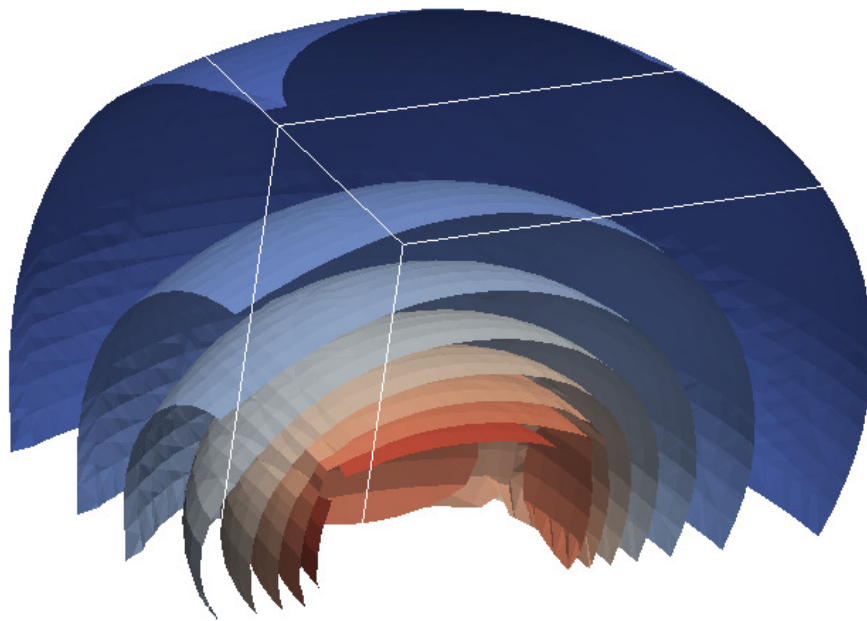Final Assignment
for the course
«CFD with OpenSource Software»

Radiation heat transfer in OpenFoam

Student: Alexey Vdovin, 850408-3737, vdovin@student.chalmers.se

Göteborg, Chalmers University of Technology, 10.12.2009

Contents

1.0 Introduction

Radiation heat transfer process is extremely important. It is concerned with the exchange of thermal energy between two or more bodies. Heat transfer through radiation takes place in form of electromagnetic waves mainly in the infrared region. In contrast with a heat transfer by conduction and convection it requires no medium. The intermediaries are photons which travel at the speed of light.

The heat transferred into or out of an object by thermal radiation is a function of several components. These include its surface reflectivity, emissivity, surface area, temperature, and geometric orientation with respect to other thermally participating objects. In turn, an object's surface reflectivity and emissivity is a function of its surface conditions (roughness, finish, etc.) and composition.

For a huge amount of different purposes it is important to be able to calculate or simulate radiation heat transfer between bodies. To do it, there are several radiation models of different complexity. In this report one can find the information about radiation heat transfer implementation in OpenFoam (Open Field Operation and Manipulation) CFD Toolbox. Version of software used is 1.6.x. This report will answer some questions about how radiation heat transfer can be added to the case, what different radiation heat transfer models are available, what their advantages and disadvantages are. In addition a deeper look at one of the model (P1 model) files will be taken.

2.0 Addition of the radiation heat transfer to the solver

Having different solvers like *buoyantSimpleFoam* that is used for steady-state solving for buoyant, turbulent flow of compressible fluids for ventilation and heat transfer, and *buoyantSimpleRadiationFoam* that is almost the same solver just with added radiation heat transfer, it becomes possible to find out how to add radiation to the existing OpenFoam solver.

The only difference in *buoyantSimpleRadiationFoam.C* in comparison to *buoyantSimpleFoam.C* is two following lines:

> *#include "radiationModel.H"*
>
> *#include "createRadiationModel.H"*

that includes header files for radiation to the compilation.

All .h files are the same for the solver with radiation except *hEqn.H* that is modified by addition of the term *Sh(thermo)* to the energy equation:

> *fvScalarMatrix hEqn*
> *(fvm::div(phi, h)*
> * - fvm::Sp(fvc::div(phi), h)*
> * - fvm::laplacian(turbulence->alphaEff( ), h)*
> * ==*
> *   fvc::div(phi/fvc::interpolate(rho)*fvc::interpolate(p))*
> * - p*fvc::div(phi/fvc::interpolate(rho))*
> * + radiation->Sh(thermo)*
> *);*

This function will be discussed when it comes to the radiation models discussion.
Also there is one extra line added to the *hEqn.H*:

> *radiation->correct( );*

That is used for recalculating radiation as it will be shown further.

