6th March 2013

# OpenFOAM 2.2.0: Thermophysical Modelling

## Thermodynamics for Multiphase

In v2.2.0, there has been significant redesign of thermophysical modelling, to enable more flexible handling of multiple materials, *e.g.* in multiphase flows, and conjugate heat transfer. Detailed changes to the thermodynamics are described in subsequent sections below. For multiphase flows, the resulting changes are:*compressibleTwoPhaseEulerFoam*

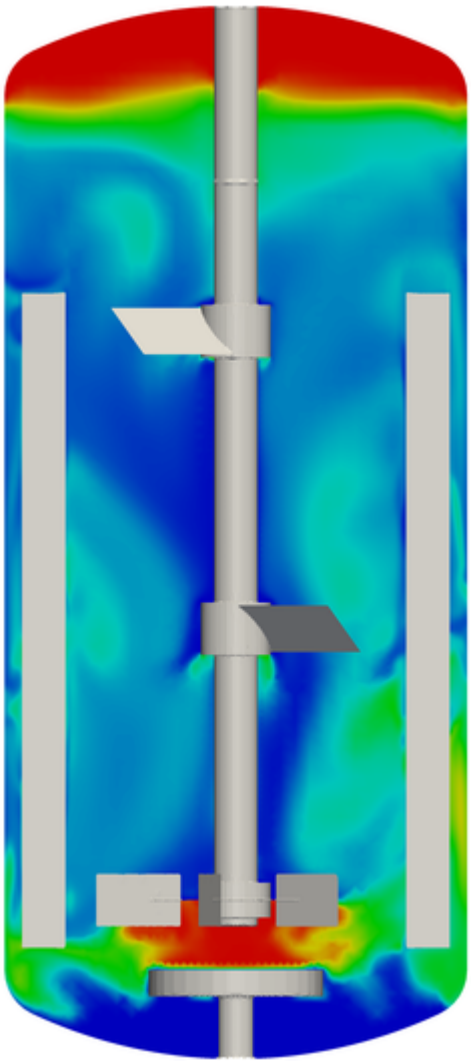
includes updated thermodynamics to use the run-time selectable form of the total energy equation for each phase, so that is possible to select different energy forms (`h` or `e`) separately for the two phases (see below for details).*compressibleInterFoam*

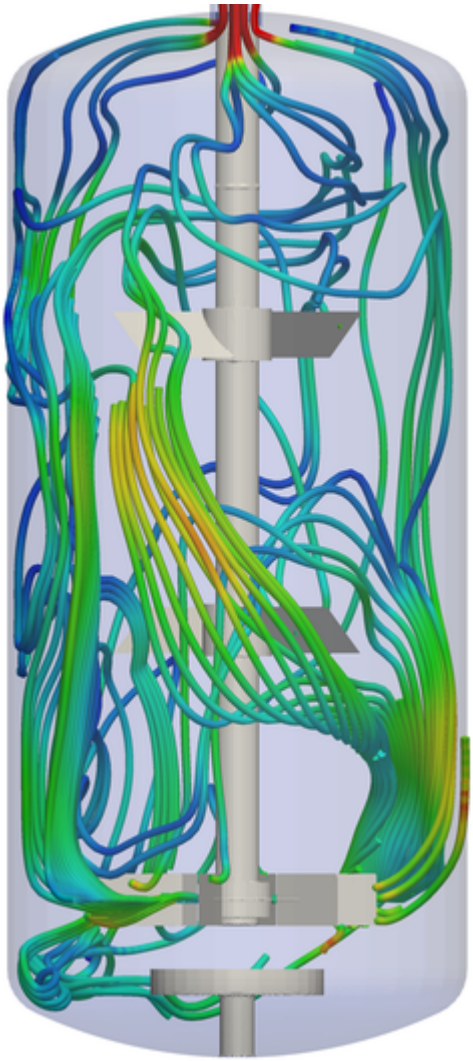
includes updated thermodynamics to use the run-time selectable form of thermodynamics, but solves for the mixture temperature equation, derived from the internal energy equation, in order to maintain boundedness and realisablility.

