. PartialMedium

- *pressure(state)*
 - *temperature(state)*
 - *density(state)*
 - *specificEnthalpy(state)*
 - *specificInternalenergy(state)*
 - *specificEntropy(state)*
 - *specificGibbsEnergy(state)*
 - *specificHelmholtzEnergy(state)*
- SingleGasNasa で宣言する
partial function
- $\text{temperature_phX}(p, h, X) = \text{temperature}(\text{setState_phX}(p, h, X))$
 - $\text{temperarure_psX}(p, s, X) = \text{temperature}(\text{setState_psX}(p, s, X))$
 - $\text{density_pTX}(p, T, X) = \text{density}(\text{setState_pTX}(p, T, X))$
 - $\text{density_phX}(p, h, X) = \text{density}(\text{setState_phx}(p, h, X))$
 - $\text{density_psX}(p, s, X) = \text{density}(\text{setState_psX}(p, s, X))$
 - $\text{specificEnthalpy_pTX}(p, T, X) = \text{specificEnthalpy}(\text{setState_pTX}(p, T, X))$
 - $\text{specificEnthalpy_psX}(p, s, X) = \text{specificEnthalpy}(\text{setState_psX}(p, s, X))$
 - $\text{specificEntropy_pTX}(p, t, X) = \text{specificEnthalpy}(\text{setState_pTX}(p, t, X))$

B. PartialPureSubstance

- $\text{temperature_ph}(p,h) = \text{temperature_phX}(p,h,\text{fill}(0,0))$
- $\text{temperarure_ps}(p,s) = \text{temperature_psX}(p,s,\text{fill}(0,0))$
- $\text{presure_dT}(d,T) = \text{pressure}(\text{setState_dTX}(d,T,\text{fill}(0,0)))$
- $\text{density_pT}(p,T) = \text{density}(\text{setState_pTX}(p,T,\text{fill}(0,0)))$
- $\text{density_ph}(p,h) = \text{density_phX}(p,h,\text{fill}(0,0))$
- $\text{density_ps}(p,s) = \text{density_psX}(p,s,\text{fill}(0,0))$
- $\text{specificEnthalpy_dT}(d,T) = \text{specificEnthalpy}(\text{setState_dTX}(d,T,\text{fill}(0,0)))$
- $\text{specificEnthalpy_ps}(p,s) = \text{specificEnthalpy_psX}(p,s,\text{fill}(0,0))$
- $\text{specificEnthalpy_pT}(p,T) = \text{specificEnthalpy_pTX}(p,T,\text{fill}(0,0))$

C. SingleGasNasa の式の中身

- `pressure(state) = state.p`
- `temperature(state) = state.T`
- `density(state) = state.p/(data.R*state.T)`
- `specificEnthalpy(state) =
Modelica.Media.IdealGases.Common.Functions.h_T(data, state.T)`
- `specificInternalEnergy(state) =
Modelica.Media.IdealGases.Common.Functions.h_T(data, state.T) -
data.R*state.T`
- `specificEntropy(state) =
Modelica.Media.IdealGases.Common.Functions.s0_T(data, state.T) -
data.R*Modelica.Math.log(state.p/reference_p)`
- `specificGibbsEnergy(state) =
Modelica.Media.IdealGases.Common.Functions.h_T(data, state.T) -
state.T*specificEntropy(state)`
- `specificHelmholtzEnergy(state) =
Modelica.Media.IdealGases.Common.Functions.h_t(data, state.T) -
data.R*state.T - state.T*specificEntropy(state)`

密度(状態方程式)

$$\rho = \frac{1}{v} = \frac{p}{RT}$$

内部エネルギー

$$u = h - pv = h - RT$$

比エントロピ

$$s(T, p) = s_0(T) - R \ln \frac{p}{p_0}$$

$p_0 = \text{reference_p}$

ギブズ自由エネルギー

$$g = h - Ts$$

ヘルムホルツ自由エネルギー

$$f = u - Ts = h - RT - Ts$$

II. 粘性率

A. PartialMedium

- *dynamicViscosity(state)*

C. SingleGasNasa

- *dynamicViscosity(state)*
- *dynamicViscosityLowPressure(T,Tc,M,Vc,w,mu,k)*

SingleGasNasa.dynamicViscosity(state)

```
function extends dynamicViscosity "Dynamic viscosity"
algorithm
  assert(fluidConstants[1].hasCriticalData,
    "Failed to compute dynamicViscosity: For the species ¥"
    + mediumName + "¥" no critical data is available.");
  assert(fluidConstants[1].hasDipoleMoment,
    "Failed to compute dynamicViscosity: For the species ¥"
    + mediumName + "¥" no critical data is available.");
  eta := Modelica.Media.IdealGases.Common.Functions.dynamicViscosityLowPressure(
    state.T,
    fluidConstants[1].criticalTemperature,
    fluidConstants[1].molarMass,
    fluidConstants[1].criticalMolarVolume,
    fluidConstants[1].acentricFactor,
    fluidConstants[1].dipoleMoment);
  annotation (smoothOrder=2);
end dynamicViscosity;
```

定数レコード
fluidConstants
のデータ

Chung, et al. (1984, 1988) の粘性率の計算方法

Bruce E. Poling, John E. Prausnitz, John P. O'Connell, "The Properties of Gases and Liquids" 5th Ed. Mc Graw Hill.