Before compilation of the solver it is also needed to add some lines to the *Make/options* file:

> *EXE_INC = *
> *   -I$(LIB_SRC)/thermophysicalModels/radiation/lnInclude *
> *EXE_LIBS = *
> *   -lradiation *

After this modification is done and the solver is compiled, it is ready to solve tasks that include radiation. The only thing left is to choose the radiation model to be used and to add some extra variables, needed by the chosen model, to the existing code.

3.0 Radiation models in the OpenFoam

There are three radiation modes available in OpenFoam. They and the abstract class for them will be discussed in this chapter.

3.1 Abstract class for radiation models

If one opens *radiation\radiationModel\radiationModel\radiationModel.C* an abstract class for *radiationModel* can be found.

The most important here are member functions *Rp(), Ru(),* and the one that generates a source term of the enthalpy equation - *Sh().*

For the calculation of *Sh()* member function it can be found:

```
volScalarField& h = thermo.h();
const volScalarField cp = thermo.Cp();
const volScalarField T3 = pow3(T_);
return
(
   Ru()
  - fvm::Sp(4.0*Rp()*T3/cp, h)
  - Rp()*T3*(T_ - 4.0*h/cp)
);
```

Ru() and Rp() member functions are defined in different radiation models. Here one can see that *fvm::Sp* function is used, it is mostly done to redistribute source terms to make the main matrix more diagonally dominant. In terms of equation being solved one can write:

$$Sh() = Ru() - 4Rp() * \frac{T^3 h}{C_P} - Rp()T^4 + 4Rp() * \frac{T^3 h}{C_P}$$

$$Sh() = Ru() - Rp()T^4$$

In other words adding and subtracting the same function does not affect the result but this significantly improves numerical stability.

Also in the abstract class one can find *correct()* member function which was seen before. Here is the code for it:

```
void Foam::radiation::radiationModel::correct()
{   if (!radiation_)
   {      return;   }
```

> *if (time_.timeIndex( ) % solverFreq_ == 0)*
>
> *{   calculate( );   }   }.*

As it can be seen in the code this function calls *calculate()* member function that is used for radiation calculation and differs for different radiation models.

3.2 NoRadiation model

First of all for testing purposes and as a fast way of disabling radiation heat transfer in the calculation there is a *noRadiation* model that puts member functions *Ru()* and *Rp()* equal to zero, so additional term for the enthalpy equation *Sh()* is becoming 0.

3.3 Finite Volume Discrete Ordinates Model

fvDOM model is another radiation model implemented in OpenFoam. In this model the radiative transfer equation is solved for a discrete number of finite solid angles σ_s.

Advantages of fvDOM model:
- Conservative method leads to heat balance for coarse discretization, the accuracy can be increased by using a finer discretization,
- Most comprehensive radiation model: Accounts for scattering, semi-transparent media, specular surfaces, and wavelength-dependent transmission using banded-gray option.

Limitations of fvDOM model:
- Solving a problem with a large number of ordinates is CPU-intensive.

3.4 P1 model

The last model implemented in OpenFooam for radiation heat transfer is P1 model. The main assumption of this model is that the directional dependence in radiative transfer equation is integrated out, resulting in a diffusion equation for incident radiation.

Advantages of P1 model:
- Radiative transfer equation easy to solve with little CPU demand,
- Includes effect of scattering, effects of particles, droplets, and soot can be included,
- Works reasonably well for applications where the optical thickness is large, $\tau = a * L > 3$, where L = distance between objects(e.g. model can be used in combustion).

Limitations of P1 model:
- Assumes all surfaces are diffuse,
- May result in loss of accuracy (depending on the complexity of the geometry) if the optical thickness is small,
- Tends to overpredict radiative fluxes from localized heat sources or sinks.

3.5 Radiation constants

There is only one constant defined for all radiation models in OpenFoam and it can be found in file *radiation\radiationConstants\radiationConstants.C:*

```
const Foam::dimensionedScalar Foam::radiation::sigmaSB
(
    Foam::dimensionedConstant
    (
        "sigmaSB",
        dimensionedScalar
        (
            "sigmaSB",
            dimensionSet(1, 0, -3, -4, 0, 0, 0),
            5.670E-08
        )
    )
);
```

This is a Stefan-Boltzmann constant that is a constant of proportionality in the Stefan–Boltzmann law: the total energy radiated per unit surface area of a black body in unit time is proportional to the fourth power of the thermodynamic temperature. The value for it is $\sigma_{SB} = 5.670 \cdot 10^{-8} \; W/_{m^2} \cdot K^4$.

