Implementation of decay heat submodel in Lagrangian library

Manohar Kampili

Institute for Energy and Climate Research (IEK-6),
Forschungszentrum Jülich,
Germany

2017-11-22

Outline

1. **Introduction**
2. Theory
3. Insight into Lagrangian library
4. Creating a new submodel
5. Solver and test case

Introduction

- Aerosol: solid/liquid particles suspended in air/gas
- Nuclear aerosol: Generated in nuclear severe accidents
- Main carriers of the fission products
- Examples: CsI, CsOH, Ag etc ..

Physics of aerosols (1)

Apart from fluid dynamics,

- Particle transport
- Heat transfer between particles and fluid
- Chemistry among particles
- Agglomeration & De-agglomeration

Physics of aerosols (2)

Additional physics involved for nuclear aerosols,

- Decay heat associated with the particles
- Fractal nature
- Shape factors
- Change of chemical composition due to radioactive decay

Selection of solver

In general, flows in reactor containment are weekly compressible and turbulent.

The particle cloud is dynamic in nature along with chemistry.

Description of `reactingParcelFoam`:

Transient solver for compressible, turbulent flow with a reacting, multiphase particle cloud, and optional sources/constraints

Additional physics need to be added to this solver.
Adding decay heat model is the focus of this tutorial.

Outline

1. Introduction
2. **Theory**
3. Insight into Lagrangian library
4. Creating a new submodel
5. Linking it to reactingParcelFoam
6. Test case

Theory - Particle transport

Particles are tracked using Lagrangian method by calculating the forces.

- Sphere Drag
- Lift
- Gravitational force
- Brownian motion

Theory - Heat transfer between particle and fluid (1)

Heat transfer mechanisms are

- Conduction (Fourier's law)
- Convection (Ranz-Marshall correlation)
- Radiation (Blackbody radiation)

For nuclear aerosol particles, additionally

- Decay heat transport

Theory - Heat transfer between particle and fluid (2)

Energy balance within the particle

$$mC_p \frac{dT}{dt} = \dot{Q}$$

m is mass of the particle,
C_p is specific heat of the particle,
T is temperature of the particle,
t is time,
$\dot{Q}$ is overall heat transfer rate from particle to fluid by all modes of heat transfer.

Theory - Heat transfer between particle and fluid (3)

By considering convection alone

$$mC_p\frac{dT}{dt} = hA(T - Tc)$$

h is convective heat transfer coefficient,
A is surface area of the particle,
Tc is carrier fluid temperature.

Analytical solution by variables separable method:

$$T = Tc + (Ti - Tc) * exp(\frac{-6ht}{\rho C_p d})$$

Ti is initial temperature of the particle,
ρ is density of the particle,
d is diameter of the particle.

Theory - Heat transfer between particle and fluid (4)

Convective heat transfer coefficient (h) calculation:

$$h = \frac{(Nu)k}{d}$$

k is thermal conductivity of the particle.

Ranz-Marshall correlation for Nusselt number:

$$Nu = 2 + 0.6 Re^{\frac{1}{2}} Pr^{\frac{1}{3}}$$

Re is particle Reynolds number,
Pr is Prandtl number.

Note: If there is no heat transfer, Nusselt number and heat transfer coefficient would be 0.

Theory - Decay heat model

Decay heat of the radioactive fission product inside aerosol particle is transported by α, β, γ.
In the present case, it is assumed that all the decay heat is absorbed within the particle.

Governing equation:

$$mC_p \frac{dT}{dt} = \dot{Q} + decayPower$$

$decayPower$ is amount of heat generated inside the particle in units $W/particle$

Outline

Classification of parcel clouds

Cloud	submodels
KinematicCloud	Forces on particles
ThermoCloud	KinematicCloud + Heat transfer
KinematicCollidingCloud	KinematicCloud + Collision models
ReactingCloud	ThermoCloud + Phase change + Variable composition (single phase)
ReactingMultiphaseCloud	ReactingCloud + Multiphase composition + Devolatilisation + Surface reactions

Classification of Lagrangian solvers

Solver	Cloud
uncoupledKinematicParcelFoam	KinematicCloud
coalChemistryFoam	ThermoCloud
DPMFoam	KinematicCollidingCloud
reactingParcelFilmFoam	ReactingCloud
reactingParcelFoam	ReactingMultiphaseCloud

Understanding heat transfer model implementation (1)

Heat transfer from particle to fluid is calculated in `calcHeatTransfer` function defined in ThermoParcel.C file.

