

# Modelica Standard Library (MSL) の MixtureGasNASA 混合物理想気体のモデルについて

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

2018/10/27 第5回 Modelica ライブラリ勉強会

finback

2018/11/04 加筆修正

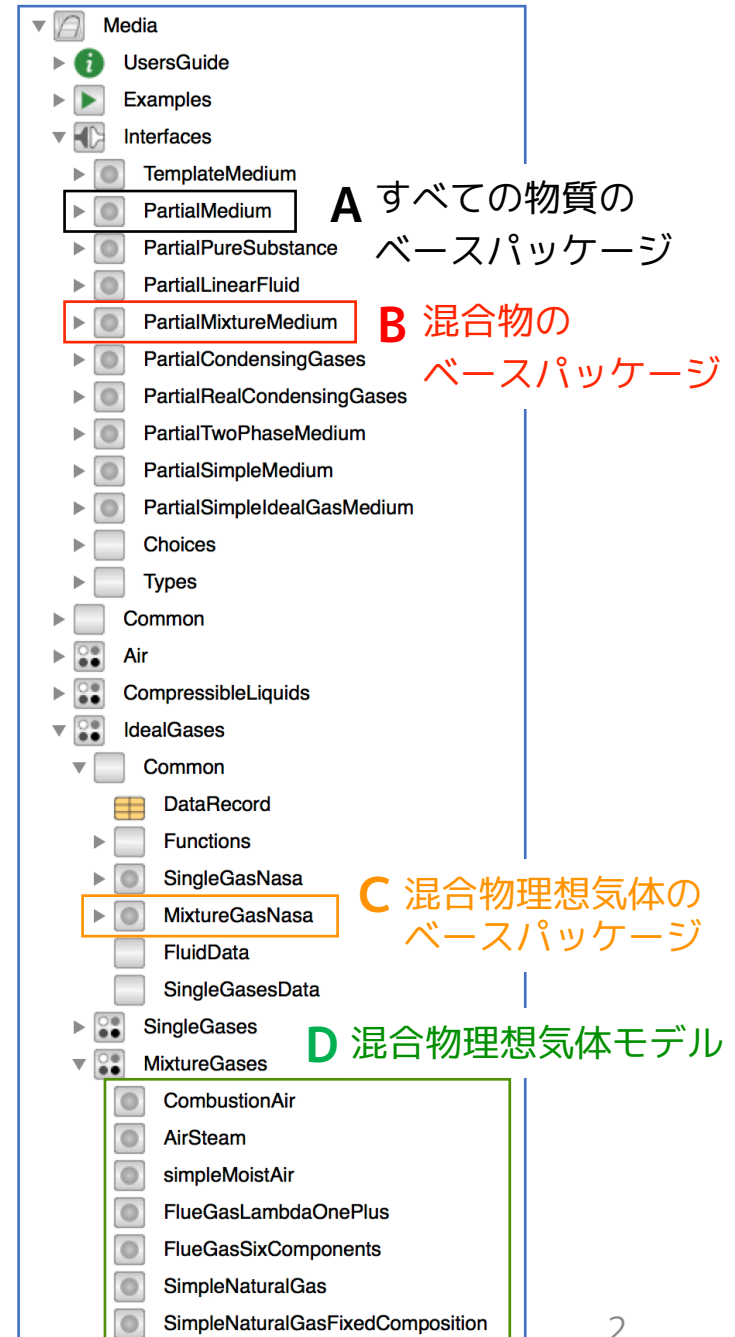
2018/11/19 加筆修正

2018/11/23 修正

# MixtureGasNasa (混合物理想気体)

package の基本構成、継承関係とクラス

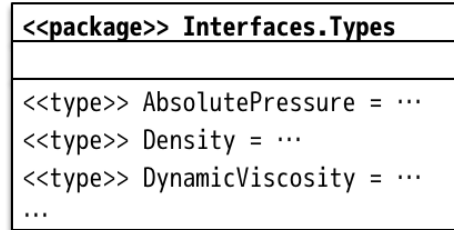
1. [Types](#)
2. [定数](#)
  - I. [定数の継承と導入](#)
3. [ThermodynamicState](#)
4. [BaseProperties](#)
  - I. [StateSelection \(状態変数選択\)](#)
  - II. [BaseProperties の方程式](#)
  - III. [比エンタルピの計算方法](#)
5. [setState XXX](#)
  - I. [setState XXX 関数](#)
  - II. [温度を求める関数](#)
6. [物性値関数](#)
  - I. [熱力学的状態変数を返す関数](#)
  - II. [粘性率\(1\)](#)
  - III. [粘性率\(2\)](#)
  - IV. [熱伝導率](#)
  - V. [モル分率、質量分率、モル質量、気体定数](#)
  - VI. [密度の偏微分](#)
  - VII. [等エントロピ過程のエンタルピ](#)
  - VIII. [その他の物性値](#)
  - IX. [setSmoothState](#)
7. [ThermoPower.Gas.CombustionChamber](#)
  - I. [MixtureGasNasa を使ったテストモデル](#)
  - II. [気体](#)
  - III. [CombustionChamber](#)
  - IV. [計算結果](#)
8. [まとめ](#)



# Modelica.Media の基本構成

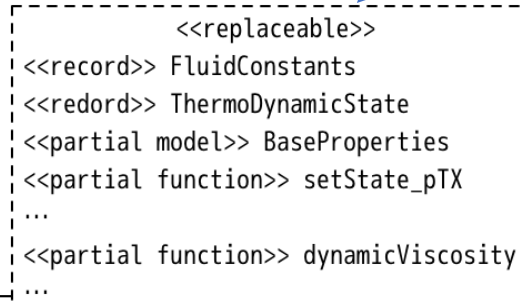
全ての物質の  
ベースパッケージ

継承



1. Types

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



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

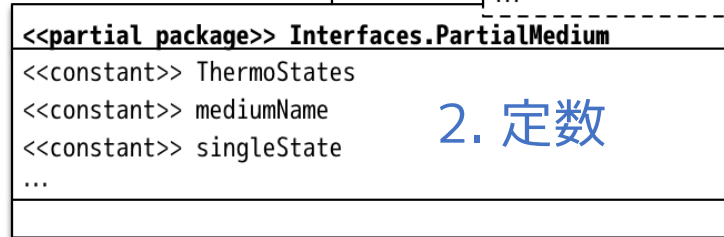
基本物性モデル

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

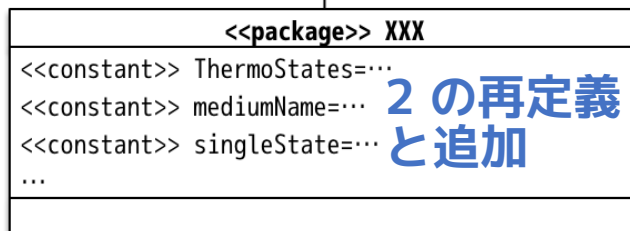
2. 定数



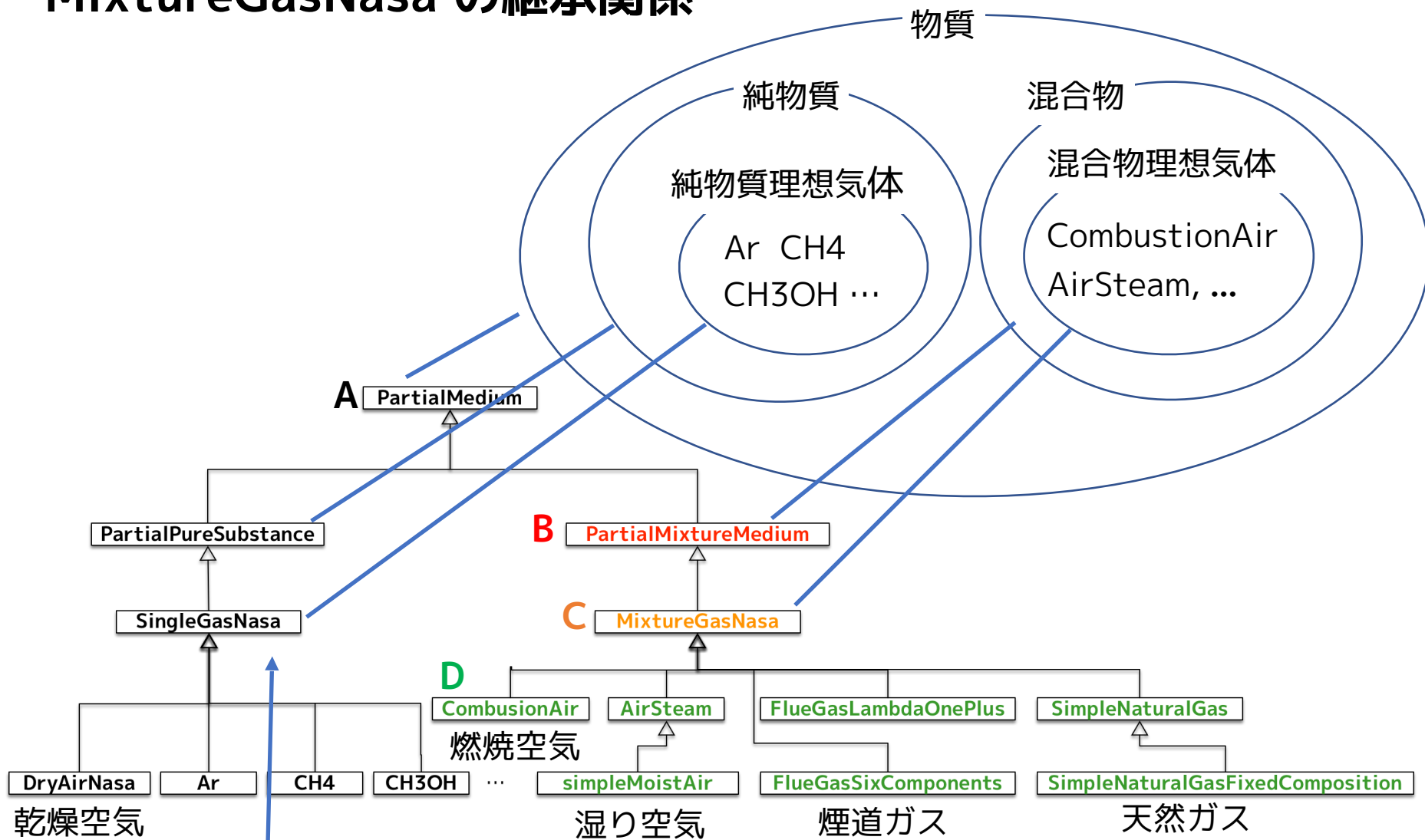
(多段階) 継承

```
<<redeclare>>
ThermoDynamicState→... 1 のパラメータ  
BaseProperties→... 変更  
setState_pTX→...  
...  
dynamicViscosity→... 3~6 の宣言  
...
```

個々の物質の  
パッケージ



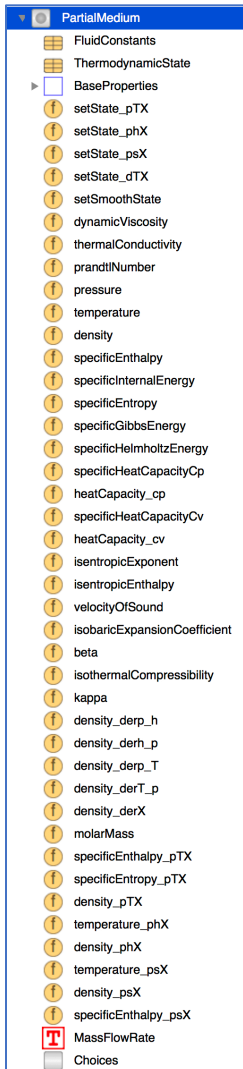
# MixtureGasNasa の継承関係



SingleGasNasa について

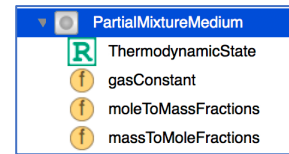
[https://www.amane.to/wp-content/uploads/2018/10/Introduction\\_SingleGasNasa\\_20181007.pdf](https://www.amane.to/wp-content/uploads/2018/10/Introduction_SingleGasNasa_20181007.pdf)

## A. PartialMedium のクラス



- [prandtlNumber](#)
- [heatCapacity\\_cp](#)
- [heatCapacity\\_cv](#)
- [beta](#)
- [kappa](#)
- [density\\_derp\\_h](#)
- [density\\_derh\\_p](#) } 未宣言 (使用しない)
- [specificEnthalpy\\_pTX](#)
- [specificEntropy\\_pTX](#)
- [density\\_pTX](#)
- [temperature\\_phX](#)
- [density\\_phX](#)
- [temperature\\_psX](#)
- [density\\_psX](#)
- [specificEnthalpy\\_psX](#)
- MassFlowRate
- Choices

## B. PartialMixtureMediumのクラス



- [FluidConstants](#)
- [ThermodynamicState](#)
- [gasConstant](#)
- [moleToMassFractions](#)
- [massToMoleFractions](#)