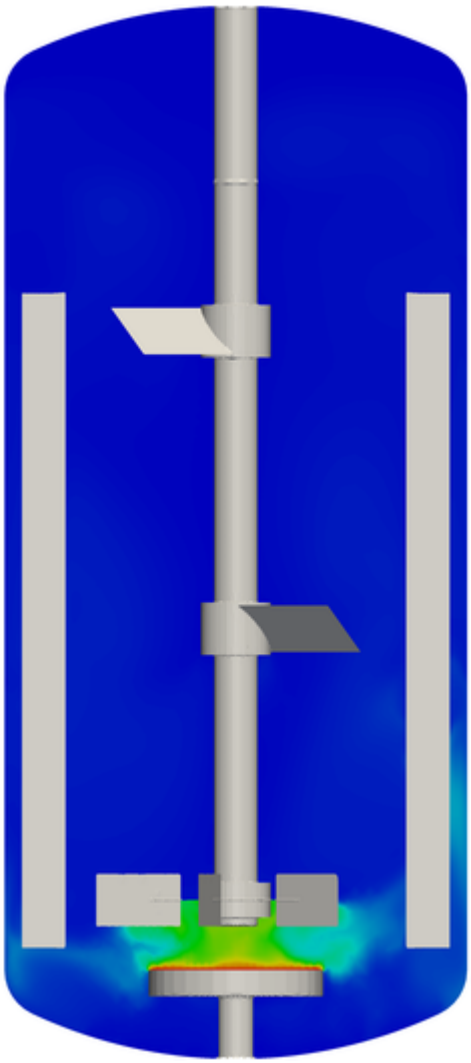
An example simulation with *compressibleTwoPhaseEulerFoam* is shown below of a mixer vessel containing cold water at 300K. Hot oil enters as 1mm droplets at 600K through a sparger towards the base of the vessel. The vessel blades rotate at 50 r.p.m., simulated using multiple reference frames (MRF).

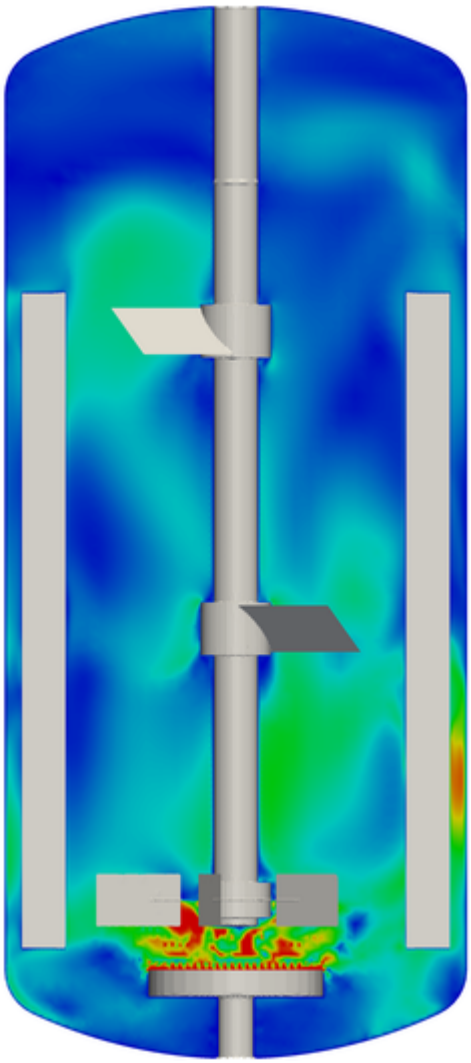
The resulting images show flow results for the dispersed oil phase:

- Top left: phase fraction between 0% (blue) and 100% (red);
- Top right: streamtubes of velocity between 0 (blue) and 1.5 m/s (red);
- Bottom left: temperature, between 300K (blue) and 600K (red);
- Bottom right: velocity, between 0 (blue) and 2 m/s (red).









### Examples

- depth charge
  `$FOAM_TUTORIALS/multiphase/compressibleInterFoam/les/depthCharge3D`
- bubble column
  `$FOAM_TUTORIALS/multiphase/compressibleTwoPhaseEulerFoam/bubbleColumn`

## Run-time Selectable Energy Solution Variable

Prior to the current release of OpenFOAM, each solver that includes an energy equation has been hard-coded to solve for a particular form of energy, *e.g.* internal energy $e$ or enthalpy $h$, and use the corresponding thermodynamics package. This limits the applicability of any solver because different fluids and classes of flow problems are solved optimally using a particular form of energy. For example, $e$ is the preferred form of energy for liquids, $h$ can be preferable for steady-state and some classes transient gaseous problems, particularly combustion.