This file located at
`$FOAM_SRC/lagrangian/intermediate/parcels/Templates/ThermoParcel/`

Since this is a very big function, it is explained in parts here.

This function returns the particle new temperature after the heat transfer from particle to fluid in a specific timestep.

Understanding heat transfer model implementation (2)

```
if (!td.cloud().heatTransfer().active())
{
    return T_;
}
```

It means, if the heat transfer model is not active (specified in `reactingCloud1Properties` in `constant` directory of case), then it returns the original temperatures of the particles as new temperatures.

Understanding heat transfer model implementation (3)

```
scalar htc = td.cloud().heatTransfer().htc(d, Re, Pr, kappa, NCpW);

if (mag(htc) < ROOTVSMALL && !td.cloud().radiation())
{
  return
      max
      (
          T_ + dt*Sh/(this->volume(d)*rho*Cp_),
          td.cloud().constProps().TMin()
      );
}
```

`htc` is heat transfer coefficient calculated from `heatTransfer` model. And the remaining code in `if` loop, calculates new temperatures of the particles if `htc` value is negligible and `radiation model` is not active. These new temperatures are due to enthalpy transfer by phase change `Sh`.

Understanding heat transfer model implementation (4)

```
htc = max(htc, ROOTVSMALL);
const scalar As = this->areaS(d);

scalar ap = Tc_ + Sh/(As*htc);
scalar bp = 6.0*(Sh/As + htc*(Tc_ - T_));
```

`scalar As` is surface area of the particle. `ap, bp` are intermediate values to calculate new temperatures using analytical solution.
For better understanding, let's assume there is no phase change, which means value of `Sh` is 0.

```
ap = Tc_
bp = 6.0*htc*(Tc_ - T_)
```

Where `Tc_` is carrier fluid temperature. `T_` is particle initial temperature.

Understanding heat transfer model implementation (5)

```
if (td.cloud().radiation())
{
    tetIndices tetIs = this->currentTetIndices();
    const scalar Gc = td.GInterp().interpolate(this->position(), tetIs)
    const scalar sigma = physicoChemical::sigma.value();
    const scalar epsilon = td.cloud().constProps().epsilon0();

    // Assume constant source
    scalar s = epsilon*(Gc/4.0 - sigma*pow4(T_));

    ap += s/htc;
    bp += 6.0*s;
}
bp /= rho*d*Cp_*(ap - T_) + ROOTVSMALL;
```

This piece of code changes the values of ap and bp, if the radiation model is active. If it is not active, then `ap = Tc_, bp = 6*htc/(rho*d*Cp)`.

Understanding heat transfer model implementation (6)

```
IntegrationScheme<scalar>::integrationResult Tres =
td.cloud().TIntegrator().integrate(T_, dt, ap*bp, bp);
```

These two lines calculates the new temperatures of the particles using the analytical solution, which is equivalent to integration during a time step.
This `integrate` function is defined in
`$FOAM_SRC/lagrangian/intermediate/IntegrationScheme/Analytical/Analytical.C`
This is exactly equivalent to

$$T = Tc + (Ti - Tc) * exp(\frac{-6ht}{\rho C_p d})$$

Understanding heat transfer model implementation (7)

```
Sph = dt*htc*As;

dhsTrans += Sph*(Tres.average() - Tc_);
```

```
dhsTrans = htc * As * dt * (Tres.average() - Tc_)
```

`Tres.average()` is average particle temperature during timestep dt. `dhsTrans` is heat transferred from particle to fluid. Value of `htc` is obtained from the heat transfer model specified in `reactingCloud1Properties` file in `constant` directory of case.
This is exactly equivalent to

$$Q = hA(T - Tc)dt$$

Understanding heat transfer model implementation (8)

Heat transfer models are located at
`$FOAM_SRC/lagrangian/intermediate/submodels/Thermodynamic/HeatTransferModel`

In `NoHeatTransfer/NoHeatTransfer.C`

```
template<class CloudType>
Foam::scalar Foam::NoHeatTransfer<CloudType>::Nu
(
    const scalar,
    const scalar
) const
{
    return 0.0;
}
```

Nusselt number value returned is always zero, which is justified since there is no heat transfer.