← 上記の他のクラスは C. MixtureGasNasa で再宣言 (redeclare) する。

## C. MixtureGasNasa のクラス

MixtureGasNasa	
<b>R</b>	ThermodynamicState
<b>M</b>	BaseProperties
<b>f</b>	setState_pTX
<b>f</b>	setState_phX
<b>f</b>	setState_psX
<b>f</b>	setState_dTX
<b>F</b>	setSmoothState
<b>F</b>	pressure
<b>F</b>	temperature
<b>F</b>	density
<b>f</b>	specificEnthalpy
<b>f</b>	specificInternalEnergy
<b>F</b>	specificEntropy
<b>f</b>	specificGibbsEnergy
<b>f</b>	specificHelmholtzEnergy
<b>f</b>	h_TX
<b>f</b>	h_TX_der
<b>F</b>	gasConstant
<b>F</b>	specificHeatCapacityCp
<b>F</b>	specificHeatCapacityCv
<b>f</b>	MixEntropy
<b>f</b>	s_TX
<b>F</b>	isentropicExponent
<b>f</b>	velocityOfSound
<b>f</b>	isentropicEnthalpyApproximation
<b>F</b>	isentropicEnthalpy
<b>f</b>	gasMixtureViscosity
<b>F</b>	dynamicViscosity
<b>f</b>	mixtureViscosityChung
<b>f</b>	lowPressureThermalConductivity
<b>F</b>	thermalConductivity
<b>F</b>	isobaricExpansionCoefficient
<b>F</b>	isothermalCompressibility
<b>F</b>	density_derp_T
<b>F</b>	density_derT_p
<b>f</b>	density_derX
<b>F</b>	molarMass
<b>f</b>	T_hX
<b>f</b>	T_psX

- [ThermodynamicState](#)
- [BaseProperties](#)
- [setState\\_pTX](#)
- [setState\\_phX](#)
- [setState\\_psX](#)
- [setState\\_dTX](#)
- [setSmoothState](#)
- [pressure](#)
- [temperature](#)
- [density](#)
- [specificEnthalpy](#)
- [specificInternalEnergy](#)
- [specificEntropy](#)
- [specificGibbsEnergy](#)
- [specificHelmholtzEnergy](#)
- [h\\_TX](#)
- [h\\_TX\\_der](#)
- [gasConstant](#)
- [specificHeatCapacityCp](#)
- [specificHeatCapacityCv](#)

- MixEntropy(未使用)
- [s\\_TX](#)
- [isentropicExponent](#)
- [velocityOfSound](#)
- [isentropicEnthalpyApproximation](#)
- [isentropicEnthalpy](#)
- [gasMixtureViscosity](#)
- [dynamicViscosity](#)
- [mixtureViscosityChung](#)
- [lowPressureThermalConductivity](#)
- [thermalConductivity](#)
- [isobaricExpansionCoefficient](#)
- [isothermalCompressibility](#)
- [density\\_derp\\_T](#)
- [density\\_derT\\_p](#)
- [density\\_derX](#)
- [molarMass](#)
- [T\\_hX](#)
- [T\\_psX](#)

# 1. Type と 2. 定数

## I. 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. PartialMixtureMediumで導入される定数

```
partial package PartialMixtureMedium
  "Base class for pure substances of several chemical substances"
  extends PartialMedium(redeclare replaceable record FluidConstants =
    Modelica.Media.Interfaces.Types.IdealGas.FluidConstants);
  ...
  constant FluidConstants[nS] fluidConstants "Constant data for the fluid";
  ...
```



SingleGasNasa と同様の FluidConstants レコードを使用する。

[https://www.amane.to/wp-content/uploads/2018/10/Introduction\\_SingleGasNasa\\_20181007.pdf](https://www.amane.to/wp-content/uploads/2018/10/Introduction_SingleGasNasa_20181007.pdf) 1.11 参照



## C. MixtureGasNasa の継承部分 (extends)で設定される定数とタイプの修正

```
partial package MixtureGasNasa
  "Medium model of a mixture of ideal gases based on NASA source"
  import Modelica.Math;
  import Modelica.Media.Interfaces.Choices.ReferenceEnthalpy;
  extends Modelica.Media.Interfaces.PartialMixtureMedium(
    ThermoStates=Modelica.Media.Interfaces.Choices.IndependentVariables.pTX,
    substanceNames=data[:].name,
    reducedX = false,
    singleState=false,
    reference_X=fill(1/nX,nX),
    SpecificEnthalpy(
      start=if referenceChoice==ReferenceEnthalpy.ZeroAt0K then 3e5 else
        if referenceChoice==ReferenceEnthalpy.UserDefined then h_offset else 0,
      nominal=1.0e5),
    Density(start=10, nominal=10),
    AbsolutePressure(start=10e5, nominal=10e5),
    Temperature(min=200, max=6000, start=500, nominal=500));
```

定数 referenceChoice によって比エンタルピのオフセットが変わるため対応して初期値も修正する。

NASA Glenn coefficients による  
比熱モデルの温度の適用範囲を設定している。

## C. MixtureGasNasa で導入される定数

```
constant Modelica.Media.IdealGases.Common.DataRecord[:] data (1)
  "Data records of ideal gas substances";
  // ={Common.SingleGasesData.N2,Common.SingleGasesData.O2}
constant Boolean excludeEnthalpyOfFormation=true
  "If true, enthalpy of formation Hf is not included in specific enthalpy h";
constant ReferenceEnthalpy referenceChoice=ReferenceEnthalpy.ZeroAt0K
  "Choice of reference enthalpy";
constant SpecificEnthalpy h_offset=0.0
  "User defined offset for reference enthalpy, if referenceChoice = UserDefined";
...
constant MolarMass[nX] MMX=data[:].MM "Molar masses of components"; (3)
constant Integer methodForThermalConductivity(min=1,max=2)=1; (4)
```

(1) data:成分物質のNasa Glenn coefficients などの熱物性データ

(2) excludeEnthalpyOfFormation, referenceChoice, h\_offset:比エンタルピのオフセット設定用定数

(3) MMX[:]: 成分物質のモル質量 [kg/mol]

(4) methodForThermalConductivity: 成分物質の熱伝導率の計算方法を決める定数

定数の意味は

[https://www.amane.to/wp-content/uploads/2018/10/Introduction\\_SingleGasNasa\\_20181007.pdf](https://www.amane.to/wp-content/uploads/2018/10/Introduction_SingleGasNasa_20181007.pdf)

2.II, 4.III, 6.III 参照

①～⑤を設定すれば、混合物理理想気体モデルを作成することができる。

## D. CombustionAir (燃焼空気)

```
package CombustionAir "Air as mixture of N2 and O2"
  extends Common.MixtureGasNasa(
    mediumName="CombustionAirN2O2",
    data={Common.SingleGasesData.N2, Common.SingleGasesData.O2},
    fluidConstants={Common.FluidData.N2, Common.FluidData.O2},
    substanceNames = {"Nitrogen", "Oxygen"},
    reference_X={0.768,0.232});
  annotation (Documentation(info="<html></html>"));
end CombustionAir;
```

① 物質名  
② DataRecord[]  
③ FluidConstant[]  
④ 成分名[]  
⑤ 質量分率[]

## D. FlueGasLambdaOnePlus (煙道ガス)

```
package FlueGasLambdaOnePlus
  "Simple flue gas for over0stoichiometric O2-fuel ratios"
  extends Common.MixtureGasNasa(
    mediumName="FlueGasLambda1plus",
    data={Common.SingleGasesData.N2, Common.SingleGasesData.O2,
          Common.SingleGasesData.H2O, Common.SingleGasesData.CO2},
    fluidConstants={Common.FluidData.N2, Common.FluidData.O2,
                    Common.FluidData.H2O, Common.FluidData.CO2},
    substanceNames = { "Nitrogen", "Oxygen", "Water", "Carbondioxide"},
    reference_X={0.768,0.232,0.0,0.0});
  annotation (Documentation(info="<html></html>"));
end FlueGasLambdaOnePlus;
```

① 物質名  
② DataRecord[]  
③ FluidConstant[]  
④ 成分名[]  
⑤ 質量分率[]

定数（まとめ）    A. PartialMedium, B. PartialMixtureMedium, C. MixtureGasNasa, D. CombustionAir

定数名	型	設定値	
mediumName	String	①	物質名
ThermoStates	IndependentVariables	pTX（圧力、温度、質量分率）	独立な熱力学的変数
singleState	Boolean	false	trueなら物性値が圧力に依存しない
substanceNames[:]	String	④	成分名の配列
reducedX	Boolean	false	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	nS	独立な質量分率の要素数
MMX	MolarMass[nX]	data[:].MM	成分物質のモル質量
extraPropertiesNames[:]	String	fill("",0)	付加的物質名(微小物質)
nC	Integer	size(extraPropertiesNames, 1)	付加的物質(微小物質)の成分数
C_nominal[nC]	Real	1.0e-6*ones(nC)	付加的物質(微小物質)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	⑤	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
fluidConstants	FluidConstants[nS]	③	成分物質の固有データ
data	DataRecord[:]	②	熱物性計算用のデータ
excludeEnthalpyOfFormation	Boolean	true	生成エンタルピを除外する
referenceChoice	ReferenceEnthalpy	ZeroAt0K	比エンタルピのオプションパラメータ
h_offset	SpecificEnthalpy	0.0	ユーザー定義のオフセット値
methodForThermalConductivity	Integer	1	熱伝導率のオプション min=1, max=2

### 3. ThermodynamicState

独立な熱力学的状態変数

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 $X[nX]$

#### B. PartialMixtureMedium.ThermodynamicState

```
redeclare replaceable record extends ThermodynamicState
  "Thermodynamic state variables"
  AbsolutePressure p "Absolute pressure of medium";
  Temperature T "Temperature of medium";
  MassFraction[nX] X(start=reference_X)
    "Mass fractions (= (component mass)/total mass  m_i/m)";
end ThermodynamicState;
```

#### C. MixtureGasNasa.ThermodynamicState

```
record extends ThermodynamicState "Thermodynamic state variables"
end ThermodynamicState;
```

独立な熱力学的状態変数として **p**, **T**, **X** を選択する。

## 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>> preferredMediumState: Boolean	
<<parameter>> standardOrderComponents: Boolean	
connector InputAbsolutePressure	
connector InputSpecificEnthalpy	
connector InputMassFraction	
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/m<sup>3</sup>] ← 数式では  $\rho$  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) preferredMediumStates の設定

## Modelica.Media.UserGuide.MediumDefinition.StaticStateSelection より

多成分物質媒体(multiple substance media) を valance volume (保存則の成り立つ容器) で使用する場合、

- 独立な質量分率  $X_i$  が  $\text{stateSelect} = \text{StateSelect.prefer}$
- 熱力学的状態変数  $p, T, d, u, h$  のうち 2 つが  $\text{stateSelect} = \text{StateSelect.prefer}$
- 残りの 3 つの熱力学的 が、この 2 つの変数の関数の形になる

のときのみ、static state selection (静的状態変数選択)が可能となる。

以下のように設定した場合を考える。

$p, T, X_i$  の  $\text{stateSelect} = \text{StateSelect.prefer}$   
 $d = f_d(p, T, X)$   
 $u = f_u(p, T, X)$   
 $h = f_h(p, T, X)$

保存則方程式は、

$$M = V * \text{medium}.d$$

$$MX_i = M * \text{medium}.X_i$$

の微分で表され、状態変数  $p, T, X$  とその微分から直接計算できる。

state selection の概要は、

[https://www.amane.to/wp-content/uploads/2018/10/Introduction\\_SingleGasNasa\\_20181007.pdf](https://www.amane.to/wp-content/uploads/2018/10/Introduction_SingleGasNasa_20181007.pdf) 4.1 参照

## MixtureGasNasa.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),  
  Xi(each stateSelect=if preferredMediumStates then StateSelect.prefer else StateSelect.default),  
  final standardOrderComponents=true)  
  "Base properties (p, d, T, h, u, R, MM, X, and Xi of NASA mixture gas"
```

**preferredMediumStates = true**  
を設定することによって静的状態変数選択が可能となる。



## II. BaseProperties の方程式

### A. PartialMedium.BaseProperties の方程式

- **reducedX, fixedX の設定方法**
- **X と Xi に関する方程式**

(1) デフォルト設定

```
standardOrderComponents = true  
reducedX = false, fixedX = false  
Xi = X[1:nXi] = X[1:nS]
```

(2) 成分を固定したい場合

```
fixedX = true  
X = reference_X
```

(3) 質量分率の和を自動的に1にしたい場合

```
reducedX=true, fixedX = false
```

```
nXi = nS - 1
```

```
Xi = X[1:nXi]
```

```
X[nX] = 1 - sum(Xi)
```

独立な質量分率の  
成分数

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;
```

```
for i in 1:nX loop
```

```
  assert(X[i] >= -1.e-5 and X[i] <= 1 + 1.e-5, "Mass fraction X[" +
```

```
  String(i) + "] = " + String(X[i]) + "of substance " +
```

```
  substanceNames[i] + "\nof medium " + mediumName +
```

```
  " is not in the range 0..1");
```

```
end for;
```

```
end if;
```

```
assert(p >= 0.0, "Pressure (= " + String(p) + " Pa) of medium \" +
```

```
mediumName + "\" is negative\n(Temperature = " + String(T) + " K)");
```

```
...
```

```
end BaseProperties;
```