$$\eta = 40.785 \frac{F_c (MT)^{1/2}}{V_c^{2/3} \Omega_v}$$

$$F_c = 1 - 0.2756 \omega + 0.059035 \mu_r^4 + \kappa$$

: 分子の形状や希薄ガスの極性の効果

$$\mu_r = 131.3 \frac{\mu}{(V_c T_c)^{1/2}}$$

: 無次元双極子モーメント
(dimensionless dipole moment)

$$\Omega_v = [A(T^*)^{-B}] + C[\exp(-DT^*)] + E[\exp(-FT^*)]$$

: 粘性衝突積分 (viscosity collision integral)

$$A = 1.16145, B = 0.14874, C = 0.52487, D = 0.77320, E = 2.16178, F = 2.43787$$

$$T^* = \frac{k}{\varepsilon} T = 1.2593 \frac{T}{T_c}$$

k : Boltzmann 定数

ε : pair-potential energy の最小値

η : 粘性率 [μP] 1 P = 0.1 Pa.s

T : 絶対温度 [K]

M : モル質量 [g/mol]

T_c : 臨界温度 (critical temperature) [K]

V_c : 臨界体積 (critical volume) [cm^3/mol]

ω : 偏心因子 (acentric factor)

κ : アルコールや酸など高い極性をもつ
物質の修正値

(special correction for highly polar
substances such as alcohols and acid)

μ : 双極子モーメント (dipole moment)[D]

debye

SI 単位への変換

$$\eta = Const1_{SI} F_c \frac{\sqrt{MT}}{V_c^{2/3} \Omega_v}$$

$$Const1_{SI} = 40.785 * 10^{-9.5}$$

$$\mu_r = Const2_{SI} \frac{\mu}{(V_c T_c)^{1/2}}$$

$$Const2_{SI} = \frac{131.3}{1000.0}$$

$$M: 1[\text{kg/mol}] = 10^3 [\text{g/mol}]$$

$$V_c: 1[\text{m}^3/\text{mol}] = 10^6 [\text{cm}^3/\text{mol}]$$

$$\frac{(10^3)^{0.5}}{(10^6)^{2/3}} = 10^{-2.5}$$

$$10^{-9.5} = 10^{-2.5} \cdot 10^{-7}$$

$$\eta: 1 [\mu\text{P}] = 10^{-7} [\text{Pa.s}]$$

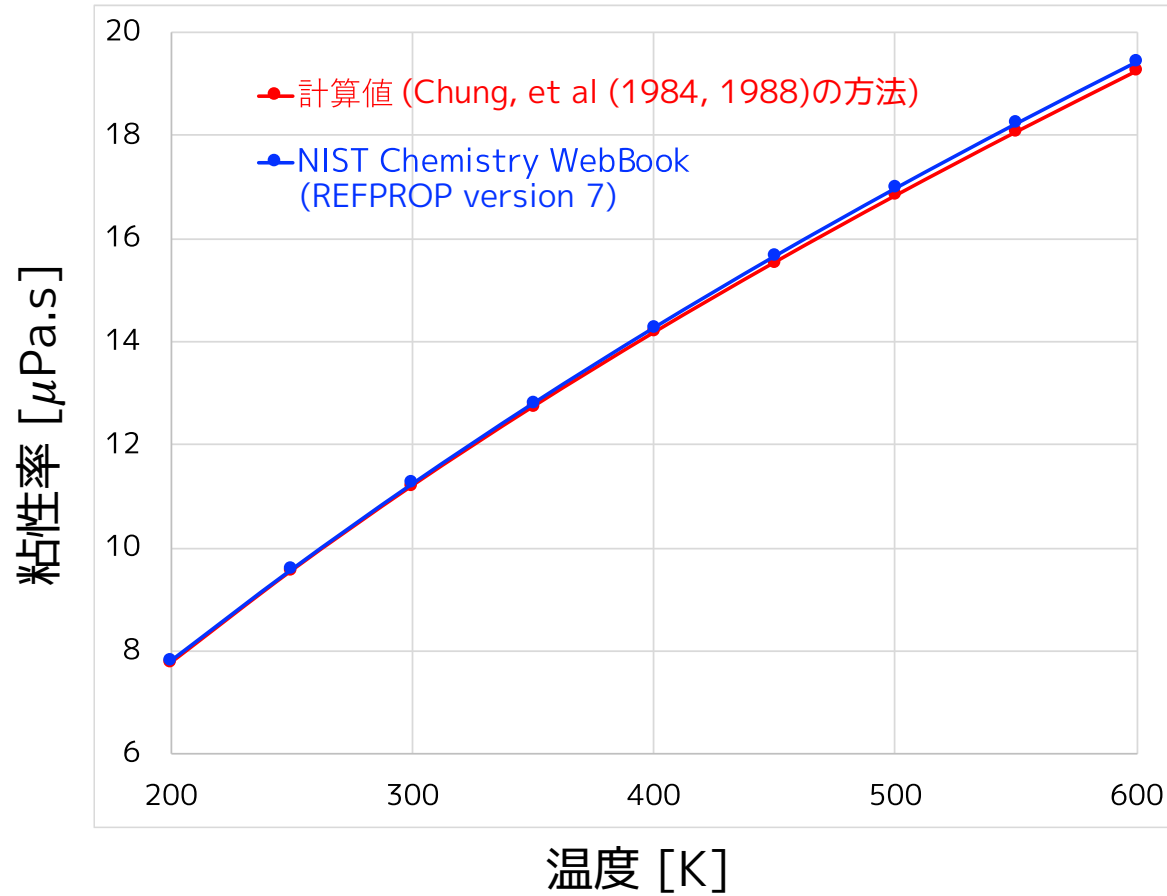
$$V_c: 1[\text{m}^3/\text{mol}] = 10^6 [\text{cm}^3/\text{mol}]$$