Understanding heat transfer model implementation (9)

Ranz-Marshall heat transfer model is located at
`$FOAM_SRC/lagrangian/intermediate/submodels/Thermodynamic/HeatTransferModel`

In `RanzMarshall.C`

```
template<class CloudType>
Foam::scalar Foam::RanzMarshall<CloudType>::Nu
(
    const scalar Re,
    const scalar Pr
) const
{
    return 2.0 + 0.6*sqrt(Re)*cbrt(Pr);
}
```

Nusselt number value returned is a function of Reynolds number and Prandtl number, which is exactly equivalent to the equation showed in Theory (slide 12).

Outline

Decay heat model implementation (1)

Decay heat of the particle is a source term to the energy balance equation.

$$mC_p \frac{dT}{dt} = \dot{Q} + decayPower$$

In order to implement the decay heat model explained in Theory, calculation `ap`, `bp` should be understood clearly for different modes of heat transfer

For convection alone:

$$ap = Tc$$

$$bp = \frac{6 * htc}{\rho * d * Cp}$$

Decay heat model implementation (2)

In case of convection heat transfer and phase change:

$$ap = Tc + \frac{Sh}{(As)(htc)}$$

$$bp = \frac{6 * htc}{\rho * d * Cp} + 6(\frac{Sh}{As})$$

Where is `Sh` is enthalpy transfer due to phase change.

In case of convection heat transfer, radiation heat transfer and phase change:

$$ap = Tc + \frac{Sh}{(As)(htc)} + \frac{\epsilon * (\frac{Gc}{4} - \sigma * T^4)}{htc}$$

$$bp = \frac{6 * htc}{\rho * d * Cp} + 6(\frac{Sh}{As}) + 6 * \epsilon * (\frac{Gc}{4} - \sigma * T^4)$$

Decay heat model implementation (3)

Following the same trend for decay heat contribution:

$$ap = Tc + \frac{Sh}{(As)(htc)} + \frac{\epsilon * (\frac{Gc}{4} - \sigma * T^4)}{htc} + \frac{decayPower}{(As)(htc)}$$

$$bp = \frac{6 * htc}{\rho * d * Cp} + 6(\frac{Sh}{As}) + 6 * \epsilon * (\frac{Gc}{4} - \sigma * T^4) + \frac{6 * decayPower}{As}$$

`decayPower` is in the units of W / particle.

Steps for developing new submodel

- User copy of Lagrangian library
- Identifying the files that need to be changed
- Modifying one by one file
- Compiling
- Linking it solver

User copy of Lagrangian library

Create an user copy of lagrangian library that is needed along with Make directory. Modify the `files` file in Make directory as following.

```
OF1706+

cp -r $FOAM_SRC/lagrangian/intermediate $WM_PROJECT_USER_DIR/src

cd $WM_PROJECT_USER_DIR/src/lagrangian/intermediate/Make

sed -i s/FOAM_LIB_BIN/FOAM_USER_LIB_BIN/g files

sed -i s/liblagrangianIntermediate/libmyLagrangianIntermediate/g files
```

Identifying files for modifying

In this tutorial, directory structure of heat transfer model is replicated for developing decay heat models.
Let's find out `heatTransfer` keyword using `grep` command.

```
cd $WM_PROJECT_USER_DIR/src/lagrangian/intermediate
grep -r -n heatTransfer .
```

This will result different instances of `heatTransfer` in following files:

- ThermoCloud.H
- ThermoCloudI.H
- ThermoCloud.C
- ThermoParcel.C
- ReactingMultiphaseParcel.C
- ReactingParcel.C
- HeatTransferModelNew.C
- HeatTransferModel.H

Modifications in ThermoCloud.H (1)

```
vi $WM_PROJECT_USER_DIR/src/lagrangian/intermediate/clouds/
Templates/ThermoCloud/ThermoCloud.H
```

- First modification is forward declaration of `DecayHeatModel` class (just below the HeatTransferModel) as following:

```
template<class CloudType>
class HeatTransferModel;

template<class CloudType>
class DecayHeatModel;
```

Modifications in ThermoCloud.H (2)