$0 - 10^{-5} < Xi[i] < 1 + 10^{-5}$  でなければエラー

絶対圧力が負ならエラー

## C. MixutureGasaNasa.BaseProperties の方程式

### T, p, X, h, u, d, MM, R に関する方程式

```
equation
  assert(T >= 200 and T <= 6000, "Temperature T (= " + String(T) + " K = 200 K) is
not in the allowed range 200 K <= T <= 6000 K required from medium model \" +
mediumName + "\".");
  MM = molarMass(state);
  h = h_TX(T, X);
  R = data.R*X;
  u = h - R*T;
  d = p/(R*T);
  // connect state with BaseProperties
  state.T = T;
  state.p = p;
  state.X = if fixedX then reference_X else X;
end BaseProperties;
```

混合物質の気体定数 [J/kg.K]

$$R = \mathbf{R} \cdot \mathbf{X} = \sum_{i=1}^{nX} data[i].R \cdot X[i]$$

比エンタルピの定義

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

理想気体の状態方程式

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

### III. 比エンタルピの計算方法

#### 混合物の比エンタルピ

$$h = \mathbf{X} \cdot \mathbf{h} = \sum_{i=1}^{nX} X_i h_i$$

$X_i$ : i 番目の成分の質量分率 [kg/kg]  
 $h_i$ : i 番目の成分の比エンタルピ [J/kg]

$$h_i = h\_T(\text{data}[i], T, \text{exclEnthForm}, \text{refChoice}, h\_off)$$

成分物質の比エンタルピ [https://www.amane.to/wp-content/uploads/2018/10/Introduction\\_SingleGasNasa\\_20181007.pdf](https://www.amane.to/wp-content/uploads/2018/10/Introduction_SingleGasNasa_20181007.pdf) 3.III

#### MixtureGasNasa.h\_TX(T, X, exclEnthForm, refChoice, h\_off) のアルゴリズム

```
algorithm
  h := (if fixedX then reference_X else X)*
    {Modelica.Media.IdealGases.Common.Functions.h_T(
      data[i], T, exclEnthForm, refChoice, h_off) for i in 1:nX};
  annotation(Inline=false, smoothOrder=2);
end h_TX;
```

#### オプションのデフォルト値 (MixtureGasNasaの定数を参照する)

- fixedX = false
  - exclEnthForm = excludeEnthalpyOfFormation = true
  - refChoice = referenceChoice = ReferenceEnthalpy.ZeroAt0K
  - h\_off = h\_offset = 0
- 0 [K] で h = 0 [J/kg]


## 5. setState\_XXX

### I. setState\_XXX 関数

#### setState\_pTX(p, T, X) のアルゴリズム

```
algorithm
state := if size(X,1) == 0 then
    ThermodynamicState(p=p,T=T,X=reference_X)
else if size(X,1) == nX then
    ThermodynamicState(p=p,T=T, X=X)
else
    ThermodynamicState(p=p,T=T, X=cat(1,X,{1-sum(X)}));
```

reducedX = true、  
Xi[nXi], nXi = nS-1  
を引数にする場合に  
X[nX]を計算している。



#### setState\_dTX(d, T, X) のアルゴリズム

```
algorithm
state := if size(X,1) == 0 then
    ThermodynamicState(p=d*(data.R*reference_X)*T,T=T,X=reference_X)
else if size(X,1) == nX then
    ThermodynamicState(p=d*(data.R*X)*T,T=T,X=X)
else
    ThermodynamicState(p=d*(data.R*cat(1,X,{1-sum(X)}))*T,T=T,
        X=cat(1,X,{1-sum(X)}));
```

理想気体の状態方程式で圧力を求める

$$p = \rho RT$$

$$R = \mathbf{R} \cdot \mathbf{X} \quad \mathbf{R} = \text{data.R}$$

混合物の気体定数

## setState\_phX(p, h, X) のアルゴリズム

```
algorithm
state := if size(X,1) == 0 then
    ThermodynamicState(p=p,T=T_hX(h,reference_X),X=reference_X)
else if size(X,1) == nX then
    ThermodynamicState(p=p,T=T_hX(h,X),X=X)
else
    ThermodynamicState(p=p,T=T_hX(h,X), X=cat(1,X,{1-sum(X)}));
```

## setState\_psX(p, s, X) のアルゴリズム

```
algorithm
state := if size(X,1) == 0 then
    ThermodynamicState(p=p,T=T_psX(p,s,reference_X),X=reference_X)
else if size(X,1) == nX then
    ThermodynamicState(p=p,T=T_psX(p,s,X),X=X)
else
    ThermodynamicState(p=p,T=T_psX(p,s,X), X=cat(1,X,{1-sum(X)}));
```

これらは、温度をそれぞれ次の非線形方程式を解く関数によって計算する。

MixtureGasNasa.T\_hX(h,X)

MixtureGasNasa.T\_psX(p,s,X)

## II. 比エンタルピや比エントロピから温度を求める関数

### MixtureGasNasa.T\_hX 比エンタルピと質量分率から温度を求める関数

温度と比エンタルピの関係を表す非線形方程式を解いて比エンタルピから温度を求める。

#### T\_hX の非線形方程式

MixtureGasNasa.T\_hX.Internal.f\_nonlinear のアルゴリズム

```
redeclare function extends f_nonlinear
algorithm
  y := h_TX(x,X);
end f_nonlinear;
```

x: 温度 [K]

y: 比エンタルピ [J/kg]

#### T\_hX の入出力変数

<<function>> T_hX	
<<input>> h: SpecificEnthalpy	}
<<input>> X: MassFraction[nX]	
<<input>> exclEnthForm : Boolean	
<<input>> refChoice : ReferenceEnthalpy	
<<input>> h_off : SpecificEnthalpy	
<<output>> T : Temperature	
<<protected>> Xfull : MassFraction[nX]	
package <b>Internal</b>	
algorithm T = <b>Internal.solve</b> (h,200,6000,1.0e5, Xfull,data[1])	

このエンタルピーのオプション引数は、Internal.f\_nonlinearに渡されないため使用されない。



MixtureGasNasa の定数

- excludeEnthalpyOfFormation
  - referenceChoice
  - h\_offset
- を参照する。

非線形方程式解法の概要は、[https://www.amane.to/wp-content/uploads/2018/10/Introduction\\_SingleGasNasa\\_20181007.pdf](https://www.amane.to/wp-content/uploads/2018/10/Introduction_SingleGasNasa_20181007.pdf) 5.II 参照

## MixtureGasNasa.T\_psX 圧力、比エントロピ、質量分率から温度を求める関数

温度と比エントロピの関係を表す非線形方程式を解くことによって比エントロピから温度を求める。

### T\_psX の非線形方程式

$$y = \mathbf{X} \cdot \mathbf{s}$$
$$= \sum_{i=1}^{nX} X_i s_{0i}(x) - \sum_{i=1}^{nX} X_i (R_i Y'(p_i))$$

$s = \{s_1, s_2, \dots, s_{nX}\},$   $s_i = s_{0i}(T) + R_i \log \frac{p_i}{p_0}$  成分のエントロピ

$R_i = \text{Modelica.Constants.R/MMX}[i]$  成分の気体定数

$p_i = Y_i p$  成分の分圧

$p_0 = \text{reference\_p} = 101325 \text{ [Pa]}$

$$Y'(p_i) = \begin{cases} \log \frac{p_i}{p_0}, & X_i \geq \varepsilon \\ Y_i, & X_i < \varepsilon \end{cases}$$

$\varepsilon = \text{Modelica.Constants.eps} = 1\text{e-}15$

$s_{TX}(x, X_{full})$  温度依存部分

圧力依存部分

$x$  : 温度 [K]  
 $y$  : 比エントロピ [J/kg.K]

$\mathbf{X} = \{X_1, X_2, \dots, X_{nX}\} = X_{full}$  質量分率

$\mathbf{Y} = \{y_1, y_2, \dots, y_{nX}\}$   
 $= \text{massToMolFractions}(X_{full}, \text{data.MM})$  モル分率

### MixtureGasNasa.T\_psX.Internal.f\_nonlinear のアルゴリズム

```
algorithm
  y := s_TX(x, Xfull) - sum(Xfull[i]*Modelica.Constants.R/MMX[i]*
    (if Xfull[i]<Modelica.Constants.eps then Y[i] else
      Modelica.Math.log(Y[i]*p/reference_p)) for i in 1:nX);
```

非線形方程式解法の概要は、[https://www.amane.to/wp-content/uploads/2018/10/Introduction\\_SingleGasNasa\\_20181007.pdf](https://www.amane.to/wp-content/uploads/2018/10/Introduction_SingleGasNasa_20181007.pdf) 5.11 参照

## MixtureGasNasa.s\_TX(T, X) 混合気体の比エントロピ（温度依存部分）

$$s_{TX}(T, X) = \sum_{i=1}^{size(X,1)} s_{0_i}(T) X_i$$

### アルゴリズム

```
algorithm
  s := sum(Modelica.Media.IdealGases.Common.Functions.s0_T(
    data[i], T)*X[i] for i in 1:size(X,1));
  annotation(Inline=true,smoothOrder=2);
end s_TX;
```



## 6. 物性値関数

### I. 熱力学的状态変数を返す関数

#### A. PartialMedium

- [pressure](#)(state)
  - [temperature](#)(state)
  - [density](#)(state)
  - [specificEnthalpy](#)(state)
  - [specificInternalenergy](#)(state)
  - [specificEntropy](#)(state)
  - [specificGibbsEnergy](#)(state)
  - [specificHelmholtzEnergy](#)(state)
- MixtureGasNasa で宣言する  
partial function
- $\text{temperature\_phX}(p, h, X) = \text{temperature}(\text{setState\_phX}(p, h, X))$
  - $\text{temperarure\_psX}(p, s, X) = \text{temperature}(\text{setState\_psX}(p, s, X))$
  - $\text{density\_pTX}(p, T, X) = \text{density}(\text{setState\_pTX}(p, T, X))$
  - $\text{density\_phX}(p, h, X) = \text{density}(\text{setState\_phx}(p, h, X))$
  - $\text{density\_psX}(p, s, X) = \text{density}(\text{setState\_psX}(p, s, X))$
  - $\text{specificEnthalpy\_pTX}(p, T, X) = \text{specificEnthalpy}(\text{setState\_pTX}(p, T, X))$
  - $\text{specificEnthalpy\_psX}(p, s, X) = \text{specificEnthalpy}(\text{setState\_psX}(p, s, X))$
  - $\text{specificEntropy\_pTX}(p, t, X) = \text{specificEnthalpy}(\text{setState\_pTX}(p, t, X))$

## C. MixtureGasNasa で宣言する式の内容

- `pressure(state) = state.p`
- `temperature(state) = state.T`
- `density(state)`

$$\text{密度 } \rho = \frac{1}{v} = \frac{p}{(\mathbf{X} \cdot \mathbf{R})T}$$

`= state.p/(state.X*data.R)*state.T)`

- `specificEnthalpy(state)`  
`= h_TX(state.T, state.X)`

$$\text{比エンタルピー } h = h(T, \mathbf{X})$$

- `specificInternalenergy(state)`

$$\text{内部エネルギー } u = h - pv = h - RT$$

`= h_TX(state.T, state.X) - gasConstant(state)*state.T`

- `specificGibbsEnergy(state)`

`= h_TX(state.T, state.X) - state.T*specificEntropy(state)`

- `specificHelmholtzEnergy(state)`

`= h_TX(state.T, state.X)`

$$\text{ギブス自由エネルギー } g = h - Ts$$

`- gasconstant(state)*state.T - state.T*specificEntropy(state)`

- `specificEntropy(state)`

$$\text{ヘルムホルツ自由エネルギー } f = u - Ts = h - RT - Ts$$

次ページ参照

`h_TX(state.T, state.X)`, `gasConstant(state)`

## MixtureGasNasa.specificEntropy(state) 混合気体の比エントロピ

$$s = \mathbf{X} \cdot \mathbf{s}$$

$$= \sum_{i=1}^{nX} X_i s_{0_i}(T) - \sum_{i=1}^{nX} X_i (R_i Y'(p_i))$$

$R_i = \text{Modelica.Constants.R/MMX}[i]$  成分の気体定数  
 $Y'(p_i) = \begin{cases} \log \frac{p_i}{p_0}, & X_i \geq \varepsilon \\ Y_i, & X_i < \varepsilon \end{cases}$  低濃度の場合の対数の発散をさける

