ThermophysicalModels library in OpenFOAM-2.3.x (or 2.4.x)

How to implement a new thermophysical model

Teaching within: CFD with OpenSource software (TME050)

Isabelle Choquet
isabelle.choquet@hv.se
University West

2015-09-28

Exemples of thermodynamic and transport properties:

- density ρ $\longrightarrow$ *equation of state (Ia, see slide 4)*
- heat capacity C_v , C_p
- internal energy *e*, enthalpy *h*

$\longrightarrow$ *thermophysical properties (Ib, see slide 5)*

- diffusivity *D*,*α* …
- viscosity μ
- thermal conductivity κ
- electric conductivity

$\longrightarrow$ *transport properties (Ic, see slide 6)*

Depend on:

- temperature *T*
- pressure *P*

$\longrightarrow$ *Pure mixture (Ia-Ic, see slide 4-6)*

- fluid (possibly solid) composition $\longrightarrow$ *Mixture (II, see slide 7)*

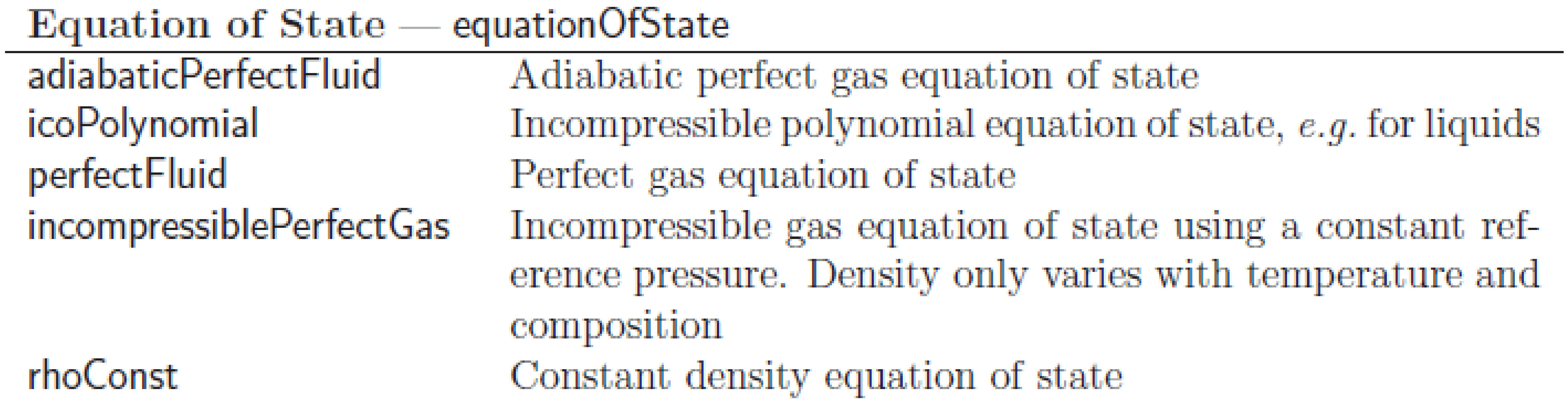
Involved in heat transfer, compressible flow, multiphase problems, combustion, etc.

Content

Slides

Models available in OpenFOAM (see UserGuide) *(1/8)*

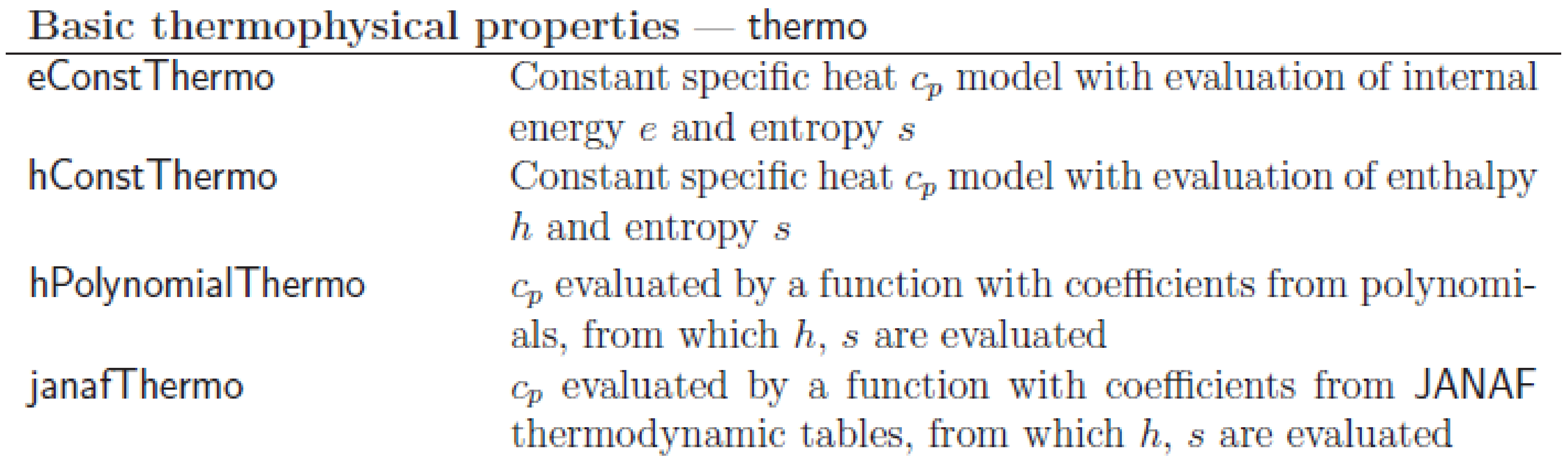
Ia



Equation of State — equationOfState	
adiabaticPerfectFluid	Adiabatic perfect gas equation of state
icoPolynomial	Incompressible polynomial equation of state, *e.g.* for liquids
perfectFluid	Perfect gas equation of state
incompressiblePerfectGas	Incompressible gas equation of state using a constant reference pressure. Density only varies with temperature and composition
rhoConst	Constant density equation of state

Models available in OpenFOAM (see UserGuide) *(2/8)*

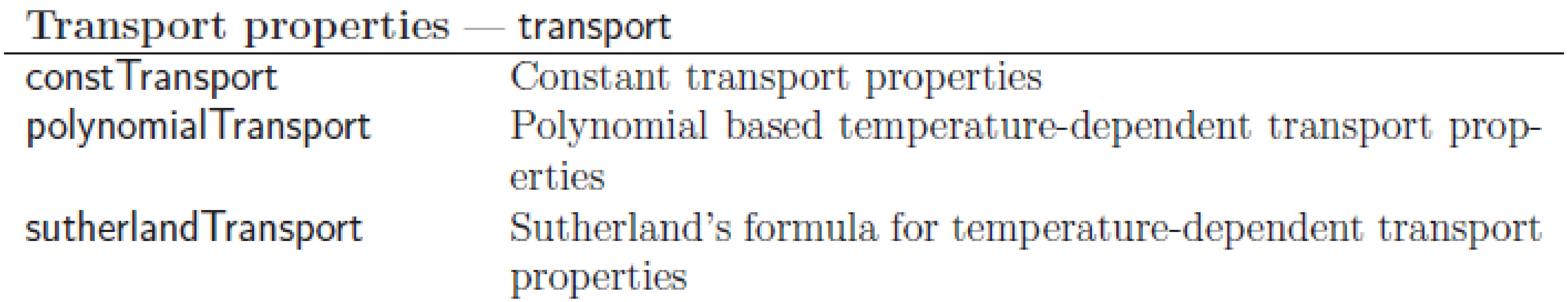
Ib



Basic thermophysical properties — thermo	
eConstThermo	Constant specific heat c_p model with evaluation of internal energy e and entropy s
hConstThermo	Constant specific heat c_p model with evaluation of enthalpy h and entropy s
hPolynomialThermo	c_p evaluated by a function with coefficients from polynomials, from which h, s are evaluated
janafThermo	c_p evaluated by a function with coefficients from JANAF thermodynamic tables, from which h, s are evaluated

Models available in OpenFOAM (see UserGuide) *(3/8)*

Ic



Transport properties — transport	
constTransport	Constant transport properties
polynomialTransport	Polynomial based temperature-dependent transport properties
sutherlandTransport	Sutherland's formula for temperature-dependent transport properties

Models available in OpenFOAM (see UserGuide) *(4/8)*

II

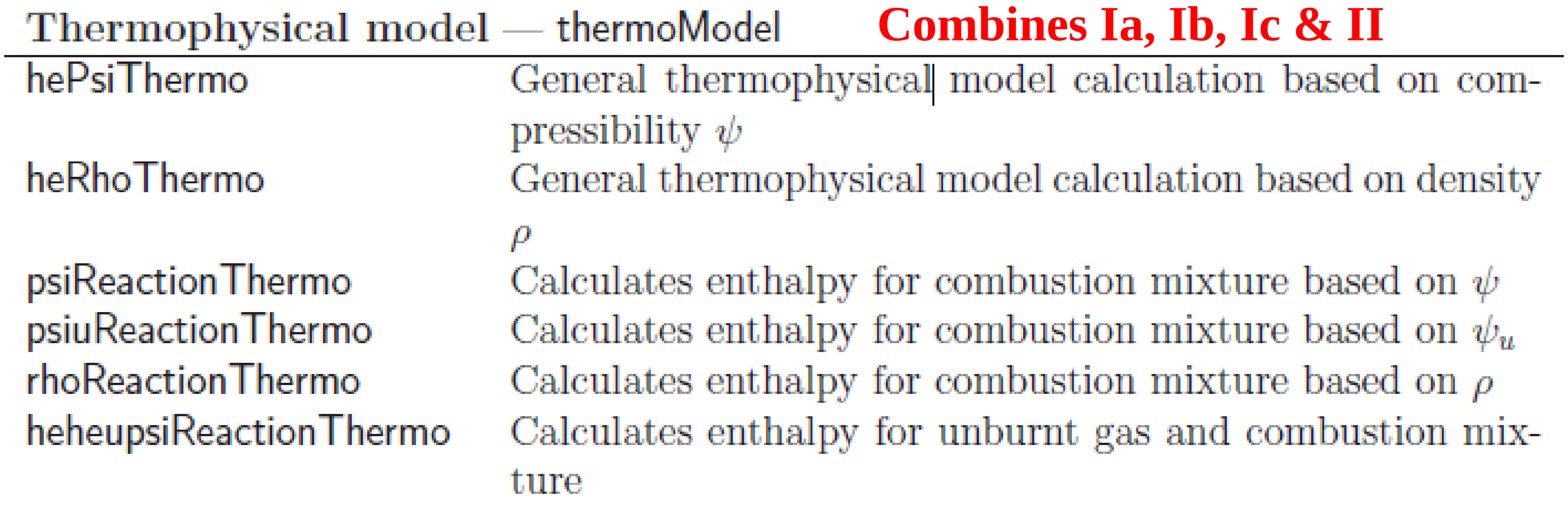
Mixture properties — mixture	
pureMixture	General thermophysical model calculation for passive gas mixtures
homogeneousMixture	Combustion mixture based on normalised fuel mass fraction b
inhomogeneousMixture	Combustion mixture based on b and total fuel mass fraction f_t
veryInhomogeneousMixture	Combustion mixture based on b, f_t and unburnt fuel mass fraction f_u
basicMultiComponentMixture	Basic mixture based on multiple components
multiComponentMixture	Derived mixture based on multiple components
reactingMixture	Combustion mixture using thermodynamics and reaction schemes
egrMixture	Exhaust gas recirculation mixture
singleStepReactingMixture	Single step reacting mixture

$\rightarrow$ **Ia, Ib & Ic**

$\rightarrow$ **Ia, Ib & Ic** for all species **&** suited mixing rules

Models available in OpenFOAM (see UserGuide) *(5/8)*

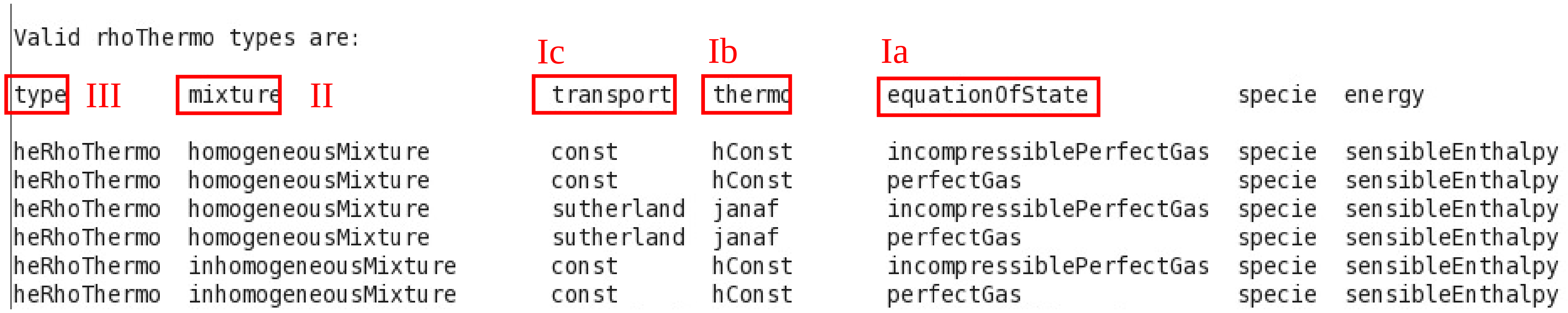
III **Combines Ia, Ib, Ic & II**



Thermophysical model — thermoModel	
hePsiThermo	General thermophysical model calculation based on compressibility ψ
heRhoThermo	General thermophysical model calculation based on density ρ
psiReactionThermo	Calculates enthalpy for combustion mixture based on ψ
psiuReactionThermo	Calculates enthalpy for combustion mixture based on ψ_u
rhoReactionThermo	Calculates enthalpy for combustion mixture based on ρ
heheupsiReactionThermo	Calculates enthalpy for unburnt gas and combustion mixture

Models available in OpenFOAM (for a given solver) *(6/8)*

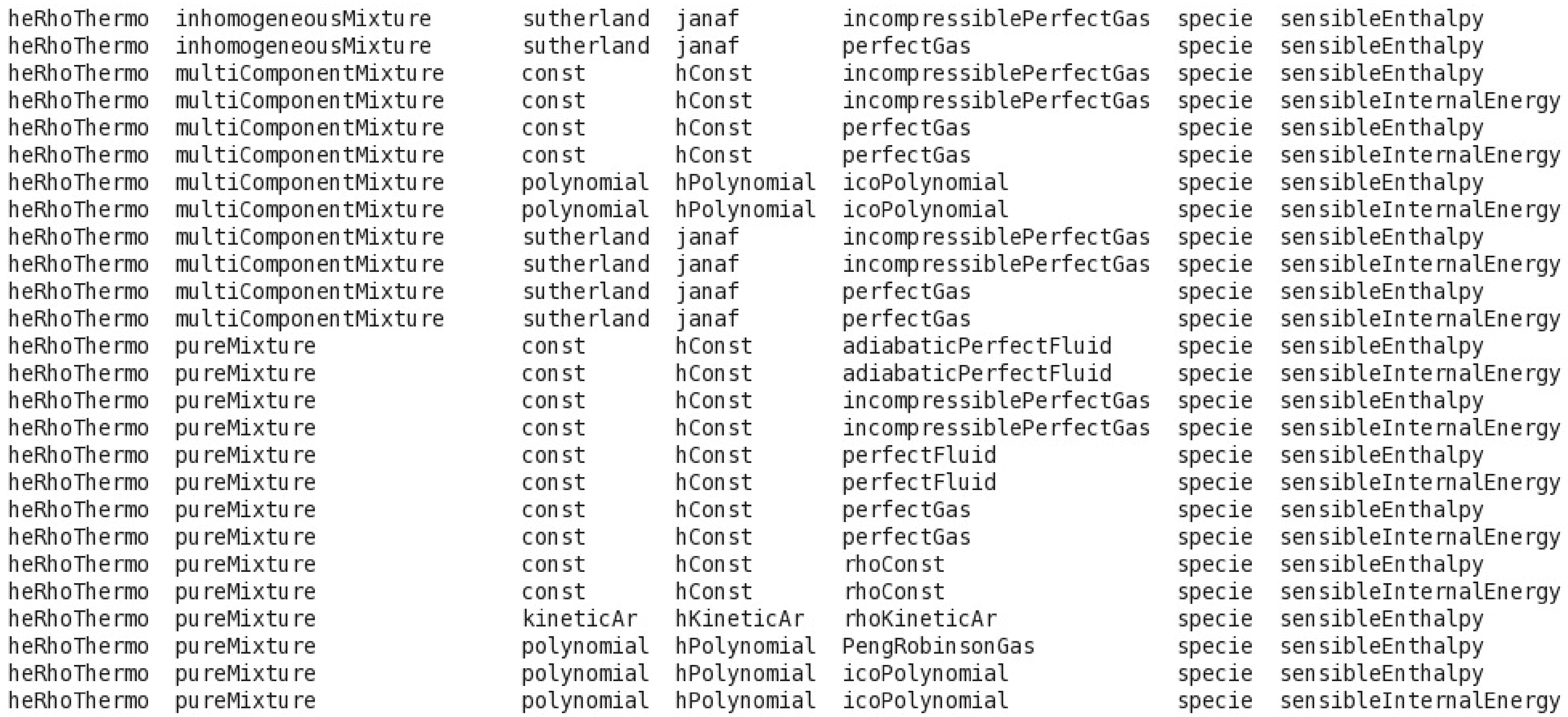
Example: go to ~~tuturials~~/heatTransfer/buoyantSimpleFoam/buoyantCavity/constant
in thermophysicalProperties change ”transport const” to ”transport dummy”
and run the solver buoyantSimpleFoam;
it returns the list of thermophysical models available for this solver:



```
Valid rhoThermo types are:

type        mixture                  transport   thermo   equationOfState            specie  energy

heRhoThermo homogeneousMixture       const       hConst   incompressiblePerfectGas   specie  sensibleEnthalpy
heRhoThermo homogeneousMixture       const       hConst   perfectGas                 specie  sensibleEnthalpy
heRhoThermo homogeneousMixture       sutherland  janaf    incompressiblePerfectGas   specie  sensibleEnthalpy
heRhoThermo homogeneousMixture       sutherland  janaf    perfectGas                 specie  sensibleEnthalpy
heRhoThermo inhomogeneousMixture     const       hConst   incompressiblePerfectGas   specie  sensibleEnthalpy
heRhoThermo inhomogeneousMixture     const       hConst   perfectGas                 specie  sensibleEnthalpy
```