$$(10^6)^{1/2} = 1000$$

Modelica.Media.IdealGases.Common.Functions.dynamicViscosityLowPressure

```
function dynamicViscosityLowPressure
  "Dynamic viscosity of low pressure gases"
  extends Modelica.Icons.Function;
  input SI.Temp_K T "Gas temperature";
  input SI.Temp_K Tc "Critical temperature of gas";
  input SI.MolarMass M "Molar mass of gas";
  input SI.MolarVolume Vc "Critical molar volume of gas";
  input Real w "Acentric factor of gas";
  input Modelica.Media.Interfaces.Types.DipoleMoment mu
    "Dipole moment of gas molecule";
  input Real k = 0.0 "Special correction for highly polar substances";
  output SI.DynamicViscosity eta "Dynamic viscosity of gas";
  Protected
    parameter Real Const1_SI=40.785*10^(-9.5)
      "Constant in formula for eta converted to SI units";
    parameter Real Const2_SI=131.3/1000.0
      "Constant in formula for mur converted to SI units";
    Real mur=Const2_SI*mu/sqrt(Vc*Tc)
      "Dimensionless dipole moment of gas molecule";
    Real Fc=1 - 0.2756*w + 0.059035*mur^4 + k
      "Factor to account for molecular shape and polarities of gas";
    Real Tstar "Dimensionless temperature defined by equation below";
    Real Ov "Viscosity collision integral for the gas";
  algorithm
    Tstar := 1.2593*T/Tc;
    Ov := 1.16145*Tstar^(-0.14874) + 0.52487*Modelica.Math.exp(-0.7732*Tstar)
      + 2.16178*Modelica.Math.exp(-2.43787*Tstar);
    eta := Const1_SI*Fc*sqrt(M*T)/(Vc^(2/3)*Ov);
    annotation (smoothOrder=2, ...);
  end dynamicViscosityLowPressure;
```

CH₄ の粘性率 (101325 Pa)



III. 熱伝導率

A. PartialMedium

- *thermalConductivity(state)*

C. SingleGasNasa

- *thermalConductivity(state, method)*
- *thermalConductivityEstimate(Cp,eta,method,data)*

どちらも

Modelica.Media.IdealGases.Common.Functions.thermalConductivityEstimate
をコールする。

SingleGasNasa.thermalConductivity(state)

```
function extends thermalConductivity
  "Thermal conductivity of gas"
  // input IdealGases.Common.DataRecord data "Ideal gas data";
  input Integer method=Functions.methodForThermalConductivity
    "1: Eucken Method, 2: Modified Eucken Method";
  algorithm    assert(fluidConstants[1].hasCriticalData,
    "Failed to compute thermalConductivity: For the species ¥" + mediumName +
    "¥" no critical data is available.");
  lambda := Modelica.Media.IdealGases.Common.Functions.thermalConductivityEstimate(
    specificHeatCapacityCp(state),
    dynamicViscosity(state), method=method,data=data);
  annotation (smoothOrder=2);
end thermalConductivity;
```

method のデフォルト値 = 1

Modelica.Media.IdealGases.Common.Functions. thermalConductivityEstimate(Cp, eta, method, data)

method = 1 Eucken Method

$$\lambda = \eta \left(C_p - R + \frac{9}{4} R \right)$$

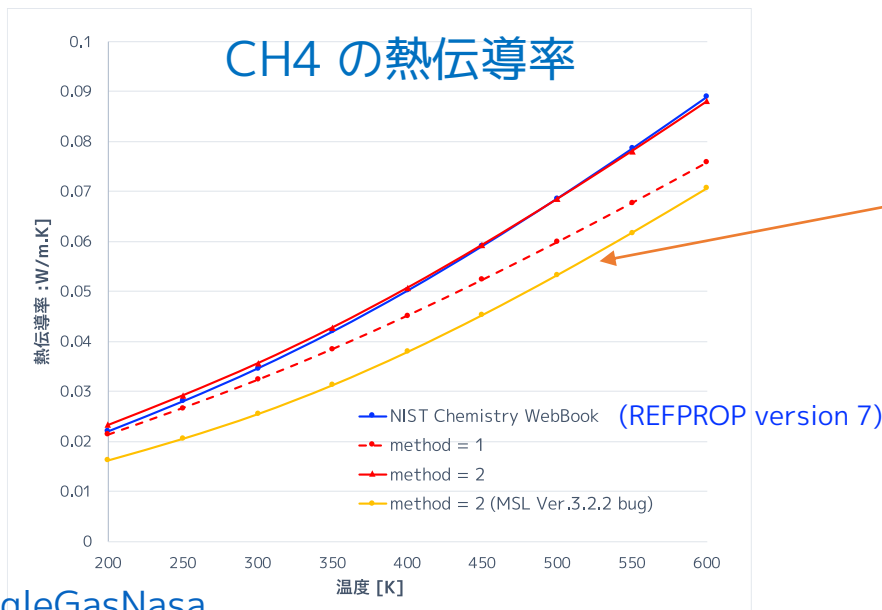
低温で有効
高温で低めに評価される

method = 2 Modified Eucken Method

$$\lambda = \eta (C_p - R) \left(1.32 + \frac{1.77}{C_p/R - 1.0} \right)$$

高温で有効
低温で高めに評価される

C_p : 比熱 [J/kg.K], M : モル質量 [kg/mol], R : 気体定数 [J/kg.K]



MSL Ver.3.2.2 ではmethod = 2 にバグあり

$$\lambda = \eta (C_p - R) \left(1.32 + \frac{1.77}{\frac{C_p}{R_{univarsal}} - 1.0} \right)$$

$R_{univarsal} = \text{Modelica.Constants.R}$
 $= 8.3144598 \text{ [J/(mol.K)]}$

修正は MSL Ver.3.2.3 milestone
になってます。

IV. 密度の偏微分

A. PartialMedium

- $density_derp_h(state)$
- $density_derh_p(state)$
- $density_derp_T(state)$
- $density_derT_p(state)$
- $density_derX(state)$

} SingleGasNasa では実装されない !