$s_{TX}(T, X)$

温度依存部分

圧力依存部分

$p_i = Y_i p$  成分の分圧

$p_0 = \text{reference\_p} = 101325 \text{ [Pa]}$  基準圧力

$\varepsilon = \text{Modelica.Constants.eps} = 1\text{e-}15$

$\mathbf{X} = \{X_1, X_2, \dots, X_{nX}\}$  質量分率

$\mathbf{Y} = \{y_1, y_2, \dots, y_{nX}\}$  モル分率

```
function extends specificEntropy "Return specific entropy"
protected
  Real[nX] Y(unit="mol/mol")=massToMoleFractions(state.X, data.MM)
  "Molar fractions";
algorithm
  s := s_TX(state.T, state.X) - sum(state.X[i]*Modelica.Constants.R/MMX[i]*
    (if state.X[i]<Modelica.Constants.eps then Y[i] else
      Modelica.Math.log(Y[i]*state.p/reference_p)) for i in 1:nX);
  annotation(Inline=true,smoothOrder=2);
end specificEntropy;
```

## II. 粘性率(1) Method of Wike (1950)

MixtureGasNasa.dynamicViscosity 成分気体の粘性率を使う方法

Method of Wike (1950)

$$\eta_m = \sum_{i=1}^n \frac{y_i \eta_i}{\sum_{j=1}^n y_j \phi_{ij}}$$

$\eta_m$  : 混合気体の粘性率 (viscosity of the mixture) [Pa.s]

$\eta_i$  : 成分 の粘性率 (component viscosities) [Pa.s]

$y_i$  : 成分 のモル分率 (mole fractions) [mol/mol]

$M_i$  : 成分 i のモル質量 (mole masses) [kg/mol]

$$\phi_{ij} = \begin{cases} \frac{\left[1 + (\eta_i/\eta_j)^{1/2}(M_j/M_i)^{1/4}\right]^2}{\left[8(1 + M_i/M_j)\right]^{1/2}}, & i = 1 \text{ or } i \leq j \\ \frac{\eta_j}{\eta_i} \frac{M_j}{M_i} \phi_{ji}, & i \neq j \text{ and } i > j \end{cases}$$

[1] Bruce E. Poling, John M. Prausnitz, John P. O'Connell, THE PROPERTIES OF GASES AND LIQUIDS, Fifth Edition, McGRAW-HILL

[2] 武田哲明, Bing HAN, 小川益郎, 多成分混合気体の熱物性値, JAERI-M,92-131

<https://jopss.jaea.go.jp/pdfdata/JAERI-M-92-131.pdf>

[3] 川口雅之, 燃焼ガスの熱物性に関する研究-鹿児島大学リポジトリ

## MixtureGasNasa.dynamicViscosity(state) のアルゴリズム

```
function extends dynamicViscosity
  "Return mixture dynamic viscosity"
protected
  DynamicViscosity[nX] etaX "Component dynamic viscosities";
algorithm
  for i in 1:nX loop                                成分の粘性率  $\eta_i$  を計算する
    etaX[i] := Modelica.Media.IdealGases.Common.Functions.dynamicViscosityLowPressure(
      state.T,
      fluidConstants[i].criticalTemperature,
      fluidConstants[i].molarMass,
      fluidConstants[i].criticalMolarVolume,
      fluidConstants[i].acentricFactor,
      fluidConstants[i].dipoleMoment);
  end for;
  eta := gasMixtureViscosity(                        関数 gasMixtureViscosity で
    massToMoleFractions(                             混合気体の粘性率  $\eta_m$  を計算する
      state.X, fluidConstants[:].molarMass), fluidConstants[:].molarMass, etaX);
  annotation (smoothOrder=2);
end dynamicViscosity;
```

成分の粘性率は

[https://www.amane.to/wp-content/uploads/2018/10/Introduction\\_SingleGasNasa\\_20181007.pdf](https://www.amane.to/wp-content/uploads/2018/10/Introduction_SingleGasNasa_20181007.pdf)

6.11 参照

## MixtureGasNasa.gasMixtureViscosity(yi,M,eta) のアルゴリズム

```
protected
  Real fi[size(yi,1),size(yi,1)];
algorithm
  for i in 1:size(eta,1) loop
    assert(fluidConstants[i].hasDipoleMoment,"Dipole moment for "
      + fluidConstants[i].chemicalFormula + " not known. Can not compute viscosity.");
    assert(fluidConstants[i].hasCriticalData, "Critical data for "
      + fluidConstants[i].chemicalFormula + " not known. Can not compute viscosity.");
    for j in 1:size(eta,1) loop
      if i==j then
         $\phi_{ij}$  を計算する
        fi[i,j] := (1 + (eta[i]/eta[j])^(1/2)*(M[j]/M[i])^(1/4))^2/(8*(1 + M[i]/M[j]))^(1/2);
      elseif j<i then
        fi[i,j] := eta[i]/eta[j]*M[j]/M[i]*fi[j,i];
      else
        fi[i,j] := (1 + (eta[i]/eta[j])^(1/2)*(M[j]/M[i])^(1/4))^2/(8*(1 + M[i]/M[j]))^(1/2);
      end if;
    end for;
  end for;
   $\eta_m$  を計算する
  etam := sum(yi[i]*eta[i]/sum(yi[j]*fi[i,j] for j in 1:size(eta,1)) for i in 1:size(eta,1));
```

### III. 粘性率(2) Chung, et al. rules (1984, 1988)

**mixtureViscosityChung** 成分気体の粘性率を使わない方法

$$\eta_m = \frac{26.69 F_{cm} (M_m T)^{\frac{1}{2}}}{\sigma_m^2 \Omega_v} = \frac{26.69 F_{cm} (M_m T)^{\frac{1}{2}}}{(\sigma_m^3)^{\frac{2}{3}} \Omega_v} \quad [\mu\text{P}]$$

(2) ↓      ↓ (3)  
(1) →      ← (4)

```
<<function>> mixtureViscosityChung
<<input>> T: Temperature
<<input>> Tc: Temperature[nX]
<<input>> Vcrit: MolarVolume[nX]
<<input>> w: Real[nX]
<<input>> mu: Real[nX]
<<input>> MolecularWeights: MolarMass[nX]
<<input>> y: MassFraction[nX]
<<input>> kappa: Real[nX] = zeros[nX]
<<output>> etaMixture: DynamicViscosity
<<protected constant>> Vc = Vcrit*1000000
<<protected constant>> M = MolecularWeight *1000
...
algorithm
```

$T$  : 温度 (temperature) [K]  
 $T_{ci}$  : 臨界温度 (critical temperature) [K]  
 $V_{crit}$  : 臨界体積 (critical volumes) [m<sup>3</sup>/mol]  
 $w_i$  : 偏心因子 (acentric factors)  
 $\mu_i$  : 双極子モーメント (dipole moments) [debyes]  
 MolecularWeights: モル質量 [kg/mol]  
 $y_i$  : モル分率 (molar fractions) [mol/mol]  
 $\kappa_i$  : 会合因子 (association factors)

$\eta_m$  : 混合気体の粘性率 [Pa.s]

$V_{ci}$  : 臨界体積 [cm<sup>3</sup>/mol] = Vcrit[i]\*1000000  
 $M_i$  : モル質量 [g/mol] = MolecularWeight[i]\*1000

[1] THE PROPERTIES OF GASES AND LIQUIDS, Fifth Edition, McGRAW-HILL

Bruce E. Poling, John M. Prausnitz, John P. O'Connell.

[2] Chung, T.-H., M. Ajlan, L. L. Lee, and K. E. Starling; Ind. Eng. Chem. Res., 27: 671 (1988).

[3] Chung, T.-H., L. L. Lee, and K. E. Starling; Ing. Eng. Chem. Fundam., 23: 3 (1984).

## (1) 混合気体の分子の大きさを表す量

$$\sigma_m^3 = \sum_{i=1}^n \sum_{j=1}^n y_i y_j \sigma_{ij}^3,$$

$$\sigma_{ij} = \begin{cases} 0.809 V_{ci}^{1/3}, & i = j \\ \xi_{ij} (\sigma_i \sigma_j)^{1/2} = ((0.809 V_{ci}^{1/3})(0.809 V_{cj}^{1/3}))^{1/2}, & i \neq j \end{cases}$$

$\xi_{ij} = 1$  2成分相互作用パラメータ(binary interaction parameter)で、通常は1とする

## (2) 希薄気体の分子形状や極性を表す因子

$$F_{cm} = 1 - \underbrace{0.275 \omega_m}_{(a)} + 0.059035 \underbrace{\mu_{rm}^4}_{(b)} + \underbrace{\kappa_m}_{(c)}$$

### (a) 混合気体の偏心因子 (acentric factor)

$$\omega_m = \frac{\sum_{i=1}^n \sum_{j=1}^n y_i y_j \omega_{ij} \sigma_{ij}^3}{\sigma_m^3}, \quad \omega_{ij} = \begin{cases} \omega_i, & i = j \\ \frac{\omega_i + \omega_j}{2}, & i \neq j \end{cases}$$



## (b) 混合気体の無次元化した双極子モーメント (dimensionless dipole moment)

$$\mu_{rm} = \frac{131.3\mu_m}{(V_{cm}T_{cm})^{1/2}}, \quad \mu_m = \left\{ \sigma_m^3 \sum_{i=1}^n \sum_{j=1}^n \frac{y_i y_j \mu_i^2 \mu_j^2}{\sigma_{ij}^3} \right\}^{1/4}, \quad V_{cm} = \frac{\sigma_m^3}{0.8093},$$

$$T_{cm} = 1.2593 \left( \frac{\varepsilon}{k} \right)_m, \quad \left( \frac{\varepsilon}{k} \right)_m = \frac{\sum_{i=1}^n \sum_{j=1}^n y_i y_j \left( \frac{\varepsilon_{ij}}{k} \right) \sigma_{ij}^3}{\sigma_m^3},$$

$$\left( \frac{\varepsilon_{ij}}{k} \right) = \begin{cases} \frac{\varepsilon_i}{k}, & i = j \\ \zeta_{ij} \left( \frac{\varepsilon_i}{k} \frac{\varepsilon_j}{k} \right)^{1/2}, & i \neq j \end{cases}, \quad \frac{\varepsilon_i}{k} = \frac{T_{ci}}{1.2593}$$

$\zeta_{ij} = 1$  2成分相互作用パラメータ(binary interaction parameter)で、通常は1とする

## (c) アルコールや酸など極性の強い物質の相関

(correlation for highly polar substances such as alcohols and acids)

$$\kappa_m = \sum_{i=1}^n \sum_{j=1}^n y_i y_j \kappa_{ij}, \quad \kappa_{ij} = \begin{cases} \kappa_i, & i = j \\ \left( \kappa_i \kappa_j \right)^{1/2}, & i \neq j \end{cases}$$

### (3) 混合気体のモル質量

$$M_m = \left[ \frac{\sum_{i=1}^n \sum_{j=1}^n y_i y_j \left( \frac{\varepsilon_{ij}}{k} \right) \sigma_{ij}^2 M_{ij}^{1/2}}{\left( \frac{\varepsilon}{k} \right)_m \sigma_m^2} \right]^2, \quad M_{ij} = \frac{2M_i M_j}{M_i + M_j}$$

### (4) 衝突積分

$$\Omega_v = A(T_m^*)^{-B} + C \exp(-DT_m^*) + E \exp(-FT_m^*),$$

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

$$T_m^* = \frac{T}{(\varepsilon/k)_m}$$

# mixtureViscosityChung のアルゴリズム①

$V_{ci}, M_i$

```
protected
  constant Real[size(y,1)] Vc = Vcrit*1000000 "Critical volumes (cm3/mol)";
  constant Real[size(y,1)] M = MolecularWeights*1000 "Molecular weights (g/mol)";
Integer n = size(y,1) "Number of mixed elements" ;
...
```

$M_{ij}, \sigma_{ij}, \left(\frac{\varepsilon_{ij}}{k}\right), \omega_{ij}, \kappa_{ij}$

```
algorithm
//combining rules
for i in 1:n loop
  for j in 1:n loop
    Mij[i,j] := 2*M[i]*M[j]/(M[i]+M[j]);
    if i==j then
      sigma[i,j] := 0.809*Vc[i]^(1/3);
      edivk[i,j] := Tc[i]/1.2593;
      wij[i,j] := w[i];
      kappa[i,j] := kappa[i];
    else
      sigma[i,j] := (0.809*Vc[i]^(1/3)*0.809*Vc[j]^(1/3))^(1/2);
      edivk[i,j] := (Tc[i]/1.2593*Tc[j]/1.2593)^(1/2);
      wij[i,j] := (w[i] + w[j])/2;
      kappa[i,j] := (kappa[i]*kappa[j])^(1/2);
    end if;
  end for;
end for;
```