3.6 Scatter models

If including the effect of scattering the model should be defined. To find available scatter models one should look at *radiation\submodels\scatterModel* folder. Currently there is only one scatter model available, it is called *constantScatter*. If looking at the code for it, one can find that there is only one member function:

```
Foam::tmp<Foam::volScalarField>
Foam::radiation::constantScatter::sigmaEff() const
{
    return tmp<volScalarField>
    (
        new volScalarField
        (
            IOobject
            (
                "sigma",
                mesh_.time().timeName(),
                mesh_,
                IOobject::NO_READ,
                IOobject::NO_WRITE,
                false
            ),
```

```
        mesh_,
        sigma_*(3.0 - C_)
    ) );
}
```

So this *constantScatter* model has an input of two coefficients *sigma* and *C* and returns

$$sigmaEff = sigma(3.0 - C)$$

This value is used for calculating incident radiation intensity in radiation models.

3.7 Absorption-Emission Models

As radiation models are dealing with absorption and emission effects, coefficient for them should be defined. There are three coefficients that are used in radiation models for that purpose:

$a \left[1/m \right] -$ *absorbtion coefficient,*

$e \left[1/m \right] -$ *emission coefficient,*

$E \left[W/m^3 \right] -$ *emission contribution.*

For calculation of them there are several absorption-emission models available in the OpenFoam, they can be found in the *radiation\submodels\absorptionEmissionModel* folder. In this tutorial *constantAbsorptionEmission* model is going to be used. For this model these three coefficients are defined in the radiation model and they are considered to have a constant value.

3.8 Radiation boundary conditions

For solving the case, boundary conditions for incident radiation intensity *G* need to be defined. There are a number of different boundary conditions that can be used; they can be found at *radiation\derivedFvPatchFields* folder.

Here one can find *MarshakRadiation* boundary condition that is going to be used in this tutorial. It uses calculated temperature from the case as a temperature for calculating radiation intensity. Another option that can be found in the folder is *MarshakRadiationFixedT*. For this boundary condition radiation temperature should be specified. And then this value should be used for calculating *G*.

4.0 P1 radiation model files

Having checked *radiation/radiationModel/P1/* folder one can find all files that are related to the P1 radiation model and its implementation in OpenFoam.

By taking a look at *P1.C* file the model behavior can be understood. Firstly, *gamma* value is being calculated using absorption coefficient *a* and the value of *sigmaEff* that is coming from the scatter model as it was described before.

```
// Construct diffusion
   const volScalarField gamma
   (
      IOobject
      (
         "gammaRad",
         G_.mesh().time().timeName(),
         G_.mesh(),
         IOobject::NO_READ,
         IOobject::NO_WRITE
      ),
      1.0/(3.0*a_ + sigmaEff)
   );
```

In terms of formulas the equation for gamma will be:

$$gamma = \frac{1}{3a + sigmaEff}$$

Secondly, incident radiation intensity *G* transport equation is being solved:

```
solve
(
   fvm::laplacian(gamma, G_)
  - fvm::Sp(a_, G_)
 ==
  - 4.0*(e_*radiation::sigmaSB*pow4(T_) + E_)
);
```
Or in terms of formulas:

$$\nabla \cdot gamma \nabla G - aG = -4(e\sigma_{SB}T^4 + E)$$

Thirdly, *Ru()* and *Rp()* member function definitions can be found. For the *Rp()* there is:

```
4.0*absorptionEmission_->eCont()*radiation::sigmaSB
```

or in terms of equations:

$$Rp(\ ) = 4e\sigma_{SB}$$

where σ_{SB} is Stefan-Boltzmann constant defined at the *radiationConstants*.

As for the *Ru( )* function one can find:

> *return a*G - 4.0*E;*

or in terms of equations:

$$Ru(\ ) = aG - 4E$$

Finally the additional term for the enthalpy equation *Sh( )* for this model will look like:

$$Sh(\ ) = Ru(\ ) - Rp(\ )T^4$$

or

$$Sh(\ ) = aG - 4(e\sigma_{SB}T^4 + E)$$

In other words this additional term represents the amount of radiation emitted subtracted from the amount of irradiation absorbed.

