



Radiative heat transfer in OpenFOAM and its non-grey implementation

CFD with OpenSource Software

Wei Chen

China-UK Low Carbon College, Shanghai Jiao Tong University

2024 1



SHANGHAI JIAO TONG
UNIVERSITY

Table of contents



SHANGHAI JIAO TONG
UNIVERSITY

- 1 Introduction
- 2 Radiation models in this study
- 3 Radiation models in OpenFOAM
- 4 Developing a non-grey radiation model in OpenFOAM
- 5 Model validation
- 6 Discussions and conclusions

- 1 Introduction
- 2 Radiation models in this study
- 3 Radiation models in OpenFOAM
- 4 Developing a non-grey radiation model in OpenFOAM
- 5 Model validation
- 6 Discussions and conclusions

Radiative heat transfer in participating media

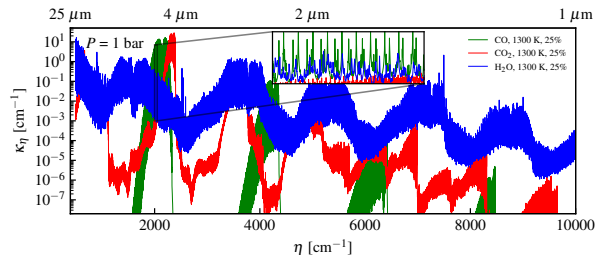
SHANGHAI JIAO TONG
UNIVERSITY

- Governing equation: RTE
- Non-local effect of radiation
- Sharp η -dependent κ_η shifts



Radiative Transfer Equation (RTE)

$$\frac{dI_\eta}{ds} = \hat{\mathbf{s}} \cdot \nabla I_\eta = \underbrace{\kappa_\eta I_{b\eta}}_{\text{Emission}} - \underbrace{\beta_\eta I_\eta}_{\text{Absorption}} + \underbrace{\frac{\sigma_{s\eta}}{4\pi} \int_{4\pi} I_\eta(\hat{\mathbf{s}}_i) \Phi_\eta(\hat{\mathbf{s}}_i, \hat{\mathbf{s}}) d\Omega_i}_{\text{Scattering}},$$



Modelling the radiative heat transfer



The radiative intensity I_η is intrinsically 6D

- 3D in space \mathbf{x}
- 2D in direction $\hat{\mathbf{s}}$
- 1D in wavenumber η

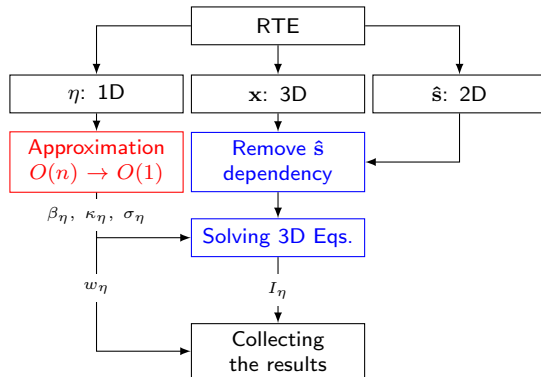
The RTE solver aims to:

- Removing RTE's angular dependency
- Solving the derived equation with given κ_η , β_η , $\sigma_{s\eta}$

The spectral model aims to:

- Mathematical manipulation in η space
- Millions' times of RTE evaluation \rightarrow several times
- Providing κ_η , β_η , $\sigma_{s\eta}$ for the RTE solver
- Providing weight w_η for each solution

Collecting the results with w_η



RTE solver

Discrete Ordinates Method (DOM)

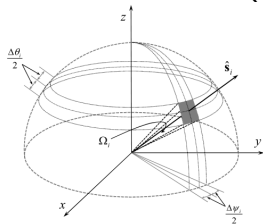


Figure: Discrete Ordinates Method (DOM) ¹

- ① Discretizing in solid angle
- ② RTE → several PDEs in 3D
- ③ Using FVM to solve the PDEs

¹Modest et al. "The Method of Discrete Ordinates (S N -Approximation)"

²https://en.wikipedia.org/wiki/Spherical_harmonics#/media/File:Sphericalfunctions.svg

Spherical Harmonics Method (PN)


l:	$P_l^m(\cos \theta) \cos(m\varphi)$	$P_l^{ m }(\cos \theta) \sin(m \varphi)$	
m:	6 5 4 3 2 1 0	-1 -2 -3 -4 -5 -6	
0 s			
1 p			
2 d			
3 f			
4 g			
5 h			
6 i			

Figure: Spherical Harmonics Method (PN) ²

- ① "2D Fourier expansion" for intensity
- ② RTE → several PDEs in 3D
- ③ Using FVM to solve the PDEs

Comparison between two models

Discrete Ordinates Method (DOM)

- Mathematically simple
- Different N_ϕ and $N_\theta \rightarrow$ similar equations
- High computational cost ($N_\phi N_\theta$)
- False scattering

Spherical Harmonics Method (PN)

- Mathematically complex, when $N \uparrow$
- Different $N \rightarrow$ different equations
- Low computational cost ($N(N+1)/2$)
- Stability issues

Governing equation for DOM

$$\hat{\mathbf{s}}_i \cdot \nabla \mathbf{I}(\mathbf{r}, \hat{\mathbf{s}}_i) = \kappa(\mathbf{r}) \mathbf{I}_b(\mathbf{r}) - \beta(\mathbf{r}) \mathbf{I}(\mathbf{r}, \hat{\mathbf{s}}_i) + \frac{\sigma_s(\mathbf{r})}{4\pi} \sum_{j=1}^n \omega_j I(\mathbf{r}, \hat{\mathbf{s}}_j) \Phi(\mathbf{r}, \hat{\mathbf{s}}_j, \hat{\mathbf{s}}_i).$$