## mixtureViscosityChung のアルゴリズム②

$$\sigma_m^3, \left(\frac{\varepsilon}{k}\right)_m, M_m, \omega_m V_{cm}, T_{cm}, \kappa_m, F_{cm}, T_m^*, \Omega_v, \eta_m, \eta_{Mixture}$$

```
//mixing rules
sigmam3 := (sum(sum(y[i]*y[j]*sigma[i,j]^3 for j in 1:n) for i in 1:n));
//(epsilon/k)m
edivkm := (sum(sum(y[i]*y[j]*edivk[i,j]*sigma[i,j]^3 for j in 1:n) for i in 1:n))/sigmam3;
Mm := ((sum(sum(y[i]*y[j]*edivk[i,j]*sigma[i,j]^2*Mij[i,j]^(1/2)
  for j in 1:n) for i in 1:n))/(edivkm*sigmam3^(2/3)))^2;
wm := (sum(sum(y[i]*y[j]*wij[i,j]*sigma[i,j]^3 for j in 1:n) for i in 1:n))/sigmam3;
mum := (sigmam3*(sum(sum(y[i]*y[j]*mu[i]^2*mu[j]^2/sigma[i,j]^3
  for j in 1:n) for i in 1:n)))^(1/4);
Vcm := sigmam3/(0.809)^3;
Tcm := 1.2593*edivkm; murm := 131.3*mum/(Vcm*Tcm)^(1/2);
kappam := (sigmam3*(sum(sum(y[i]*y[j]*kappaij[i,j] for j in 1:n) for i in 1:n)));
Fcm := 1 - 0.275*wm + 0.059035*murm^4 + kappam;
Tmstar := T/edivkm;
omegav := 1.16145*(Tmstar)^(-0.14874) + 0.52487*Math.exp(-0.77320*Tmstar) +
2.16178*Math.exp(-2.43787*Tmstar);
etam := 26.69*Fcm*(Mm*T)^(1/2)/(sigmam3^(2/3)*omegav);
etaMixture := etam*1e7;
  annotation (smoothOrder=2,);
end mixtureViscosityChung;
```

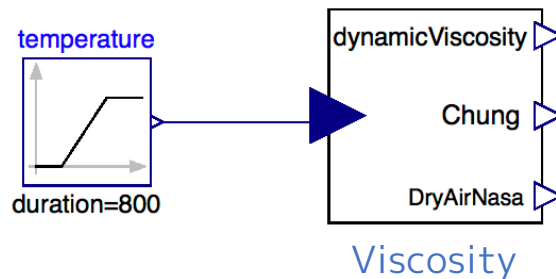
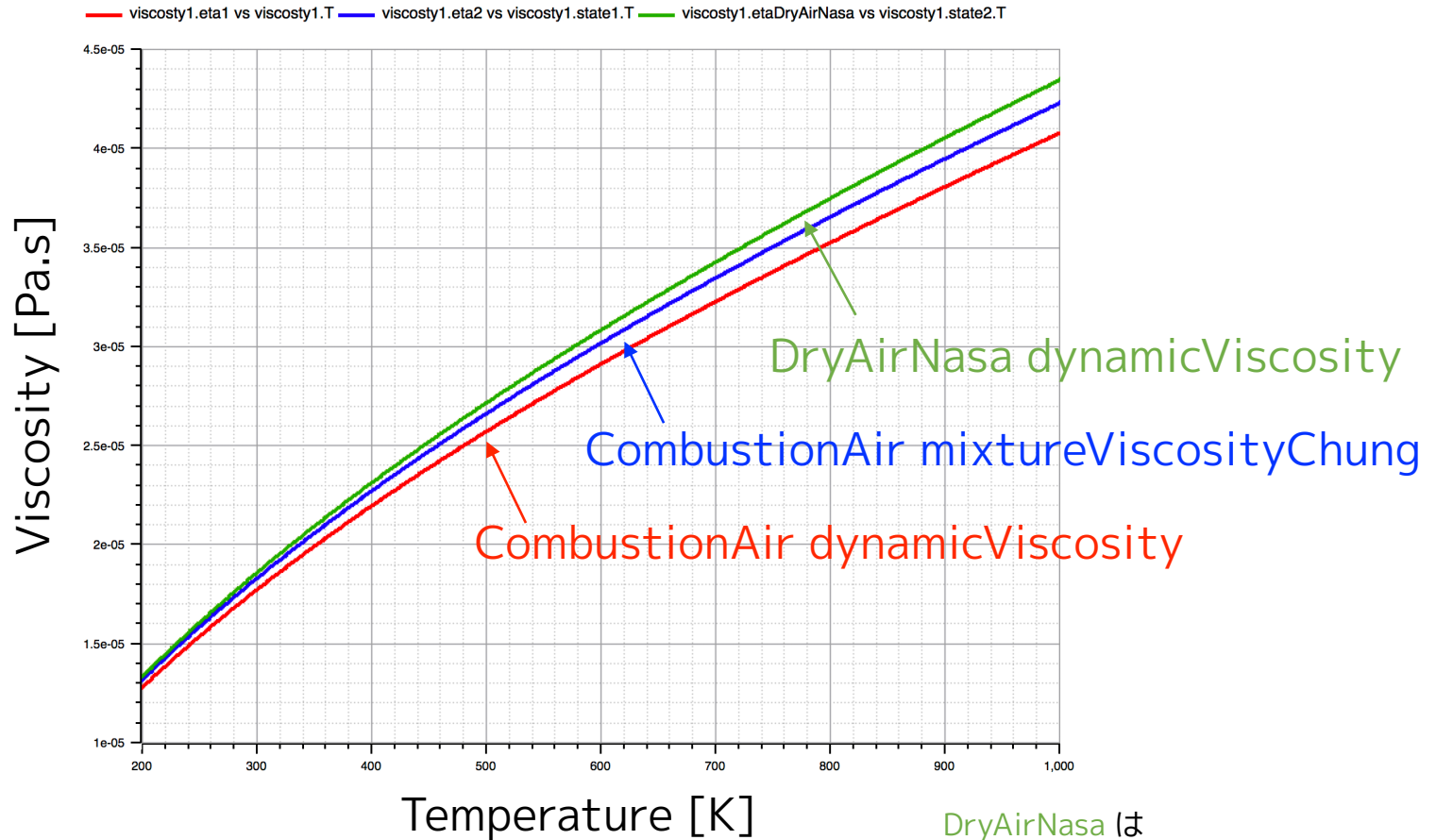
正しくは  $\text{etam} \cdot 1\text{e-7}$

MSL 3.2.2 のバグです。

修正は MSL 3.2.3 milestone になってます。

<https://github.com/modelica/ModelicaStandardLibrary/issues/2764>

# 空気の粘性率の比較



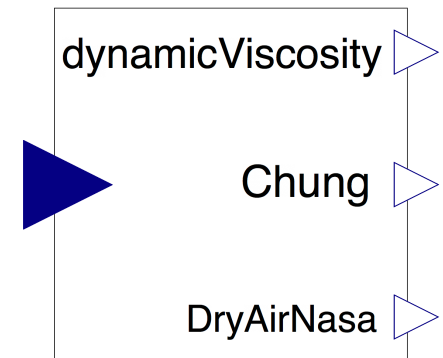
DryAirNasa は  
[https://www.amane.to/wp-content/uploads/2018/10/Introduction\\_SingleGasNasa\\_20181007.pdf](https://www.amane.to/wp-content/uploads/2018/10/Introduction_SingleGasNasa_20181007.pdf) 7 参照

```

model Viscosty
  replaceable package Medium1 = Modelica.Media.IdealGases.MixtureGases.CombustionAir;
  replaceable package Medium2 = Modelica.Media.Air.DryAirNasa;
  Medium1.ThermodynamicState state1;
  Medium2.ThermodynamicState state2;
  Modelica.Blocks.Interfaces.RealInput T annotation( ...);
  Modelica.Blocks.Interfaces.RealOutput eta1 annotation( ...);
  Modelica.Blocks.Interfaces.RealOutput eta2 annotation( ...);
  Modelica.Blocks.Interfaces.RealOutput etaDryAirNasa annotation( ...);
equation
  state1 = Medium1.setState_pTX(p = 101325, T = T, X = Medium1.reference_X);
  eta1 = Medium1.dynamicViscosity(state1);
  eta2 = Medium1.mixtureViscosityChung(T = T,
    Tc = Medium1.fluidConstants.criticalTemperature,
    Vcrit = Medium1.fluidConstants.criticalMolarVolume,
    w = Medium1.fluidConstants.acentricFactor,
    mu = Medium1.fluidConstants.dipoleMoment,
    MolecularWeights = Medium1.data.MM,
    y = Medium1.massToMoleFractions(state1.X, Medium1.MMX))*1e-14;
  state2 = Medium2.setState_pT(p = 101325, T = T);
  etaDryAirNasa = Medium2.dynamicViscosity(state2);
  annotation( ...);
end Viscosty;

```

↑  
バグ対応！



## IV. 熱伝導率

### Wassiljew (1904)

$$\lambda_m = \sum_{i=1}^n \frac{y_i \lambda_i}{\sum_{j=1}^n y_j A_{ij}}$$

$\lambda_m$ : 混合気体の熱伝導率

$y_i$ : 成分のモル分率

(mole fraction of components)

$\lambda_i$ : 成分の熱伝導率

(thermal conductivity of components)

### Mason and Saxena Modification (1958)

$$A_{ij} = \frac{\varepsilon \left[ 1 + \left( \frac{\lambda_{tri}}{\lambda_{trj}} \right)^{1/2} \left( \frac{M_i}{M_j} \right)^{1/4} \right]^2}{\left[ 8 \left( 1 + \frac{M_i}{M_j} \right) \right]^{1/2}}$$

$\varepsilon = 1$  1に近い数値的定数 (numerical constant near unity)

$$\frac{\lambda_{tri}}{\lambda_{trj}} = \frac{\Gamma_j [\exp(0.0464T_{ri}) - \exp(-0.2412T_{ri})]}{\Gamma_i [\exp(0.0464T_{rj}) - \exp(-0.2412T_{rj})]}$$

上式の単位は、

$P_{ci}$  [Pa]

$M_i$  [kg/mol]

を用いるが比をとるので問題ない。

### Roy and Thodos (1968, 1970)

$$\Gamma_i = 210 \left( \frac{T_{ci} M_i^3}{P_{ci}^4} \right)^{1/6} \quad \text{reduced inverse thermal conductivity [W/m.K]}^{-1}$$

$$T_{ri} = \frac{T}{T_{ci}} \quad \text{reduced temperature}$$

$T_{ci}$ : 臨界温度 [K],

$P_{ci}$ : 臨界圧力 [bar],

$M_i$ : モル質量 [g/mol]

[1] THE PROPERTIES OF GASES AND LIQUIDS, Fifth Edition, McGRAW-HILL

Bruce E. Poling, John M. Prausnitz, John P. O'Connell.

## MixtureGasNasa.thermalConductivity(state, method)のアルゴリズム

```
function extends thermalConductivity
  "Return thermal conductivity for low pressure gas mixtures"
  input Integer method=methodForThermalConductivity
  "Method to compute single component thermal conductivity"
protected
  ThermalConductivity[nX] lambdaX "Component thermal conductivities";
  DynamicViscosity[nX] eta "Component thermal dynamic viscosities";
  SpecificHeatCapacity[nX] cp "Component heat capacity";
algorithm
  for i in 1:nX loop
    assert(fluidConstants[i].hasCriticalData, "Critical data for "+
      fluidConstants[i].chemicalFormula +
      " not known. Can not compute thermal conductivity.");
    eta[i] :=Modelica.Media.IdealGases.Common.Functions.dynamicViscosityLowPressure(
      state.T, fluidConstants[i].criticalTemperature,
      fluidConstants[i].molarMass, fluidConstants[i].criticalMolarVolume,
      fluidConstants[i].acentricFactor, fluidConstants[i].dipoleMoment);
    cp[i] :=Modelica.Media.IdealGases.Common.Functions.cp_T(data[i], state.T);
    lambdaX[i] :=Modelica.Media.IdealGases.Common.Functions.thermalConductivityEstimate(
      Cp=cp[i], eta=eta[i], method=method, data=data[i]);
  end for;
  lambda := lowPressureThermalConductivity(massToMoleFractions(state.X,
    fluidConstants[:].molarMass), state.T,
    fluidConstants[:].criticalTemperature,
    fluidConstants[:].criticalPressure,
    fluidConstants[:].molarMass,
    lambdaX);
  annotation (smoothOrder=2);
end thermalConductivity;
```

成分気体の  
粘性率  $\eta_i$  と比熱  $C_{p_i}$  を求め  
熱伝導率  $\lambda_i$  を計算する。

混合気体の熱伝導率  $\lambda_m$  を求める。

[https://www.amane.to/wp-content/uploads/2018/10/Introduction\\_SingleGasNasa\\_20181007.pdf](https://www.amane.to/wp-content/uploads/2018/10/Introduction_SingleGasNasa_20181007.pdf) 6.III 参照



## MixtureGas.Nasa.lowPressureThermalConductivity(y, T, Tc, Pc, M, lambda) のアルゴリズム

```
algorithm
  for i in 1:size(y,1) loop
    gamma[i] := 210*(Tc[i]*M[i]^3/Pc[i]^4)^(1/6);    $\Gamma_i, T_{ri}$  を計算する
    Tr[i] := T/Tc[i];
  end for;
  for i in 1:size(y,1) loop
    for j in 1:size(y,1) loop
      A[i,j] := epsilon*(1 +
        (gamma[j]*(Math.exp(0.0464*Tr[i]) - Math.exp(-0.2412*Tr[i]))/
        (gamma[i]*(Math.exp(0.0464*Tr[j]) - Math.exp(-0.2412*Tr[j]))))^(1/2)
        *(M[i]/M[j])^(1/4))^2/(8*(1 + M[i]/M[j]))^(1/2);
       $A_{ij}$  を計算する
    end for;
  end for;
  lambdam := sum(y[i]*lambda[i]/(sum(y[j]*A[i,j] for j in 1:size(y,1))) for i in 1:size(y,1));
  annotation (smoothOrder=2, ...);
end lowPressureThermalConductivity;
```