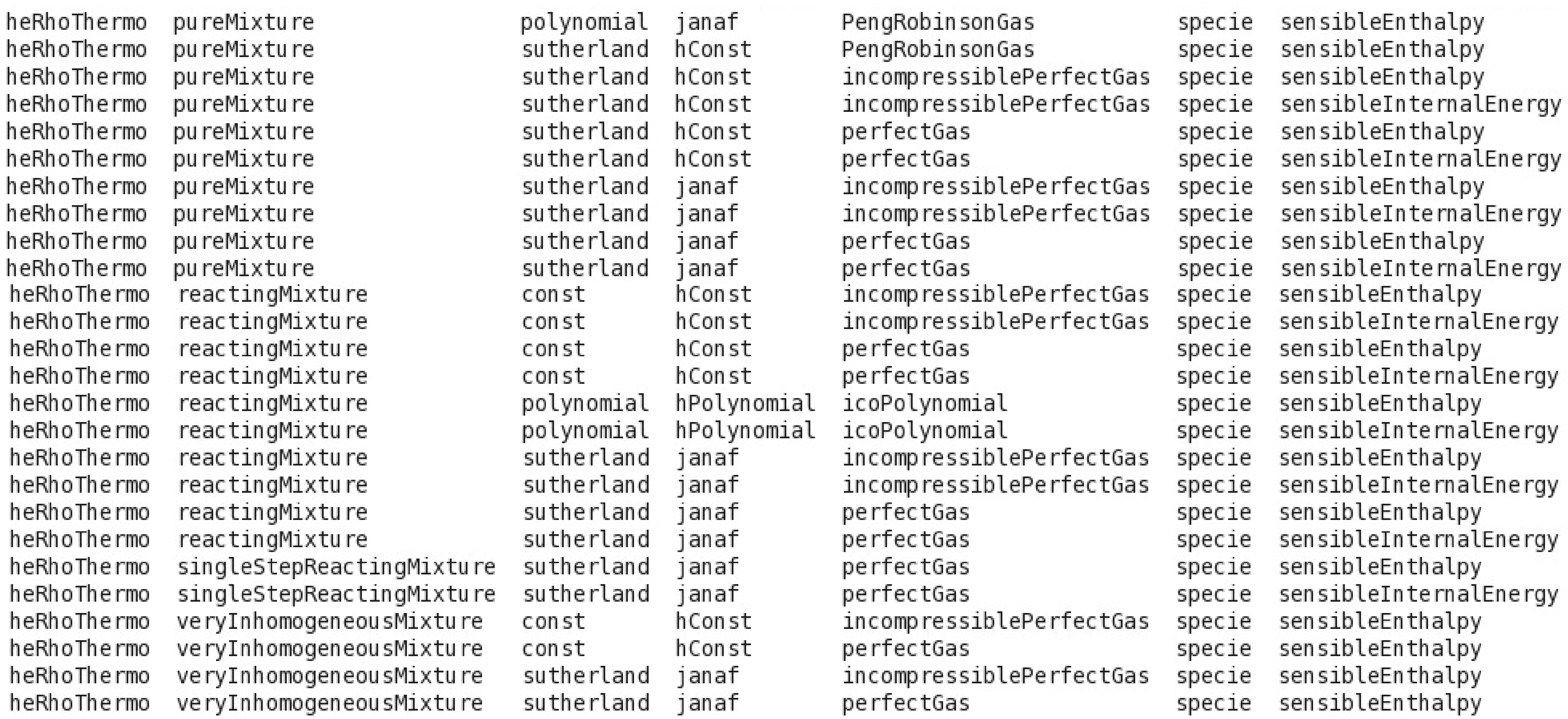
… *(+ 2 following slides*)

Models available in OpenFOAM (for a given solver) *(7/8)*



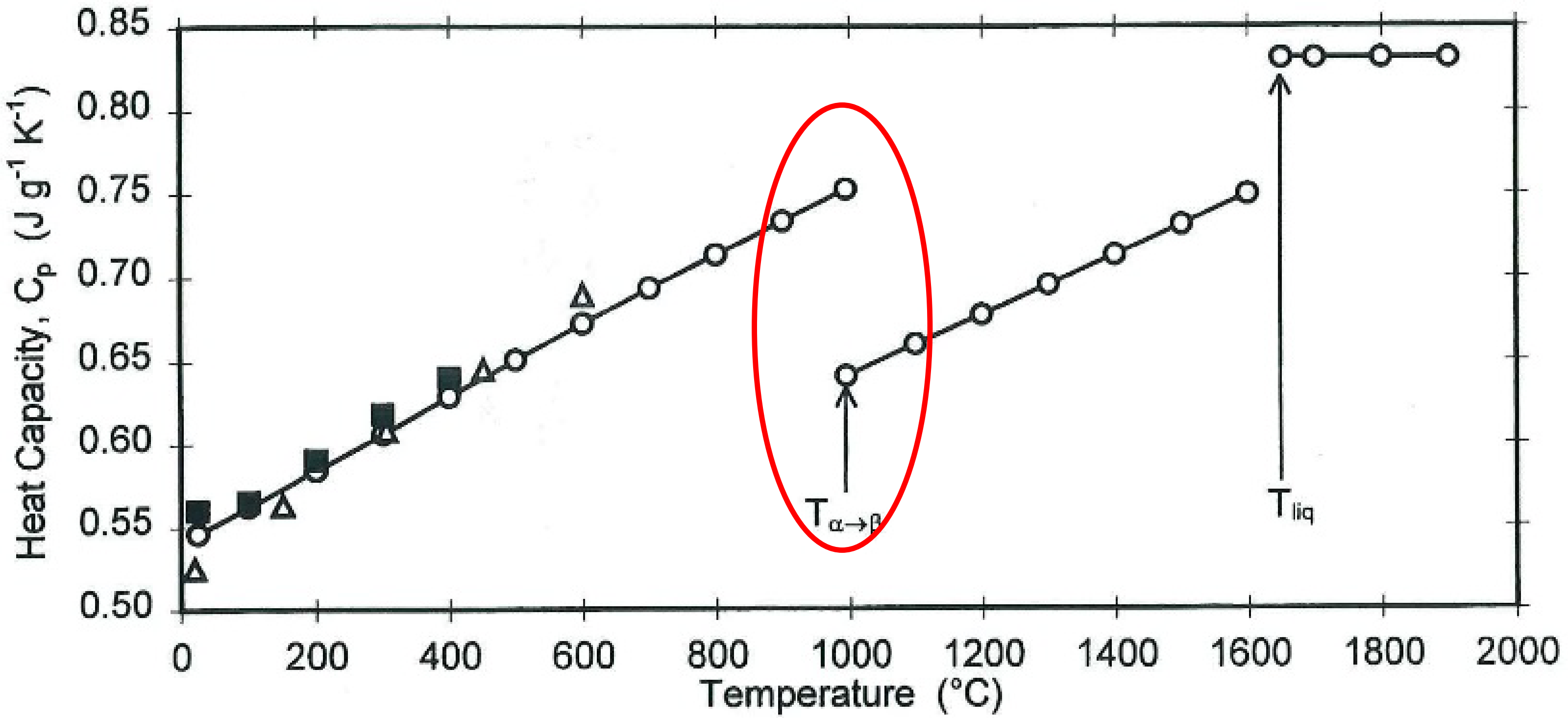
```
heRhoThermo  inhomogeneousMixture       sutherland  janaf        incompressiblePerfectGas  specie  sensibleEnthalpy
heRhoThermo  inhomogeneousMixture       sutherland  janaf        perfectGas                specie  sensibleEnthalpy
heRhoThermo  multiComponentMixture      const       hConst       incompressiblePerfectGas  specie  sensibleEnthalpy
heRhoThermo  multiComponentMixture      const       hConst       incompressiblePerfectGas  specie  sensibleInternalEnergy
heRhoThermo  multiComponentMixture      const       hConst       perfectGas                specie  sensibleEnthalpy
heRhoThermo  multiComponentMixture      const       hConst       perfectGas                specie  sensibleInternalEnergy
heRhoThermo  multiComponentMixture      polynomial  hPolynomial  icoPolynomial             specie  sensibleEnthalpy
heRhoThermo  multiComponentMixture      polynomial  hPolynomial  icoPolynomial             specie  sensibleInternalEnergy
heRhoThermo  multiComponentMixture      sutherland  janaf        incompressiblePerfectGas  specie  sensibleEnthalpy
heRhoThermo  multiComponentMixture      sutherland  janaf        incompressiblePerfectGas  specie  sensibleInternalEnergy
heRhoThermo  multiComponentMixture      sutherland  janaf        perfectGas                specie  sensibleEnthalpy
heRhoThermo  multiComponentMixture      sutherland  janaf        perfectGas                specie  sensibleInternalEnergy
heRhoThermo  pureMixture                const       hConst       adiabaticPerfectFluid     specie  sensibleEnthalpy
heRhoThermo  pureMixture                const       hConst       adiabaticPerfectFluid     specie  sensibleInternalEnergy
heRhoThermo  pureMixture                const       hConst       incompressiblePerfectGas  specie  sensibleEnthalpy
heRhoThermo  pureMixture                const       hConst       incompressiblePerfectGas  specie  sensibleInternalEnergy
heRhoThermo  pureMixture                const       hConst       perfectFluid              specie  sensibleEnthalpy
heRhoThermo  pureMixture                const       hConst       perfectFluid              specie  sensibleInternalEnergy
heRhoThermo  pureMixture                const       hConst       perfectGas                specie  sensibleEnthalpy
heRhoThermo  pureMixture                const       hConst       perfectGas                specie  sensibleInternalEnergy
heRhoThermo  pureMixture                const       hConst       rhoConst                  specie  sensibleEnthalpy
heRhoThermo  pureMixture                const       hConst       rhoConst                  specie  sensibleInternalEnergy
heRhoThermo  pureMixture                kineticAr   hKineticAr   rhoKineticAr              specie  sensibleEnthalpy
heRhoThermo  pureMixture                polynomial  hPolynomial  PengRobinsonGas           specie  sensibleEnthalpy
heRhoThermo  pureMixture                polynomial  hPolynomial  icoPolynomial             specie  sensibleEnthalpy
heRhoThermo  pureMixture                polynomial  hPolynomial  icoPolynomial             specie  sensibleInternalEnergy
```

Models available in OpenFOAM (for a given solver) *(8/8)*



```
heRhoThermo  pureMixture                 polynomial  janaf        PengRobinsonGas            specie  sensibleEnthalpy
heRhoThermo  pureMixture                 sutherland  hConst       PengRobinsonGas            specie  sensibleEnthalpy
heRhoThermo  pureMixture                 sutherland  hConst       incompressiblePerfectGas   specie  sensibleEnthalpy
heRhoThermo  pureMixture                 sutherland  hConst       incompressiblePerfectGas   specie  sensibleInternalEnergy
heRhoThermo  pureMixture                 sutherland  hConst       perfectGas                 specie  sensibleEnthalpy
heRhoThermo  pureMixture                 sutherland  hConst       perfectGas                 specie  sensibleInternalEnergy
heRhoThermo  pureMixture                 sutherland  janaf        incompressiblePerfectGas   specie  sensibleEnthalpy
heRhoThermo  pureMixture                 sutherland  janaf        incompressiblePerfectGas   specie  sensibleInternalEnergy
heRhoThermo  pureMixture                 sutherland  janaf        perfectGas                 specie  sensibleEnthalpy
heRhoThermo  pureMixture                 sutherland  janaf        perfectGas                 specie  sensibleInternalEnergy
heRhoThermo  reactingMixture             const       hConst       incompressiblePerfectGas   specie  sensibleEnthalpy
heRhoThermo  reactingMixture             const       hConst       incompressiblePerfectGas   specie  sensibleInternalEnergy
heRhoThermo  reactingMixture             const       hConst       perfectGas                 specie  sensibleEnthalpy
heRhoThermo  reactingMixture             const       hConst       perfectGas                 specie  sensibleInternalEnergy
heRhoThermo  reactingMixture             polynomial  hPolynomial  icoPolynomial              specie  sensibleEnthalpy
heRhoThermo  reactingMixture             polynomial  hPolynomial  icoPolynomial              specie  sensibleInternalEnergy
heRhoThermo  reactingMixture             sutherland  janaf        incompressiblePerfectGas   specie  sensibleEnthalpy
heRhoThermo  reactingMixture             sutherland  janaf        incompressiblePerfectGas   specie  sensibleInternalEnergy
heRhoThermo  reactingMixture             sutherland  janaf        perfectGas                 specie  sensibleEnthalpy
heRhoThermo  reactingMixture             sutherland  janaf        perfectGas                 specie  sensibleInternalEnergy
heRhoThermo  singleStepReactingMixture   sutherland  janaf        perfectGas                 specie  sensibleEnthalpy
heRhoThermo  singleStepReactingMixture   sutherland  janaf        perfectGas                 specie  sensibleInternalEnergy
heRhoThermo  veryInhomogeneousMixture    const       hConst       incompressiblePerfectGas   specie  sensibleEnthalpy
heRhoThermo  veryInhomogeneousMixture    const       hConst       perfectGas                 specie  sensibleEnthalpy
heRhoThermo  veryInhomogeneousMixture    sutherland  janaf        incompressiblePerfectGas   specie  sensibleEnthalpy
heRhoThermo  veryInhomogeneousMixture    sutherland  janaf        perfectGas                 specie  sensibleEnthalpy
```

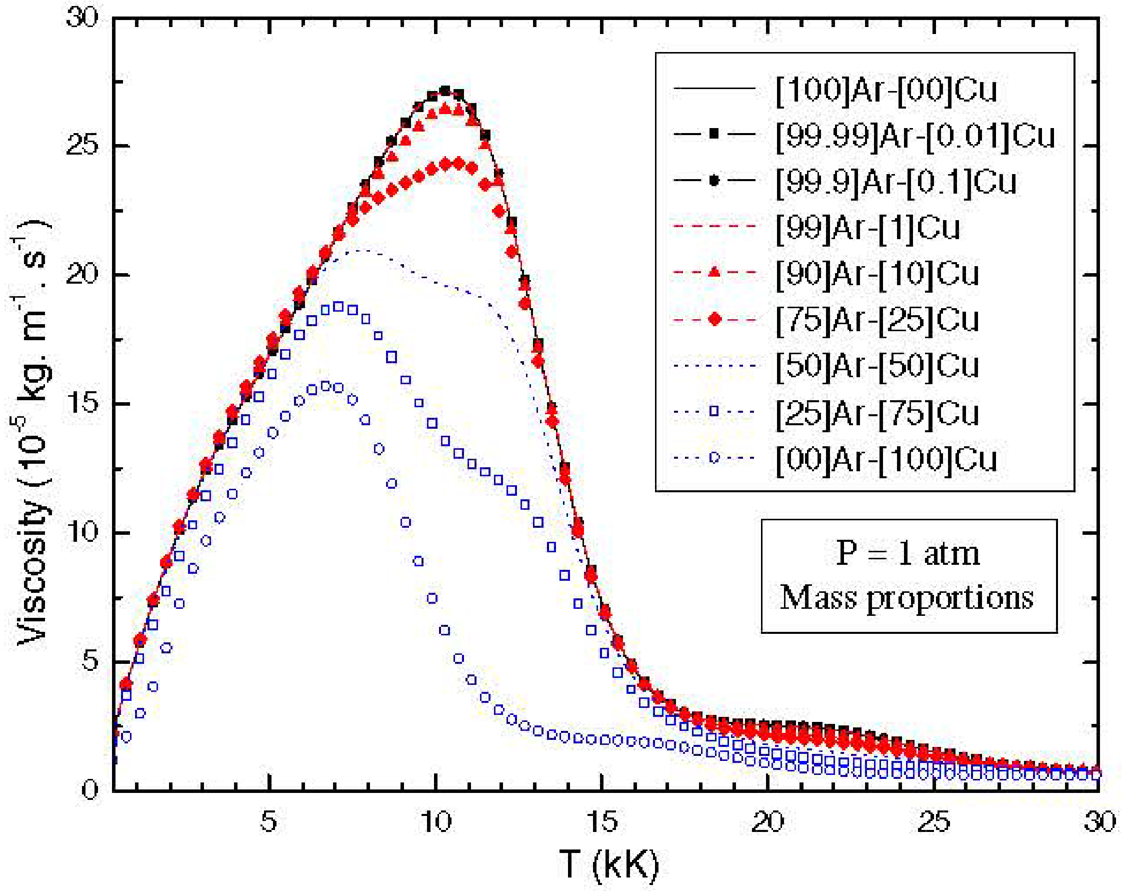
Other example of thermophysical model : *phase change in solid state* *(1/3)*



Heat capacity of Ti/6Al/4V as a function of temperature; ——, o, recommended values; Δ, Bros [3]; ■, Richardson [4].

From "Recommended values of thermophysical properties for selected commetcial alloys", Kenneth C Mills, ASM International, ISBN 0-87170-753-5, (page 213)

Other example of thermophysical model : *thermal plasma* *(2/3)*



. Viscosity of Ar–Cu mixtures at atmospheric pressure.

From “Thermal plasma properties for Ar–Cu, Ar–Fe and Ar–Al mixtures used in welding plasmas processes: II. Transport coefficients at atmospheric pressure”, Y Cressault, A B Murphy, Ph Teulet, A Gleizes and M Schnick, J. Phys. D: Appl. Phys. **46** (2013) 415207

Other example of thermophysical model : *thermal plasma* *(3/3)*

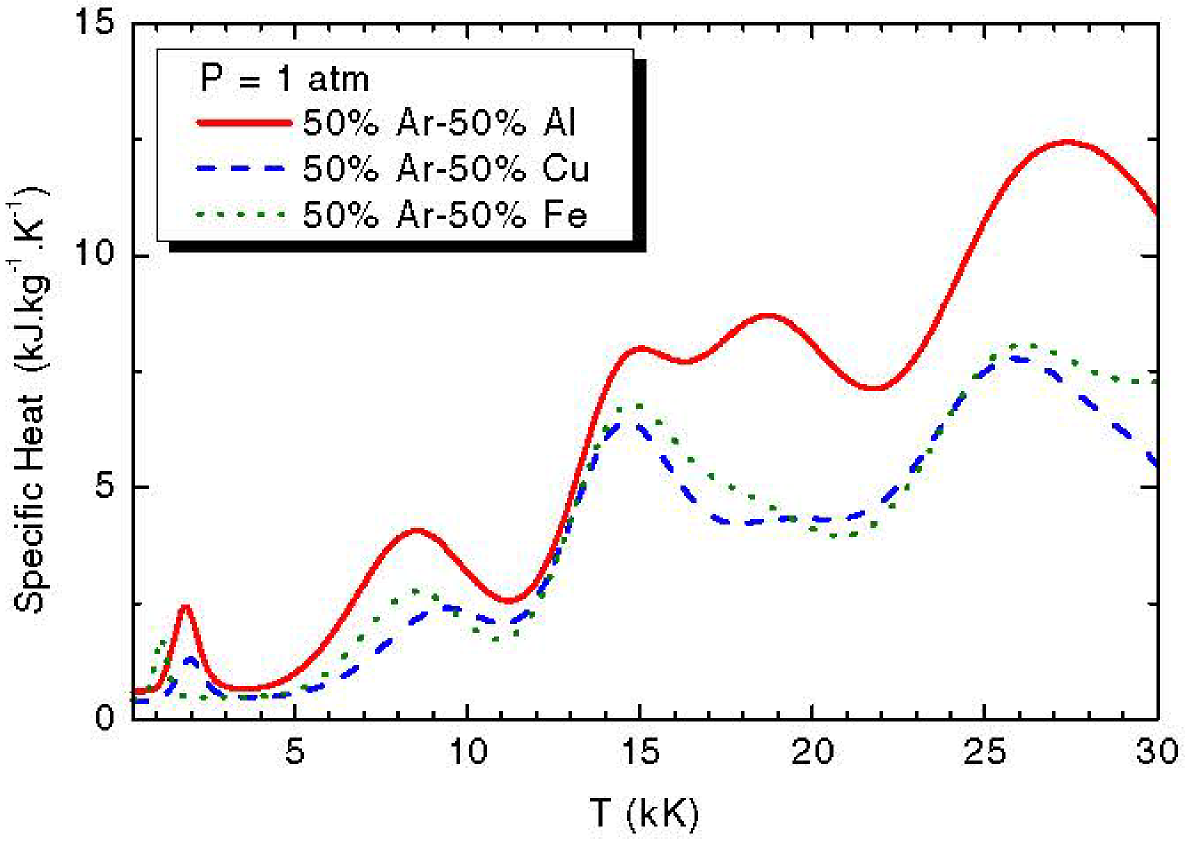


Figure 1. Specific heat at constant pressure for 50%Ar–50% metal vapour mixtures by mass at atmospheric pressure.