Governing equation for PN

$$\sum_{k=1}^3 \left\{ (\mathcal{L}_{xx} - \mathcal{L}_{yy}) \left[(1 + \delta_{n2}) \omega_k^{nm} J_{n+4-2k}^{m-2} + \frac{\delta_{n1}}{2} \omega_k^{nm} J_{n+4-2k}^{m-1} + e_k^{nm} J_{n+4-2k}^{m+2} \right] + (\mathcal{L}_{xx} + \mathcal{L}_{yy}) \left[(1 + \delta_{n1}) \omega_k^{nm} J_{n+4-2k}^{m-1} + d_k^{nm} J_{n+4-2k}^{m+1} \right] + (\mathcal{L}_{xy} + \mathcal{L}_{yx}) \left[-(1 - \delta_{n2}) \omega_k^{nm} J_{n+4-2k}^{-(m-2)} + \frac{\delta_{n1}}{2} \omega_k^{nm} J_{n+4-2k}^{-(m-1)} + e_k^{nm} J_{n+4-2k}^{-(m+2)} \right] + (\mathcal{L}_{xx} - \mathcal{L}_{yy}) \left[-(1 - \delta_{n1}) \omega_k^{nm} J_{n+4-2k}^{-(m-1)} + d_k^{nm} J_{n+4-2k}^{-(m+1)} \right] + (\mathcal{L}_{xx} + \mathcal{L}_{yy} - 2\mathcal{L}_{zz}) \omega_k^{nm} J_{n+4-2k}^{m-3} \right\} + [\mathcal{L}_{xx} - (1 - \omega) \mathcal{I}_b] I_n^m - (1 - \omega) \mathcal{I}_b \mathcal{I}_n = 0$$

$(V_n^m: n = 0, 2, \dots, N-1, 0 \leq m \leq n)$

Second order operator: $\mathcal{L}_{xy} = \frac{1}{2} \frac{\partial}{\partial x} \left(\frac{1}{\rho} \frac{\partial}{\partial y} \right)$, $\mathcal{L}_{xx} = \frac{1}{2} \frac{\partial}{\partial x} \left(\frac{1}{\rho} \frac{\partial}{\partial x} \right)$
Boundary conditions: $(1 + \delta_{n,0}) \pi \sum_{m=0}^N \rho_{n,2m-1}^m \bar{I}_n^m = \int_0^{2\pi} \int_0^\pi I_n Y_{2m-1}^m d\theta d\psi$

Spectral model



- Line-by-line model (LBL): Evaluate RTE at each wavenumber (benchmark model)
- Band model: limited to black walls and non-scattering media
- Global model
 - Weighted sum of grey gases model (WSGG): Assume homogeneous mixture
 - Full spectrum correlated k -distribution model (**FSCK**): The state-of-art spectral model
 - Spectral-line-based WSGG (SLW): Mathematically identical to the FSCK model ¹
- Gray gas model: May result in larger errors compared to ignoring radiation

In this study, **FSCK** is implemented into OpenFOAM's radiation model framework.

¹See also: Section 19.11 in Modest et al. "Solution Methods for Nongray Extinction Coefficients"

Spectral model in CFD software

Official implementation

- 🔒 OpenFOAM: gray gas model and band model
- 🔒 Fluent: gray gas model, band model, WSGG
- 🔒 CFX: gray gas model, band model, WSGG

WSGG in ANSYS Fluent is not the one commonly recognized in the radiative heat transfer community. It is a simplified version of the WSGG model with only 1 times RTE evaluation (works not bad for combustion applications).

¹Wang et al. "Full-spectrum k-distribution look-up table for nonhomogeneous gas-soot mixtures"

²Ge et al. "Development of high-order PN models for radiative heat transfer in special geometries and boundary conditions"

³Ren et al. "Monte Carlo Simulation for Radiative Transfer in a High-Pressure Industrial Gas Turbine Combustion Chamber"

⁴Sun et al. "A hybrid non-gray gas radiation heat transfer solver based on OpenFOAM"

⁵Guo et al. "A full spectrum k-distribution based weighted-sum-of-grey-gases model for oxy-fuel combustion"

⁶Long et al. "Development and validation of a full-spectrum correlated k-distribution radiation model for CO₂-H₂O-CO-soot mixtures in ANSYS-Fluent"

Third party implementation

- 🔒 A compact radiation model in OpenFOAM 2.2x by Michael Modest's group ¹²³.
- 🔒 WSGG in OpenFOAM (unknown version) by Sun et al ⁴.
- 🔒 Limited FSCK UDF in Fluent 16.0 by Guo et al ⁵.
- 🔒 Limited FSCK UDF in Fluent 2022 by Wang et al ⁶.

I have access to **these codes**, which could be used to verify my implementation.

Outline

How to use it:

- How to use the radiation model in OpenFOAM with a focus on the combustion application.
- How to choose the radiation model in OpenFOAM.

The theory of it:

- The theory of radiative heat transfer.
- The theory of the radiative transfer equation (RTE) solution methods.
- The theory of the spectral models.

How it is implemented:

- How the radiation model is implemented in OpenFOAM.
- How the `greyMeanAbsorptionEmission` model cooperate with the RTE solver.

How to modify it:

- How to model the non-grey radiative heat transfer in OpenFOAM.



Introduction



Radiation models in this study



Radiation models in OpenFOAM



Developing a non-grey radiation model in OpenFOAM



Model validation



Discussions and conclusions

The P1 model: The expansion of radiative intensity



The spherical harmonics can be written as,

$$Y_l^m(\theta, \psi) = \frac{(-1)^l}{2^l l!} \sqrt{\frac{(2l+1)(l+m)!}{4\pi(l-m)!}} e^{im\psi} P_l^m(\cos \theta).$$

By expanding the intensity into the series of spherical harmonics,

$$I(\mathbf{r}, \hat{\mathbf{s}}) = \sum_{l=0}^{\infty} \sum_{m=-l}^l I_l^m(\mathbf{r}) Y_l^m(\hat{\mathbf{s}}).$$

If only expand to the first order, it can be written as,

$$I(\mathbf{r}, \theta, \psi) = I_0^0 + I_1^0 \cos \theta - I_1^{-1} \sin \theta \sin \psi - I_1^1 \sin \theta \cos \psi.$$

About P_n^m

It is called as associated Legendre polynomial and is expressed as,

$$P_n^m(\mu) = (-1)^m \frac{(1-\mu^2)^{|m|/2}}{2^n n!} \frac{d^{n+|m|}}{d\mu^{n+|m|}} (\mu^2 - 1)^n.$$

Two important properties,

- Orthogonalities:

$$\int_{-1}^1 P_l(\mu) P_m(\mu) d\mu = \frac{2\delta_{lm}}{2m+1} = \begin{cases} 0 & \text{for } m \neq l, \\ \frac{2}{2m+1} & \text{for } m = l, \end{cases}$$

- Recursion relation:

$$(2l+1)\mu P_l(\mu) = lP_{l-1}(\mu) + (l+1)P_{l+1}(\mu)$$

The P1 model: Formulation



$$I(\mathbf{r}, \theta, \psi) = I_0^0 + I_1^0 \cos \theta - I_1^{-1} \sin \theta \sin \psi - I_1^1 \sin \theta \cos \psi.$$

By defining the incident radiation as,

$$G(\mathbf{r}) = \int_{4\pi} I(\mathbf{r}, \hat{\mathbf{s}}) d\Omega,$$

the approxiamted RTE can be written as,

$$\nabla \cdot (\Gamma \nabla G) - aG = -4\epsilon\sigma T^4 - E,$$

where $\Gamma = \frac{1}{3a + \sigma_s + a_0}$ is the diffusivity of the equation, a is the absorption coefficient, σ_s is the linear scattering factor, ϵ is the emission coefficient, and E is the emission coefficient.