C. SingleGasNasa

- $density_derp_T(state)$
- $density_derT_p(state)$
- $density_derX(state)$

状態方程式

$$\rho = \frac{1}{v} = \frac{p}{RT}$$

$$\left(\frac{\partial \rho}{\partial p}\right)_T = \frac{1}{RT}$$

$$\left(\frac{\partial \rho}{\partial T}\right)_p = -\frac{p}{RT^2}$$

$$\left(\frac{\partial \rho}{\partial X_i}\right) = 0$$

V. 等エントロピ過程のエンタルピ

refState の状態から圧力 $p_{\text{downstream}}$ の状態に等エントロピ的に変化したときの比エンタルピを計算する。

A. PartialMedium

- *isentropicEnthalpy(p_downstream, refState)*

C. SingleGasNasa

- isentropicEnthalpy(p_downstream, refState)
- isentropicEnthalpyApproximation(p2, state, exclEnthForm, refChoice, h_off)

SingleGasNasa.isentropicEnthalpy

```
function extends isentropicEnthalpy "Return isentropic enthalpy"
  input Boolean exclEnthForm=Functions.excludeEnthalpyOfFormation
    "If true, enthalpy of formation Hf is not included in specific enthalpy h";
  input ReferenceEnthalpy refChoice=Functions.referenceChoice
    "Choice of reference enthalpy";
  input SpecificEnthalpy h_off=Functions.h_offset
    "User defined offset for reference enthalpy, if referenceChoice = UserDefined";
  algorithm
    h_is :=
      isentropicEnthalpyApproximation(p_downstream, refState, exclEnthForm, refChoice, h_off);
    annotation(Inline=true, smoothOrder=2);
end isentropicEnthalpy;
```

SingleGasNasa.isentropicEnthalpyApproximation のアルゴリズム

```
protected
  IsentropicExponent gamma = isentropicExponent(state) "Isentropic exponent";
algorithm
  h_is := Modelica.Media.IdealGases.Common.Functions.h_T(
    data,state.T,exclEnthForm,refChoice,h_off) +
    gamma/(gamma - 1.0)*state.p/density(state)*((p2/state.p)^((gamma - 1)/gamma) - 1.0);
annotation(Inline=true,smoothOrder=2);end isentropicEnthalpyApproximation;
```

(p, v) は (p_1, v_1) から等エントロピ的に変化する。

$$\frac{p}{p_1} = \left(\frac{v_1}{v}\right)^\gamma \quad \Rightarrow \quad v = v_1 p_1^{1/\gamma} p^{-\frac{1}{\gamma}} = \frac{p_1^{1/\gamma}}{\rho_1} p^{-\frac{1}{\gamma}}$$

S-P 表示による比エンタルピの全微分

$$dh = T ds + v dp$$

等エントロピ過程 $ds = 0$ より、 $dh = v dp = \frac{p_1^{1/\gamma}}{\rho_1} p^{-\frac{1}{\gamma}} dp$

$$\begin{aligned} \Delta h &= \frac{p_1^{1/\gamma}}{\rho_1} \int_{p_1}^{p_2} p^{-\frac{1}{\gamma}} dp = \frac{p_1^{1/\gamma}}{\rho_1} \left[\frac{\gamma}{\gamma - 1} p^{\frac{\gamma-1}{\gamma}} \right]_{p_1}^{p_2} \\ &= \frac{p_1^{1/\gamma}}{\rho_1} \frac{\gamma}{\gamma - 1} \left(p_2^{\frac{\gamma-1}{\gamma}} - p_1^{\frac{\gamma-1}{\gamma}} \right) = \frac{p_1}{\rho_1} \frac{\gamma}{\gamma - 1} \left(\left(\frac{p_2}{p_1} \right)^{(\gamma-1)/\gamma} - 1 \right) \\ h_{is} &= h + \Delta h \end{aligned}$$

VI その他の物性値

A. PartialMedium

- *specificHeatCapacityCp(state)*
- *specificHeatCapacityCv(state)*
- *isentropicExponent(state)*
- *velocityOfSound(state)*
- *isobaricExpansionCoefficient(state)*
- *isothermalCompressibility(state)*
- *molarMass(state)*
- *heatCapacity_cp(state) = specificHeatCapacityCp(state)*
- *heatCapacity_cv(state) = specificHeatCapacityCv(state)*
- *beta(state) = isobaricExpansionCoefficient(state)*
- *Kappa(state) = isothermalCompressibility(state)*
- *prandtlNumber(state)*

SingleGasNasa で宣言する
partial function

C. SingleGasNasa

- *specificHeatCapacityCp(state)*
- *specificHeatCapacityCv(state)*
- *isentropicExponent(state)*
- *velocityOfSound(state)*
- *isobaricExpansionCoefficient(state)*
- *isothermalCompressibility(state)*
- *molarMass(state)*