From "Thermal plasma properties for Ar–Cu, Ar–Fe and Ar–Al mixtures used in welding plasmas processes: II. Transport coefficients at atmospheric pressure", Y Cressault, A B Murphy, Ph Teulet, A Gleizes and M Schnick, J. Phys. D: Appl. Phys. **46** (2013) 415207

Example of problem needing a new thermophysical model : *(1/16)*

thermal conduction in a high temperature argon gas

Governing equation

$$\frac{\partial(\rho\, C_v T)}{\partial t} = \nabla(\kappa\, \nabla . T)$$

with ρ, C_v and κ function of T obtained from kinetic theory for $T \in [200; 20000]K$

Rk: need a heatTransfer solver

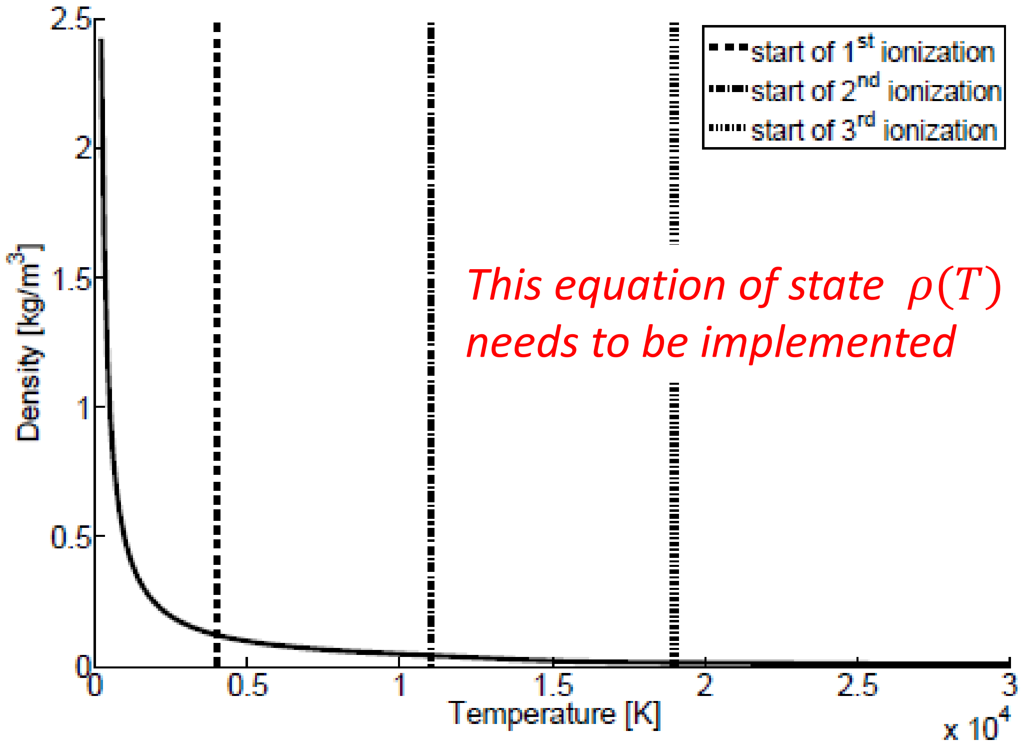


Figure 2.2: Density of an argon gas versus temperature .

From "Plasma arc welding simulation with openFOAM", M. Sass-Tisovskaya, Lic. thesis, Dept. Applied Mechanics, Chalmers University of Technology, 2009 (page 30)

Example of problem needing a new thermophysical model : *(2/16)*

thermal conduction in a high temperature argon gas

This thermodynamic property $C_p(T)$ needs to be implemented

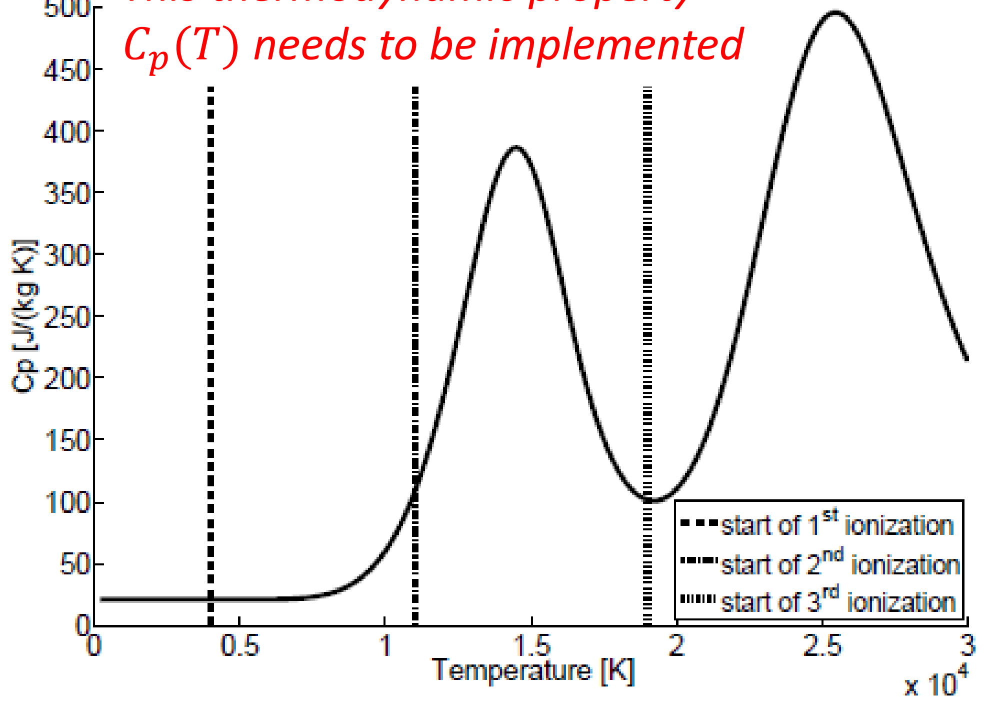


Figure 2.5: Heat capacity at constant pressure of an argon gas versus temperature.

This transport property $\kappa(T)$ needs to be implemented

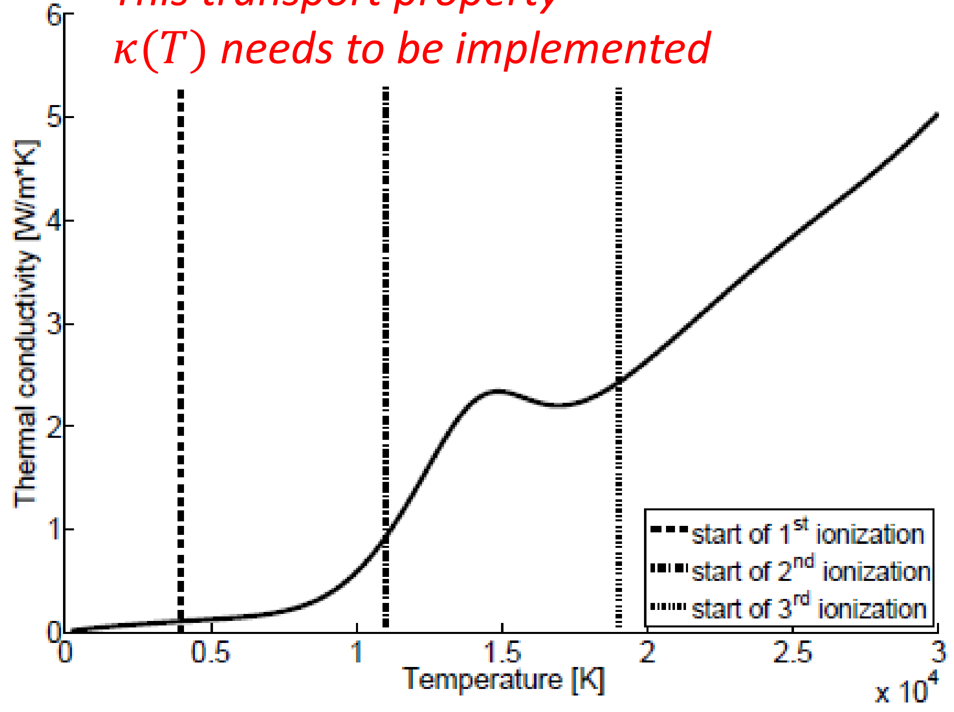


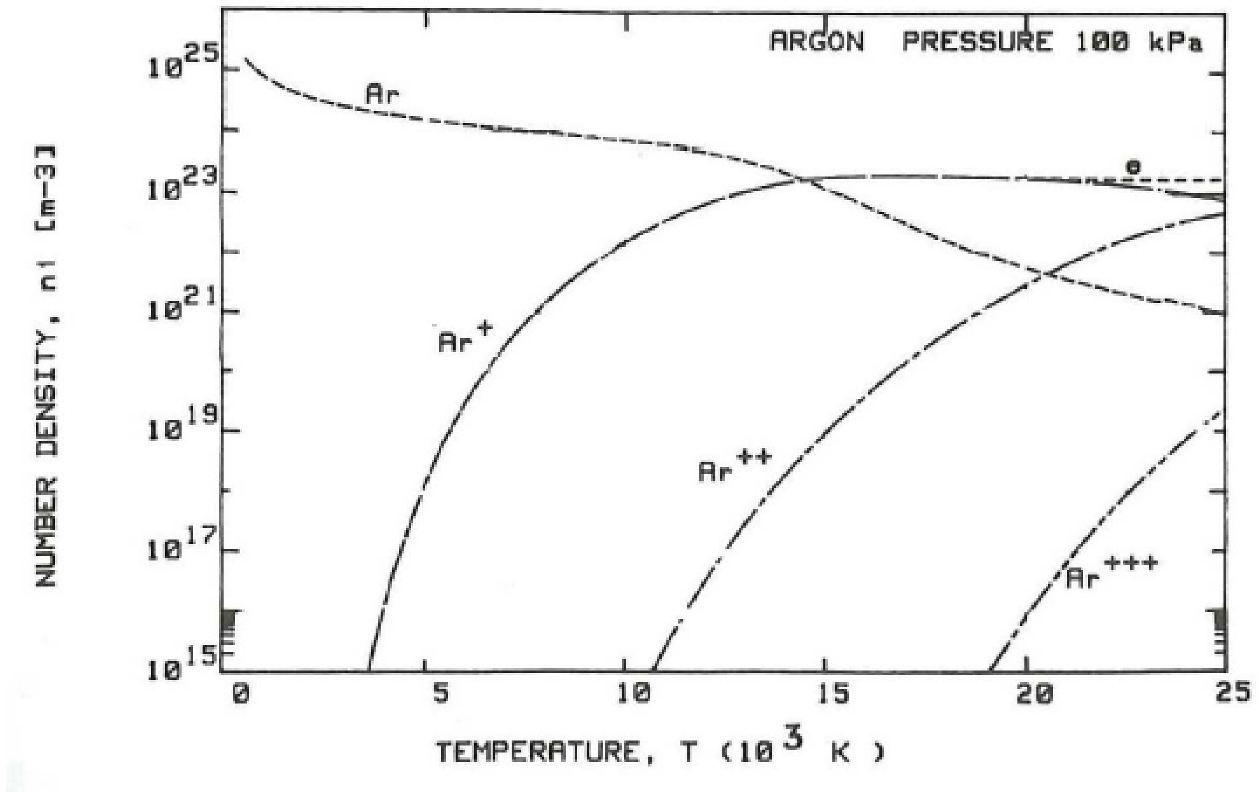
Figure 2.4: Thermal conductivity of an argon gas versus temperature.

From "Plasma arc welding simulation with openFOAM", M. Sass-Tisovskaya, Lic. thesis, Dept. Applied Mechanics, Chalmers University of Technology, 2009 (page 31)

Example of problem needing a new thermophysical model : *(3/16)*

thermal conduction in a high temperature argon gas

Why these peaks in C_p and κ ?
Because the fluid is not a pure substance: its composition changes with T.



Temperature dependence of the equilibrium composition (species number densities) of an argon plasma at atmospheric pressure (starting from one mole of Ar at room temperature [21].)

From "Thermal Plasmas, Fundamentals and Applications", Boulos M.I., Fauchais P. and Pfender E.: Vol. 1, Plenum Press, New York, 1994 (page 235)

Example of problem needing a new thermophysical model : *(4/16)*

thermal conduction in a high temperature argon gas

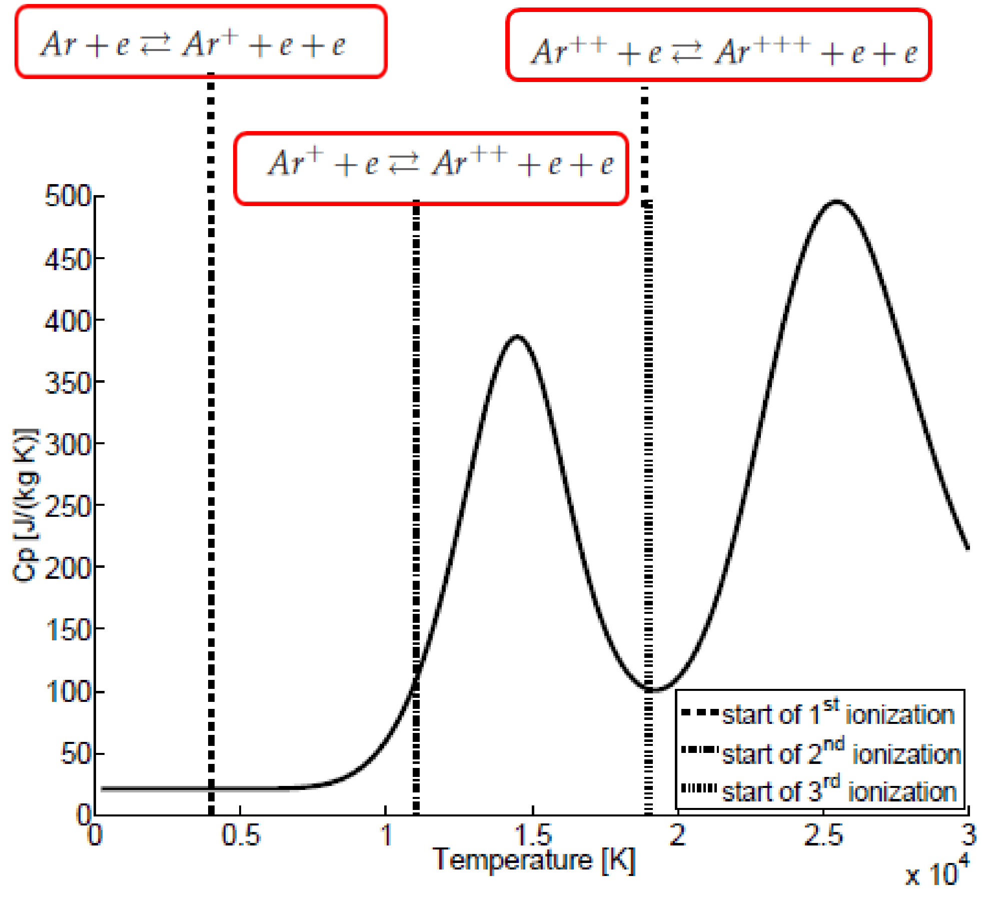


Figure 2.5: Heat capacity at constant pressure of an argon gas versus temperature.

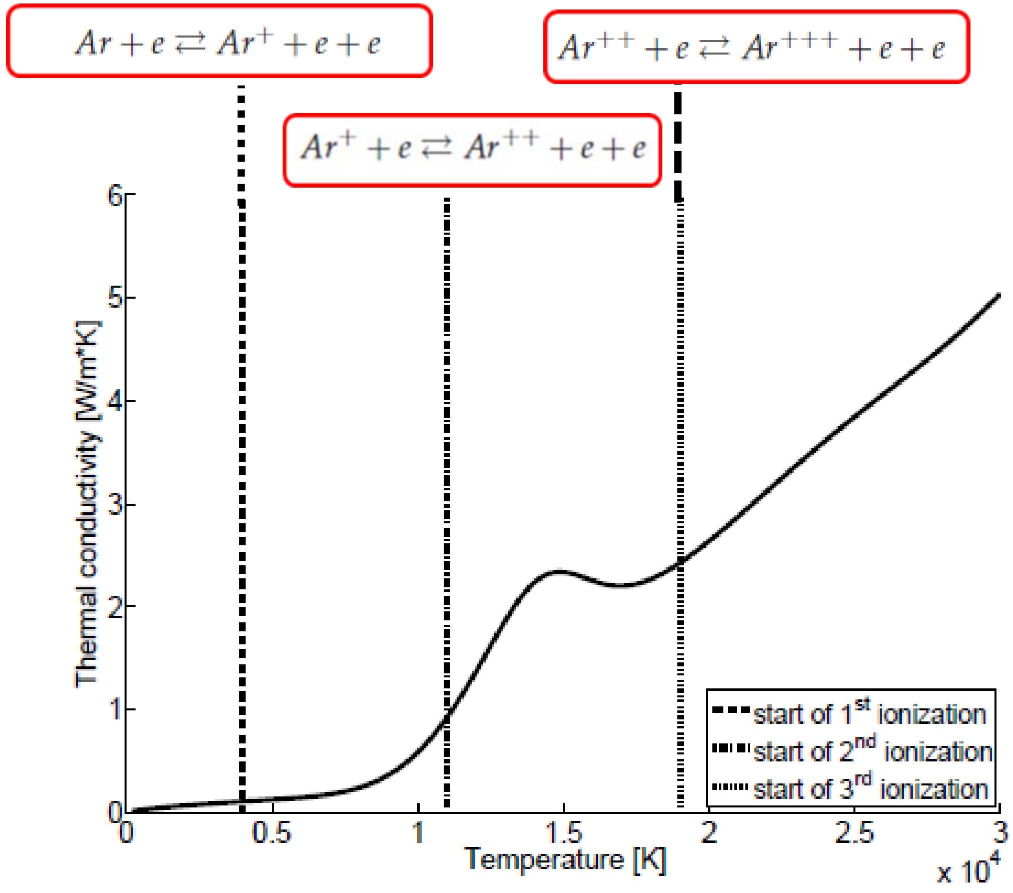
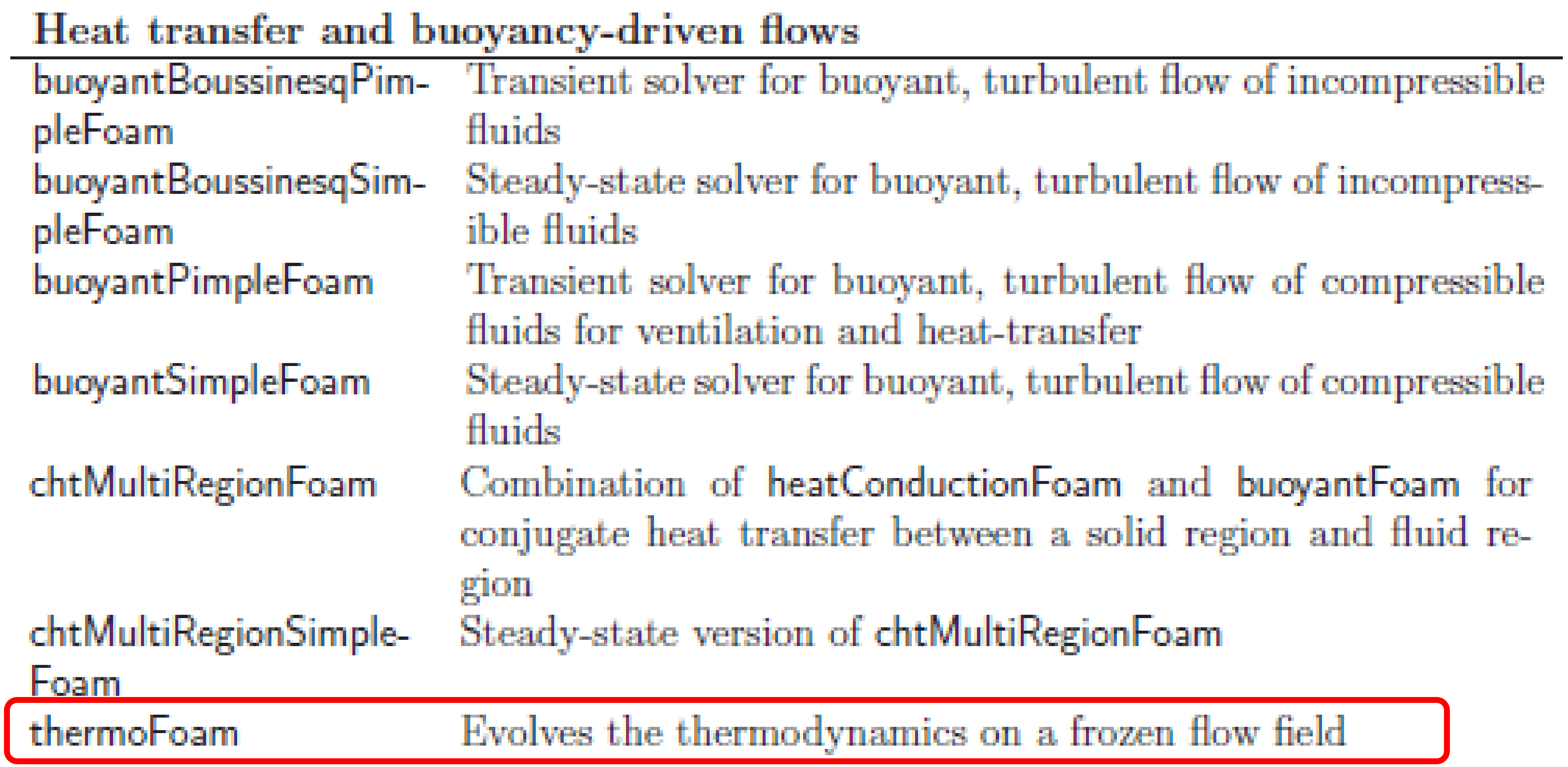


Figure 2.4: Thermal conductivity of an argon gas versus temperature.