Formulation of the P1 model in OpenFOAM
(\$FOAM.RADIATION/radiationModels/P1/P1.C)

```

1 void Foam::radiation::nonGreyP1::calculate()
2 {
3     absorptionEmission_->correct(G_, Gg_);
4
5     const dimensionedScalar a0("a0", a_.dimensions(),
6         ROOTVSMALL);
7     // ...
8     solve
9     (
10         fvm::laplacian(gamma, G_)
11         - fvm::Sp(a_, G_)
12         ==
13         - 4.0*(e_*physicoChemical::sigma*pow4(T_)) - E_
14     );
15     // ...
16 }
```

The P1 model: Boundary condition



The boundary condition for the P1 model is the Marshak's boundary condition. The generalized form is given by,

$$\int_{\hat{\mathbf{n}} \cdot \hat{\mathbf{s}} > 0} I(\mathbf{r}_w, \hat{\mathbf{s}}) \bar{Y}_{2i-1}^m(\hat{\mathbf{s}}) d\Omega = \int_{\hat{\mathbf{n}} \cdot \hat{\mathbf{s}} > 0} I_w(\hat{\mathbf{s}}) \bar{Y}_{2i-1}^m(\hat{\mathbf{s}}) d\Omega, \quad i = 1, 2, \dots, \frac{1}{2}(N+1),$$

For P1, it is.

$$-\frac{2(2-\epsilon)}{\epsilon} \Gamma(\hat{\mathbf{n}} \cdot \nabla G) + G = 4\sigma T^4.$$

Formulation of the Marshak boundary condition in OpenFOAM

```
1 void Foam::radiation::MarshakRadiationFvPatchScalarField::updateCoeffs()
2 {
3     // ...
4     // Re-calc reference value
5     refValue() = 4.0*constant::physicoChemical::sigma.value()*pow4(Tp);
6     // ...
7     // Set value fraction
8     valueFraction() = 1.0/(1.0 + gamma*patch().deltaCoeffs()/Ep);
9     // ...
10    mixedFvPatchScalarField::updateCoeffs();
11 }
```



The FSCK model: Procedures

Principles: Transferring the RTE from the wavenumber η space into so-called g space by reordering the absorption coefficient κ_η based on some laws.

Benefits: Millions times RTE evaluation \rightarrow at most 32 times.

How to: By replacing the absorption coefficient with the k_n and the “emission coefficient” with the $k_n a_n$ in the RTE, the RTE can be written as,

$$\text{RTE}(k_n, a_n, G_{g_n}) = 0, \quad (n \in [1, nq])$$

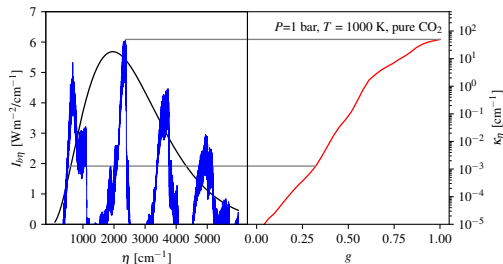
k_n is the k value at each quadrature point, a_n is called as non-gray stretching coefficient and G_{g_n} is the G value at each quadrature point.

Collecting the results from all quadrature points, $G = \sum_{n=1}^{nq} w_n G_{g_n}$, the radiative heat source can be written as,

$$\nabla \cdot q = \sum_{n=1}^{nq} w_n \nabla \cdot q_{g_n} = \sum_{n=1}^{nq} w_n (4\pi a I_b - G_{g_n}).$$

¹Wang et al. “Efficient full-spectrum correlated-k-distribution look-up table”

²Zhou et al. “A machine learning based full-spectrum correlated k-distribution model for nonhomogeneous gas-soot mixtures”



k_n , a_n and w_n values?

Reference volume fraction: $\mathbf{x}_{\text{ref}} = \frac{1}{V} \int_V \mathbf{x} dV$.

Reference temperature:

$$k(p, T, T_{\text{ref}}, \mathbf{x}_{\text{ref}}) I_b(T_{\text{ref}}) = \frac{1}{V} \int_V k(p, T, T, \mathbf{x}) I_b(T) dV$$

k is a function of p , T , T_{ref} , \mathbf{x} , which can be obtained by a look-up table ¹ or a neural network ². a can be derived from k . w_n is only a function of nq .

The FSCK model: Procedures



Principles: Transferring the RTE from the wavenumber η space into so-called g space by reordering the absorption coefficient κ_η based on some laws.

Benefits: Millions times RTE evaluation \rightarrow at most 32 times.

How to: By replacing the absorption coefficient with the k_n and the “emission coefficient” with the $k_n a_n$ in the RTE, the RTE can be written as,

$$\text{RTE}(k_n, a_n, G_{g_n}) = 0, \quad (n \in [1, nq])$$

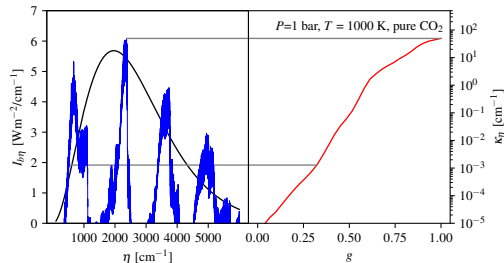
k_n is the k value at each quadrature point, a_n is called as non-gray stretching coefficient and G_{g_n} is the G value at each quadrature point.

Collecting the results from all quadrature points, $G = \sum_{n=1}^{nq} w_n G_{g_n}$, the radiative heat source can be written as,

$$\nabla \cdot q = \sum_{n=1}^{nq} w_n \nabla \cdot q_{g_n} = \sum_{n=1}^{nq} w_n (4\pi a I_b - G_{g_n}).$$

¹Wang et al. “Efficient full-spectrum correlated-k-distribution look-up table”

²Zhou et al. “A machine learning based full-spectrum correlated k-distribution model for nonhomogeneous gas-soot mixtures”



k_n , a_n and w_n values?

Reference volume fraction: $\mathbf{x}_{\text{ref}} = \frac{1}{V} \int_V \mathbf{x} dV$.