Prandtl Number (プラントル数)

$$Pr = \eta \frac{C_p}{\lambda}$$

η : 粘性率
 C_p : 定圧比熱
 λ : 熱伝導率

specificHeatCapacityCp(state) 定圧比熱

```
function extends specificHeatCapacityCp
  "Return specific heat capacity at constant pressure"
algorithm
  cp := Modelica.Media.IdealGases.Common.Functions.cp_T(data, state.T);
  annotation(Inline=true,smoothOrder=2);
end specificHeatCapacityCp;
```

specificHeatCapacityCv(state) 定積比熱

```
function extends specificHeatCapacityCv
  "Compute specific heat capacity at constant volume from temperature and gas data"
algorithm
  cv := Modelica.Media.IdealGases.Common.Functions.cp_T(
    data, state.T) - data.R;
  annotation(Inline=true,smoothOrder=2);
end specificHeatCapacityCv;
```

マイヤーの関係式

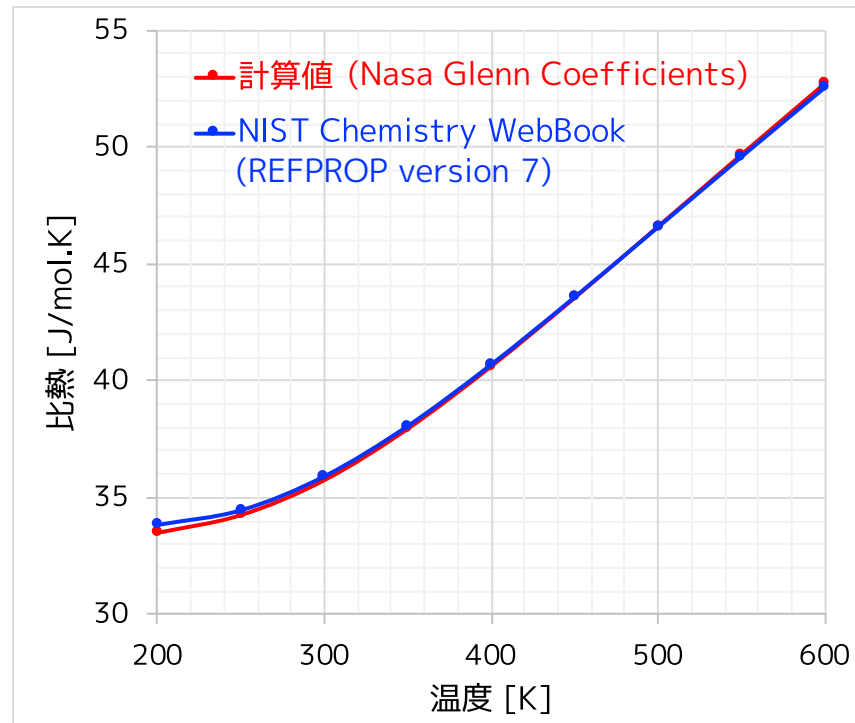
$$c_v = c_p - R$$

Modelica.Media.IdealGases.Common.Functions. specificHeatCapacityCp(state)

$$c_p = \frac{R}{T^2} \left(a_1 + T \left(a_2 + T \left(a_3 + T \left(a_4 + T (a_5 + T (a_6 + a_7 T)) \right) \right) \right) \right)$$
$$= R \sum_{i=1}^7 a_i T^{i-3} \quad [\text{J/kg.K}]$$

$$a_i = \begin{cases} a_{low}[i] & T < T_{limit} \\ a_{high}[i] & T \geq T_{limit} \end{cases}$$

CH₄ の比熱



isentropicExponent(state) 等エントロピー指数 (理想気体では比熱比)

```
function extends isentropicExponent "Return isentropic exponent"  
algorithm  
  gamma := specificHeatCapacityCp(state)/specificHeatCapacityCv(state);  
  annotation(Inline=true,smoothOrder=2);  
end isentropicExponent;
```

$$\gamma = \frac{c_p}{c_v}$$

velocityOfSound(state) 音速

```
function extends velocityOfSound "Return velocity of sound"  
  extends Modelica.Icons.Function;  
algorithm  
  a := sqrt(max(0,data.R*state.T*Modelica.Media.IdealGases.Common.Functions.cp_T(  
    data, state.T)/specificHeatCapacityCv(state)));  
  annotation(Inline=true,smoothOrder=2);  
end velocityOfSound;
```

$$a = \sqrt{\gamma RT} = \sqrt{\frac{c_p}{c_v} RT}$$

isobaricExpantionCoefficient 定圧膨張係数

```
function extends isobaricExpansionCoefficient
  "Returns overall the isobaric expansion coefficient beta"
algorithm
  beta := 1/state.T;
  annotation(Inline=true,smoothOrder=2);
end isobaricExpansionCoefficient;
```

$$v = \frac{RT}{p}$$
$$\beta \equiv \frac{1}{v} \left(\frac{\partial v}{\partial T} \right)_p = \frac{p}{RT} \frac{R}{p} = \frac{1}{T}$$

isothermalCompressibility(state) 等温圧縮率

```
function extends isothermalCompressibility
  "Returns overall the isothermal compressibility factor"
algorithm
  kappa := 1.0/state.p;
  annotation(Inline=true,smoothOrder=2);
end isothermalCompressibility;
```

$$v = \frac{RT}{p}$$
$$\kappa \equiv -\frac{1}{v} \left(\frac{\partial v}{\partial p} \right)_T = -\frac{p}{RT} \left(-\frac{RT}{p^2} \right) = \frac{1}{p}$$