5.0 Case set up and modification of *hotRoom* tutorial to make it work with radiation

5.1 Case set up

According to \constant\polyMesh\blockMeshDict there is a room with dimensions of 10x6x2 meters and a box that will represent a heat source 1x1x0.5 meters in dimensions. The temperature of walls, ceiling and floor is set to 300 K and the temperature of heater to 500 K.
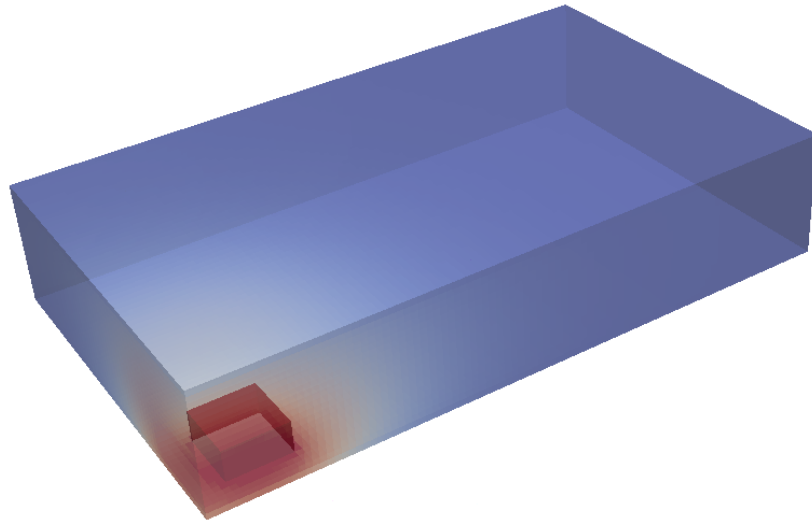

Figure 1. HotRoom case setup

5.2 Modification of *hotRoom* tutorial

To make the case work with radiation not only the different solver should be used (*buoyantSimpleRadiationFoam* instead of *buoyantSimpleFoam* in this case), but several changes need to be done either.

Firstly there is \constant\radiationProperties file, where one can find following lines:

radiation   on;
radiationModel P1;
…
absorptionEmissionModel constantAbsorptionEmission;
constantAbsorptionEmissionCoeffs
{   a   a [0 -1 0 0 0 0 0] 0.5;
*   e   e [0 -1 0 0 0 0 0] 0.5;*
*   E   E [1 -1 -3 0 0 0 0] 0;}*

scatterModel   constantScatter;

constantScatterCoeffs
{   sigma   sigma [0 -1 0 0 0 0 0] 0;
* C   C [0 0 0 0 0 0 0] 0;*
}

As it can be seen radiation model is defined here, it can be set to be "none" to use *NoRadiation* model ,"P1" for *P1* model or ""fvDOM" for finite volume discrete ordinates model. Few lines after it an absortionEmissionModel is selected, so the model with constant coefficients for absorbtion and emission is to be used. The equal value of $0.5\,m^{-1}$ is set for both of them. The emission contribution *E* is set to 0. Also here *scatterModel* is selected to be *constantScatter* and coefficients for it are defined.

Another thing to be done is to add boundary conditions for the incident radiation intensity *G*. So at */0/* folder file */0/G* should be created with following lines incide:

FoamFile
{   version   2.0;
*   format   ascii;*
*   class   volScalarField;*
*   object   G;}*
*// * //*
dimensions   [1 0 -3 0 0 0];
internalField   uniform 0;

boundaryField
{
*   floor*
*   {   type   MarshakRadiation;*
*   T   T;*
*   emissivity   1;*
*   value   uniform 0;   }*

*   fixedWalls*
*   {   type   MarshakRadiation;*
*   T   T;*
*   emissivity   1;*
*   value   uniform 0;   }*

*   ceiling*
*   {   type   MarshakRadiation;*
*   T   T;*
*   emissivity   1;*
*   value   uniform 0;*
*   }*

*   box*
*   {   type   MarshakRadiation;*

```
    T            T;
    emissivity     1;
    value         uniform 0;
  }
}
```

As it can be seen, same conditions are set for all faces for room and for heater.
In addition to everything said before *system/fvSolution* file should be modified. Firstly the solver for G variable should be defined:

```
solvers
{
  ...
  G
  {
    solver        PCG;
    preconditioner  DIC;
    tolerance      1e-05;
    relTol        0.1;
  }
}
```

and relaxation factor value should be set:

```
relaxationFactors
{
...
G            0.7;
}
```

As all this changes are done the case is ready to work with the radiation heat transfer. So the results for the simulation with and without radiation are compared in the following chapter.