Reference temperature:

$$k(p, T, T_{\text{ref}}, \mathbf{x}_{\text{ref}}) I_b(T_{\text{ref}}) = \frac{1}{V} \int_V k(p, T, T, \mathbf{x}) I_b(T) dV$$

k is a function of p , T , T_{ref} , \mathbf{x} , which can be obtained by a look-up table ¹ or a neural network ². a can be derived from k . w_n is only a function of nq .

The FSCK model: Procedures

Principles: Transferring the RTE from the wavenumber η space into so-called g space by reordering the absorption coefficient κ_η based on some laws.

Benefits: Millions times RTE evaluation \rightarrow at most 32 times.

How to: By replacing the absorption coefficient with the k_n and the “emission coefficient” with the $k_n a_n$ in the RTE, the RTE can be written as,

$$\text{RTE}(k_n, a_n, G_{g_n}) = 0, \quad (n \in [1, nq])$$

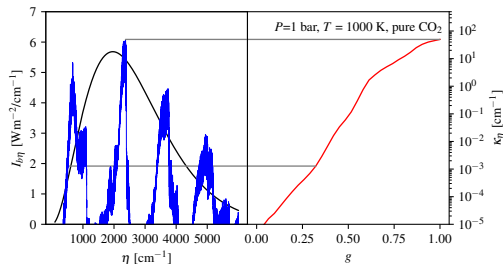
k_n is the k value at each quadrature point, a_n is called as non-gray stretching coefficient and G_{g_n} is the G value at each quadrature point.

Collecting the results from all quadrature points, $G = \sum_{n=1}^{nq} w_n G_{g_n}$, the radiative heat source can be written as,

$$\nabla \cdot q = \sum_{n=1}^{nq} w_n \nabla \cdot q_{g_n} = \sum_{n=1}^{nq} w_n (4\pi a I_b - G_{g_n}).$$

¹Wang et al. “Efficient full-spectrum correlated-k-distribution look-up table”

²Zhou et al. “A machine learning based full-spectrum correlated k-distribution model for nonhomogeneous gas-soot mixtures”



k_n , a_n and w_n values?

Reference volume fraction: $\mathbf{x}_{\text{ref}} = \frac{1}{V} \int_V \mathbf{x} dV$.

Reference temperature:

$$k(p, T, T_{\text{ref}}, \mathbf{x}_{\text{ref}}) I_b(T_{\text{ref}}) = \frac{1}{V} \int_V k(p, T, T, \mathbf{x}) I_b(T) dV$$

k is a function of p , T , T_{ref} , \mathbf{x} , which can be obtained by a look-up table ¹ or a neural network ². a can be derived from k . w_n is only a function of nq .

The FSCK model: Procedures



Principles: Transferring the RTE from the wavenumber η space into so-called g space by reordering the absorption coefficient κ_η based on some laws.

Benefits: Millions times RTE evaluation \rightarrow at most 32 times.

How to: By replacing the absorption coefficient with the k_n and the “emission coefficient” with the $k_n a_n$ in the RTE, the RTE can be written as,

$$\text{RTE}(k_n, a_n, G_{g_n}) = 0, \quad (n \in [1, nq])$$

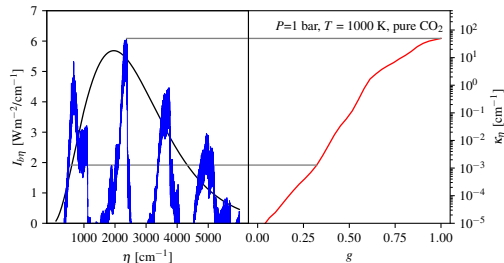
k_n is the k value at each quadrature point, a_n is called as non-gray stretching coefficient and G_{g_n} is the G value at each quadrature point.

Collecting the results from all quadrature points, $G = \sum_{n=1}^{nq} w_n G_{g_n}$, the radiative heat source can be written as,

$$\nabla \cdot q = \sum_{n=1}^{nq} w_n \nabla \cdot q_{g_n} = \sum_{n=1}^{nq} w_n (4\pi a I_b - G_{g_n}).$$

¹Wang et al. “Efficient full-spectrum correlated-k-distribution look-up table”

²Zhou et al. “A machine learning based full-spectrum correlated k-distribution model for nonhomogeneous gas-soot mixtures”



k_n , a_n and w_n values?

Reference volume fraction: $\mathbf{x}_{\text{ref}} = \frac{1}{V} \int_V \mathbf{x} dV$.

Reference temperature:

$$k(p, T, T_{\text{ref}}, \mathbf{x}_{\text{ref}}) I_b(T_{\text{ref}}) = \frac{1}{V} \int_V k(p, T, T, \mathbf{x}) I_b(T) dV$$

k is a function of p , T , T_{ref} , \mathbf{x} , which can be obtained by a look-up table ¹ or a neural network ². a can be derived from k . w_n is only a function of nq .



Introduction



Radiation models in this study



Radiation models in OpenFOAM



Developing a non-grey radiation model in OpenFOAM



Model validation



Discussions and conclusions

How to use the OpenFOAM's radiation model?



- Add fvOptions under constant folder
- Add radiationProperties under constant folder
- Add boundaryRadiationProperties under constant folder
- Setup boundary conditions

fvOptions



```
1  /*----- C++ -----*/
2  | ===== |
3  | \\      / F ield      | OpenFOAM: The Open Source CFD Toolbox |
4  | \\      / O peration  | Version: v2212 |
5  | \\      / A nd        | Website: www.openfoam.com |
6  | \\      / M anipulation |
7  /*-----*/
8  FoamFile
9  {
10     version      2.0;
11     format        ascii;
12     class         dictionary;
13     object        fvOptions;
14 }
15 // *****
16
17 radiation
18 {
19     type          radiation;
20     libs (radiationModels);
21 }
22 // *****
23
```

radiationProperties



```

radiation on;
radiationModel P1;
P1Coeffs
{
    C                C [0 0 0 0 0 0] 0;
}
// Number of flow iterations per radiation iteration
solverFreq 1;
absorptionEmissionModel greyMeanAbsorptionEmission;
greyMeanAbsorptionEmissionCoeffs
{
    lookUpTableFileName    none;
    EhrrCoeff              0.0;
    CO2
    {
        Tcommon           200;    //Common Temp
        invTemp            true;    //Is the polynomio using
        inverse temperature.
        Tlow               200;    //Low Temp
        Thigh              2500;    //High Temp

        loTcoeffs          //coefss for T < Tcommon
        (
            0              // a0          +
            0              // a1*T        +
            0              // a2*T^(+/-)2  +