$\lambda_m$  を計算する

# V. モル分率、質量分率、気体定数、モル質量

## (1) 混合気体のモル分率

### PartialMixtureMedium.massToMoleFractions(X, MMX) のアルゴリズム

1kg に含まれる成分  $i$  のモル数 [mol/kg]

$$y_i = \frac{\frac{X_i}{M_i}}{\sum_i \frac{X_i}{M_i}} = M_{mix} \frac{X_i}{M_i},$$

1 kg に含まれる全モル数[mol/kg]

$y_i = \text{moleFraction}[i]$  [mol/mol]  
 $M_i = \text{MMX}[i]$  : モル質量 [kg/mol]  
 $X_i$  : 質量分率 [kg/kg]

$$M_{mix} = \frac{1}{\sum_i \frac{X_i}{M_i}} = 1/(X * \text{invMMX}) : \text{混合物のモル質量 [kg/mol]}$$

```
algorithm
  for i in 1:size(X, 1) loop
    invMMX[i] := 1/MMX[i];
  end for;
  Mmix := 1/(X*invMMX);
  for i in 1:size(X, 1) loop
    moleFractions[i] := Mmix*X[i]/MMX[i];
  end for;
```

## (2) 混合気体の質量分率

### PartialMixtureMedium.moleToMassFractions(moleFractions, MMX) のアルゴリズム

$$X_i = \frac{y_i M_i}{\sum_i y_i M_i} = \frac{y_i M_i}{M_{mix}}$$

1 mol に含まれる成分 の質量 [kg/mol]

混合物のモル質量 [kg/mol]

1 mol に含まれる全成分の質量 [kg/mol]

$y_i$  = moleFractions[i] : モル分率 [mol/mol]

$M_i$  = MMX[i] : モル質量 [kg/mol]

```
fractions of gas mixture";
protected
    MolarMass Mmix=moleFractions*MMX "Molar mass of mixture";
algorithm
    for i in 1:size(moleFractions, 1) loop
        X[i] := moleFractions[i]*MMX[i]/Mmix;
    end for;
```

### (3) 混合物のモル質量

#### MixtureGasNasa.molarMass(state)

$$MM = \frac{1}{\sum_j \frac{X_j}{M_j}} \quad : \text{混合物 1kg に含まれる全モル数 [mol/kg]}$$

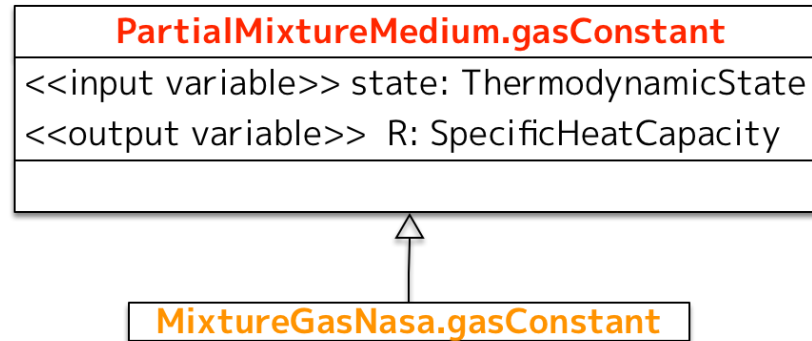
$M_j = \text{data}[j].M$  : j 番目の成分のモル質量[kg/mol]

$X_j$  : j 番目の成分の質量分率 [kg/kg]

```
function extends molarMass "Return molar mass of mixture"  
algorithm  
  MM := 1/sum(state.X[j]/data[j].MM for j in 1:size(state.X, 1));  
  annotation(Inline=true,smoothOrder=2);  
end molarMass;
```

## (4) 混合物の気体定数

### MixtureGas.Nasa.gasConstant(state)



$R = \mathbf{R} \cdot \mathbf{X}$        $\mathbf{R} = \text{data.R}$       成分物質の気体定数のベクトル  
 $\mathbf{X} = \text{state.X}$       成分物質の質量分率のベクトル

```
function extends gasConstant "Return gasConstant"  
algorithm    R := data.R*state.X;  
            annotation(Inline = true, smoothOrder = 3);  
end gasConstant;
```

## VI. 密度の偏微分

### MixtureGasNasa.density\_derp\_T(state) のアルゴリズム

理想気体状態方程式

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

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

algorithm

```
ddpT := 1/(state.T*gasConstant(state));
```

### MixtureGasNasa.density\_derT\_p(state) のアルゴリズム

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

algorithm

```
ddTp := -state.p/(state.T*state.T*gasConstant(state));
```

### MixtureGasNasa.density\_derX(state) のアルゴリズム

$$\frac{\partial \rho}{\partial X_i} = -\frac{p}{RT} \frac{M}{M_i} \quad \text{次ページ参照}$$

algorithm

```
dddX := {-state.p/(state.T*gasConstant(state))*molarMass(state)/data[  
  i].MM for i in 1:nX};
```

## 密度を質量分率で微分する

### 成分 の気体定数

$$R_i = \frac{R_{mol}}{M_i} \quad R_{mol} = 8.314598 \text{ [J/mol.K]} \text{ モル気体定数}$$
$$M_i = \text{data}[i].MM: \text{成分 } i \text{ のモル質量 [kg/mol]}$$

### 混合気体の気体定数

$$R = \sum_i R_i X_i = \sum_i \frac{R_{mol}}{M_i} X_i = R_{mol} \sum_i \frac{X_i}{M_i}$$

### 混合気体の質量分率による偏微分


$$\frac{\partial R}{\partial X_i} = \frac{R_{mol}}{M_i}$$

### 混合気体のモル質量

### 密度の偏微分

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

$$\frac{\partial \rho}{\partial X_i} = -\frac{p}{TR^2} \frac{\partial R}{\partial X_i} = -\frac{p}{TR} \frac{\frac{\partial R}{\partial X_i}}{R} = -\frac{p}{TR} \frac{\frac{R_{mol}}{M_i}}{R_{mol} \sum_i \frac{X_i}{M_i}} = -\frac{p}{TR} \frac{M}{M_i}$$

$$M = \frac{1}{\sum_i \frac{X_i}{M_i}}$$


## VII. 等エントロピー過程のエンタルピー

気体が `refState` の状態から等エントロピー的に変化して `p_downstream` になった場合の比エンタルピーを求める。

- `exact = true`  
関数 `specificEnthalpy_psX` を使用して、圧力 `p_downstream`、比エントロピー `specificEntropy(refState)` の状態の比エンタルピーを直接求める。
- `exact = false`  
等エントロピーを仮定した関係式から比エンタルピーを求める。

### `MixtureGasNasa.isentropicEnthalpy(p_downstream, refState, exact)` のアルゴリズム

```
function extends isentropicEnthalpy "Return isentropic enthalpy"  
  input Boolean exact = false  
  "Flag whether exact or approximate version should be used"; algorithm  
  h_is := if exact then  
    specificEnthalpy_psX(p_downstream, specificEntropy(refState), refState.X)  
  else  
    isentropicEnthalpyApproximation(p_downstream, refState);  
    annotation(Inline=true, smoothOrder=2);  
end isentropicEnthalpy;
```



## MixtureGasNasa.isentropicEnthalpyApproximation(p2, state) のアルゴリズム

$h = \mathbf{h} \cdot \mathbf{X}$        $\mathbf{h}$ :成分物質の比エンタルピー     $\mathbf{X}$ :質量分率

$$h_{is} = h + \frac{p_1}{\rho_1} \frac{\gamma}{\gamma - 1} \left( \left( \frac{p_2}{p_1} \right)^{(\gamma-1)/\gamma} - 1 \right) = h + RT \frac{\gamma}{\gamma - 1} \left( \left( \frac{p_2}{p_1} \right)^{(\gamma-1)/\gamma} - 1 \right)$$

上式の導出は、

[https://www.amane.to/wp-content/uploads/2018/10/Introduction\\_SingleGasNasa\\_20181007.pdf](https://www.amane.to/wp-content/uploads/2018/10/Introduction_SingleGasNasa_20181007.pdf) 6.V 参照

algorithm

```
X := if reducedX then cat(1,state.X,{1-sum(state.X)}) else state.X;
h_component := {Modelica.Media.IdealGases.Common.Functions.h_T(
  data[i], state.T, excludeEnthalpyOfFormation,
  referenceChoice, h_offset) for i in 1:nX};
h := h_component*X;
h_is := h + gamma/(gamma - 1.0)*(state.T*gasConstant(state))*
  ((p2/state.p)^((gamma - 1)/gamma) - 1.0);
annotation(smoothOrder=2);
end isentropicEnthalpyApproximation;
```

## VIII. その他の物性値

### A. PartialMedium

- `heatCapacity_cp(state) = specificHeatCapacityCp(state)`
- `heatCapacity_cv(state) = specificHeatCapacityCv(state)`
- `beta(state) = isobaricExpansionCoefficient(state)`
- `kappa(state) = isothermalCompressibility(state)`
- `prandtlNumber(state)`

Prandtl Number (プラントル数)

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

$\eta$  : 粘性率

$C_p$  : 定圧比熱

$\lambda$  : 熱伝導率

### C. MixtureGasNasa

- `h_TX_der(T,X,exclEnthForm, refChoice,h_off,dT,dX)`
- `specificHeatCapacityCp(state)`
- `specificHeatCapacityCv(state)`
- `isentropicExponent(state)`
- `velocityOfSound(state)`
- `isobaricExpansionCoefficient(state)`
- `isothermalCompressibility(state)`

## (1) 比エンタルピの微分

$$h = \mathbf{h} \cdot \mathbf{X} = \sum_{i=1}^{nX} h_i X_i \quad \text{より}$$

$$\frac{dh}{dt} = \sum_{i=1}^{nX} X_i \frac{dh_i}{dt} + \sum_{i=1}^{nX} h_i \frac{dX_i}{dt} = \sum_{i=1}^{nX} (X_i C_{p_i}) \frac{dT}{dt} + \sum_{i=1}^{nX} \left( h_i \frac{dX_i}{dt} \right)$$

$$\text{fixedX} = \text{true} \text{ なら, } \quad \frac{dh}{dt} = \sum_{i=1}^{nX} (X_i C_{p_i}) \frac{dT}{dt}, \quad \frac{dX_i}{dt} = 0$$

## h\_TX\_der(T,X,exclEnthForm, refChoice,h\_off,dT,dX) のアルゴリズム

```
algorithm                                dT: 温度の微分, dX: 質量分率の微分
  h_der := if fixedX then
    dT*sum((Modelica.Media.IdealGases.Common.Functions.cp_T(
      data[i], T)*reference_X[i]) for i in 1:nX)
  else
    dT*sum((Modelica.Media.IdealGases.Common.Functions.cp_T(
      data[i], T)*X[i]) for i in 1:nX)+
    sum((Modelica.Media.IdealGases.Common.Functions.h_T(
      data[i], T)*dX[i]) for i in 1:nX);
  annotation (Inline = false, smoothOrder=1);
end h_TX_der;
```

## (2) 定圧比熱 **specificHeatCapacityCp(state)** のアルゴリズム

$$Cp = \mathbf{Cp} \cdot \mathbf{X}, \mathbf{Cp} = \{Cp_1, Cp_2, \dots, Cp_n\}$$

algorithm

```
cp := {Modelica.Media.IdealGases.Common.Functions.cp_T(  
  data[i], state.T) for i in 1:nX}*state.X;
```

## (3) 定積比熱 **specificHeatCapacityCv(state)** のアルゴリズム

$$Cv = \mathbf{Cp} \cdot \mathbf{X} - \mathbf{R} \cdot \mathbf{X}$$

algorithm

```
cv := {Modelica.Media.IdealGases.Common.Functions.cp_T(  
  data[i], state.T) for i in 1:nX}*state.X - data.R*state.X;
```

## (4) 等エントロピー指数 **isentropicExponent(state)** のアルゴリズム

$$\gamma = \frac{Cp}{Cv}$$

algorithm

```
gamma := specificHeatCapacityCp(state)/specificHeatCapacityCv(state);
```

## (5) 音速 `velocityOfSound(state)` のアルゴリズム

$$a = \sqrt{\gamma RT} = \sqrt{RT \frac{C_p}{C_v}}$$

algorithm      音速が負にならないようにしている。  
a := `sqrt(max(0, gasConstant(state)*state.T*specificHeatCapacityCp(state)  
                  /specificHeatCapacityCv(state)))`;

## (6) 定圧膨張係数 `isobaricExpansionCoefficient(state)` のアルゴリズム

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

algorithm  
beta := `1/state.T`;