From "Plasma arc welding simulation with openFOAM", M. Sass-Tisovskaya,
Lic. thesis, Dept. Applied Mechanics, Chalmers University of Technology, 2009 (page 31)

Heat transfer solvers in OpenFOAM: (see UserGuide) : *(5/16)*



Heat transfer and buoyancy-driven flows

buoyantBoussinesqPimpleFoam	Transient solver for buoyant, turbulent flow of incompressible fluids
buoyantBoussinesqSimpleFoam	Steady-state solver for buoyant, turbulent flow of incompressible fluids
buoyantPimpleFoam	Transient solver for buoyant, turbulent flow of compressible fluids for ventilation and heat-transfer
buoyantSimpleFoam	Steady-state solver for buoyant, turbulent flow of compressible fluids
chtMultiRegionFoam	Combination of heatConductionFoam and buoyantFoam for conjugate heat transfer between a solid region and fluid region
chtMultiRegionSimpleFoam	Steady-state version of chtMultiRegionFoam
thermoFoam	Evolves the thermodynamics on a frozen flow field

OpenFOAM solver heatTransfer/thermoFoam: *(6/16)*

- **Copy the solver in your user directory and rename it**:

```
cd $WM_PROJECT_DIR
cp -r --parents applications/solvers/heatTransfer/thermoFoam $WM_PROJECT_USER_DIR
cd $WM_PROJECT_USER_DIR/applications/solvers/heatTransfer
mv thermoFoam myThermoFoam
cd myThermoFoam
mv thermoFoam.C myThermoFoam.C
```

- **Modify Make/files to**

```
myThermoFoam.C
EXE = $(FOAM_USER_APPBIN)/myThermoFoam
```

- **Clean and compile**

```
 wclean
 rm -r Make/linux*
 wmake
```

(to clean also the Debug version in the system)

OpenFOAM solver heatTransfer/thermoFoam: *(7/16)*

Parts of thermoFoam involving a thermophysical model

```
createFields.H
EEqn.H
Make
myThermoFoam.C
setAlphaEff.H
```

```
Info<< "Reading thermophysical properties\n" << endl;

autoPtr<rhoThermo> pThermo(rhoThermo::New(mesh));
rhoThermo& thermo = pThermo();
thermo.validate(args.executable(), "h", "e");

volScalarField rho
(
  ...
```

Create the object "thermo"

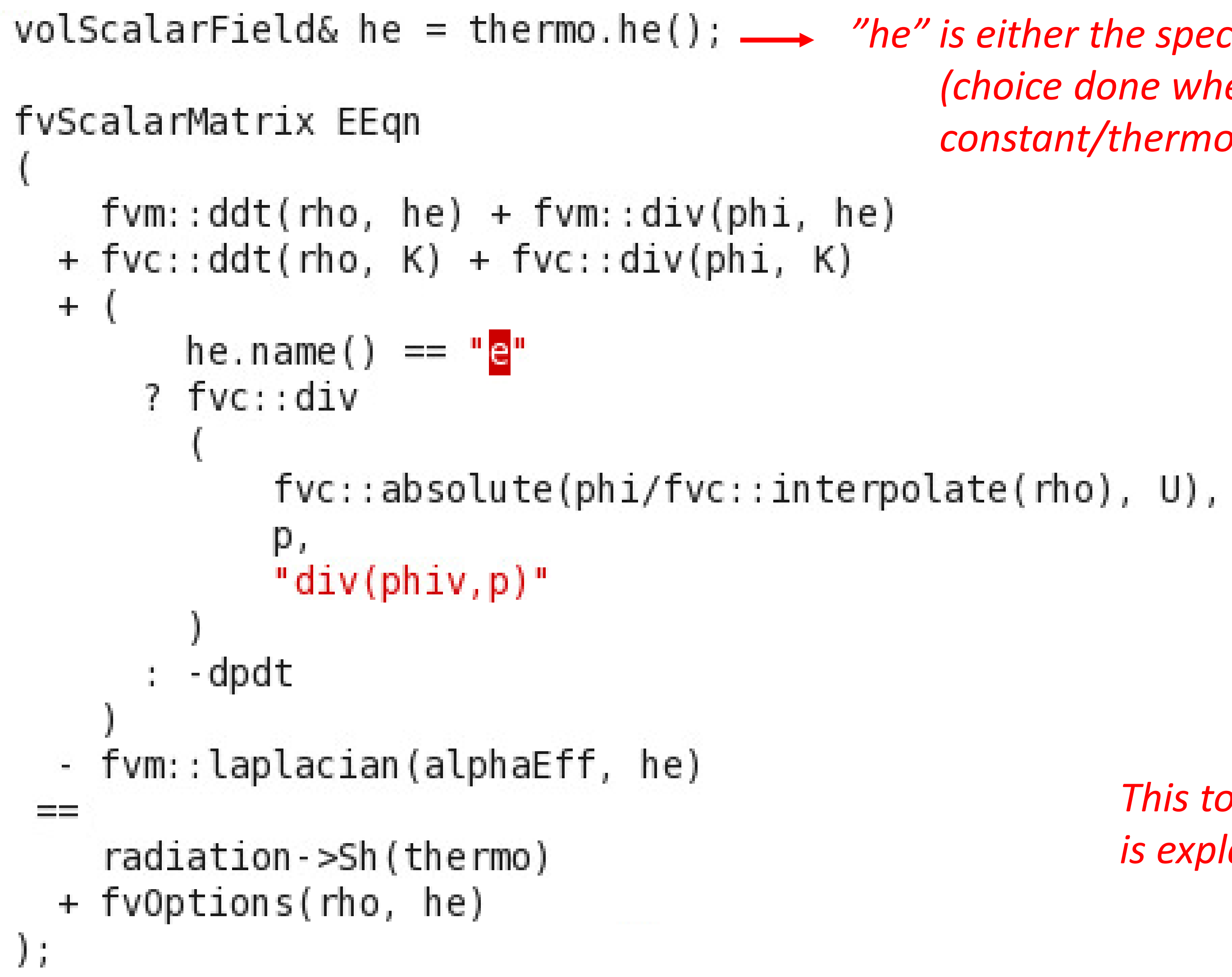
```
...
#include "rhoThermo.H"
...
#include "EEqn.H"
...
```

Include a thermophysic library; that object is used on the next slide.

Include an energy conservation equation (see next slide)

OpenFOAM solver heatTransfer/thermoFoam:

Part of the energy conservation equation EEqn.H



```
volScalarField& he = thermo.he();

fvScalarMatrix EEqn
(
    fvm::ddt(rho, he) + fvm::div(phi, he)
  + fvc::ddt(rho, K) + fvc::div(phi, K)
  + (
        he.name() == "e"
      ? fvc::div
        (
            fvc::absolute(phi/fvc::interpolate(rho), U),
            p,
            "div(phiv,p)"
        )
      : -dpdt
    )
  - fvm::laplacian(alphaEff, he)
 ==
    radiation->Sh(thermo)
  + fvOptions(rho, he)
);
```

”he” is either the specific internal energy or the specific enthalpy (choice done when preparing a case, in the dictionary constant/thermophysicalProperties)

This total energy conservation equation is explained in the next slides

Energy conservation -total energy of flowing fluid: $\rho(\hat{h} + K)$

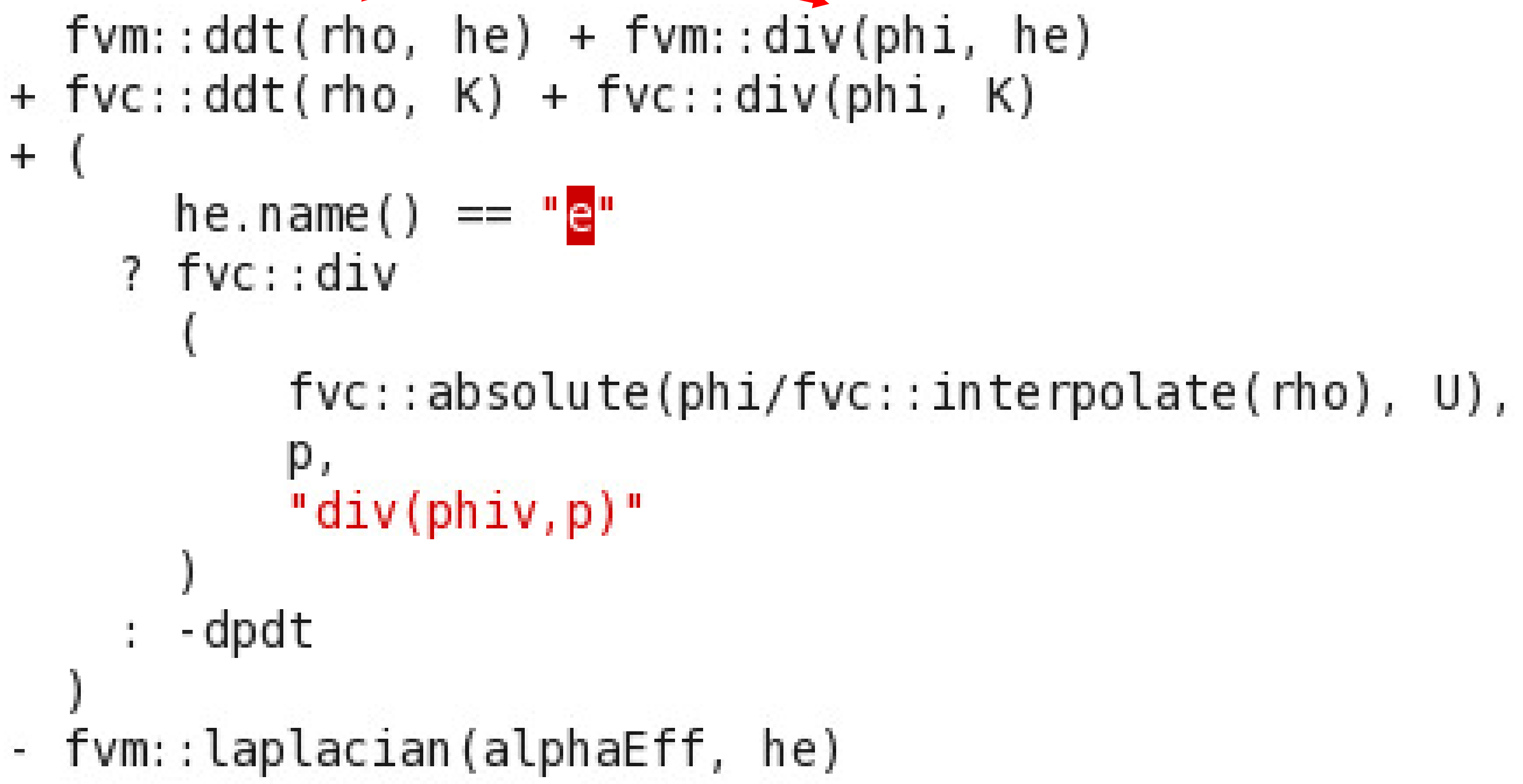
$$\frac{\partial}{\partial t}\Big(\boxed{\rho(\hat{h}} + K)\Big) + \nabla \cdot \Big(\boxed{\rho \boldsymbol{v}(\hat{h}} + K)\Big) - \frac{\partial p}{\partial t} - \nabla \cdot q = 0 \quad \text{where} \quad q = \alpha \nabla \hat{h}$$

Enthalpy : $\hat{h} = \hat{e} + \frac{p}{\rho}$

Kinetic energy: *K*

Thermal diffusivity : α

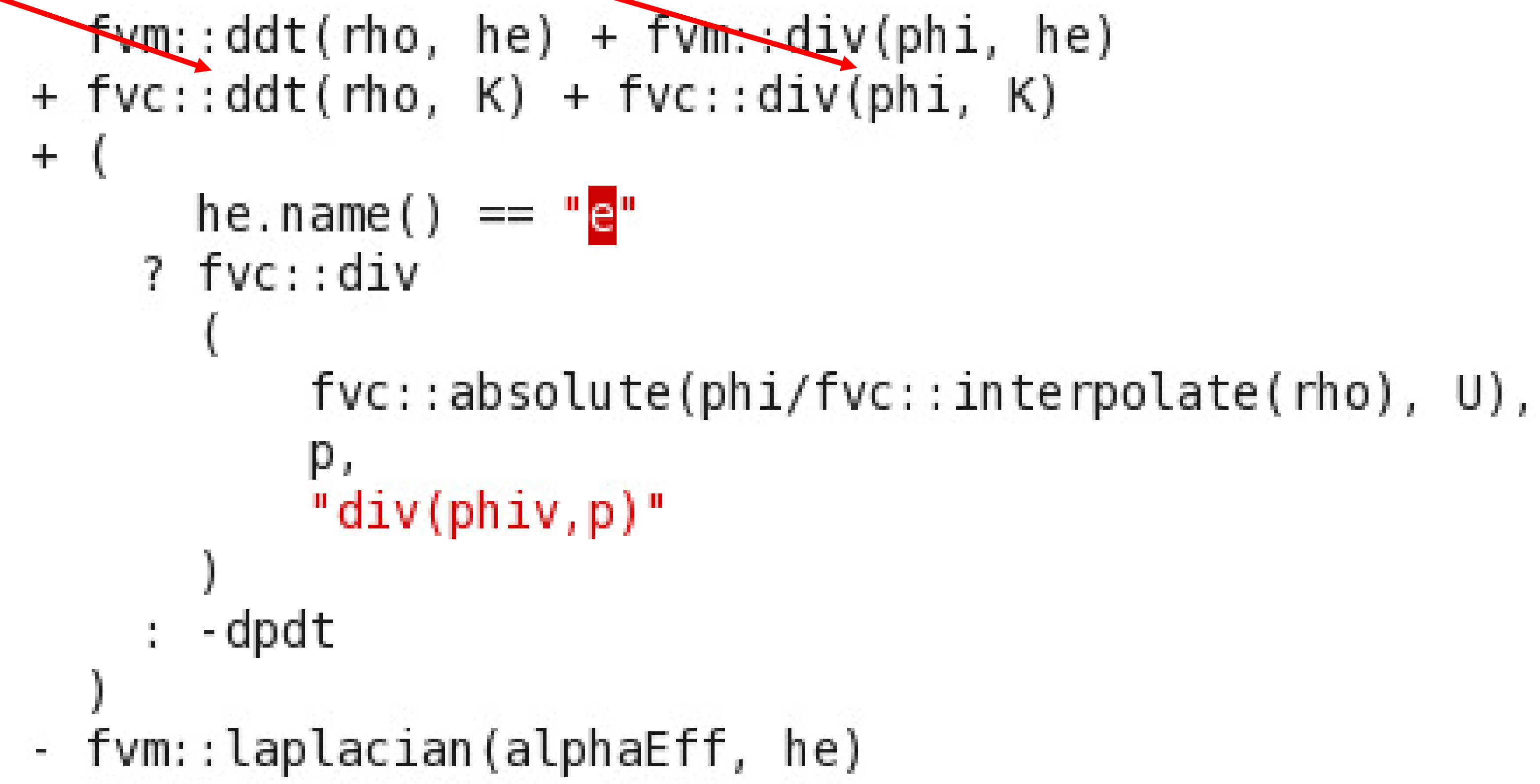
specific enthalpy $\hat{h}$ *or specific internal energy* $\hat{e}$


```
  fvm::ddt(rho, he) + fvm::div(phi, he)
+ fvc::ddt(rho, K) + fvc::div(phi, K)
+ (
      he.name() == "e"
    ? fvc::div
      (
          fvc::absolute(phi/fvc::interpolate(rho), U),
          p,
          "div(phiv,p)"
      )
    : -dpdt
  )
- fvm::laplacian(alphaEff, he)
```

Energy conservation -total energy of flowing fluid: $\rho(\hat{h}+K)$

$$\frac{\partial}{\partial t}\left(\boxed{\rho}(\hat{h}+\boxed{K})\right)+\nabla\cdot\left(\boxed{\rho}\boldsymbol{v}(\hat{h}+\boxed{K})\right)-\frac{\partial p}{\partial t}-\nabla\cdot q=0 \quad \text{where} \quad q=\alpha\nabla\hat{h}$$

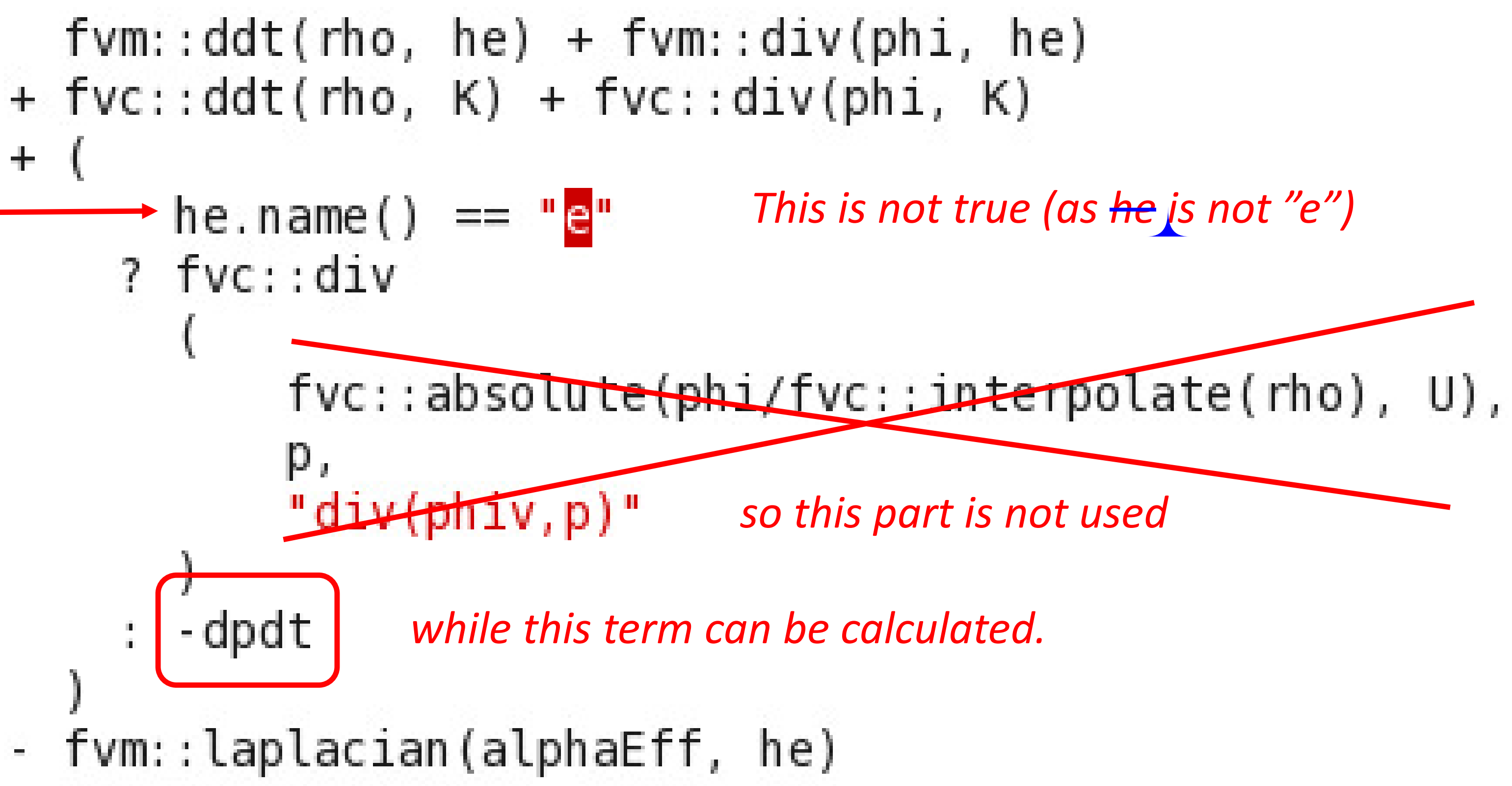
specific kinetic energy K



```
  fvm::ddt(rho, he) + fvm::div(phi, he)
+ fvc::ddt(rho, K) + fvc::div(phi, K)
+ (
      he.name() == "e"
    ? fvc::div
      (
          fvc::absolute(phi/fvc::interpolate(rho), U),
          p,
          "div(phiv,p)"
      )
    : -dpdt
  )
- fvm::laplacian(alphaEff, he)
```