```

```

            0              // a3*T^(+/-)3  +
            0              // a4*T^(+/-)4  +
            0              // a5*T^(+/-)5  +
        );
        hiTcoeffs          //coefss for T > Tcommon
        (
            18.741
            -121.31e3
            273.5e6
            -194.05e9
            56.31e12
            -5.8169e15
        );
    }
    H2O // ...
    CH4 // ...
    O2  // ...
    N2  // ...
}
scatterModel    none;
sootModel        none;
transmissivityModel none;

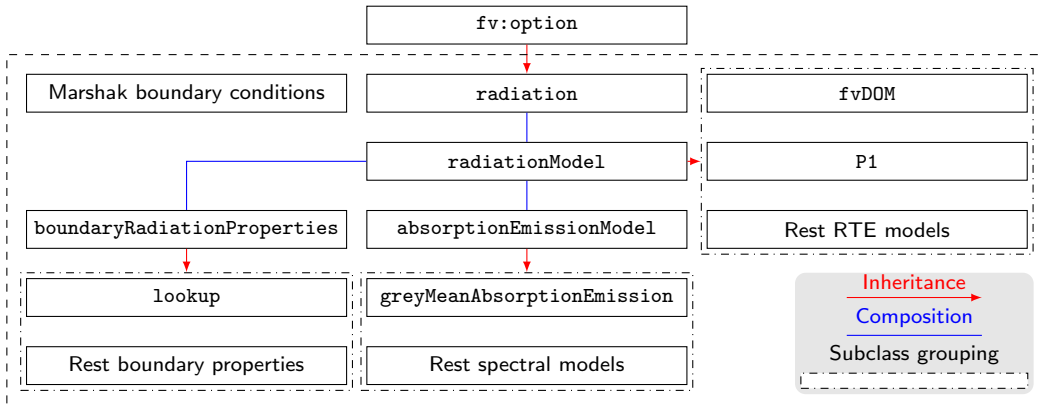
```

boundaryRadiationProperties



```
1  /*----- C++ -----*/
2  | ===== |
3  | \ \ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
4  | \ \ / O p e r a t i o n | Version: v2212 |
5  | \ \ / A n d | Website: www.openfoam.com |
6  | \ \ / M a n i p u l a t i o n | |
7  /*-----*/
8  FoamFile
9  {
10     version      2.0;
11     format        ascii;
12     class         dictionary;
13     object        boundaryRadiationProperties;
14 }
15 // *****
16
17 ".*"
18 {
19     type          lookup;
20     emissivity     1;
21     absorptivity   0;
22 }
23 // *****
24
```

Structure



radiation class



Energy equation in reactingFOAM

```
1 fvScalarMatrix EEqn
2 (
3     fvm::ddt(rho, he) + mvConvection->fvmDiv(phi, he)
4     // ...
5     ==
6     Qdot
7     + fvOptions(rho, he)
8 );
```

The definition of radiation

```
1 class radiation
2 :
3     public fv::option
4 {
5     // Private Data
6     // ..
7 };
```

- No stand-alone radiation solver
- Add source term to the energy equation
- Inherits from `fv::option`
- Main work done by `radiationModel`

The constructor of radiation

```
1 Foam::fv::radiation::radiation
2 (
3     // ...
4 )
5 :
6     fv::option(sourceName, modelType, dict, mesh)
7 {
8     // ...
9     radiation_ = Foam::radiation::radiationModel::New(thermo.T());
10 }
```



Calling radiationModel class in radiation class

The addSup function in radiation

```
1 void Foam::fv::radiation::addSup
2 (
3     const volScalarField& rho,
4     fvMatrix<scalar>& eqn,
5     const label fieldi
6 )
7 {
8     const auto& thermo = mesh_.lookupObject<basicThermo>(basicThermo::dictName);
9
10    radiation_->correct();
11
12    eqn += radiation_->Sh(thermo, eqn.psi());
13 }
```

- Override addSup in fv::option
- Perform calculation in radiation_->correct()
- Add source term: radiation_->Sh(...)

radiationModel class: solving the RTE



RTE is solved when necessary

```
1 void Foam::radiation::radiationModel::correct()
2 {
3     if (!radiation_)
4     {
5         return;
6     }
7
8     if (firstIter_ || (time_.timeIndex() % solverFreq_ == 0))
9     {
10         calculate();
11         firstIter_ = false;
12     }
13
14     if (soot_)
15     {
16         soot_->correct();
17     }
18 }
```

- The RTE is solved in the virtual method calculate
- correct only calls calculate when necessary

Calculating the source term



Enthalpy source term

```
1 Foam::tmp<Foam::fvScalarMatrix> Foam::radiation::radiationModel::Sh
2 (
3     const basicThermo& thermo,
4     const volScalarField& he
5 ) const
6 {
7     const volScalarField Cp(thermo.Cp());
8     const volScalarField T3(pow3(T_));
9
10    return
11    (
12        Ru()
13        - fvm::Sp(4.0*Rp()*T3/Cp, he)
14        - Rp()*T3*(T_ - 4.0*he/Cp)
15    );
16 }
17
```

- The source term contains T^4 , which have to be linearized.
- Virtual methods Ru and Rp have to be overridden in the subclass.

P1 model in OpenFOAM



In the P1:calculate method:

- ① Get spectral parameter from `absorptionEmission_` pointer.
- ② Define equation parameter Γ
- ③ Solve G transport equation
- ④ Calculate radiative heat flux on boundaries

We may find out:

- ① It is a pure gray RTE solver.
- ② It only solves RTE one times in each iteration.
- ③ It can not even couple with the built-in multi-band spectral model.

P1::calculate method



```
void Foam::radiation::P1::calculate()
{
    a_ = absorptionEmission_->a();
    e_ = absorptionEmission_->e();
    E_ = absorptionEmission_->E();
    const volScalarField sigmaEff(scatter_->sigmaEff());

    const dimensionedScalar a0("a0", a_.dimensions(),
        ROOTVSMALL);

    // 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 + a0)
    );

    // Solve G transport equation
    solve
```

```
(
    fvm::laplacian(gamma, G_)
    - fvm::Sp(a_, G_)
    ==
    - 4.0*(e_*physicoChemical::sigma*pow4(T_)) - E_
);

// Calculate radiative heat flux on boundaries.
volScalarField::Boundary& qrBf = qr_.boundaryFieldRef();
const volScalarField::Boundary& GBf = G_.boundaryField()
;
const volScalarField::Boundary& gammaBf = gamma.
    boundaryField();

forAll(mesh_.boundaryMesh(), patchi)
{
    if (!GBf[patchi].coupled())
    {
        qrBf[patchi] = -gammaBf[patchi]*GBf[patchi].
            snGrad();
    }
}
}
```

greyMeanAbsorptionEmission class



- Absorption coefficient is a function of temperature polynomial
- It assumes absorption coefficient equal to the “emission coefficient”
- It is a pure grey spectral model.
- Method for getting absorption coefficient has band index as argument.