- Second modification is adding `decayHeatModel_` in protected member data

```
//- Heat transfer model
autoPtr<HeatTransferModel<ThermoCloud<CloudType>>>
heatTransferModel_;
//- decay heat model
autoPtr<DecayHeatModel<ThermoCloud<CloudType>>>
decayHeatModel_;
```

- Third modification is adding `decayHeat()` member function

```
// Sub-models
//- Return reference to heat transfer model
    inline const HeatTransferModel<ThermoCloud<CloudType>>&
    heatTransfer() const;
//- Return reference to decay heat model
    inline const DecayHeatModel<ThermoCloud<CloudType>>&
    decayHeat() const;
```

Modification in ThermoCloudI.H (1)

- Defining the inline function `decayHeat()`:

```
template<class CloudType>
inline const Foam::HeatTransferModel<Foam::ThermoCloud<CloudType>>&
Foam::ThermoCloud<CloudType>::heatTransfer() const
{
    return heatTransferModel_;
}

template<class CloudType>
inline const Foam::DecayHeatModel<Foam::ThermoCloud<CloudType>>&
Foam::ThermoCloud<CloudType>::decayHeat() const
{
    return decayHeatModel_;
}
```

Modifications in ThermoCloud.C (1)

- First modification is including DecayHeatModel.H in the header files

```
#include "HeatTransferModel.H"
#include "DecayHeatModel.H"
```

- Second modification is adding `decayHeatModel_` in `cloudReset` member function

```
    heatTransferModel_.reset(c.heatTransferModel_.ptr());

    decayHeatModel_.reset(c.decayHeatModel_.ptr());
```

Modifications in ThermoCloud.C (2)

- Third modification is in the first member function, as following:

```
heatTransferModel_.reset
(
    HeatTransferModel<ThermoCloud<CloudType>>::New
    (
        this->subModelProperties(),
        *this
    ).ptr()
);

decayHeatModel_.reset
(
    DecayHeatModel<ThermoCloud<CloudType>>::New
    (
        this->subModelProperties(),
        *this
    ).ptr()
);
```

Modifications in ThermoCloud.C (3)

- Modification in first constructor

```
heatTransferModel_(nullptr),
decayHeatModel_(nullptr),
```

- Modification in second constructor

```
heatTransferModel_(c.heatTransferModel_->clone()),
decayHeatModel_(c.decayHeatModel_->clone()),
```

- Modification in third constructor

```
heatTransferModel_(nullptr),
decayHeatModel_(nullptr),
```

Modification in ThermoParcel.C

In `calcHeatTransfer` function, modify the few lines in the middle as following:

```
..
..
htc = max(htc, ROOTVSMALL);
const scalar As = this->areaS(d);
const scalar Vs = this ->volume(d);
scalar decay= td.cloud().decayHeat().decayPower();

scalar ap = Tc_ + (Sh/(As*htc)) + (decay/(As*htc));
scalar bp = 6.0*(Sh/As + htc*(Tc_ - T_)+(decay/As));
if (td.cloud().radiation())
{
    tetIndices tetIs = this->currentTetIndices();
    const scalar Gc = td.GInterp().interpolate(this->position(), tetIs)
..
..
```

Creating DecayHeatModel directory

HeatTransferModel directory is taken as basis for creating DecayHeatModel directory. In these files, replace all the instances of `heatTransfer` with `decayHeat`, `HeatTransfer` with `DecayHeat`, `htm` with `dhm`, `RanzMarshall` with `constantDecayHeat`.

```
cd $WM_PROJECT_USER_DIR/src/intermediate/submodels/Thermodynamic

cp -r HeatTransferModel DecayHeatModel

cd DecayHeatModel

sed -i s/heatTransfer/decayHeat/g *

sed -i s/HeatTransfer/DecayHeat/g *

sed -i s/htm/dhm/g *

sed -i s/RanzMarshall/constantDecayHeat/g *
```

DecayHeatModel directory

In this `DecayHeatModel` directory, we have 3 sub-directories with following files:

- DecayHeatModel
  - DecayHeatModel.H
  - DecayHeatModel.C
  - DecayHeatModelNew.C
- NoDecayHeat
  - NoDecayHeat.H
  - NoDecayHeat.C
- constantDecayHeat
  - constantDecayHeat.H
  - constantDecayHeat.C