6.0 Results and discussions

Several test cases were run to figure out how the adding of radiation affects the simulation. In the following figure one can see the results for the case described without radiation.
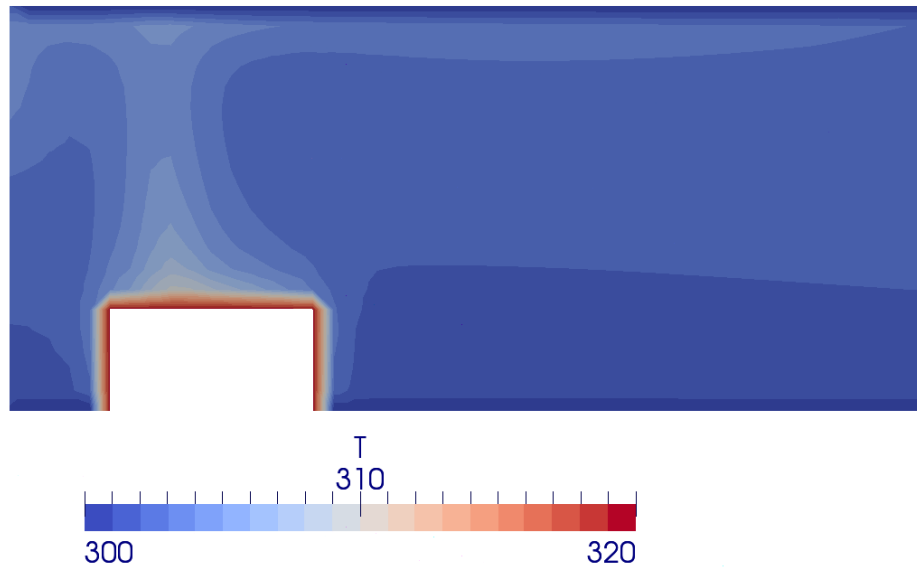

Figure 2. Temperature gradient without radiation heat transfer

Here the hot air moving upwards can be seen, but the temperature gradient is not big, in just few centimeters from the heat source the temperature drops from 500 K almost to the temperature of surrounding air or 300 K. This will differ a lot when one uses the model with P1 radiation model added. The result can be seen in the figure 3.
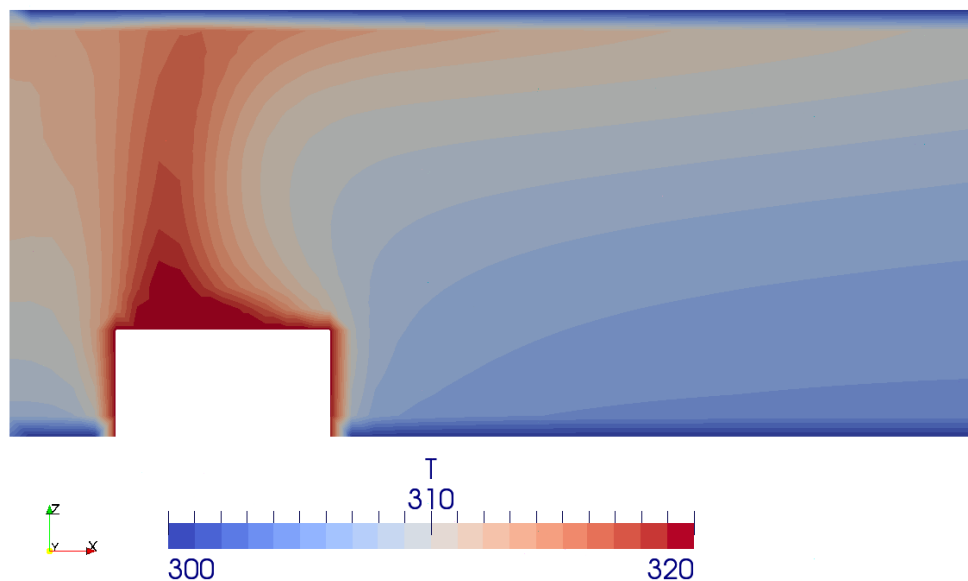

Figure 3. Temperature gradient with radiation heat transfer

The temperature scale for these two pictures is the same so one can see the big difference. As the temperature for the air is increased the air speeds will grow so all case results will be different.