- 1 Introduction
- 2 Radiation models in this study
- 3 Radiation models in OpenFOAM
- 4 Developing a non-grey radiation model in OpenFOAM**
- 5 Model validation
- 6 Discussions and conclusions

Overview



- 1 Calculate the reference state for FSCK model.
- 2 Get the k values and the non-grey stretching coefficients a for the whole field.
- 3 Solve the RTE based on k values and a values at each quadrature points.
- 4 Collect the results from all quadrature points.
- 5 Calculate the radiative heat source.

Obtaining the FSCK parameters



Model by Zhou et al. ¹:

- It can predict k values from three MLPs.
- Each MLP covers a pressure range.
- MLP takes $p, T, T_{\text{ref}}, \mathbf{x}$ as input.
- Non-grey stretching coefficient a is derived from k .
- Model outputs k for **all** quadrature points (32).
- Interpolation is required for less quadrature points.
- Model is hard-coded into Fortran code ².

Another way to obtain FSCK parameters is by using a lookup table. While the lookup table for FSCK model is ca. 12.45 GB, which may largely limit the application of this model.

In this report:

- I store the MLP parameters in JSON.
- I use `json.hpp` ³ to read MLP's parameters.
- I write MLP class to perform forward propagation.
- I write MLPManager class to manage MLPs.
- I write `support_func.cpp`, which can
 - Interpolate k and a to any number of quadrature points.
 - Calculate a from k
 - Calculate w

Source code and MLP parameters are stored together under `fsckMLPModel` folder. It is compiled by `make` but not `wmake`

¹Zhou et al. "A machine learning based full-spectrum correlated k-distribution model for nonhomogeneous gas-soot mixtures"

²: https://github.com/ZY-LHY/Machine_learning_based_FSCK_model_soot

³: <https://github.com/nlohmann/json>, Lohmann *JSON for Modern C++*



fsckMLP class

Overall, this class does the following things:

- It stores k and a values for each quadrature points in `PtrList<volScalarField>`
- It calculates the reference state.
- It updates k and a values by calling the `MLPManager::get_prediction`

Some aspects are important to be mentioned:

- To ensure only one instance is created and not modify the higher level code, I use singleton pattern.
- `PtrList<volScalarField>` needs to be initialized.
- CFD software uses mass fraction, while radiation models uses volume fraction.

Initializing fields for k and a



```
void Foam::radiation::fsckMLP::initKA()
{
    ki_.setSize(nBands_);
    ai_.setSize(nBands_);
    for (label i=0; i<nBands_; i++)
    {
        Info << "Initializing k and a for band" << i << endl;
        ki_.set(i, new volScalarField
        (
            IOobject
            (
                "k" + std::to_string(i),
                mesh_.time().timeName(),
                mesh_,
                IOobject::NO_READ,
                IOobject::AUTO_WRITE
            ),
            mesh_,
```

```
        dimensionedScalar(dimless/dimLength, ROOTVSMALL)
    ));
    ai_.set(i, new volScalarField
    (
        IOobject
        (
            "a" + std::to_string(i),
            mesh_.time().timeName(),
            mesh_,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        mesh_,
        dimensionedScalar(dimless/dimLength, ROOTVSMALL)
    ));
    }
}
```


Caculating volume fractions



```
1 scalar invWt = 0.0;
2 scalar xco2_cell, xh2o_cell, xco_cell;
3 forAll(mixture.Y(), s)
4 {
5     invWt += mixture.Y(s)[celli]/mixture.W(s);
6 }
7 xco2_cell = mixture.Y("CO2")[celli]/(mixture.W(mixture.species()["CO2"])*invWt);
8 xh2o_cell = mixture.Y("H2O")[celli]/(mixture.W(mixture.species()["H2O"])*invWt);
9 xco_cell = mixture.Y("CO")[celli]/(mixture.W(mixture.species()["CO"])*invWt);
10
11 xco2 = xco2 + xco2_cell * mesh_.cellVolumes()[celli];
12 xh2o = xh2o + xh2o_cell * mesh_.cellVolumes()[celli];
13 xco = xco + xco_cell * mesh_.cellVolumes()[celli];
```

$$x_i = Y_i MW_{\text{mix}} / MW$$

Update k and a values



Update k and a for internal field

```
forAll(T, celli)
{
    //...
    double a[NqDB];
    afun(gNqDB, k_Tref.data(), k_T.data(), wNqDB, a);
    double k_new[nBands_], a_new[nBands_];
    simple_interp(NqDB, nBands_, gNqDB, k_Tref.data(), gNq,
        k_new);
    simple_interp(NqDB, nBands_, gNqDB, a, gNq, a_new);

    forAll(ki_, bandI)
    {
        ki_[bandI][celli] = k_new[bandI]*100.0;
        ai_[bandI][celli] = a_new[bandI];
    }
}
```

Update k and a for boundaries

```
forAll(T.boundaryField(), patchi)
{
    forAll (T.boundaryField()[patchi], facei)
    {
        // ...

        forAll(ai_, bandI)
        {
            ai_[bandI].boundaryFieldRef()[patchi][facei] =
                a_new[bandI];
            ki_[bandI].boundaryFieldRef()[patchi][facei] =
                k_new[bandI]*100.0;
        }
    }
}
```

Key takeaway

Using `mesh_.boundary()` to iterate through the boundary faces is fine for 3D cases. However, when trying to set value on empty boundary, it will cause segmentation fault.

nonGreyMeanAbsorptionEmission class



- It reads the number of quadrature points from the radiationProperties dict under constant folder.
- It creates fsckMLP instance by providing the number of quadrature points.
- It updates reference temperature, k and a values in the correct method.
- It returns corresponding k and a values by the index of quadrature points.

nonGreyP1 class



In the calculate method

- ① It calls `nonGreyMeanAbsorptionEmission::correct` at the beginning
- ② It iterates through each quadrature points
- ③ It solves the RTE just like the original P1 class, but uses k and a values from `nonGreyMeanAbsorptionEmission`
- ④ It collects the results by sum up the incident radiation at each quadrature points with w_n .