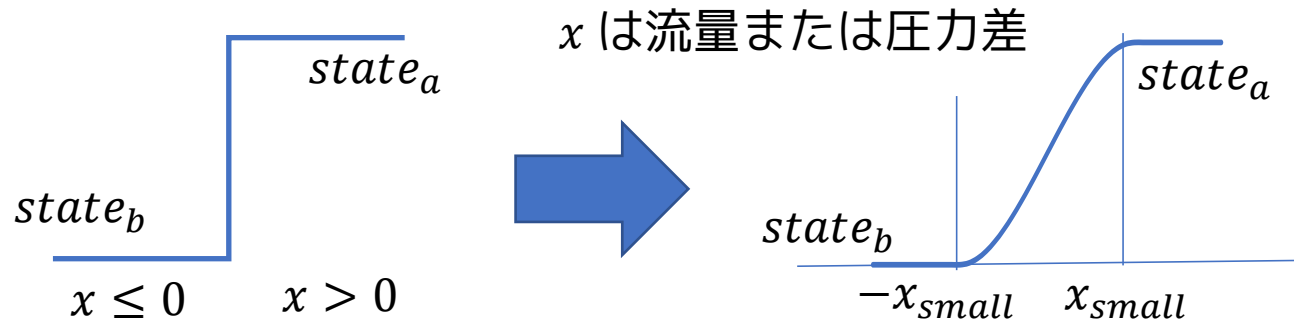
molarMass モル質量

```
function extends molarMass "Return the molar mass of the medium"
algorithm
  MM := data.MM;
  annotation(Inline=true,smoothOrder=2);
end molarMass;
```

VII setSmoothState

$x=0$ で不連続な熱力学的状態 $state_a$, $state_b$ を滑らかに近似する。

$$y = \begin{cases} state_a, & x > 0 \\ state_b, & x \leq 0 \end{cases} \quad y = setSmoothState(x, state_a, state_b, x_{small})$$



partialMedium SimpleGasNasa でアルゴリズムを実装する。

- `setSmoothState(x, state_a, state_b, x_small(min=0))`

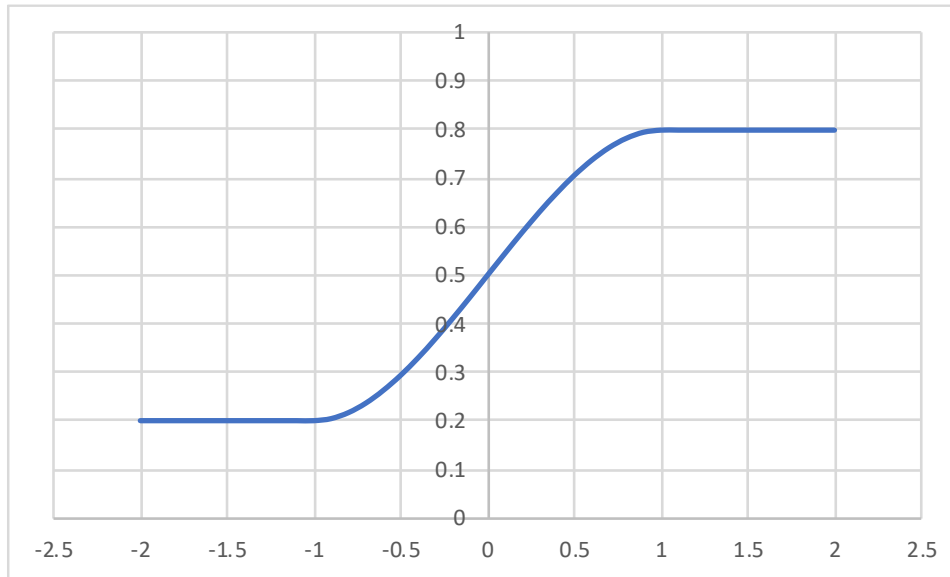
SingleGasNasa.setSmoothState

```
function extends setSmoothState
"Return thermodynamic state so that it smoothly approximates: if x > 0 then state_a else state_b"
algorithm
  state := ThermodynamicState(p=Media.Common.smoothStep(x, state_a.p, state_b.p, x_small),
                              T=Media.Common.smoothStep(x, state_a.T, state_b.T, x_small));
  annotation(Inline=true,smoothOrder=2);
end setSmoothState;
```

Modelica.Media.Common.smoothStep

ステップ関数を滑らか（連続かつ微分可能）に近似する。

$$y = \begin{cases} y_1, & x > x_{small} \\ y_2, & x < -x_{small} \\ \frac{x}{x_{small}} \left(\left(\frac{x}{x_{small}} \right)^2 - 3 \right) \frac{y_2 - y_1}{4} + \frac{y_1 + y_2}{2}, & x_{small} \geq x \geq -x_{small} \end{cases}$$



$x_{small} = 0$ のときは

$$y = \frac{y_1 + y_2}{2}$$

$$y_1 = 0.8, y_2 = 0.2, x_{small} = 1$$

7. Modelica.Media.Air.DryAirNasa 乾燥空気

```
within Modelica.Media.Air;  
package DryAirNasa "Air: Detailed dry air model as ideal gas (200..6000 K)"  
  extends Modelica.Icons.MaterialProperty;  
  extends IdealGases.Common.SingleGasNasa(  
    mediumName="Air",    data=IdealGases.Common.SingleGasesData.Air,  
    fluidConstants={IdealGases.Common.FluidData.N2});  
  
  redeclare function dynamicViscosity      乾燥空気の粘性率  
    ...  
  end dynamicViscosity;  
  
  redeclare function thermalConductivity    乾燥空気の熱伝導率  
    ...  
  end thermalConductivity;  
  
  annotation (Documentation(info="<html> ...));  
end DryAirNasa;
```