For better understanding of the P1 model that was used, it can be valuable to plot the distribution of incident radiation intensity G. This is done in the figure 4.
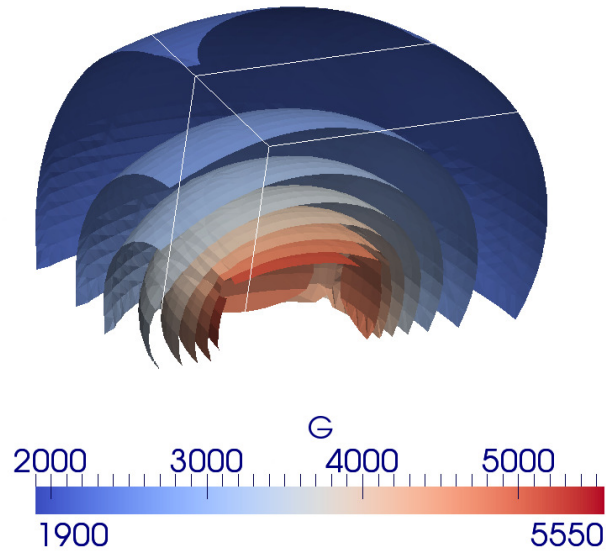

Figure 4. Incident radiation intensity

As one can see G is inversely as the square of the distance from the heat source. So as it was predictable the air around the box is additionally heated by the radiation heat transfer.

An additional simulation was performed. By setting the gravity constant close to zero at *constant/g* file, the convection effects were decreased. The result of this simulation can be seen below.
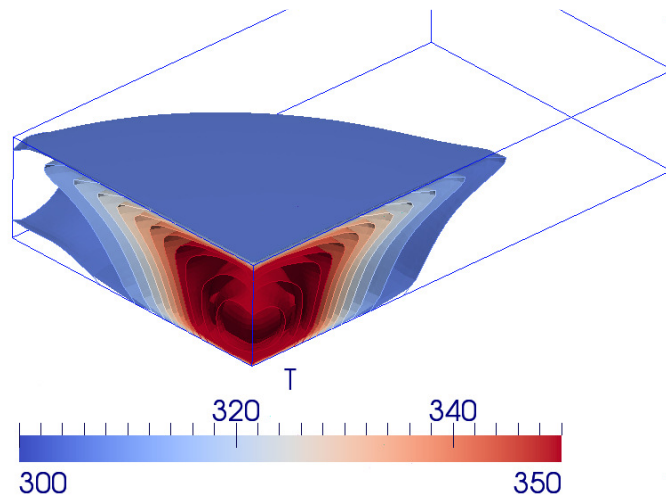

Figure 5. Temperature gradient for the case without convection

As predictable, without convection temperatures are much higher because heated air is not going upwards and it is not being replaced by cold one.

15

7.0   Conclusions

This is obvious that the effect of radiation heat transfer cannot be ignored and it needs to be added to any cases where it can be predicted to have a significant effect. It was shown that the addition of radiation heat transfer to the solver and case files is not a very complicated task. Depending on the complexity of the task being solved and the computational power available different radiation heat transfer models can be selected. One should also keep in mind that there is a need to select a proper scatter model, boundary conditions and absorption-emission model. After all assumptions are done and the values for all needed coefficients are set it is only a matter of time to get the results.

8.0 References

1. Frank P. Incropera, David P.DeWitt, - "Fundamentals of Heat and Mass Transfer", School of Mechanical Engineering Purdue University,

2. S. S. Sazhin, E. M. Sazhina, O. Faltsi-Saravelou and P. Wild – "The P-1 model for thermal radiation transfer: advantages and limitations", Fuel Vol. 75 No. 3, pp. 289-294, 1996,

3. H. Knaus, R. Schneider, X. Han, J. Ströhle, U. Schnell, K.R.G. Hein, - "Comparison of Different Radiative Heat Transfer Models and their Applicability to Coal-Fired Utility Boiler Simulations",

4. FLUENT 6.3 Tutorial Guide, "Modeling Radiation and Natural Convection" chapter,

5. http://www.cfd-online.com/Forums/