Non-grey marshak boundary conditions



There are two grey marshak boundary conditions in OpenFOAM, namely,

- MarshakRadiationFvPatchScalarField class
- MarshakRadiationFixedTemperatureFvPatchScalarField class

To make them become non-grey version, you just need to times non-grey stretching factor a on the refValue

```
fsckMLP* fsck = fsckMLP::getInstance();

const word& aName_("a" + std::to_string(fsck->getBandI()));

// Nongrey stretching factor field
const scalarField& ap =
    patch().lookupPatchField<volScalarField, scalar>(aName_);

// Temperature field
const scalarField& Tp =
    patch().lookupPatchField<volScalarField, scalar>(TName_);

// Re-calc reference value
refValue() = 4.0*constant::physicoChemical::sigma.value()*pow4(Tp)*ap;
```



Introduction



Radiation models in this study



Radiation models in OpenFOAM



Developing a non-grey radiation model in OpenFOAM



Model validation



Discussions and conclusions

Overview



Benchmark code:

- Modest group's radiation models ¹²³ in OpenFOAM 2.2x
- Ren group's FSCK model ⁴ in ANSYS Fluent 2022

Validation cases:

- 1D homogeneous slab (compared with Modest's code)
- 1D non-homogeneous slab (compared with Modest's code)
- 2D homogeneous plate (compared with Tao's code)
- 2D non-homogeneous plate (compared with Tao's code)

¹Wang et al. "Full-spectrum k-distribution look-up table for nonhomogeneous gas-soot mixtures"

²Ge et al. "Development of high-order PN models for radiative heat transfer in special geometries and boundary conditions"

³Ren et al. "Monte Carlo Simulation for Radiative Transfer in a High-Pressure Industrial Gas Turbine Combustion Chamber"

⁴Long et al. "Development and validation of a full-spectrum correlated k-distribution radiation model for CO₂-H₂O-CO-soot mixtures in ANSYS-Fluent"

1D homogeneous slab



1 bar, 1200 K, 27.32% CO₂, 3.477% CO, and 6.298% H₂O.

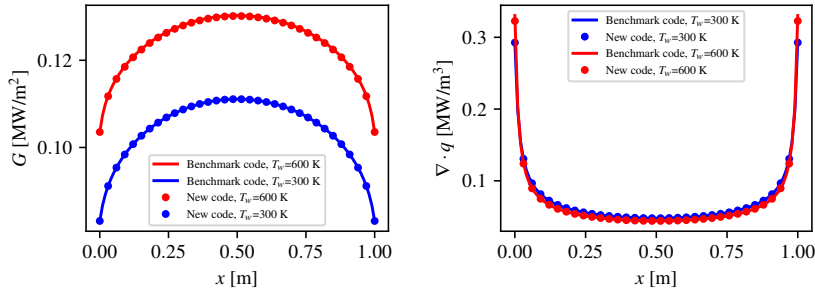


Figure: Incident radiation (left) and radiative heat source (right) for 1D homogeneous slab

1D nonhomogeneous slab

$$p = 1 \text{ bar}, T = \left(1600 \exp \left(-\frac{(x-0.2)^2}{0.3^2} \right) + 400 \right) \text{ K},$$

$$x_{\text{CO}_2} = 0.15 \exp \left(-\frac{(x-0.2)^2}{0.3^2} \right), x_{\text{H}_2\text{O}} = 0.15 \exp \left(-\frac{(x-0.2)^2}{0.3^2} \right), x_{\text{CO}} = 0.075 \exp \left(-\frac{(x-0.2)^2}{0.3^2} \right).$$

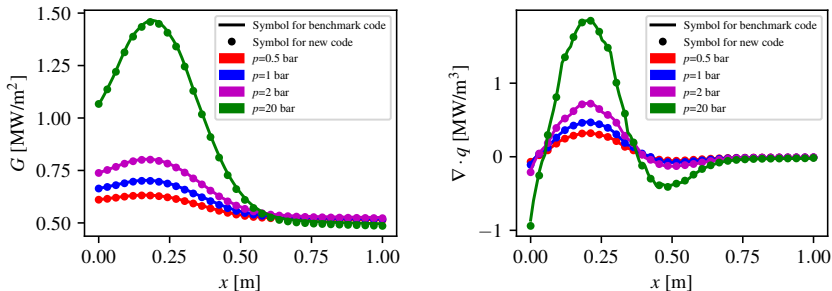


Figure: Incident radiation (left) and radiative heat source (right) for a 1D non-homogeneous slab at different pressures.

2D homogeneous plate



1 bar, 1200 K, 30% CO₂, 20% CO, and 50% H₂O with cold black wall

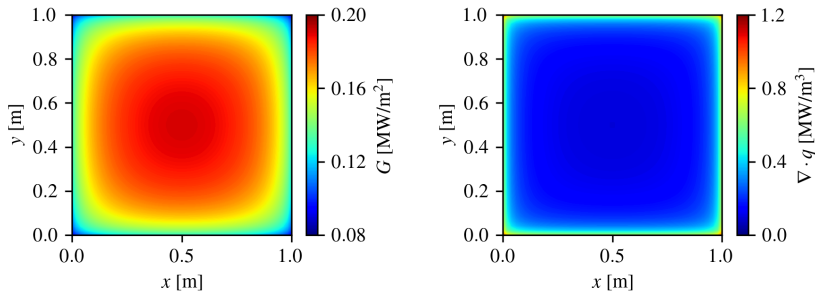


Figure: Incident radiation (left) and radiative heat source (right) for 2D homogeneous enclosure predicted by the new code

2D homogeneous plate: Comparison

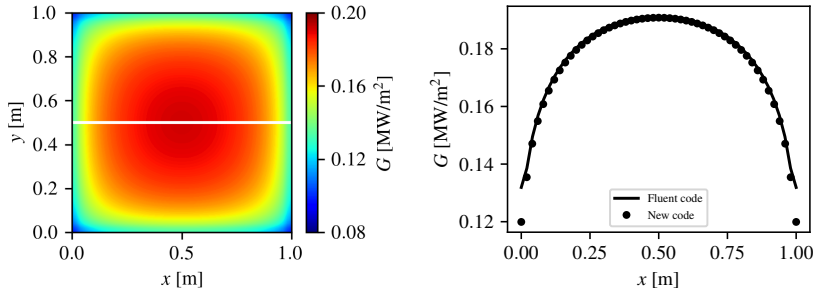


Figure: Incident radiation for 2D homogeneous enclosure predicted by the in-house UDF code (left) and the comparison with the new code sampling at the middle white line (right)

2D nonhomogeneous plate: Setup



- With the profile: $\phi(x, y) = \max\left(0, 1 - \frac{\|\mathbf{pos}(x, y) - \mathbf{c}\|}{r}\right)$.
- $\mathbf{c} = (0.5, 0.5)$ m, $r = 0.5$ m.
- Temperature varies from 0 K to 1200 K.
- x_{CO_2} varies from 0 to 0.1.
- $x_{\text{H}_2\text{O}}$ varies from 0 to 0.05.
- x_{CO} varies from 0 to 0.03.