Energy conservation -total energy of flowing fluid: $\rho\left(\hat{h}+K\right)$ *(11/16)*

$$\frac{\partial}{\partial t}\left(\rho\left(\hat{h}+K\right)\right)+\nabla\cdot\left(\rho\boldsymbol{v}\left(\hat{h}+K\right)\right)\boxed{-\frac{\partial p}{\partial t}}-\nabla\cdot q=0 \quad \text{where} \quad q=\alpha\nabla\hat{h}$$



```
  fvm::ddt(rho, he) + fvm::div(phi, he)
+ fvc::ddt(rho, K) + fvc::div(phi, K)
+ (
      he.name() == "e"
    ? fvc::div
      (
          fvc::absolute(phi/fvc::interpolate(rho), U),
          p,
          "div(phiv,p)"
      )
    : -dpdt
  )
- fvm::laplacian(alphaEff, he)
```

If "he" is the specific enthalpy "$\hat{h}$" $\longrightarrow$

This is not true (as ~~he~~ is not "e")

so this part is not used

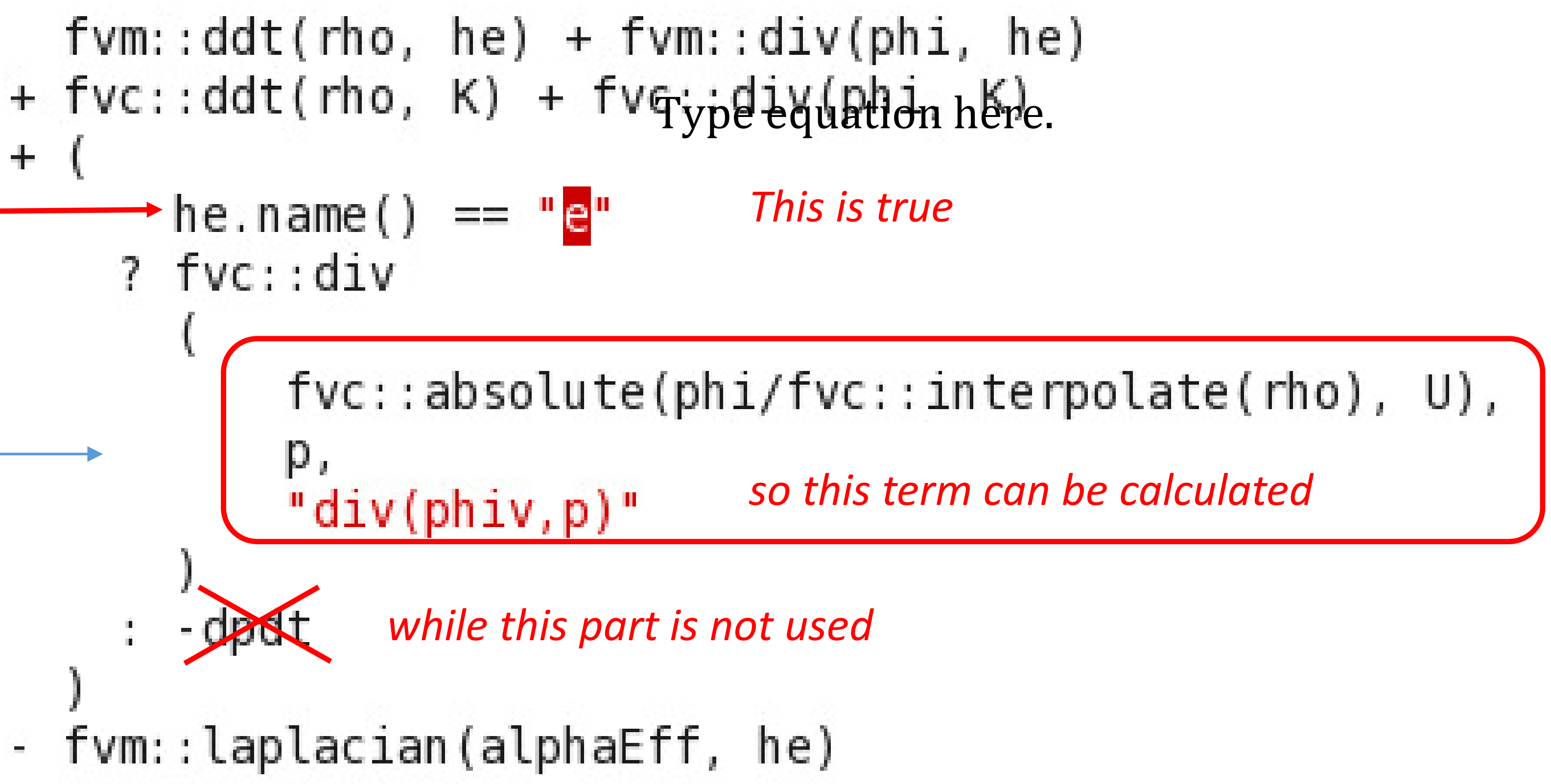
while this term can be calculated.

In fact in thermoFOAM dpdt (created in createFields) it is set to zero since this is a frozen field

Energy conservation -total energy of flowing fluid: $\rho(\hat{h}+K)$ - in a control volume moving at $\mathbf{v_b}$ *(12/16)*

$$\frac{\partial}{\partial t}\left(\rho\left(\hat{h}+K\right)\right)+\nabla\cdot\left(\rho\boldsymbol{v}\left(\hat{h}+K\right)\right)-\frac{\partial p}{\partial t}-\nabla\cdot q=0 \quad \text{where} \quad q=\alpha\nabla\hat{h}$$

Enthalpy : $\hat{h}=\hat{e}+\frac{p}{\rho}$



```
  fvm::ddt(rho, he) + fvm::div(phi, he)
+ fvc::ddt(rho, K) + fvc::div(phi, K)
+ (
      he.name() == "e"
    ? fvc::div
      (
          fvc::absolute(phi/fvc::interpolate(rho), U),
          p,
          "div(phiv,p)"
      )
    : -dpdt
  )
- fvm::laplacian(alphaEff, he)
```

Type equation here.

If ”he” is the specific internal energy $\hat{e}$

This is true

so this term can be calculated

while this part is not used

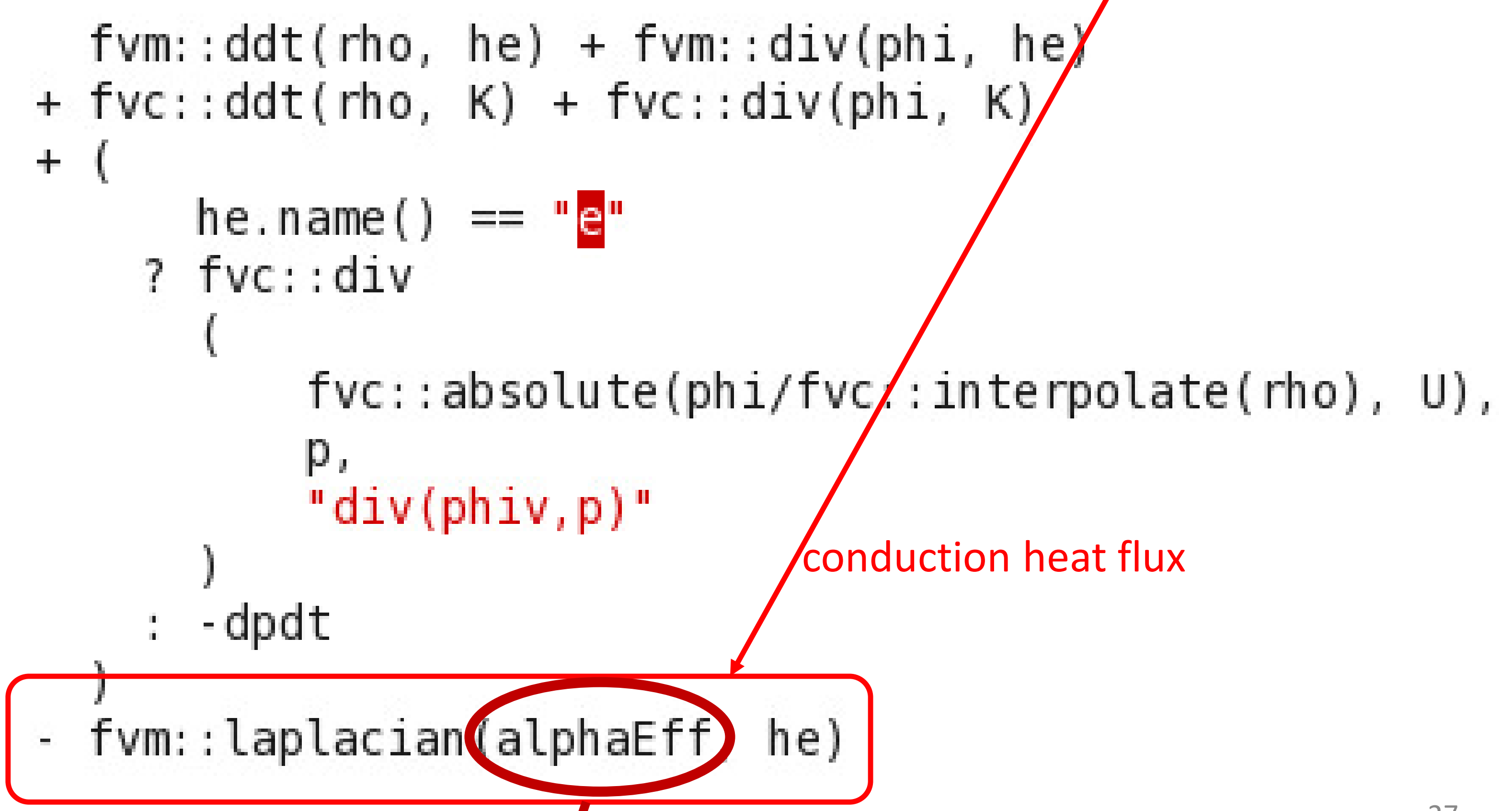
Energy conservation -total energy of flowing fluid: $\rho(\hat{h}+K)$ - in a control volume moving at $\mathbf{v_b}$ *(13/16)*

$$\frac{\partial}{\partial t}\left(\rho\left(\hat{h}+K\right)\right)+\nabla\cdot\left(\rho\boldsymbol{v}\left(\hat{h}+K\right)\right)-\frac{\partial p}{\partial t}\boxed{-\nabla\cdot q}=0 \quad \text{where} \quad \boxed{q=\alpha\nabla\hat{h}}$$

Enthalpy : $\hat{h}=\hat{e}+\frac{p}{\rho}$

Kinetic energy: K

Thermal diffusivity : α



```
  fvm::ddt(rho, he) + fvm::div(phi, he)
+ fvc::ddt(rho, K) + fvc::div(phi, K)
+ (
      he.name() == "e"
    ? fvc::div
      (
          fvc::absolute(phi/fvc::interpolate(rho), U),
          p,
          "div(phiv,p)"
      )
    : -dpdt
  )
- fvm::laplacian(alphaEff, he)
```

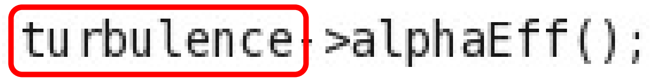
conduction heat flux

alphaEff (the thermal diffusivity) is made of 2 contributions:

alphaEff = **alpha laminar** + *alpha turbulent*

It is set in solver/heatTransfer/thermoFoam/setAlphaEff.H

via the turbulence library : src/turbulenceModels



```
turbulence->alphaEff();
```

the ”turbulenceModels” library is linked to the thermo library src/thermophysicalModels

so if the file names of thermophysical library(.so) are changes,*
also the turbulenceModels need to be recompiled with links
to the new library names

alpha turbulent is defined in *src/turbulenceModels*

alpha laminar is defined in **src/thermophysicalModels** (as transport property)

$$\alpha = \kappa/(\rho c_p)$$

What do we need to implement ?

The heat flux is : $q = -\,\mathrm{alphaEff}\ \nabla\, he$

where the laminar part of alphaEff is either:

$\kappa\,/C_p$ if *"he"* represents the specific enthalpy *h* *(we will work with this case and from now assume he=h)*

or

$\kappa\,/C_v$ if *"he"* represents the specific internal energy *e*

So we **need to implement the thermal conductivity** $\kappa = \kappa(T)$ - *plotted slide 16 (right).*
the **density, and the specific heat capacity,** *respectively plotted slide 15 and 16*, so that $C_p = \rho(T).\,c_p(T)$.

But this is not sufficient since the conservative variable in EEqn.H is the specific enthalpy *h*
while the termodynamic and transport properties depend on another variable: the temperature *T*.
So we also need to determine *T* from *h*. This is done solving (with an iterative procedure already implemented in OpenFOAM) the equation of state

$$\Delta h = \int_{T_{ref}}^{T} c_p(T)\, dT$$

It implies that we **also need to implement the specific enthalpy** *h = h(T).*

This implementation can be done in 2 parts

1. First, the implementation of the new transport property κ (the thermal conductivity) for a temperature range from 200K to 20kK. It will be implemented, compiled and tested.

   Rk. The resultant thermophysical model will not be consistent from a physical point of view (as it will be associated with constant enthalpy, constant specific heat, …). But it runs and has the advantage of allowing splitting the implementation work in smaller parts (easier to debug).

2. Implement the new thermodynamic properties C_p and h and the new equation of state ρ.
   These must be implemented together to be able to derive the temperature T from *h(T)* using $c_p(T)$.

   Rk. The resultant thermophysical model will then be consistent from a physical point of view. It can be observed that the consistent model runs faster than the non-consistent model implemented in the 1st part.

Implement a new transport property K: main steps *(1/12)*

Step 0 : copy and rename suited parts of the library ”thermophysicalModels”

Step 1 : declare the new transport property model (see Ic)*

Step 2 : define the new transport property model (see Ic)*

Step 3: declare the new thermophysical model (see III)**

Step 4 : link the new thermophysical model to the solver

Step 5 : call the new thermophysical model in a test case

Slide 6* ** * Slide 7*

Implement a new transport property K: step 0 *(2/12)*

Prepare your library thermophysicalModels/specie

- **Copy the folder "specie" of the library in your user directory** :

```
 foam
 cp -r --parents src/thermophysicalModels/specie $WM_PROJECT_USER_DIR
 cd $WM_PROJECT_USER_DIR/src/thermophysicalModels
 cd specie
```

Contrary to the usual recommendation, it is not renamed

- **Modify the Make/files to:**

Now the executable is in your working space, and your "libspecie" will be accessed in priority (instead of the OpenFOAM executable in $FOAM_LIBBIN, even if the name is the same)

LIB = $(**FOAM_USER_LIBBIN**)/libspecie

Then this name of executable can be kept unchanged. Doing so, no need to import the turbulence library, no need to rename its links (in Make/options) to your thermo library, and no need to recompile the turbulence library (See slide 28):
the $FOAM_LIBBIN turbulence library will link to your own thermo library.

- **Clean & compile**

You should be in the directory specie

wclean lib
rm –r Make/~~linux~~*
wmake libso

Implement a new transport property K: step 0

Prepare your library thermophysicalModels/basic

- **Copy the folder "basic" of the library in your user directory and rename it**:

```
foam
cp -r --parents src/thermophysicalModels/basic $WM_PROJECT_USER_DIR
cd $WM_PROJECT_USER_DIR/src/thermophysicalModels
cd basic
```

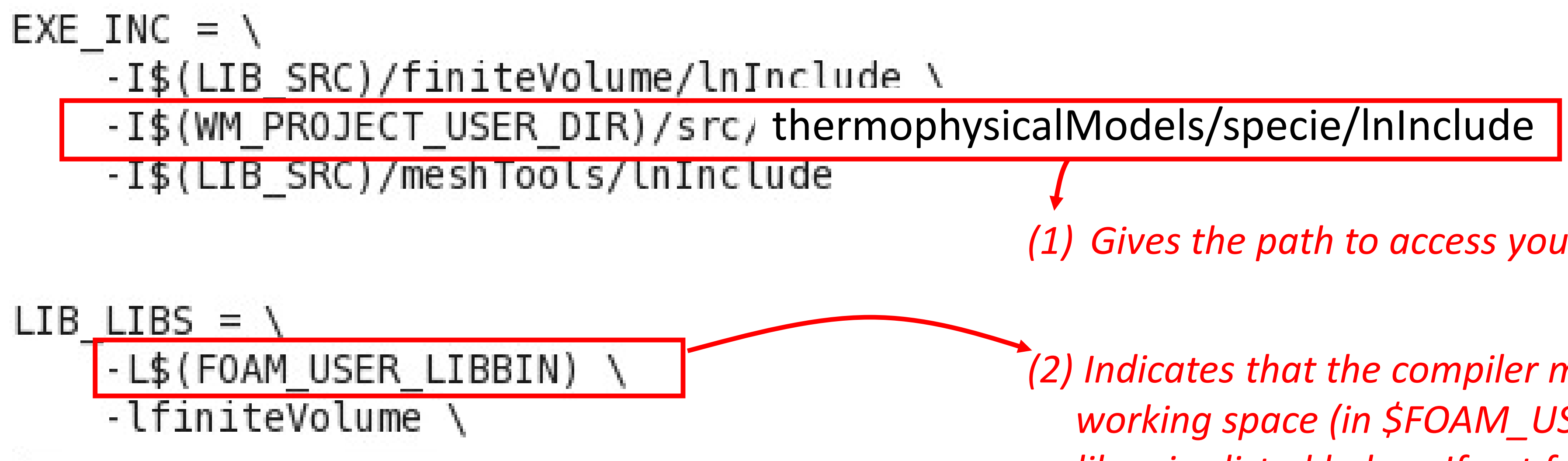
- **Modify Make/files to:**

Similar to the previous slide

LIB = $(**FOAM_USER_LIBBIN**)/libfluidThermophysicalModels

Implement a new transport property K: step 0 *(4/12)*

- **Modify the Make/options file to** *(since basic needs to be linked to specie at compilation)*



```
EXE_INC = \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(WM_PROJECT_USER_DIR)/src/thermophysicalModels/specie/lnInclude
    -I$(LIB_SRC)/meshTools/lnInclude

LIB_LIBS = \
    -L$(FOAM_USER_LIBBIN) \
    -lfiniteVolume \
```

(1) Gives the path to access your own files located in specie

(2) Indicates that the compiler must 1st look in your own working space (in $FOAM_USER_LIBBIN) to pick the libraries listed below. If not found there (ex. may be lfiniteVolume is not in your space $FOAM_USER_LIBBIN) the compiler will next look in the OpenFOAM space (in $FOAM_LIBBIN).
If still not found it will complain.