## (7) 定圧膨張係数 `isothermalCompressibility(state)` のアルゴリズム

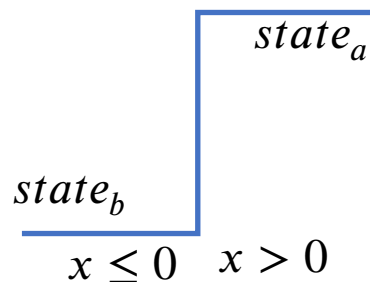
$$v = \frac{RT}{p} \quad \text{より} \quad \kappa \equiv -\frac{1}{v} \left( \frac{\partial v}{\partial p} \right)_T = -\frac{p}{RT} \left( -\frac{RT}{p^2} \right) = \frac{1}{p}$$

algorithm  
kappa := `1.0/state.p`;

# IX. setSmoothState

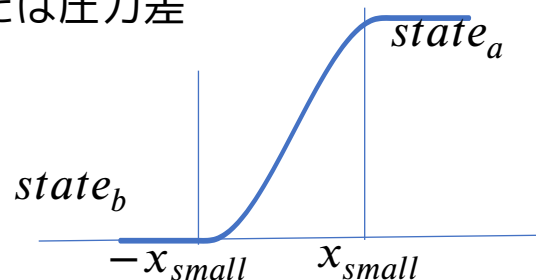
$x=0$  で不連続な熱力学的状態  $state\_a$ ,  $state\_b$  を滑らかに補間する。

$$y = \begin{cases} state_a, & x > 0 \\ state_b, & x \leq 0 \end{cases}$$



$$y = setSmoothState(x, state_a, state_b, x_{small})$$

$x$  は流量または圧力差



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

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

## MixtureGasNasa.setSmoothState のアルゴリズム

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),  
  X=Media.Common.smoothStep(x, state_a.X, state_b.X, x_small));
```

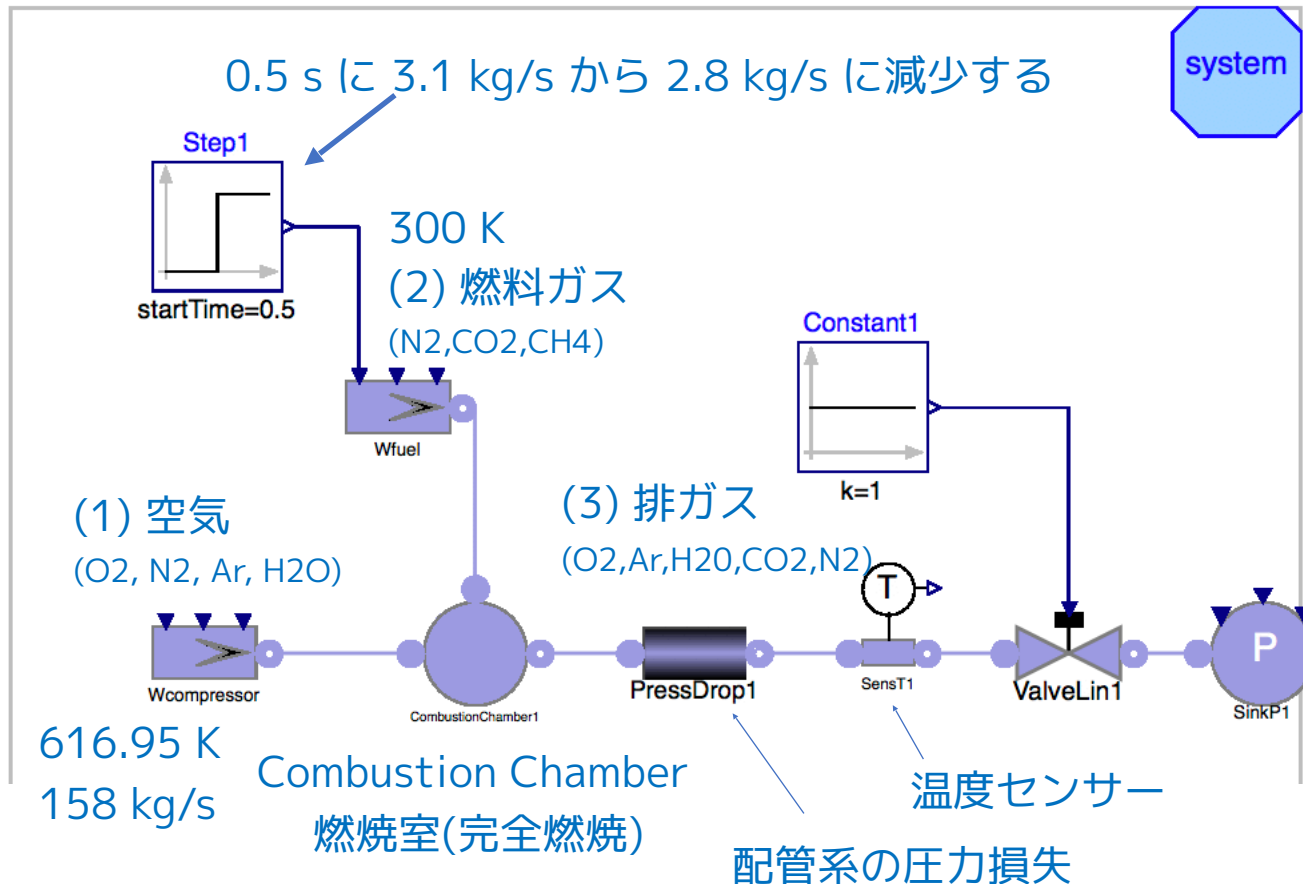
smoothStep については

[https://www.amane.to/wp-content/uploads/2018/10/Introduction\\_SingleGasNasa\\_20181007.pdf](https://www.amane.to/wp-content/uploads/2018/10/Introduction_SingleGasNasa_20181007.pdf) 6.VII 参照

# 7. ThermoPower.Gas.CombustionChamber

## I. MixtureGasNasa を使ったテストモデル

ThermoPower.Test.GasComponents.TestCC



初期化オプション

initOpt=ThermoPower.Choices.Init.Options.steadyState

## II. 気体

### (1) 空気 ThermoPower.Media.Air

$O_2, H_2O, Ar, N_2$

```
package Air "Air as mixture of O2, N2, Ar and H2O"  
extends Modelica.Media.IdealGases.Common.MixtureGasNasa(  
  mediumName="Air",  
  data={Modelica.Media.IdealGases.Common.SingleGasesData.O2,  
        Modelica.Media.IdealGases.Common.SingleGasesData.H2O,  
        Modelica.Media.IdealGases.Common.SingleGasesData.Ar,  
        Modelica.Media.IdealGases.Common.SingleGasesData.N2},  
  fluidConstants={  
    Modelica.Media.IdealGases.Common.FluidData.O2,  
    Modelica.Media.IdealGases.Common.FluidData.H2O,  
    Modelica.Media.IdealGases.Common.FluidData.Ar,  
    Modelica.Media.IdealGases.Common.FluidData.N2},  
  substanceNames={"Oxygen", "Water", "Argon", "Nitrogen"},  
  reference_X={0.23, 0.015, 0.005, 0.75},  
  referenceChoice=Modelica.Media.Interfaces.Choices.ReferenceEnthalpy.ZeroAt25C);  
end Air;
```

### ThermoPowerのデフォルト

- `excludeEnthalpyOfFormation = true`
- `referenceChoice = Modelica.Media.Interfaces.Choices.ReferenceEnthalpy.ZeroAt25C`



## (2) 燃料ガス ThermoPower.Media.NaturalGas

$N_2$ ,  $CO_2$ ,  $CH_4$

```
package NaturalGas "Mixture of N2, CO2, and CH4"  
  extends Modelica.Media.IdealGases.Common.MixtureGasNasa(  
    mediumName="NaturalGas",  
    data={Modelica.Media.IdealGases.Common.SingleGasesData.N2,  
          Modelica.Media.IdealGases.Common.SingleGasesData.CO2,  
          Modelica.Media.IdealGases.Common.SingleGasesData.CH4},  
    fluidConstants={  
      Modelica.Media.IdealGases.Common.FluidData.N2,  
      Modelica.Media.IdealGases.Common.FluidData.CO2,  
      Modelica.Media.IdealGases.Common.FluidData.CH4},  
    substanceNames={"Nitrogen", "Carbondioxide", "Methane"},  
    reference_X={0.02, 0.012, 0.968},  
    referenceChoice=Modelica.Media.Interfaces.Choices.ReferenceEnthalpy.ZeroAt25C);  
end NaturalGas;
```

### (3) 排ガス ThermoPower.Media.FlueGas

$O_2, Ar, H_2O, CO_2, N_2$

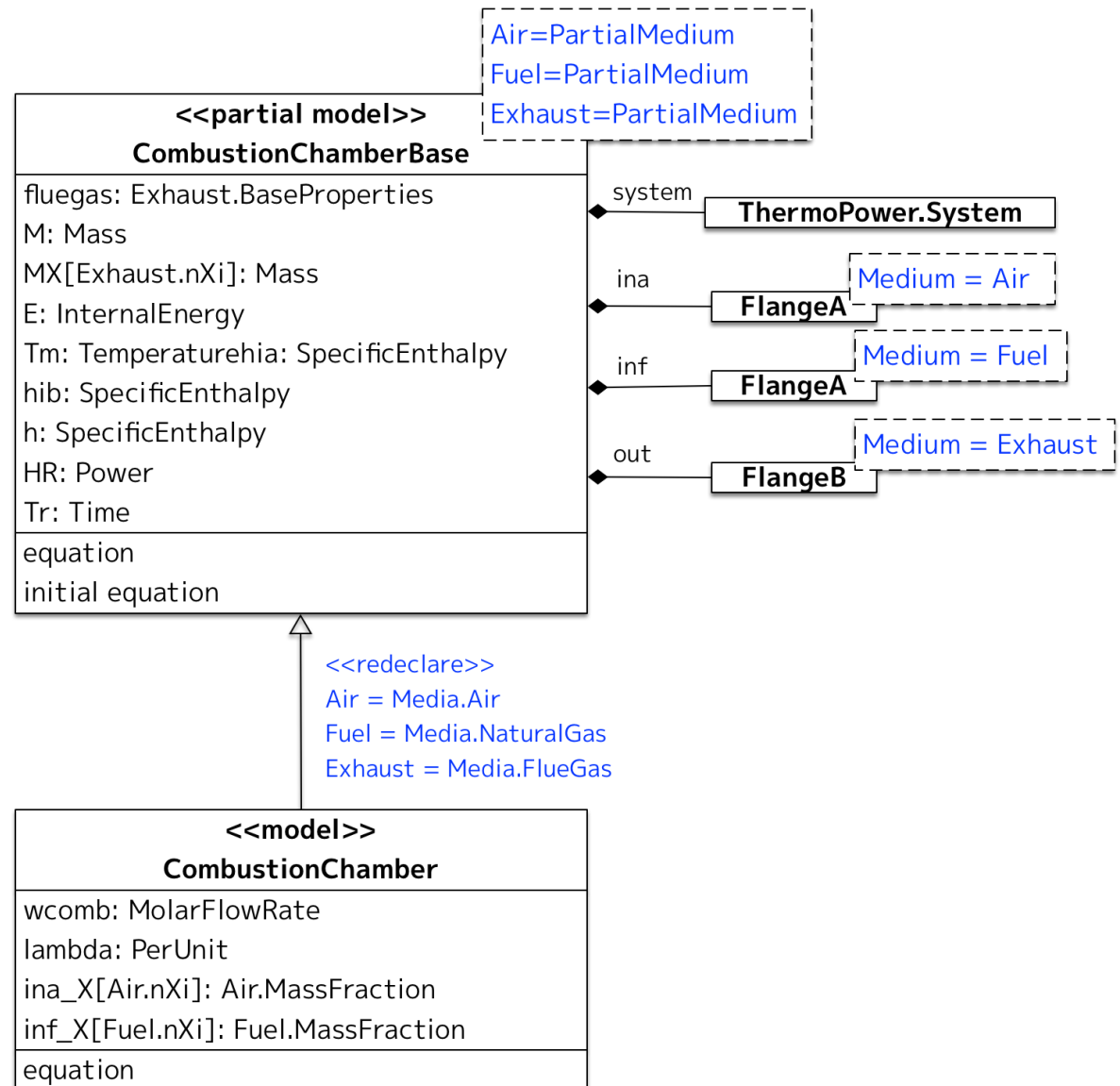
```
package FlueGas "flue gas"
  extends Modelica.Media.IdealGases.Common.MixtureGasNasa(
    mediumName="FlueGas",
    data={Modelica.Media.IdealGases.Common.SingleGasesData.O2,
          Modelica.Media.IdealGases.Common.SingleGasesData.Ar,
          Modelica.Media.IdealGases.Common.SingleGasesData.H2O,
          Modelica.Media.IdealGases.Common.SingleGasesData.CO2,
          Modelica.Media.IdealGases.Common.SingleGasesData.N2},
    fluidConstants={
      Modelica.Media.IdealGases.Common.FluidData.O2,
      Modelica.Media.IdealGases.Common.FluidData.Ar,
      Modelica.Media.IdealGases.Common.FluidData.H2O,
      Modelica.Media.IdealGases.Common.FluidData.CO2,
      Modelica.Media.IdealGases.Common.FluidData.N2},
    substanceNames={"Oxygen", "Argon", "Water", "Carbondioxide", "Nitrogen"},
    reference_X={0.23, 0.02, 0.01, 0.04, 0.7},
    referenceChoice=Modelica.Media.Interfaces.Choices.ReferenceEnthalpy.ZeroAt25C);
end FlueGas;
```

# III. CombustionChamber

## 継承関係

燃焼室全体の  
質量保存と  
エネルギー保存

成分物質の  
質量保存



## CombustionChamber のパラメータ

parameters	クラス	TestCC の設定値	概要
V	Volume	0.1	燃焼室の容積 [m3]
S	Area	0.1	壁面の伝熱面積 [m2]
gamma	CoefficientOfHeatTransfer	0	壁面の熱伝達係数 [W/m2.K]
Cm	HeatCapacity	0	壁面の熱容量 [J/K]
Tmstart	Temperature	300	壁面の初期温度 [K]
HH	SpecificEnthalpy	41.6e6	低位発熱量 [J/kg]
allowFlowReversal	Boolean	system.allowFlowReversal = true	逆流の有無
pstart	Air.AbsolutePressure	11.2e5	燃焼室の初期圧力
Tstart	Air.Temperaute	300	燃焼室の初期温度
Xstart[Exhaust.nX]	Air.MassFraction	Exhaust.reference_X	燃焼室の初期質量分率
initOpt	Choices.Init.Options	steadyState	初期化オプション
noInitialPressure	Boolean	false	圧力の初期化の有無

# CombustionChamberBase

- 容積一定の燃焼室のモデルベースである。
- 金属壁は温度と熱伝導率が一様で、外部に対して断熱である。
- 壁面の熱容量  $C_m$  または熱伝達率  $\gamma$  がゼロなら壁面熱伝達は無視される。
- 正しいエネルギーバランスを得る方法は、次の2通りがある。

- HH に低位発熱量を設定し、全ての気体で `excludeEnthalpyOfFormation = true` にする。
- HH = 0 とし、全ての気体で `excludeEnthalpyOfFormation = false` にする。

生成エンタルピーを使用して発熱量を計算する。

## CombustionChamberBase の方程式 (燃焼室全体の保存則)

equation

`M = fluegas.d*V "Gas mass";`

`E = fluegas.u*M "Gas energy";`

`MX = fluegas.Xi*M "Component masses";`

`HR = inf.m_flow*HH;`

`der(M) = ina.m_flow + inf.m_flow + out.m_flow "Gas mass balance";`

`der(E) = ina.m_flow*hia + inf.m_flow*hif + out.m_flow*ho`

`+ HR - gamma*S*(fluegas.T - Tm) "Gas energy balance";`

`if Cm > 0 and gamma > 0 then`

`Cm*der(Tm) = gamma*S*(fluegas.T - Tm) "Metal wall energy balance";`

`else`

`Tm = fluegas.T;`

`end if;`

燃焼室内の質量とエネルギー

成分物質の質量 (ベクトル)

燃焼による発熱量

質量保存則

エネルギー保存則

熱伝達による壁面温度変化の式

壁面の熱容量と熱伝達率がゼロなら、壁面温度と燃焼室内ガス温度が同じ

## CombustionChamberBase の方程式 (コネクタとの接続式)