2D nonhomogeneous plate: Results

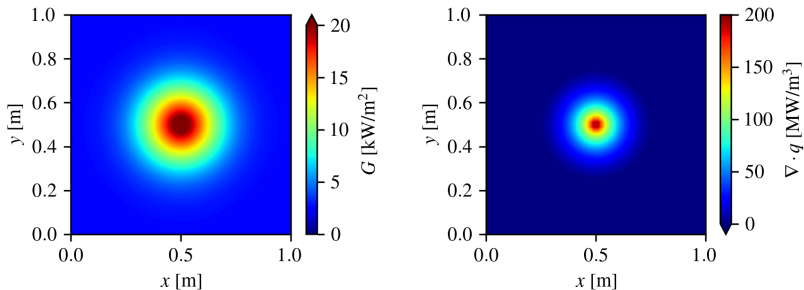


Figure: Incident radiation (left) and radiative heat source (right) for 2D nonhomogeneous enclosure predicted by the new code

2D nonhomogeneous plate: Comparison

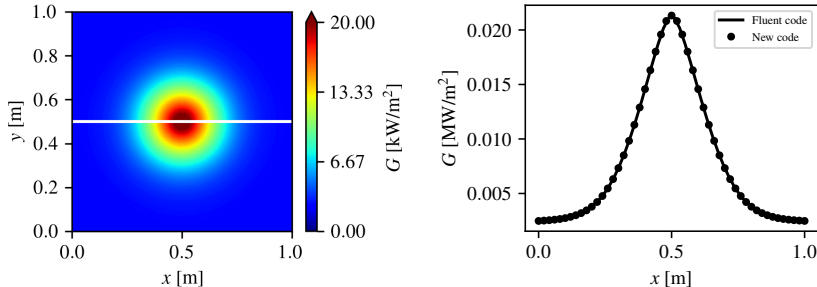


Figure: Incident radiation for 2D nonhomogeneous enclosure predicted by the in-house UDF code (left) and the comparison with the new code sampling at the middle white line (right)



Introduction



Radiation models in this study



Radiation models in OpenFOAM



Developing a non-grey radiation model in OpenFOAM



Model validation



Discussions and conclusions

Difference between Modest's code and Tao's code



Modest's code

- Modest's code implements the radiation model in a stand-alone library.
- To integrate with solvers, modification on the source code is necessary.
- Radiative heat source is not linearized.
- Modest's code depends on many libraries written in Fortran.
- Modest's code is not open-source.

Tao's code

- The boundary condition can not be properly treated because of the absence of API.
- The extensibility is largely limited by the UDF interface.

Why does it work?



Iterate through mesh_.boundaryMesh()

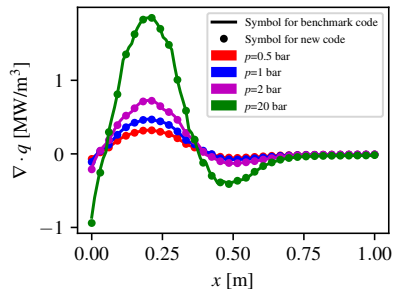
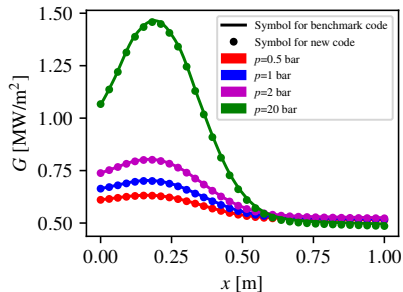
```
1 forAll(Gg_, bandI)
2 {
3     Gg_ += Gg_[bandI]*wNq[bandI];
4     forAll(mesh_.boundaryMesh(), patchi)
5     {
6         Gg_.boundaryFieldRef()[patchi] == Gg_.boundaryFieldRef()
7         [patchi] +
8         Gg_[bandI].boundaryField()[patchi]*wNq[bandI];
9     }
```

Iterate through T.boundaryField()

```
1 forAll(T.boundaryField(), patchi)
2 {
3     forAll (T.boundaryField()[patchi], facei)
4     {
5         // ...
6
7         forAll(ai_, bandI)
8         {
9             ai_[bandI].boundaryFieldRef()[patchi][facei] =
10             a_new[bandI];
11             ki_[bandI].boundaryFieldRef()[patchi][facei] =
12             k_new[bandI]*100.0;
13         }
14     }
```

A loop through faces may lead to segmentation fault when the boundary is empty.

Radiative heat loss is not smooth



Radiative heat loss is not smooth: Possible reason

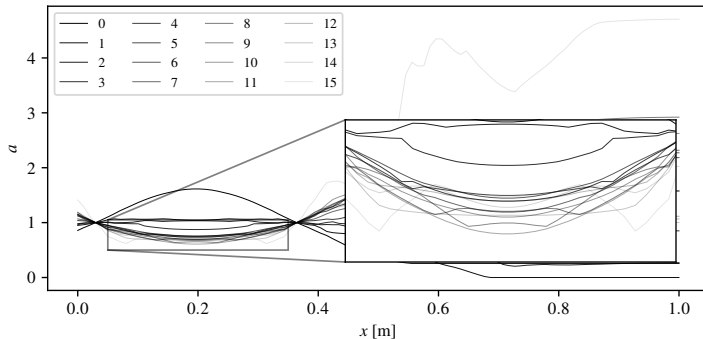


Figure: Non-grey stretching coefficient at different quadrature points

Conclusions and future work

Conclusions

- Theory of radiative heat transfer, RTE solvers and spectral models are reviewed.
- A brief tutorial of the usage of OpenFOAM's radiation model is given.
- The guideline for the choice of RTE solver is given.
- A walk-through of OpenFOAM's radiation model is given with a focus on the P1 model and the greyMeanAbsorptionEmission.
- The FSCK model is implemented in the existing OpenFOAM radiation model framework.

Future work

- Predicting k and a in a single network.
- Training a smaller network to reduce the computational cost.
- Integrating the FSCK model with the existing OpenFOAM's fvDOM model.
- Validating the model with some real cases.
- Comparing with the experiment data.

Acknowledgments



I would like to express my gratitude to the following individuals for their contributions:

- Tao Ren, Saeed Salehi, and Chit Yan Toe for their effort in the peer-review process.
- Håkan Nilsson and Saeed Salehi for their effort in preparing the course.

Thank You



SHANGHAI JIAO TONG
UNIVERSITY