Modifications in DecayHeatModel.H and DecayHeatModel.C

- Declare a member function `decayPower()` and delete old member functions if any in `DecayHeatModel.H`.

```
// Member Functions
    // Evaluation
       //- Retrun decay Power
        virtual scalar decayPower() const = 0;
```

- Define the member function `decayPower()` as following in `DecayHeatModel.C`:

```
// * * * * * * *  Member Functions  * * * * * *  //

template<class CloudType>
Foam::scalar Foam::DecayHeatModel<CloudType>::decayPower() const
{
   const scalar decayPower_ = this -> decayPower();
   return decayPower_;
}
```

Modification in DecayHeatModelNew.C

Change the error message as following:

```
FatalErrorInFunction
            << "Unknown decay heat model type"
            << modelType << nl << nl
            << "Valid decay heat model types are:" << nl
            << dictionaryConstructorTablePtr_->sortedToc()
            << exit(FatalError);
```

Modifications in NoDecayHeat.H and NoDecayHeat.C

- Declare a member function `decayPower()` and delete old member functions in `NoDecayHeat.H`.

```
// Member Functions

//- Flag to indicate whether model activates heat transfer model
virtual bool active() const;

//- decay Power
virtual scalar decayPower() const;
```

- Define the member function `decayPower()` as following in `NoDecayHeat.C`: This function should return zero always.

```
// * * * * * * *  Member Functions  * * * * * *  //

template<class CloudType>
Foam::scalar Foam::NoDecayHeat<CloudType>::decayPower() const
{
    return 0.0;
}
```

Modifications in constantDecayHeat.H

- Declare a new member function decayPower() and delete old member functions in constantDecayHeat.H.

```
// Member Functions

//- Flag to indicate whether model activates heat transfer model
virtual bool active() const;

//- decay Power
virtual scalar decayPower() const;
```

- Declare a new member data decayPower_

```
template<class CloudType>
class constantDecayHeat
:
    public DecayHeatModel<CloudType>
{
    //private data

    scalar decayPower_;
```

Modification in constantDecayHeat.C (1)

- Initialize the decayPower_ with input value in the costructor as following:

```
// * * * * * * * * * * * *  Constructors   * * * * * * * * * * * //

template<class CloudType>
Foam::exponentialDecayHeat<CloudType>::constantDecayHeat
(
    const dictionary& dict,
    CloudType& cloud
)
:
    DecayHeatModel<CloudType>(dict, cloud, typeName),
    decayPower_(readScalar(this->coeffDict().lookup("decayPower")))
{}
```

Modification in constantDecayHeat.C (2)

- Define the member function decayPower() as following in constantDecayHeat.C to return private member data decayPower_

```
// * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * *

template<class CloudType>
bool Foam::constantDecayHeat<CloudType>::active() const
{
    return false;
}

template<class CloudType>
Foam::scalar Foam::constantDecayHeat<CloudType>::decayPower() const
{
    return decayPower_;
}
```

Compiling decay heat models

Heat transfer models are compiled using `makeParcelHeatTransferModels.H` located at

`$WM_PROJECT_USER_DIR/src/intermediate/parcels/include/`

Following the same trend, create a file `makeParcelDecayHeatModels.H` in the same directory.

Code in makeParcelDecayHeatModels.H

```
#ifndef makeParcelDecayHeatModels_H
#define makeParcelDecayHeatModels_H
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
#include "NoDecayHeat.H"
#include "constantDecayHeat.H"
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//

#define makeParcelDecayHeatModels(CloudType)                \
                                                             \
    makeDecayHeatModel(CloudType);                          \
                                                             \
    makeDecayHeatModelType(NoDecayHeat, CloudType);         \
    makeDecayHeatModelType(constantDecayHeat, CloudType);

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
#endif
```

Linking ParcelDecayHeatModels to ThermoParcelSubmodels

In order to link the decay heat models to ThermoParcelSubmodels, there are two modifications required in **makeBasicThermoParcelSubmodels.C**, located in the directory at

`$WM_PROJECT_USER_DIR/src/lagrangian/intermediate/parcels/derived/basicThermoParcel/`

- Including header file `makeParcelDecayHeatModels.H`

```
// Thermodynamic
#include "makeParcelHeatTransferModels.H"
#include "makeParcelDecayHeatModels.H"
```