A new framework has been built in version 2.2.0 to enable solvers to be rewritten in terms of a general form of energy, referred to as `he` (meaning "*h* or *e*"). The framework allows the user to select the form of energy and corresponding thermodynamics package at run-time. In addition, in most solvers the energy equation is written in a form to conserve total energy at convergence, *e.g.* see *rhoPimpleFoam/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(turbulence->alphaEff(), he)
 ==
    fvOptions(rho, he)
);
```

In this code the energy `he` might be chosen to be either internal energy `e` or enthalpy `h` at run-time. Depending on the form of energy, the appropriate source term is selected from the name of the energy through `he.name()`. The choice of energy type is checked against the options the solver support in *createFields.H*:

```
thermo.validate(args.executable(), "h", "e");
```

### Source code

- Energy equation `$FOAM_SOLVERS/compressible/rhoPimpleFoam/EEqn.H`

## Sensible and Absolute Energies

In addition to being able to select the form of energy, *e.g. e* or *h*, the user may select at run-time whether the energy includes heat of formation $\Delta h_f$ or not. We refer to *absolute* energy where heat of

formation is included, and *sensible* energy where it is not. For example absolute enthalpy *h* is related to sensible enthalpy $h_s$ by

$$h = h_s + \sum_i Y_i \Delta h_f^i$$

where $Y_i$ and $h_f^i$ are the mass fraction and heat of formation, respectively, of specie *i*. In most cases, we use the sensible form of energy, for which it is easier to account for energy change due to reactions.

## Dictionary-based Thermodynamics Selection

In previous releases, the selection of a thermodynamics package used a single, complex string derived from the C++ templating used in the creation of the thermodynamic package itself, *e.g.*

```
thermoType hPsiThermo<pureMixture<constTransport<specieThermo<hConstThermo>>>>;
```

To improve clarity, the thermodynamic package selection mechanism has been re-written to use a sub-dictionary of individual entries to select the component parts of the package *e.g.*

```
thermoType
{
    type            hePsiThermo;
    mixture         pureMixture;
    transport       const;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleEnthalpy;
}
```

The syntax include the **`energy`** keyword, in which the user specifies the form of energy to be used in the solution, *e.g.* **`sensibleEnthalpy`**, **`sensibleInternalEnergy`**, **`absoluteEnthalpy`**. If an error is made in this specification or a combination of models is chosen which is not currently available in the libraries linked into the application, the valid options are printed in tabular form *e.g.*

```
--> FOAM FATAL ERROR:
Unknown psiThermo type
thermoType
{
    type            hePsiThermoTest;
    mixture         pureMixture;
```

```
    transport       const;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleEnthalpy;
}

Valid psiThermo types are:

type          mixture                transport   thermo  equationOfState  specie  e

hePsiThermo   homogeneousMixture     const       hConst  perfectGas       specie  s
hePsiThermo   homogeneousMixture     sutherland  hConst  perfectGas       specie  s
hePsiThermo   homogeneousMixture     sutherland  janaf   perfectGas       specie  s
hePsiThermo   inhomogeneousMixture   const       hConst  perfectGas       specie  s
hePsiThermo   inhomogeneousMixture   sutherland  hConst  perfectGas       specie  s
hePsiThermo   inhomogeneousMixture   sutherland  janaf   perfectGas       specie  s
hePsiThermo   multiComponentMixture  const       hConst  perfectGas       specie  s

hePsiThermo   multiComponentMixture  sutherland  janaf   perfectGas       specie  s
hePsiThermo   pureMixture            const       eConst  perfectGas       specie  s
...
...
```

If the particular combination of models is not available in the list, that combination can be compiled into a library or the application (as always).

## Thermal Baffles

OpenFOAM can emulate heat transfer across thin solid structures, or “baffles”. Baffles are represented as **boundary patches of the mesh** and can be created as part of the mesh generation process, *e.g.* with ***snappyHexMesh***. Heat transfer through baffles are therefore implemented in OpenFOAM as boundary conditions, which have been consolidated in the latest version, using the new thermodynamics packages (see above). The available thermal baffle models are included in the following boundary conditions:

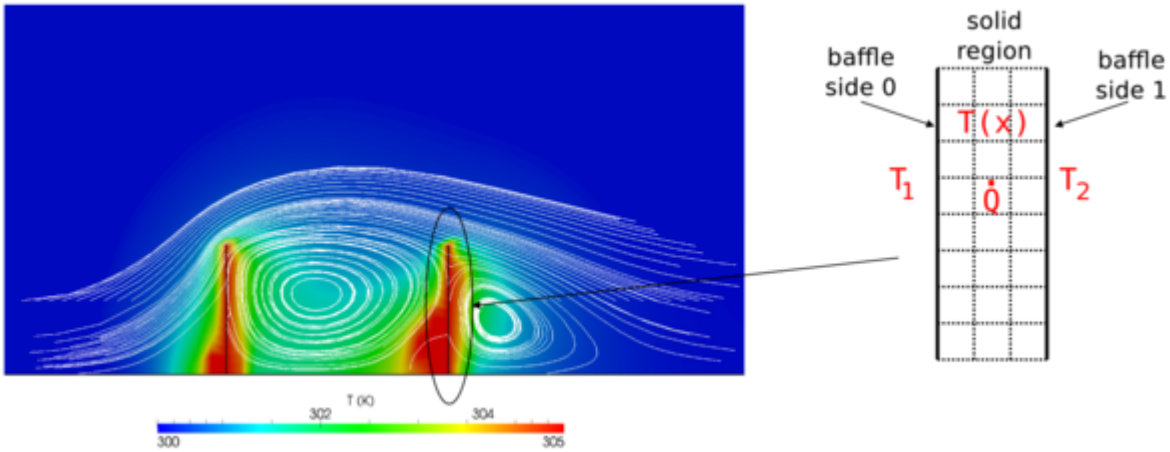
### `thermalBaffle`

This boundary condition can be applied to transfer thermal energy between both sides of the baffle. Heat transfer is solved for across a 3D region created by *extrudeToRegionMesh*, so in effect, the thermalBaffle has zero physical thickness in the flow domain, but non-zero thickness for thermal calculations. Across the encapsulated mesh region the boundary condition solves for a transient 3D heat equation during every solver iteration. The user can now select the thermodynamic models

including radiation, with an option to specify a volumetric heat source (W/mˆ3) within the baffle region. An example setup for the 3D thermal baffle an image from a simple case demonstrating its use are shown below.

### `thermalBaffle1D`

This boundary condition is a 1D approximation of the **`thermalBaffle`** boundary condition solving a steady-state analytical model for heat transfer across the baffle.