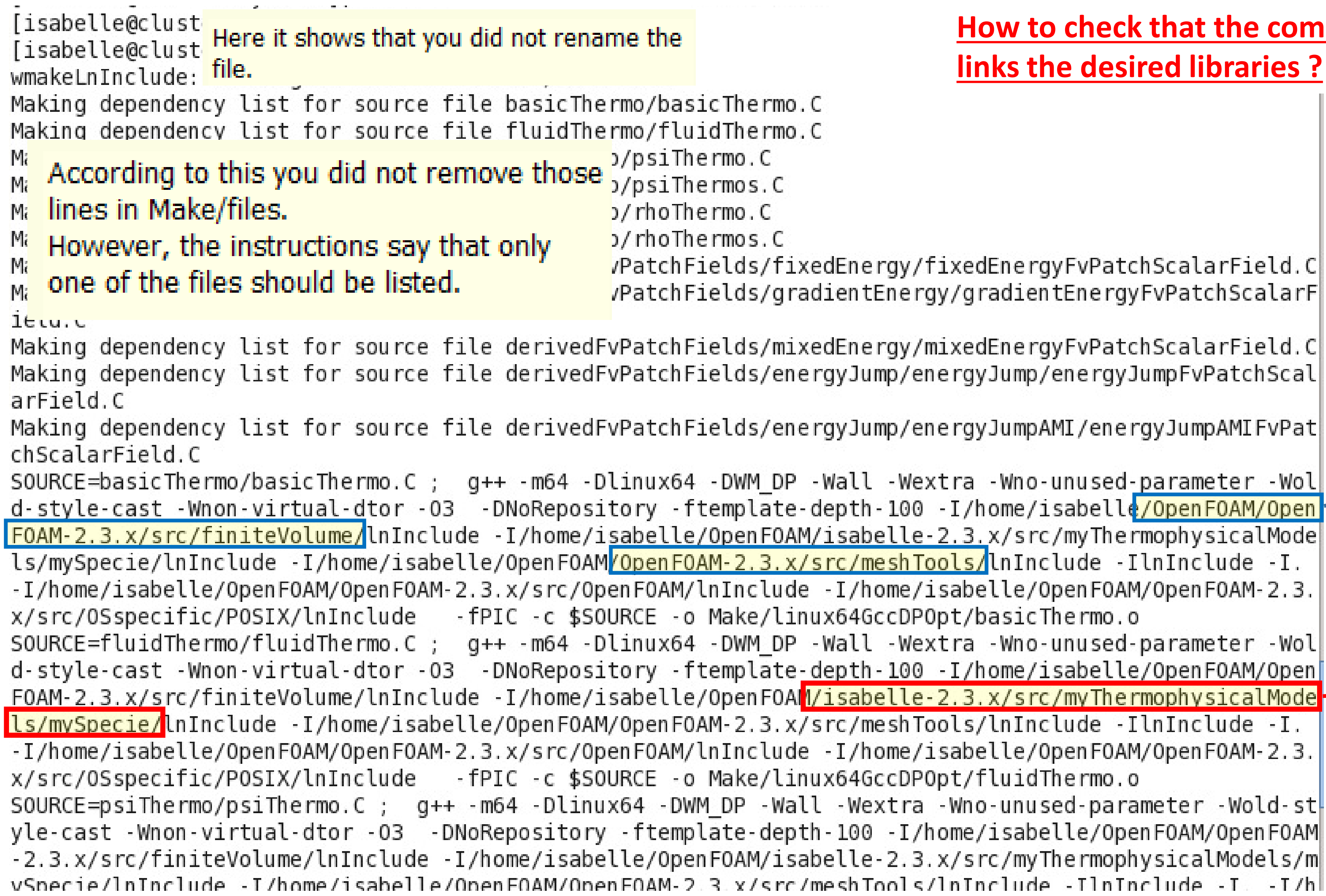
- **Clean the dependencies and compile**

You should be in the directory basic
wclean lib
rm –r Make/linux*
wmake libso

(3) Can check the path used by reading the messages written on the screen during compilation (see next slide)

How to check that the compiler links the desired libraries ?

(5/12)



```
[isabelle@clust
[isabelle@clust
wmakeLnInclude:
Making dependency list for source file basicThermo/basicThermo.C
Making dependency list for source file fluidThermo/fluidThermo.C
Ma                                                  o/psiThermo.C
Ma                                                  o/psiThermos.C
Ma                                                  o/rhoThermo.C
Ma                                                  o/rhoThermos.C
Ma                                                  vPatchFields/fixedEnergy/fixedEnergyFvPatchScalarField.C
Ma                                                  vPatchFields/gradientEnergy/gradientEnergyFvPatchScalarF
ield.C
Making dependency list for source file derivedFvPatchFields/mixedEnergy/mixedEnergyFvPatchScalarField.C
Making dependency list for source file derivedFvPatchFields/energyJump/energyJump/energyJumpFvPatchScal
arField.C
Making dependency list for source file derivedFvPatchFields/energyJump/energyJumpAMI/energyJumpAMIFvPat
chScalarField.C
SOURCE=basicThermo/basicThermo.C ;  g++ -m64 -Dlinux64 -DWM_DP -Wall -Wextra -Wno-unused-parameter -Wol
d-style-cast -Wnon-virtual-dtor -O3  -DNoRepository -ftemplate-depth-100 -I/home/isabelle/OpenFOAM/Open
FOAM-2.3.x/src/finiteVolume/lnInclude -I/home/isabelle/OpenFOAM/isabelle-2.3.x/src/myThermophysicalMode
ls/mySpecie/lnInclude -I/home/isabelle/OpenFOAM/OpenFOAM-2.3.x/src/meshTools/lnInclude -IlnInclude -I.
-I/home/isabelle/OpenFOAM/OpenFOAM-2.3.x/src/OpenFOAM/lnInclude -I/home/isabelle/OpenFOAM/OpenFOAM-2.3.
x/src/OSspecific/POSIX/lnInclude   -fPIC -c $SOURCE -o Make/linux64GccDPOpt/basicThermo.o
SOURCE=fluidThermo/fluidThermo.C ;  g++ -m64 -Dlinux64 -DWM_DP -Wall -Wextra -Wno-unused-parameter -Wol
d-style-cast -Wnon-virtual-dtor -O3  -DNoRepository -ftemplate-depth-100 -I/home/isabelle/OpenFOAM/Open
FOAM-2.3.x/src/finiteVolume/lnInclude -I/home/isabelle/OpenFOAM/isabelle-2.3.x/src/myThermophysicalMode
ls/mySpecie/lnInclude -I/home/isabelle/OpenFOAM/OpenFOAM-2.3.x/src/meshTools/lnInclude -IlnInclude -I.
-I/home/isabelle/OpenFOAM/OpenFOAM-2.3.x/src/OpenFOAM/lnInclude -I/home/isabelle/OpenFOAM/OpenFOAM-2.3.
x/src/OSspecific/POSIX/lnInclude   -fPIC -c $SOURCE -o Make/linux64GccDPOpt/fluidThermo.o
SOURCE=psiThermo/psiThermo.C ;  g++ -m64 -Dlinux64 -DWM_DP -Wall -Wextra -Wno-unused-parameter -Wold-st
yle-cast -Wnon-virtual-dtor -O3  -DNoRepository -ftemplate-depth-100 -I/home/isabelle/OpenFOAM/OpenFOAM
-2.3.x/src/finiteVolume/lnInclude -I/home/isabelle/OpenFOAM/isabelle-2.3.x/src/myThermophysicalModels/m
ySpecie/lnInclude -I/home/isabelle/OpenFOAM/OpenFOAM-2.3.x/src/meshTools/lnInclude -IlnInclude -I. -I/h
```

Here it shows that you did not rename the file.

According to this you did not remove those lines in Make/files.
However, the instructions say that only one of the files should be listed.

Here it can be checked that the library "finiteVolume" is picked in ***~~/home/…/OpenFOAM2.3.x/src~~***

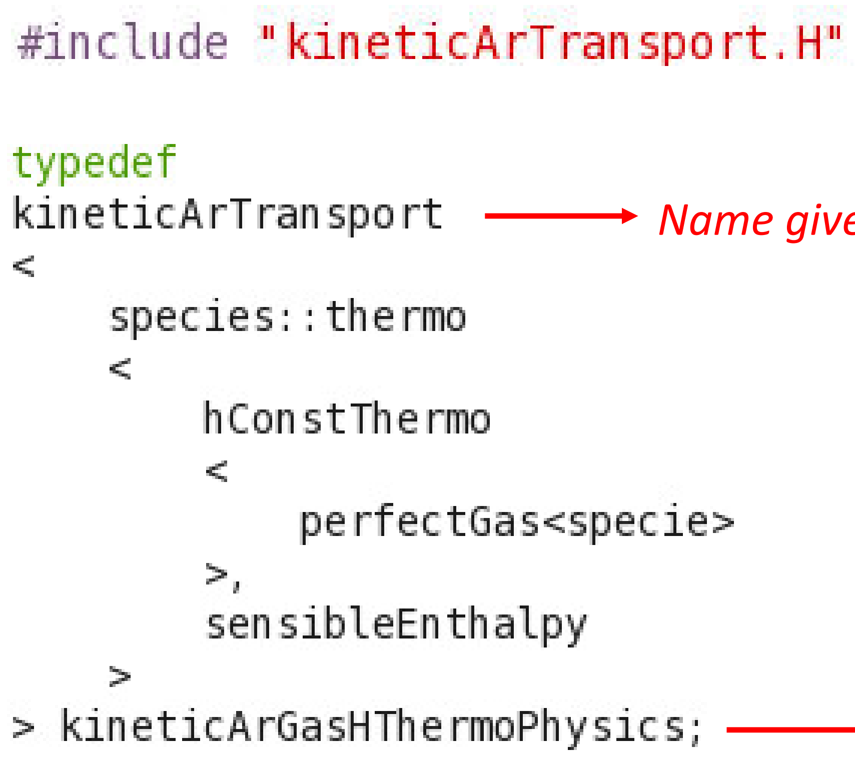
There it can be checked that "myThermophysicalModels/specie" is picked in ***~~/home/…/isabelle-2.3.x/src~~***

…

Implement a new transport property K: step 1 *(6/12)*

Declare the new transport model in user src/thermophysicalModels/specie

- **In specie/include/thermoPhysicsTypes.H add the following lines**



```
#include "kineticArTransport.H"

typedef
kineticArTransport
<
    species::thermo
    <
        hConstThermo
        <
            perfectGas<specie>
        >,
        sensibleEnthalpy
    >
> kineticArGasHThermoPhysics;
```

$\longrightarrow$ *To access the new transport model*

$\longrightarrow$ *Name given to the new transport model*

$\longrightarrow$ *Name given to the new thermophysical model*

Implement a new transport property K: step 2

Define the new transport model in user src/thermophysicalModels/specie

- **Copy and rename an existing model:** *(Prepare the structure, you should be in specie)*

```
cd transport
cp -r const kineticAr
cd kineticAr
mv constTransport.C kineticArTransport.C
mv constTransport.H kineticArTransport.H
mv constTransportI.H kineticArTransportI.H
```

open the files one by one and replace
"constTransport" (NOT just "const" !) with "kineticArTransport"

Use:
```
sed –i.old s/constTransport/kineticArTransport/g*
```

update the "instantiated type name" in kineticArTransport.H

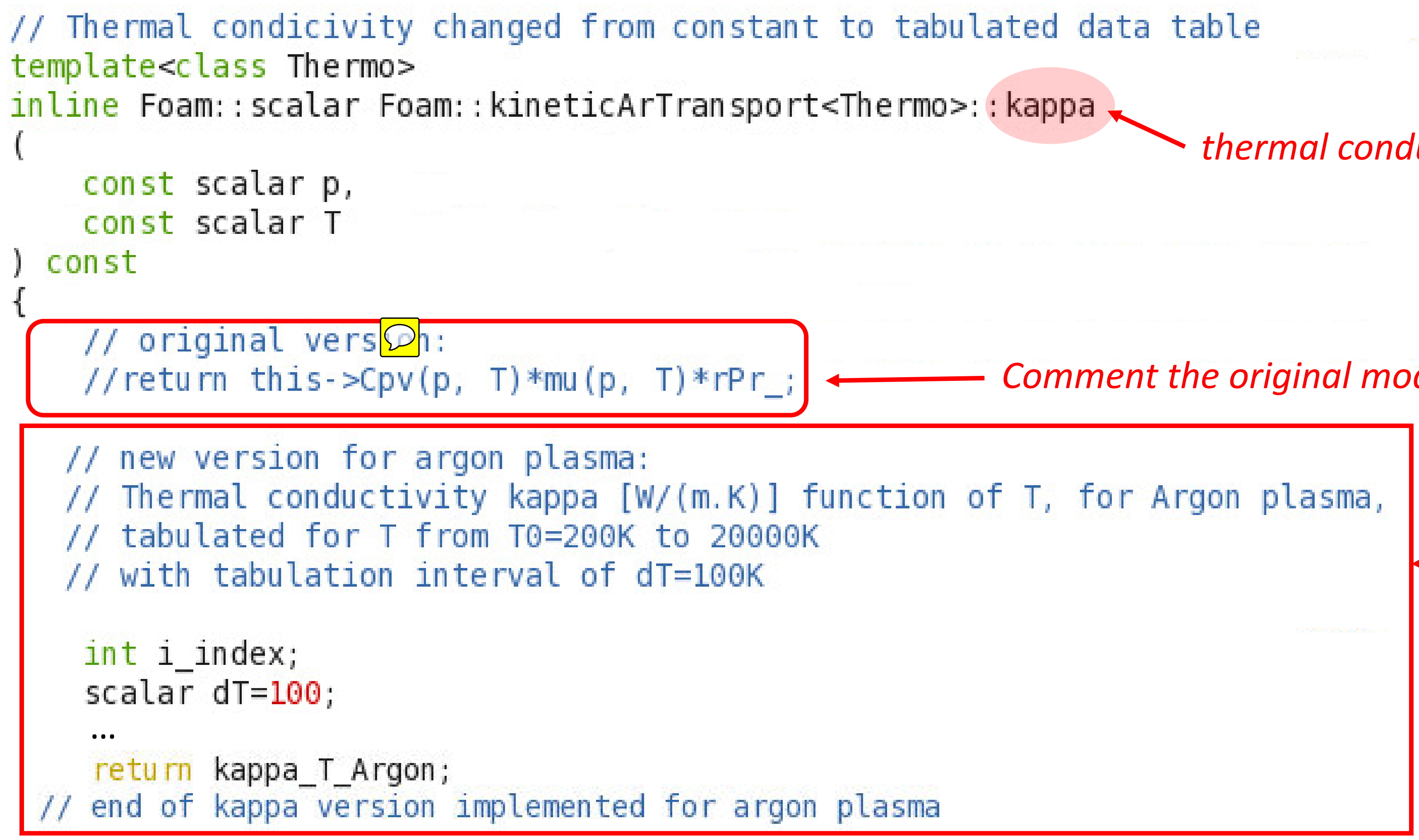
so look for "instantiated" and below (only there!) replace "return "const" " with " return "kineticAr" "

- **Clean the dependencies and compile specie**

```
wclean lib
rm –r Make/linux*
wmake libso
```

Implement a new transport property K: step 2

- **Go in the directory kineticAr and open** kineticArTransportI.H



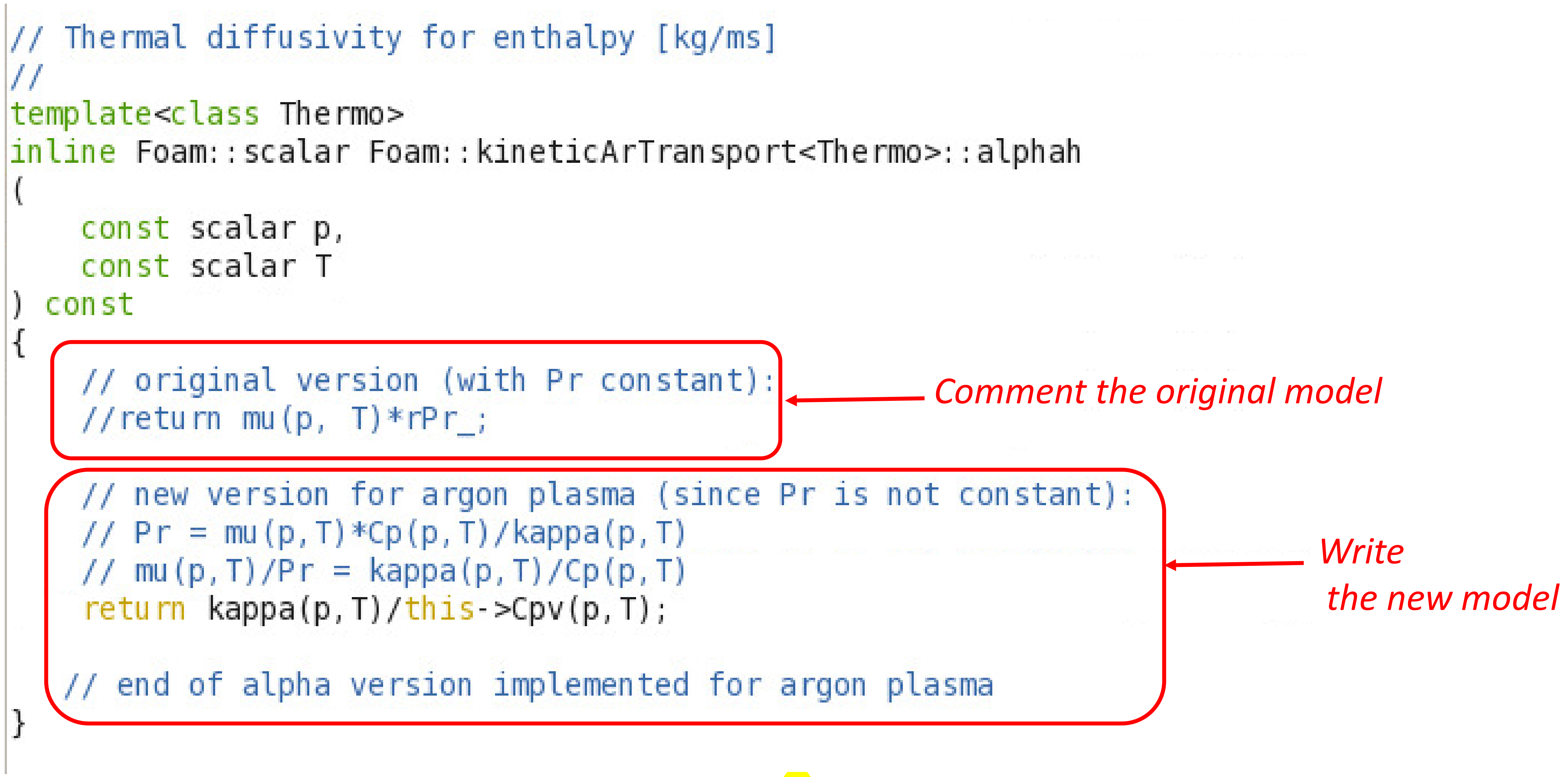
```
// Thermal condicivity changed from constant to tabulated data table
template<class Thermo>
inline Foam::scalar Foam::kineticArTransport<Thermo>::kappa
(
    const scalar p,
    const scalar T
) const
{
    // original version:
    //return this->Cpv(p, T)*mu(p, T)*rPr_;

   // new version for argon plasma:
   // Thermal conductivity kappa [W/(m.K)] function of T, for Argon plasma,
   // tabulated for T from T0=200K to 20000K
   // with tabulation interval of dT=100K

    int i_index;
    scalar dT=100;
    …
     return kappa_T_Argon;
  // end of kappa version implemented for argon plasma
```

thermal conductivity

Comment the original model

insert** the new model provided in **the file Ar_Data_ThermalConduct

Implement a new transport property K: step 2 *(9/12)*



```
// Thermal diffusivity for enthalpy [kg/ms]
//
template<class Thermo>
inline Foam::scalar Foam::kineticArTransport<Thermo>::alphah
(
    const scalar p,
    const scalar T
) const
{
    // original version (with Pr constant):
    //return mu(p, T)*rPr_;

    // new version for argon plasma (since Pr is not constant):
    // Pr = mu(p,T)*Cp(p,T)/kappa(p,T)
    // mu(p,T)/Pr = kappa(p,T)/Cp(p,T)
    return kappa(p,T)/this->Cpv(p,T);

   // end of alpha version implemented for argon plasma
}
```