- Adding thermo variant of parcelDecayHeatModels as following:

```
// Thermo sub-models
makeParcelHeatTransferModels(basicThermoCloud);

// decay heat models
makeParcelDecayHeatModels(basicThermoCloud);
```

Linking ParcelDecayHeatModels to ReactingMultiphaseParcelSubmodels

To link the decay heat models to `ReactingMultiphaseParcelSubmodels`, there are two modifications required in **makeBasicReactingMultiphaseParcelSubmodels.C**, located in the directory at

`$WM_PROJECT_USER_DIR/src/lagrangian/intermediate/parcels/derived/basicReactingMultiphaseParcel/`

- Including header file `makeParcelDecayHeatModels.H`

```
// Thermodynamic
#include "makeParcelHeatTransferModels.H"
#include "makeParcelDecayHeatModels.H"
```

- Adding thermo variant of parcelDecayHeatModels as following:

```
// Thermo sub-models
makeParcelHeatTransferModels(basicReactingMultiphaseCloud);

// decay heat models
makeParcelDecayHeatModels(basicReactingMultiphaseCloud);
```

Modifications in Make/options file for compilation

Modifications required in `options` file in order to include all the files related to `decayHeatModels`.

```
EXE_INC = \
-I. \
-Ilagrangian/intermediate/lnInclude \
..
..
```

and

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

Outline

Solver and test case

Creating myReactingParcelFoam:

```
cp -r $FOAM_SOLVERS/lagrangian/reactingParcelFoam $WM_PROJECT_USER_DIR/
applications

cd $WM_PROJECT_USER_DIR/applications

rm -rf reactingParcelFoam/simpleReactingParcelFoam

sed -i s/reactingParcelFoam/myReactingParcelFoam/g *

sed -i s/FOAM_APPBIN/FOAM_USER_APPBIN/g *
```

Now user version of `reactingParcelFoam` is ready to compile with name `myReactingParcelFoam`.

Compiling solver with Lagrangian library

- Copy the new Lagrangian library into solver:

```
cp -r $WM_PROJECT_USER_DIR/src/lagrangian $WM_PROJECT_USER_DIR/
applications/myReactingParcelFoam

cd $WM_PROJECT_USER_DIR/applications/myReactingParcelFoam
```

- Modify options file in Make directory of `myReactingParcelFoam`

```
EXE_INC = \
-I. \
-Ilagrangian/intermediate/lnInclude \
```

and

```
EXE_LIBS = \
-L$(FOAM_USER_APPBIN) \
-lmyLagrangianIntermediate \
```

- Compile

```
wmake all
```

verticalChannel case without decayHeat (1)

```
cd $WM_PROJECT_USER_DIR/run

cp -r $FOAM_TUTORIALS/lagrangian/reactingParcelFoam/verticalChannel
verticalChannelWithoutDecay

cd verticalChannelWithoutDecay
```

- Modifications in `constant/reactingCloud1Properties` file:

```
constantProperties
{
    rho0            4510;
    T0              473;
    Cp0             200;

    constantVolume  true;
}
```

verticalChannel case without decayHeat (2)

- Modifications in `injectionModels` dictionary of `constant/reactingCloud1Properties` file:

```
injectionModels
{
    model1
    {
        type            patchInjection;
        massTotal       2.36143e-5;
        parcelBasisType fixed;
        SOI             0.0;
        duration   0.1;
        nParticle 1;
        parcelsPerSecond 100;
        patch inletCentral;
        U0              (0 1.08 0);
        flowRateProfile constant 1;
        sizeDistribution
        {
            type uniform;
            uniformDistribution
            {
                minValue       1e-3;
                maxValue       1e-3;
            }
        }
     }
```

verticalChannel case without decayHeat (3)

- Modifications in `constant/reactingCloud1Properties` file:

```
 decayHeatModel none;

 singleMixtureFractionCoeffs
{
    phases
    (
        gas
        {
        }
        liquid
        {
        }
        solid
        {
            ash 1;
        }
    );
    YGasTot0        0;
    YLiquidTot0     0;
    YSolidTot0      1;
}
```

verticalChannel case with decayHeat

```
cd $WM_PROJECT_USER_DIR/run

cp -r verticalChannelWithoutDecay verticalChannelWithDecay

vi verticalChannelWithDecay/constant/reactingCloud1Properties
```