粘性率と熱伝導率だけ SingleGasNasa で定義されたものから置き換えられている。

DryAirNasa.dynamicViscosity のアルゴリズム

5 次多項式の係数
 $p_i, i = 1, \dots, 6$

algorithm

```
eta := 1e-6*Polynomials_Temp.evaluateWithRange(  
  {9.7391102886305869E-15, -3.1353724870333906E-11, 4.3004876595642225E-08,  
   -3.8228016291758240E-05, 5.0427874367180762E-02, 1.7239260139242528E+01},  
  Cv.to_degC(123.15),  $T_{min}$   
  Cv.to_degC(1273.15),  $T_{max}$   
  Cv.to_degC(state.T));  $T$   
  annotation (smoothOrder=2, ...);  
end dynamicViscosity;
```

$$\eta = \sum_{j=1}^6 p_j T^{n-j}, T_{min} \leq T \leq T_{max}$$

- 温度 $T_{min} = 123.15$ [K] = -150[°C]と $T_{max} = 1273.15$ [K] = 1000 [°C]の間の乾燥空気の粘性率を温度 T [°C] の 5 次多項式で近似する。
- 範囲外は線形外挿する。

Source: VDI Waermeatlas, 8th edition

DryAirNasa.thermalConductivity のアルゴリズム

algorithm

```
lambda := 1e-3*Polynomials_Temp.evaluateWithRange(  
    {6.5691470817717812E-15, -3.4025961923050509E-11, 5.3279284846303157E-08,  
    -4.5340839289219472E-05, 7.6129675309037664E-02, 2.4169481088097051E+01},  
    Cv.to_degC(123.15), Tmin  
    Cv.to_degC(1273.15), Tmax  
    Cv.to_degC(state.T));  
annotation (smoothOrder=2, ...);  
end DryAirNasa;
```

5 次多項式の係数
 $p_i, i = 1, \dots, 6$

$$\lambda = \sum_{j=1}^6 p_j T^{n-j}, T_{min} \leq T \leq T_{max}$$

- 温度 $T_{min} = 123.15 \text{ [K]} = -150[^\circ\text{C}]$ と $T_{max} = 1273.15 \text{ [K]} = 1000 [^\circ\text{C}]$ の間の乾燥空気の熱伝導率を温度 $T [^\circ\text{C}]$ の 5 次多項式で近似する。
- 範囲外は線形外挿する。

Source: VDI Waermeatlas, 8th edition

Modelica.Media.Incompressible.TableBased. Polynomials_Temp.evaluateWithRange

線形外挿付きn次多項式

$$\begin{aligned} \text{evaluate}(\mathbf{p}, u) &= \sum_{j=1}^n p_j u^{n-j} \\ \text{evaluate_der}(\mathbf{p}, u, \Delta u) &= \left(\frac{d \text{evaluate}(\mathbf{p}, u)}{du} \right)_{u=u} \Delta u \end{aligned}$$

evaluateWithRange

$$y = \begin{cases} u_{\min} - \text{evaluate_der}(\mathbf{p}, u_{\min}, u_{\min} - u), & u < u_{\min} \\ u_{\max} + \text{evaluate_der}(\mathbf{p}, u_{\max}, u - u_{\max}), & u > u_{\max} \\ \text{evaluate}(\mathbf{p}, u), & u_{\min} \leq u \leq u_{\max} \end{cases} \quad \left. \begin{array}{l} \text{線形外挿} \\ \text{部分} \end{array} \right\}$$

Polynomials_Temp

Copyright © 2004-2016, Modelica Association and DLR.

*This package is **free** software. It can be redistributed and/or modified under the terms of the **Modelica license**, see the license conditions and the accompanying **disclaimer** in the documentation of package Modelica in file "Modelica/package.mo".*

Polynomials_Temp.evaluate のアルゴリズム

$$y_j \equiv \begin{cases} p_1, & j = 1 \\ p_j + y_{j-1} u, & j = 2, \dots, n \end{cases} \quad \begin{array}{l} \text{ソースコードは左の漸化式のループ} \\ \text{になっている。} \end{array}$$