- **Clean the dependencies** ("wclean lib" **and** "rm –r Make/linux*") **and compile mySpecie** ("wmake libso")

Implement a new transport property K: step 3 *(10/12)*

Declare the new thermophysical model in user src/thermophysicalModels/basic

- **In basic/rhoThermo/rhoThermos.C add the following lines**

```
#include "kineticArTransport.H"
 . . .
makeThermo
(
    rhoThermo,
    heRhoThermo,
    pureMixture,
    kineticArTransport,
    sensibleEnthalpy,
    hConstThermo,
    perfectGas,
    specie
);
```

In the header

New combination of Ia, Ib, Ic, II and III (see slides 3 to 7) defining a new thermophysical model

- **Clean the dependencies** ("wclean lib" and " rm –r Make/linux* ")
- **Compile myBasic** ("wmake libso")

Implement a new transport property K: step 4 *(11/12)*

Link the new thermophysicalModel library to the solver

- **In myThermoFoam/Make/options do the following changes to access the new library**

```
-I$(LIB_SRC)/thermophysicalModels/basic/lnInclude \
```

$\downarrow$

```
-I$(WM_PROJECT_USER_DIR)/thermophysicalModels/basic/lnInclude \
```

```
EXE_LIBS = \
    -lfiniteVolume \
```

$\downarrow$

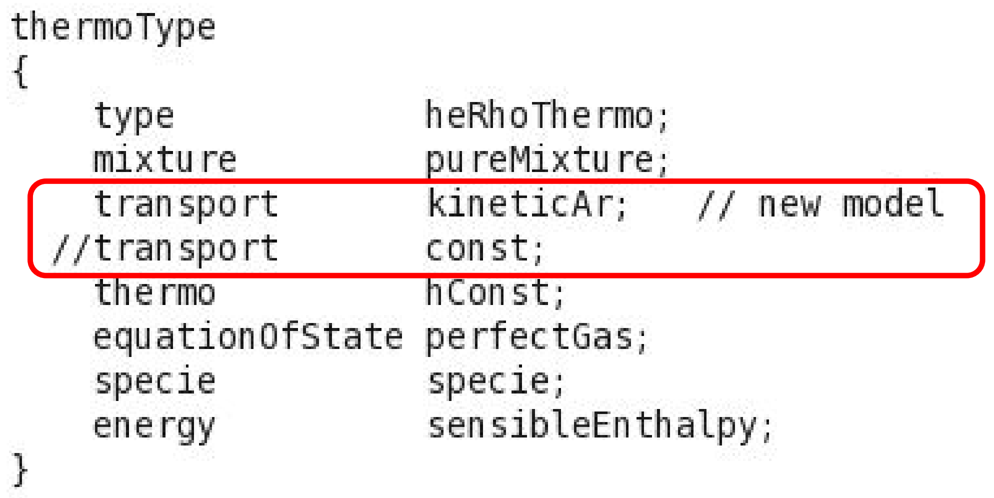
```
EXE_LIBS = \
    -L$(FOAM_USER_LIBBIN) \
    -lfiniteVolume \
```

Check that the solver is using your library:
ldd `which ~~thermoFoam~~` | grep specie
~~ldd `which thermoFoam` | grep specie~~

Implement a new transport property K: step 5 *(12/12)*

Call the new thermophysicalModel in a test case

- **Use the test case provided: blockThermoFoamCase.tgz**
- **Run blockMesh**
- **Run this case with the solver thermoFoam (the original one)**
- **Copy blockThermoFoamCase to blockNewThermoFoamCase and clean**
- **Update constant/thermophysicalPropersties to**



```
thermoType
{
    type            heRhoThermo;
    mixture         pureMixture;
    transport       kineticAr;    // new model
  //transport       const;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleEnthalpy;
}
```

- **Run this case with the solver myThermoFoam linked to the new thermophysical library**
- **Compare the results : do a plotOverLine of temperature for both cases**

This implementation can be done in 2 parts

1. First, the implementation of the new transport property κ (the thermal conductivity) for a temperature range from 200K to 20kK. It will be implemented, compiled and tested.

   Rk. The resultant thermophysical model will not be consistent from a physical point of view (as it will be associated with constant enthalpy, constant specific heat, …). But it runs and has the advantage of allowing splitting the implementation work in smaller parts (easier to debug).

2. Implement the new thermodynamic properties C_p and h and the new equation of state ρ.
   These must be implemented together since the aim is to be able to derive the temperature T from *h(T)* and to calculate the heat capacity $C_p = \rho(T).c_p(T)$

   Rk. The resultant thermophysical model will then be consistent from a physical point of view.
   It can also be observed that the consitent model runs faster than the non-consistent model implemented in the 1st step

Part 1 is done. We now start this 2^{nd}part

Implement new thermodynamic properties and a new equation of state

Step 1 : declare (see Ia, b) #,* the new thermodynamic properties and the new equation of state

Step 2 : define (see Ia) # the new equation of state

Step 3 : define (see Ib)* the new thermodynamic properties

Step 4: declare (see III)** the new thermophysical model

Step 5 : link the new thermophysical model to the solver

Step 6 : call the new thermophysical model in a test case and run

#*See slide 4* *See slide 5* ** *See slide 7*

Implement new thermodynamic properties and equation of state: step 1

Declare the new thermophysical model in user src/thermophysicalModels/basic

- **In specie/include/thermoPhysicsTypes.H add in the header**

```
#include "hKineticArThermo.H"
#include "rhoKineticAr.H"
```

To access the new thermodynamic model & the new equation of state

and modify

```
typedef
kineticArTransport
<
    species::thermo
    <
        hConstThermo
        <
            perfectGas<specie>
        >,
        sensibleEnthalpy
    >
> kineticArGasHThermoPhysics;
```

Name given to the new

thermodynamic model
&
equation of state

to

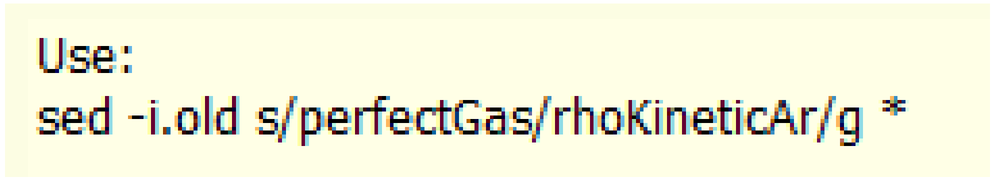
```
typedef
kineticArTransport
<
    species::thermo
    <
        hKineticArThermo
        <
            rhoKineticAr<specie>
        >,
        sensibleEnthalpy
    >
> kineticArGasHThermoPhysics;
```

Implement new thermodynamic properties and equation of state: step 2 *(4/16)*

Define a new equation of state model in user src/thermophysicalModels/specie

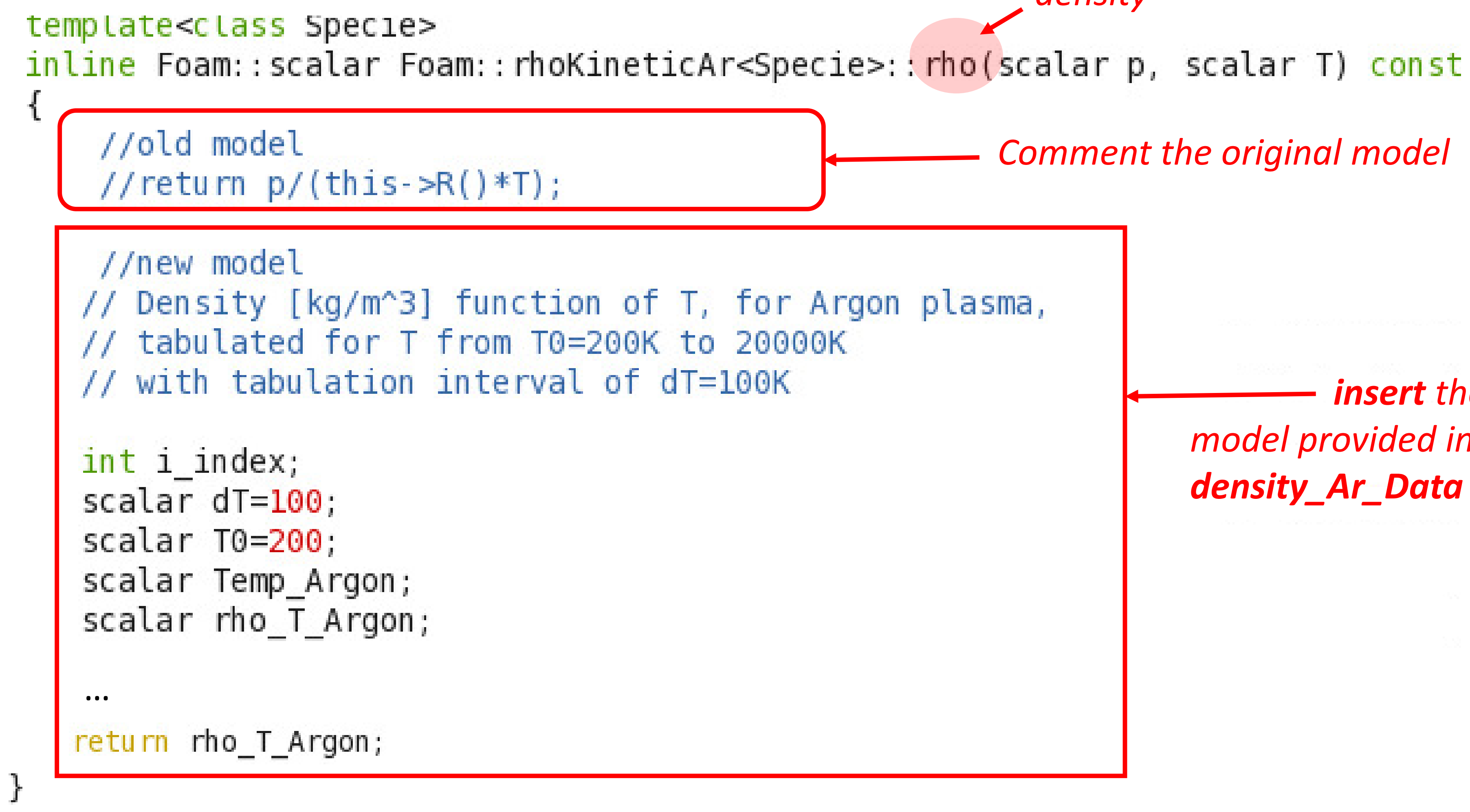
- **Copy and rename an existing model:**
  ```
  cd equationOfState
  cp -r perfectGas rhoKineticAr
  cd rhoKineticAr
  mv perfectGas.C rhoKineticAr.C
  mv perfectGas.H rhoKineticAr.H
  mv perfectGastI.H rhoKineticArI.H
  ```
  open the files one by one and replace
  “perfectGas” with “rhoKineticAr”

Prepare the structure


Use:
```
sed -i.old s/perfectGas/rhoKineticAr/g *
```

Implement new thermodynamic properties and equation of state: step 2 *(5/16)*

- **Open** rhoKineticArI.H and do the following modifications:



```
template<class Specie>
inline Foam::scalar Foam::rhoKineticAr<Specie>::rho(scalar p, scalar T) const
{
    //old model
    //return p/(this->R()*T);

    //new model
   // Density [kg/m^3] function of T, for Argon plasma,
   // tabulated for T from T0=200K to 20000K
   // with tabulation interval of dT=100K

   int i_index;
   scalar dT=100;
   scalar T0=200;
   scalar Temp_Argon;
   scalar rho_T_Argon;

   …
  return rho_T_Argon;
}
```

density

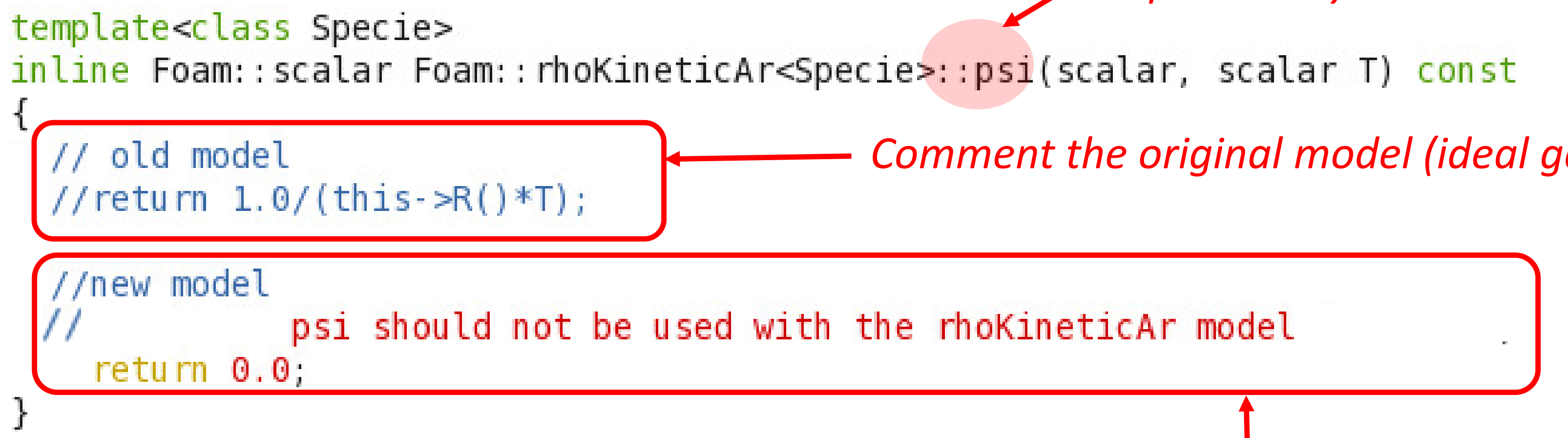
Comment the original model

insert** the new model provided in **the file density_Ar_Data

Implement new thermodynamic properties and equation of state: step 2 *(6/16)*

- **Modify also**

compressibility



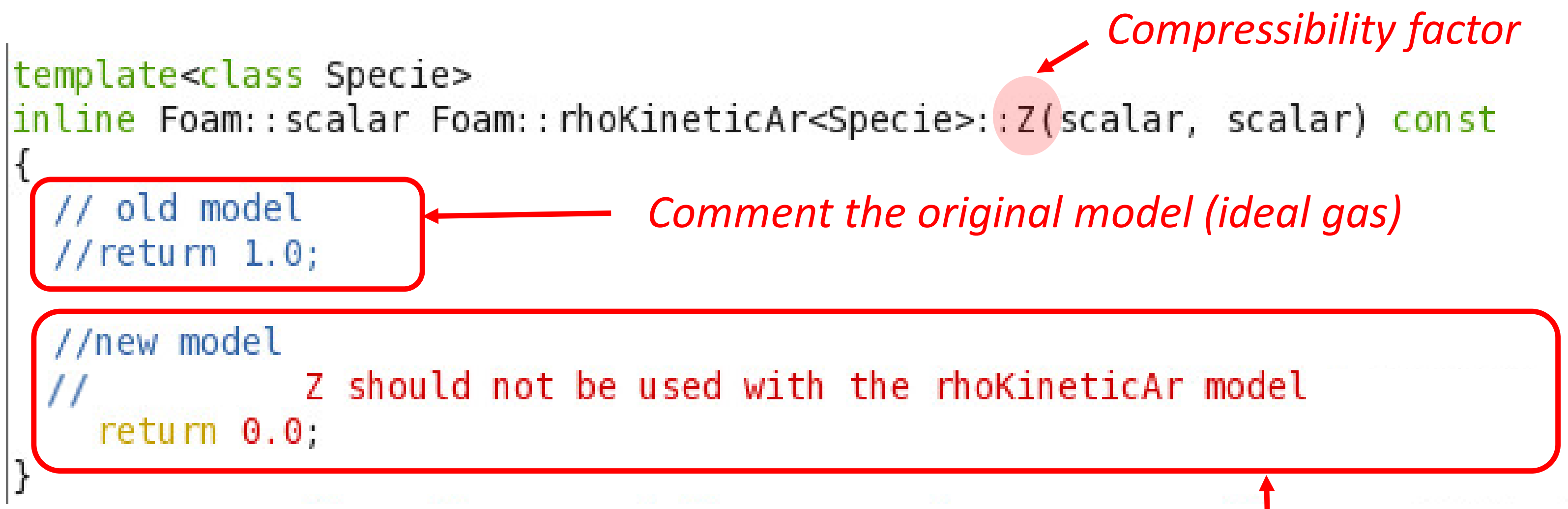
```
template<class Specie>
inline Foam::scalar Foam::rhoKineticAr<Specie>::psi(scalar, scalar T) const
{
    // old model
    //return 1.0/(this->R()*T);

    //new model
   //          psi should not be used with the rhoKineticAr model
      return 0.0;
}
```

Comment the original model (ideal gas)

Write the new model
Rk: psi is set to zero since the plasma model implemented here is mechanically incompressible, and thermaly expansible: $\rho(P,T) = \rho(T)$.