- Modifications in `constant/reactingCloud1Properties` file:

```
decayHeatModel constantDecayHeat;

constantDecayHeatCoeffs
{
    decayPower 100000;
}
```

Here decayPower 100000 W/particle is taken as a random value to see the effect of `decayHeat` in 0.1 s. Run both the cases with myReactingParcelFoam solver

```
cd verticalChannelWithoutDecay

myReactingParcelFoam >& log &

cd ..

cd verticalChannelWithDecay

myReactingParcelFoam >& log &
```

PostProcessing with foamLog and gnuplot (1)

Tracking the highest and lowest temperatures of the particles

- copy paste the foamLog.db file to case

```
cp $WM_PROJECT_DIR/bin/tools/foamLog.db verticalChannelWithoutDecay

vi verticalChannelWithoutDecay/foamLog.db
```

- Add the following lines in the end to existing foamLog.db file:

```
#particle minimum Temp
particleMinTemp/Temperature min/max             =

#particle maximum Temp
particleMaxTemp/Temperature min/,
```

```
cp verticalChannelWithoutDecay/foamLog.db vertcialChannelWithDecay
```

- Running foamLog in both cases

```
foamLog -case verticalChannelWithoutDecay
foamLog -case verticalChannelWithDecay
```

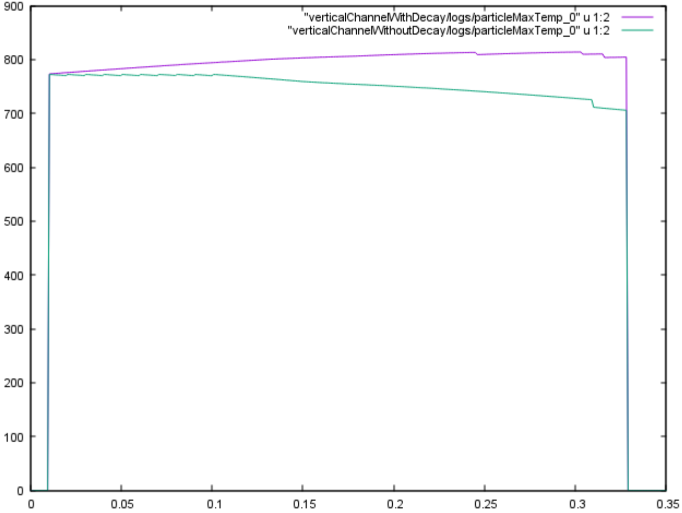
PostProcessing with foamLog and gnuplot (2)

Tracking the highest and lowest temperatures of the particles

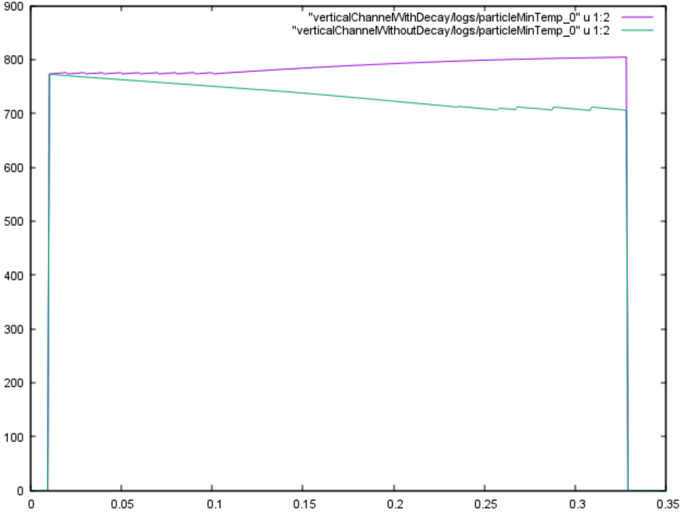
- Plotting with gnuplot

```
gnuplot

plot "verticalChannelWithoutDecay/postProcessing/logs/particleMinTemp.xy" u 1:2 w l,
"verticalChannelWithDecay/postProcessing/logs/particleMinTemp.xy" u 1:2 w l

plot "verticalChannelWithoutDecay/postProcessing/logs/particleMaxTemp.xy" u 1:2 w l,
"verticalChannelWithDecay/postProcessing/logs/particleMaxTemp.xy u 1:2 w l
```

Maximum temperatures comparison



Minimum temperatures comparison



Thank you