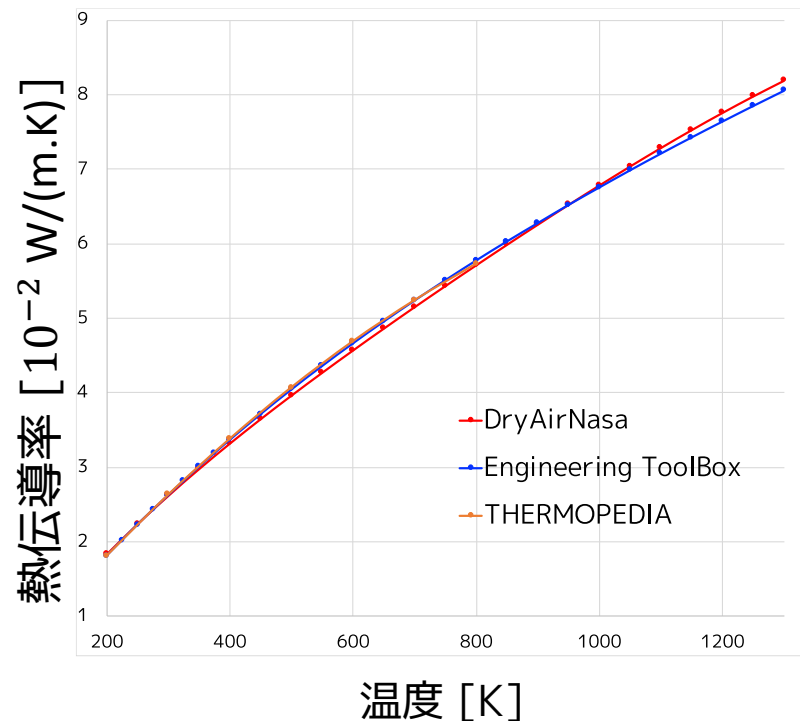
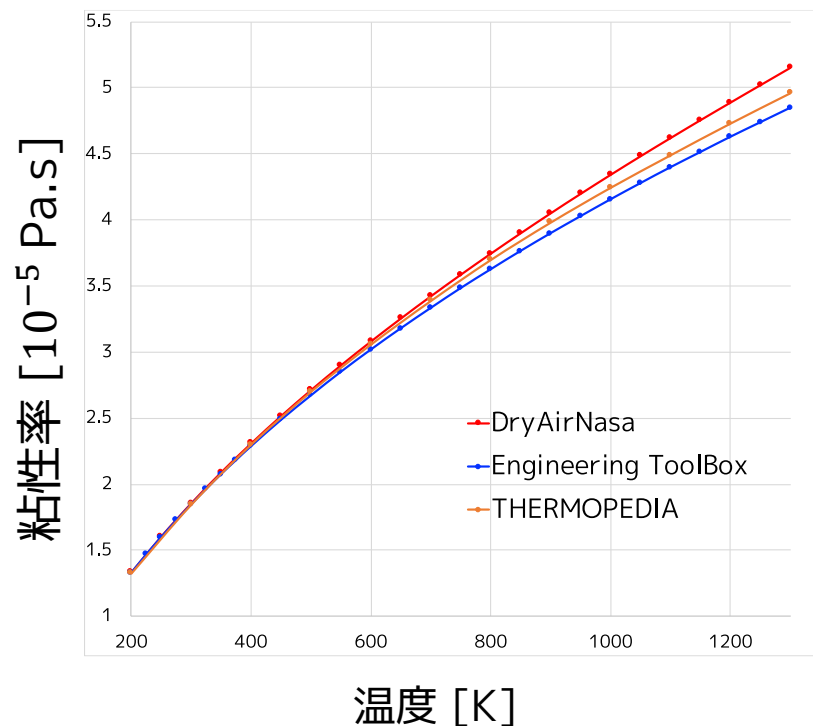
$$\begin{aligned} \text{evaluate}(\mathbf{p}, u) &= y_n && \text{代入してまとめると次のようになる。} \\ &= p_n + (p_{n-1} + (p_{n-2} + (\dots (p_3 + (p_2 + p_1 u)u) \dots)u)u)u \\ &= p_n + p_{n-1}u + p_{n-2}u^2 + \dots + p_3u^{n-3} + p_2u^{n-2} + p_1u^{n-1} \\ &= \sum_{j=1}^n p_j u^{n-j} \end{aligned}$$

Polynomials_Temp.evaluate_der のアルゴリズム

$$dy_j \equiv \begin{cases} (n-1)p_1, & j = 1 \\ (n-j)p_j + dy_{j-1}u, & j = 2, \dots, n-1 \end{cases} \quad \begin{array}{l} \text{ソースコードは左の漸化式のループ} \\ \text{になっている。} \end{array}$$

$$\begin{aligned} \text{evaluate_der}(\mathbf{p}, u, \Delta u) &&& \text{代入してまとめると次のようになる。} \\ &= dy_{n-1} \cdot \Delta u \\ &= \{p_{n-1} + (2p_{n-2} + (\dots ((n-3)p_3 + ((n-2)p_2 + (n-1)p_1 u)u) \dots)u)u\} \Delta u \\ &= \{p_{n-1} + 2p_{n-2}u + \dots + (n-3)p_3u^{n-4} + (n-2)p_2u^{n-3} + (n-1)p_1u^{n-2}\} \Delta u \\ &= \left(\frac{d \text{evaluate}(\mathbf{p}, u)}{du} \right)_u \Delta u \end{aligned}$$

空気の粘性率・熱伝導率の比較



The Engineering ToolBox

https://www.engineeringtoolbox.com/dry-air-properties-d_973.html

THERMOPEDIA

<http://thermopedia.com/content/553/>

[SingleGasNasa](http://singlegas.nasa.gov)

まとめ

SingleGasNasa は、

- NASA Glenn coefficients による熱物性モデル
- Chung, et al. (1984, 1988) による粘性率モデル
- (Modified) Eucken method による熱伝導率モデル
- 理想気体状態方程式などの熱力学的関係式

をベースとして構成され、

- preferredMediumStates パラメータによる状態変数選択
- エントロピやエンタルピから温度を求めるための非線形方程式ソルバー

などの機能をもつ。

Licensed by Amane Tanaka under the Modelica License 2

Copyright(c) 2018, Amane Tanaka

- The purpose of this document is introducing the SingleGasNasa package which is included in the Modelica Standard Library (MSL). This document uses libraries, software, figures, and documents included in MSL and those modifications. Licenses and copyrights of those are written in next page.
- This document is free and the use is completely at your own risk; it can be redistributed and/or modified under the terms of the Modelica license 2, see the license conditions (including the disclaimer of warranty) at <http://www.modelica.org/licenses/ModelicaLicense2>

Modelica Standard Library License

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

BSD 3-Clause License

Copyright (c) 1998-2018, ABB, Austrian Institute of Technology, T. Bödrich, DLR, Dassault Systèmes AB, ESI ITI, Fraunhofer, A. Haumer, C. Kral, Modelon, TU Hamburg-Harburg, Politecnico di Milano, and XRG Simulation
All rights reserved.

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

- * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- * Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

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