Implement new thermodynamic properties and equation of state: step 2

- **Modify also**

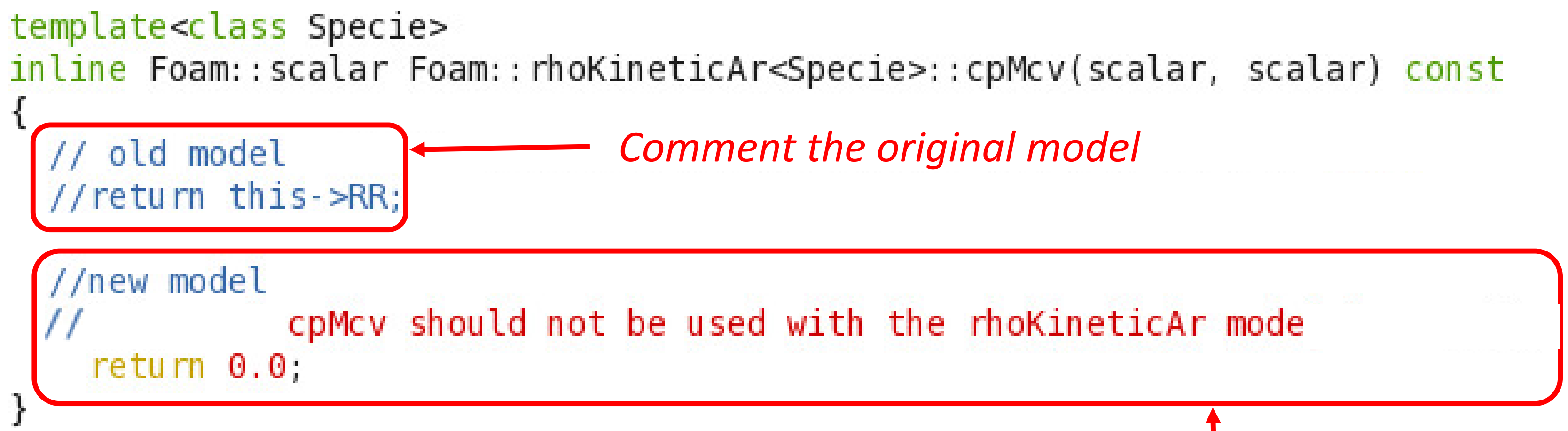


```
template<class Specie>
inline Foam::scalar Foam::rhoKineticAr<Specie>::Z(scalar, scalar) const
{
  // old model
  //return 1.0;

  //new model
 //          Z should not be used with the rhoKineticAr model
    return 0.0;
}
```

Write the new model

Implement new thermodynamic properties and equation of state: step 2 *(8/16)*

- **Modify also**



```
template<class Specie>
inline Foam::scalar Foam::rhoKineticAr<Specie>::cpMcv(scalar, scalar) const
{
  // old model
  //return this->RR;

  //new model
  //          cpMcv should not be used with the rhoKineticAr mode
    return 0.0;
}
```

Comment the original model

write the new model

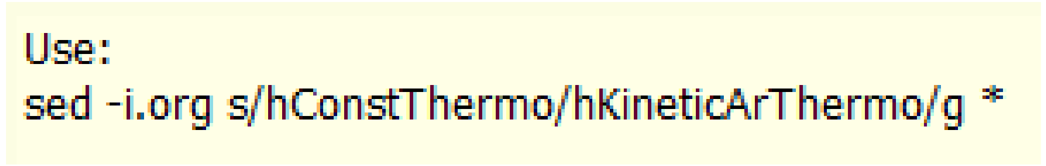
Implement new thermodynamic properties and equation of state: step 3 *(9/16)*

Define the new properties h, C_p in user src/thermophysicalModels/specie

- **Copy and rename an existing model:**
  cd thermo
  cp -r hConst hKineticAr
  cd hKineticAr
  mv hConstThermo.C hKineticArThermo.C
  mv hConstThermo.H hKineticArThermo.H
  mv hConstThermoI.H hKineticArThermoI.H

Prepare the structure

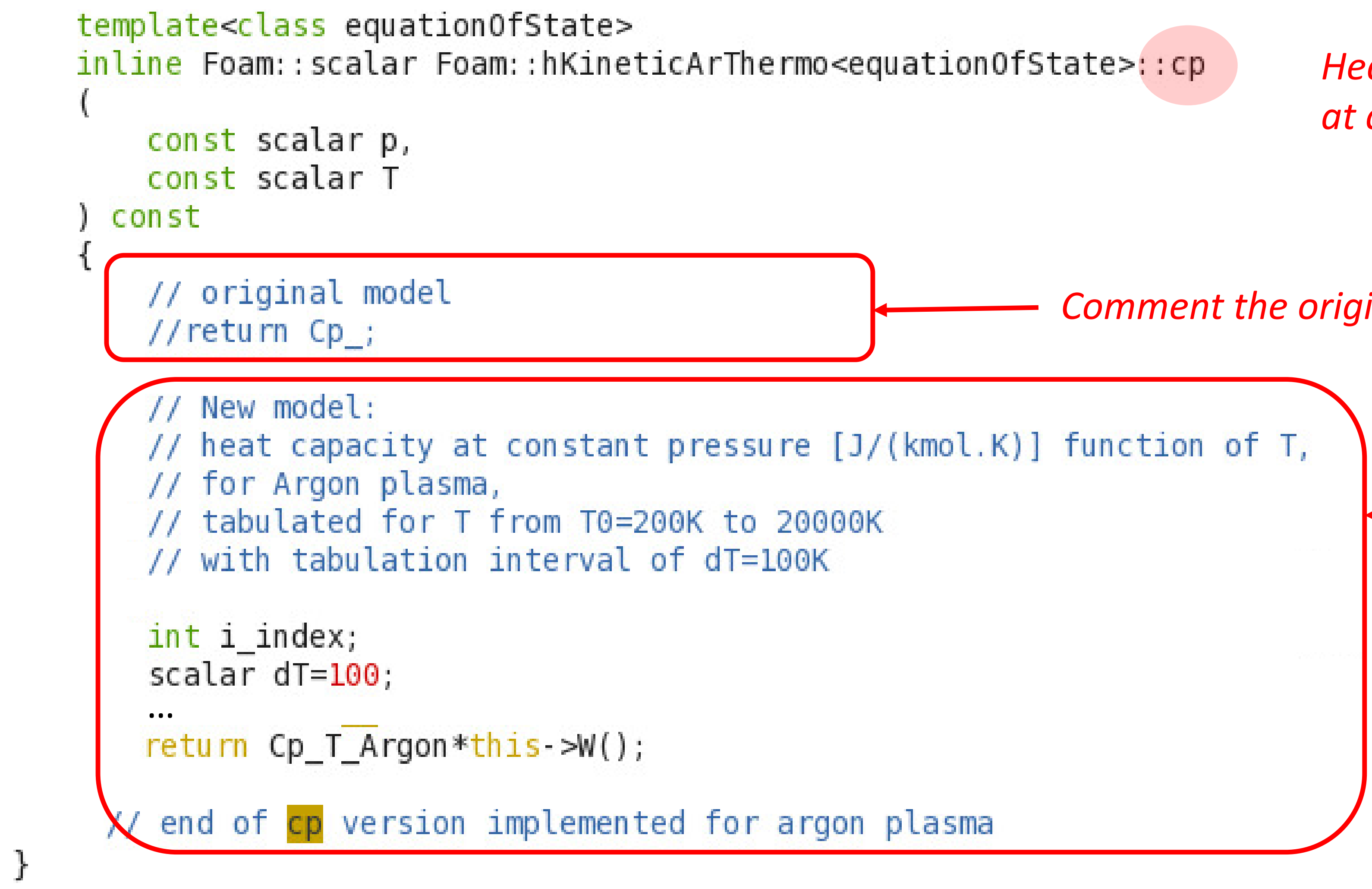
open the files one by one and replace
“hConstThermo” with “hKineticArThermo”


Use:
sed -i.org s/hConstThermo/hKineticArThermo/g *

update the “instantiated type name” in hKineticArThermo.H
so look for “instantiated” and below replace “return “hConst<” “
with “ return “hKineticAr<” “

Implement new thermodynamic properties and equation of state: step 2 *(10/16)*

- **Open hKineticArI.H and do the following modifications**:



```
template<class equationOfState>
inline Foam::scalar Foam::hKineticArThermo<equationOfState>::cp
(
    const scalar p,
    const scalar T
) const
{
    // original model
    //return Cp_;

    // New model:
    // heat capacity at constant pressure [J/(kmol.K)] function of T,
    // for Argon plasma,
    // tabulated for T from T0=200K to 20000K
    // with tabulation interval of dT=100K

    int i_index;
    scalar dT=100;
    ...
    return Cp_T_Argon*this->W();

  // end of cp version implemented for argon plasma
}
```

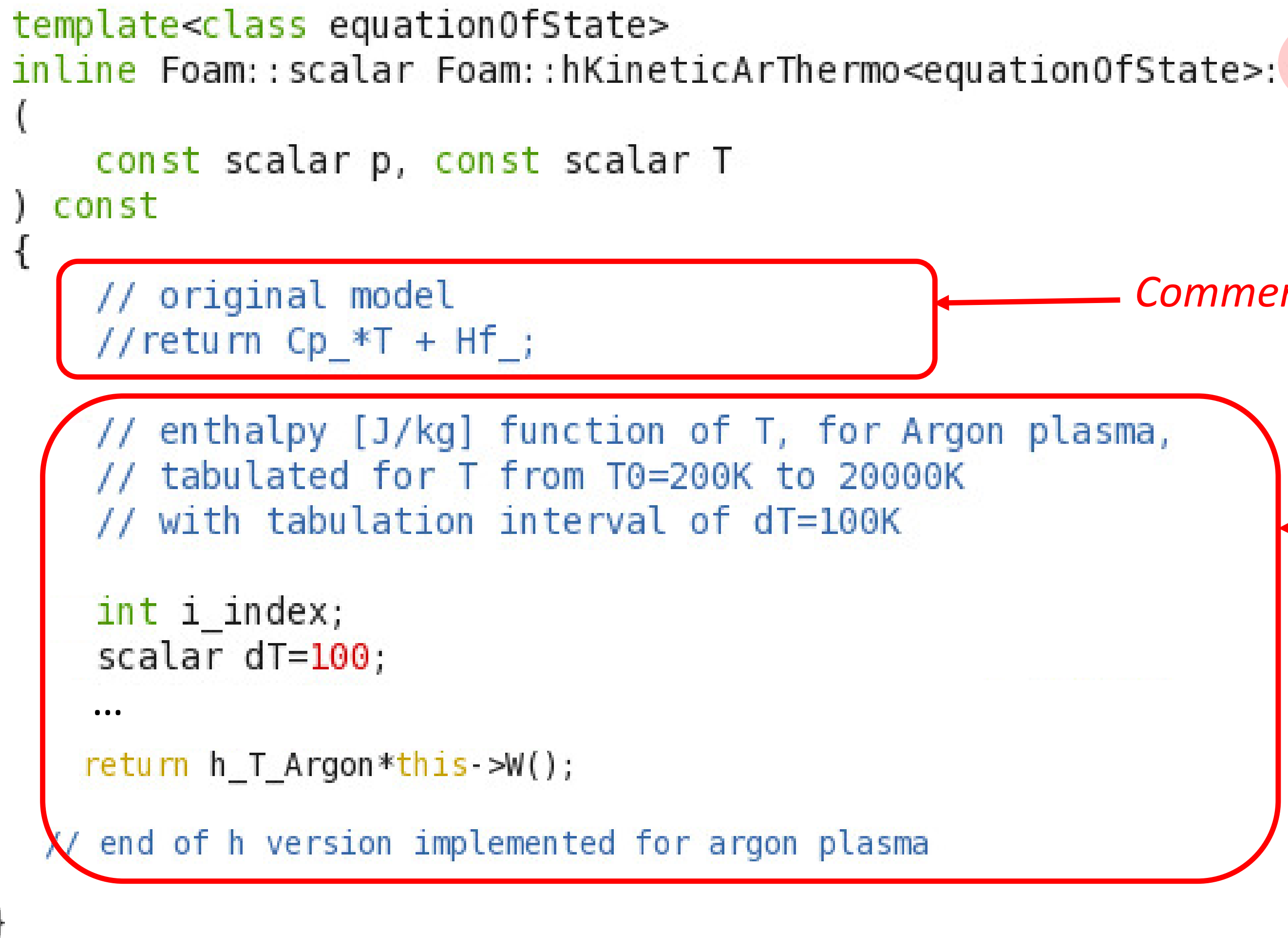
Heat capacity at constant pressure

Comment the original model

insert** the new model provided in **the file heatCapacity_Cp_Data

Implement new thermodynamic properties and equation of state: step 3 *(11/16)*

- **Modify also**



```
template<class equationOfState>
inline Foam::scalar Foam::hKineticArThermo<equationOfState>::ha
(
    const scalar p, const scalar T
) const
{
    // original model
    //return Cp_*T + Hf_;

    // enthalpy [J/kg] function of T, for Argon plasma,
    // tabulated for T from T0=200K to 20000K
    // with tabulation interval of dT=100K

    int i_index;
    scalar dT=100;
    …
   return h_T_Argon*this->W();

  // end of h version implemented for argon plasma

}
```

absolute enthalpy

Hf is the enthalpy of formation

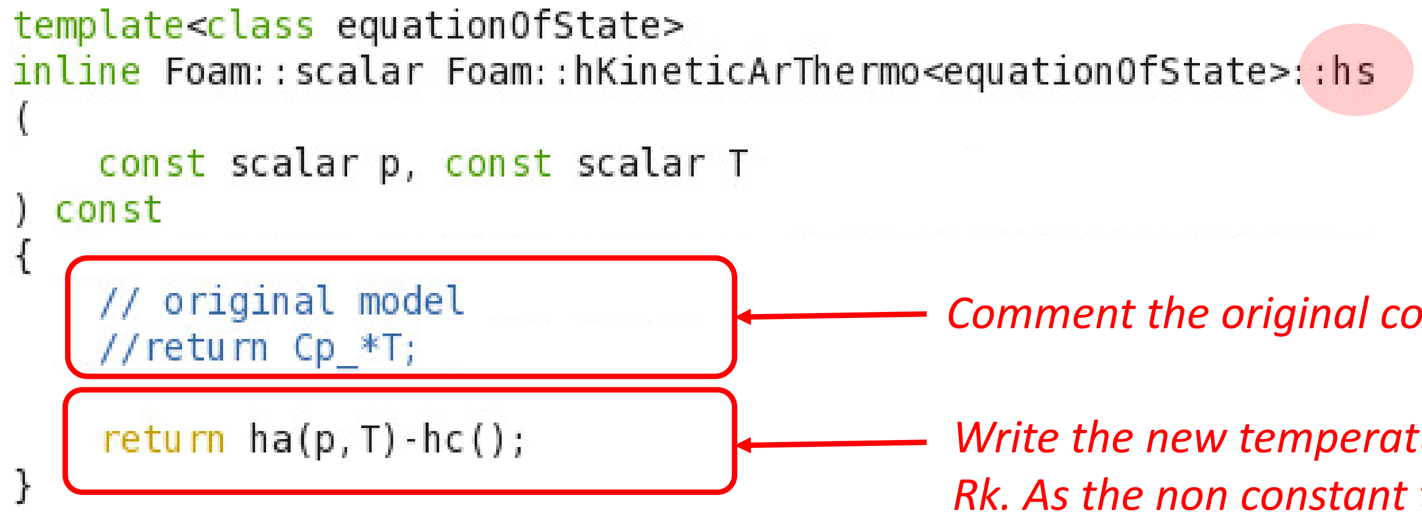
Comment the original model

insert** the new model provided in **the file enthalpy_Data.

Rk. The reference temperature was set so that Hf is zero.

Implement new thermodynamic properties and equation of state: step 3

- **Modify also**



```
template<class equationOfState>
inline Foam::scalar Foam::hKineticArThermo<equationOfState>::hs
(
    const scalar p, const scalar T
) const
{
    // original model
    //return Cp_*T;

    return ha(p,T)-hc();
}
```

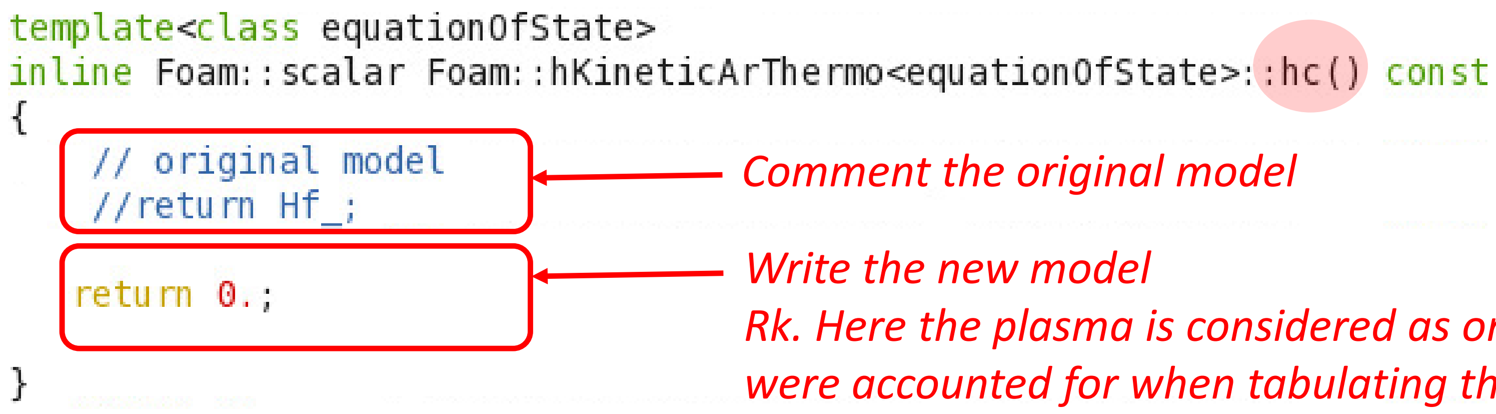
sensible enthalpy

Comment the original constant model

Write the new temperature dependent model.
Rk. As the non constant thermodynamic models in openFOAM depend on both pressure p and temperature, we write ha(p,T) although p is not used.

Implement new thermodynamic properties and equation of state: step 3 *(13/16)*

- **Modify also**



```
template<class equationOfState>
inline Foam::scalar Foam::hKineticArThermo<equationOfState>::hc() const
{
    // original model
    //return Hf_;

    return 0.;

}
```

chemical enthalpy

Comment the original model

Write the new model

Rk. Here the plasma is considered as one-fluid. The ionization reactions were accounted for when tabulating the absolute enthalpy ha. For a one-fluid model hc is the enthalpy of formation. Here it is set to zero (see the remark on slide 53).

- **Clean and compile**

wclean lib
rm –r Make/linux*
wmake libso

Implement new thermodynamic properties and equation of state: step 4 *(14/16)*

Declare the new thermophysical model in user src/thermophysicalModels/basic

- **In basic/rhoThermo/rhoThermos.C add the following lines in the header**

```
#include "hKineticArThermo.H"
#include "rhoKineticAr.H"
```

To access the new thermodynamic model & the new equation of state

and change

```
 makeThermo
 (
     rhoThermo,
     heRhoThermo,
     pureMixture,
     kineticArTransport,
     sensibleEnthalpy,
     hConstThermo,
     perfectGas,
     specie
 );
```

Name given to the new thermodynamic model & equation of state

to:

```
makeThermo
(
    rhoThermo,
    heRhoThermo,
    pureMixture,
    kineticArTransport,
    sensibleEnthalpy,
    hKineticArThermo,
    rhoKineticAr,
    specie
);
```

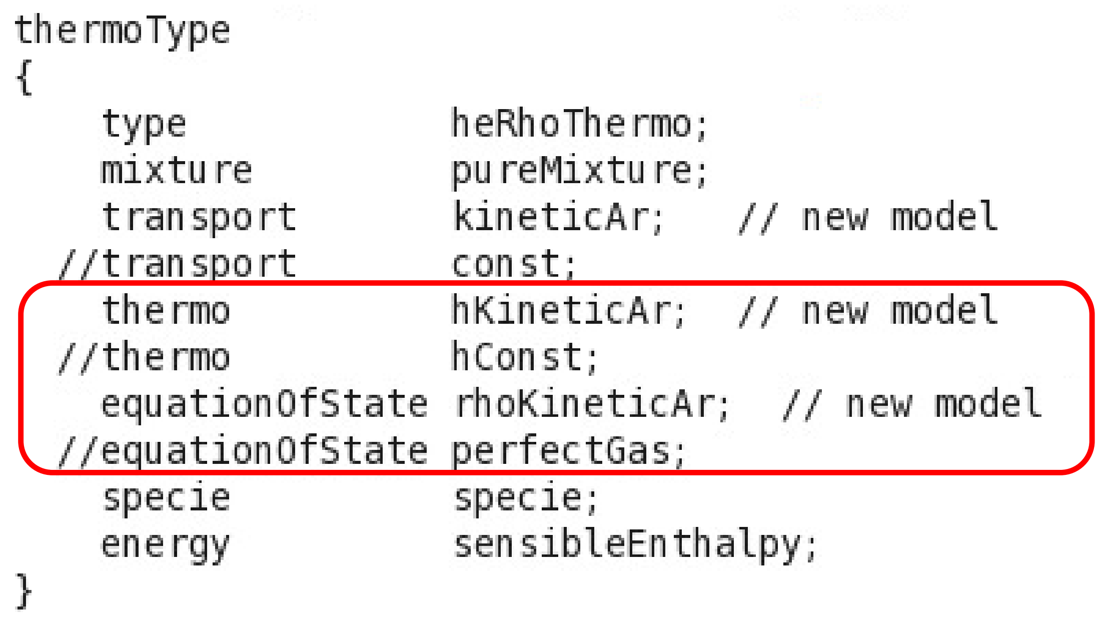
- **Clean and compile**

```
wclean lib
 rm –r Make/linux*
wmake libso
```

Implement new thermodynamic properties and equation of state: step 6

Call the new thermophysical model in a test case and run

- **Copy blockThermoFoamCase to blockKineticArThermoFoamCase and clean (wclean)**
- **Update constant/thermophysicalProperties to**



```
thermoType
{
    type            heRhoThermo;
    mixture         pureMixture;
    transport       kineticAr;    // new model
  //transport       const;
    thermo          hKineticAr;   // new model
  //thermo          hConst;
    equationOfState rhoKineticAr;   // new model
  //equationOfState perfectGas;
    specie          specie;
    energy          sensibleEnthalpy;
}
```

- **Run this case with the solver MyThermoFoam (now linked to your new thermophysical library)**
- **Compare the results : do a plotOverLine of temperature for both cases**