```
baffle_master
{
    type            compressible::thermalBaffle;
    neighbourFieldName T;
    kappa           fluidThermo;
    kappaName       none;
    thermalBaffleModel thermalBaffle;
    regionName      baffleRegion;
    infoOutput      no;
    active          yes;
    thermalBaffleCoeffs {}
    thermoType
    {
        type            heSolidThermo;
        mixture         pureMixture;
        transport       constIso;
        thermo          hConst;
        equationOfState rhoConst;
        specie          specie;
        energy          sensibleEnthalpy;
    }
    mixture
    {
        specie
```

```
        {
            nMoles          1;
            molWeight       20;
        }
        transport
        {
            kappa           0.01;
        }
        thermodynamics
        {
            Hf              0;
            Cp              15;
        }
        equationOfState
        {
            rho             80;
        }
    }
    radiation
    {
        radiationModel  opaqueSolid;
        absorptionEmissionModel none;
        scatterModel    none;
    }
    value           uniform 300;
}

baffle_slave
{
    type            compressible::thermalBaffle;
    value           uniform 300;
}
```

## Example

- Circuit board cooling
  `$FOAM_TUTORIALS/heatTransfer/buoyantSimpleFoam/circuitBoardCooling`

## Source Code

- `thermalBaffle` boundary condition
  `$FOAM_SRC/regionModels/thermalBaffleModels/derivedFvPatchFields/thermal`
- `thermalBaffle1D` boundary condition
  `$FOAM_SRC/turbulenceModels/compressible/turbulenceModel/derivedFvPatchF`

## Pressure-work in Enthalpy Equations

Under certain conditions it is preferable to solve for enthalpy but without including the pressure-work term (`dpdt`), *e.g.* when solving steady-state gaseous flow. An optional `dpdt` switch is available in the *thermophysicalProperties* file that will deactivate the pressure-work term if set to `no`; by default, it is set to `yes`.

```
dpdt            no;
```

### Example

- vertical channel
  `$FOAM_TUTORIALS/lagrangian/LTSReactingParcelFoam/verticalChannel`

6th March 2013 Chris Greenshields 2.2.0

**ARTICLES**

Learn Effective CFD

Getting the Best OpenFOAM Training

Productive CFD in OpenFOAM

OpenFOAM User Guide

OpenFOAM Open Day 2018

**RECENT TWEETS**

RT @CFDdirect: Improved specification of dimensionedConstants in the case controlDict file in #OpenFOAM-dev https://t.co/iGYeS9zAMFabout 1 hour ago

RT @CFDdirect: New reaction base class provides reaction specie coefficients without the need for a thermodynamics model in #OpenFOAM-dev h…yesterday

RT @CFDdirect: The streamline function object now writes "age", i.e. the total integration time from the starting point of the streamline,…yesterday

The OpenFOAM Foundation Ltd
Incorporated in England
Company No. 9012603
VAT Reg No. GB 211 0914 63

Directors:
Henry Weller
Chris Greenshields
Cristel de Rouvray

Address:
PO Box 56676
London, W13 3DB
United Kingdom