```
// Set gas properties
out.p = fluegas.p;
out.h_outflow = fluegas.h;
out.Xi_outflow = fluegas.Xi;
```

} 流出側の接続

```
// Boundary conditions
ina.p = fluegas.p;
ina.h_outflow = 0;
ina.Xi_outflow = Air.reference_X[1:Air.nXi];
inf.p = fluegas.p;
inf.h_outflow = 0;
inf.Xi_outflow = Fuel.reference_X[1:Fuel.nXi];
```

流入する空気、燃料ガスの圧力は  
燃焼室内と同じ

} 流入側の接続

```
assert(ina.m_flow >= 0, "The model does not support flow reversal");
hia = inStream(ina.h_outflow);           空気の比エンタルピ (流入)
assert(inf.m_flow >= 0, "The model does not support flow reversal");
hif = inStream(inf.h_outflow);           燃料ガスの比エンタルピ(流入)
assert(out.m_flow <= 0, "The model does not support flow reversal");
ho = fluegas.h;                          排ガスの比エンタルピ (燃焼室内と同じ)
Tr = noEvent(M/max(abs(out.m_flow), Modelica.Constants.eps));
```

## CombustionChamberBase の方程式 (初期化方程式)

```
initial equation // Initial conditions
  if initOpt == Choices.Init.Options.noInit then
    // do nothing
  elseif initOpt == Choices.Init.Options.fixedState then
    if not noInitialPressure then
      fluegas.p = pstart;
    end if;
    fluegas.T = Tstart;
    fluegas.Xi = Xstart[1:Exhaust.nXi];
  elseif initOpt == Choices.Init.Options.steadyState then
    if not noInitialPressure then
      der(fluegas.p) = 0;
    end if;
    der(fluegas.T) = 0;
    der(fluegas.Xi) = zeros(Exhaust.nXi);
    if (Cm > 0 and gamma > 0) then
      der(Tm) = 0;
    end if;
  elseif initOpt == Choices.Init.Options.steadyStateNoP then
    der(fluegas.T) = 0;
    der(fluegas.Xi) = zeros(Exhaust.nXi);
    if (Cm > 0 and gamma > 0) then
      der(Tm) = 0;
    end if;
  else
    assert(false, "Unsupported initialisation option");
  end if;
  annotation ( ... );
end CombustionChamberBase;
```

### 基本的な使い方

- ① **noInit**  
何も初期化しない
- ② **fixedState**  
p=pstart  
T=Tstart  
X=Xstart
- ③ **steadyState**  
 $dp/dt = 0$   
 $dT/dt = 0$   
 $dXi/dt = 0$   
 $dTm/dt = 0$
- ④ **steadyStateNoP**  
 $dT/dt = 0$   
 $dXi/dt = 0$   
 $dTm/dt = 0$

圧力を初期化しないときは  
**noInitialPressure = true**  
にする。

## CombustionChamber

燃料ガスに含まれる  $CH_4$  が完全燃焼する。



```
model CombustionChamber "Combustion Chamber"
  extends BaseClasses.CombustionChamberBase(
    redeclare package Air = Media.Air "O2, H2O, Ar, N2",
    redeclare package Fuel = Media.NaturalGas "N2, CO2, CH4",
    redeclare package Exhaust = Media.FlueGas "O2, Ar, H2O, CO2, N2");
  Real wcomb(final quantity="MolarFlowRate", unit="mol/s") "Molar Combustion rate (CH4)";
  SI.PerUnit lambda
    "Stoichiometric ratio (>1 if air flow is greater than stoichiometric)";
protected
  Air.MassFraction ina_X[Air.nXi]=inStream(ina.Xi_outflow);
  Fuel.MassFraction inf_X[Fuel.nXi]=inStream(inf.Xi_outflow);
equation
  wcomb = inf.m_flow*inf_X[3]/Fuel.data[3].MM "Combustion molar flow rate";
  lambda = (ina.m_flow*ina_X[1]/Air.data[1].MM)/(2*wcomb);
  assert(lambda >= 1, "Not enough oxygen flow");
  der(MX[1]) = ina.m_flow*ina_X[1] + out.m_flow*fluegas.X[1] - 2*wcomb*Exhaust.data[1].MM "oxygen";
  der(MX[2]) = ina.m_flow*ina_X[3] + out.m_flow*fluegas.X[2] "argon";
  der(MX[3]) = ina.m_flow*ina_X[2] + out.m_flow*fluegas.X[3] + 2*wcomb*Exhaust.data[3].MM "water";
  der(MX[4]) = inf.m_flow*inf_X[2] + out.m_flow*fluegas.X[4] + wcomb*Exhaust.data[4].MM "carbondioxide";
  der(MX[5]) = ina.m_flow*ina_X[4] + out.m_flow*fluegas.X[5] + inf.m_flow*inf_X[1] "nitrogen";
  annotation ( ... );
end CombustionChamber;
```

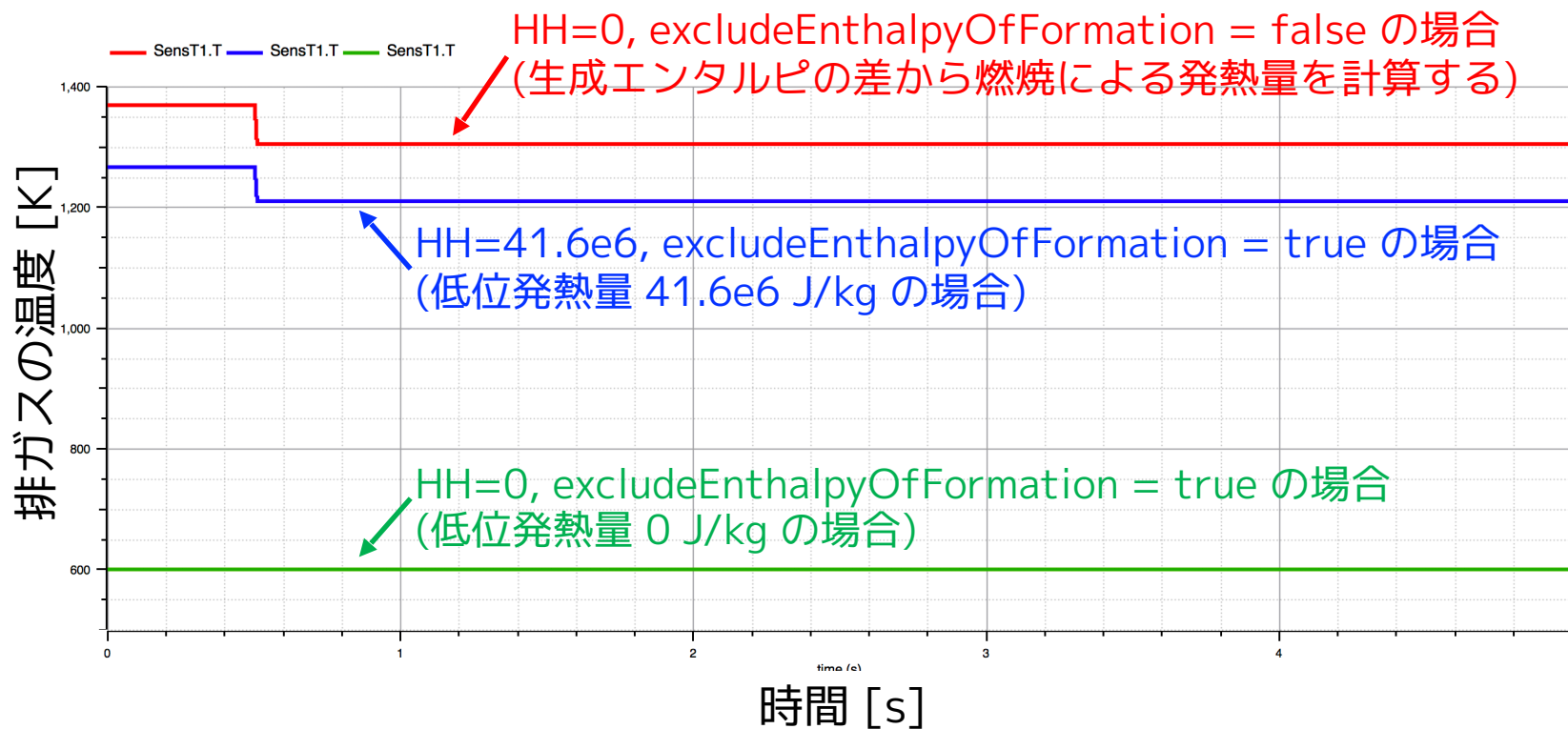
$CH_4$  の燃焼速度 (mol/s)  
(完全燃焼)

$-2O_2 + 2H_2O + CO_2$

成分物質の質量保存則



## IV 計算結果



# まとめ

MixtureGasNasa は、SingleGasNasa でモデル化される成分物質とその質量分率で定義できる混合物の理想気体モデルであり、

- Wike(1950)による方法とChungら(1984,1988)による方法の粘性率モデル
- Wassiljew(1904), Mason, Saxena (1958), Roy, Thodos(1968,1970)らによる熱伝導率モデル
- 成分気体のモデルから質量分率やモル分率を考慮して導かれた混合気体の熱物性モデルと理想気体の熱力学的関係式

などをベースとして構成される。以下のようなオプションの設定が可能である。

- reducedX, finxedX による質量分率の計算方法
- excludeEnthalpyOfFormation, referenceChoice, h\_offset による比エンタルピの計算方法

**Licensed by Amane Tanaka under the Modelica License 2**

**Copyright(c) 2018, Amane Tanaka**

- The purpose of this document is introducing the MixtureGasNasa package which is included in the Modelica Standard Library (MSL). This document uses libraries, software, figures, and documents included in MSL and ThermoPower Library, 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.

The ThermoPower package is licensed by Politecnico di Milano under the Modelica License 2.  
Copyright © 2002-2014, Politecnico di Milano.