RadLib: a radiative heat transfer model library for CFD

Victoria B. Stephens, Sally Jensen, David O. Lignell*

Department of Chemical Engineering, Brigham Young University, Provo, UT 84602, United States

Abstract

Ca. 100 words

Keywords: radiative heat transfer, reacting flows, CFD

Required Metadata

Current code version

Ancillary data table required for subversion of the codebase. Kindly replace examples in right column with the correct information about your current code, and leave the left column as it is.

Nr.	Code metadata description	Please fill in this column
C1	Current code version	TODO 2.1
C2	Permanent link to code/repository used for this code version	*github.com/BYUignite/RadLib*
C3	Code Ocean compute capsule	N/A
C4	Legal Code License	MIT license (MIT)
C5	Code versioning system used	Git
C6	Software code languages, tools, and services used	C++, Python 3
C7	Compilation requirements, operating environments & dependencies	TODO
C8	If available Link to developer documentation/manual	TODO
C9	Support email for questions	davidlignell@byu.edu

Table 1: Code metadata (mandatory)

*Corresponding author.

Email address: `davidlignell@byu.edu` (David O. Lignell)

1. Motivation and significance

Why did we make this? Radiation is hard to deal with in CFD. It's complicated. You can't really get analytic solutions to practical problems. In lots of cases, that's fine because radiation can be safely neglected; it's not usually the dominant mode of heat transfer. However, we do combustion CFD. Radiation isn't always important to combustion problems, but when it is, simulations can get grossly inaccurate. So we need a nice way to account for radiation in our CFD simulations.

When is radiation important to combustion simulations? In high-temperature systems, radiation can often be neglected in the beginning and middle of the simulation (assuming, say, a jet flame or a counterflow flame or something like that). At those times (which I'm going to call "early flame"), convective heat transfer dominates pretty heavily. However, radiation becomes more important for late-stage flame phenomena, either after some time has gone by (radiation time scales?), or physically high up in a flame where convection doesn't dominate as much as below, or just in areas that are relatively far away from the flame sheet itself where the reactions are happening. There's also soot, which involves radiation heavily, but soot particles are so big and slow that their time and length scales (and their radiative time scales) mean that radiation doesn't become important until late in the flame evolution.

So what's the problem? Why hasn't this been done already? Radiation is complicated. So is combustion. Combustion in particular can be so complex that direct simulations are too computationally expensive for us to simulate configurations that are practical for engineering systems. Direct simulations are usually used as a research tool. Other modeling approaches exist (LES, ODT, etc.) that lower the computational cost and allow us to simulate practical things. So far so good.

Radiation is a little like combustion in that its core mechanisms are complex, physically and mathematically. In addition, it's directional AND depends on the wavelength of the energy involved (other heat transfer doesn't have the wavelength dependence); its governing equations integrate over direction and wavelength, which makes things extra complicated. RADIATION GOVERNING EQUATIONS HERE. Convective heat transfer follows easy rules, but radiation doesn't. There are simple systems in which we can boil things down to analytic solutions, but most of the time, the simple equations don't apply, and that's often true in combustion systems where there are so many different length and time scales involved. The fundamental equations of radiative heat transfer are big and mathematically complex; they don't have analytic solutions except in the simplest geometries. So what do we do? One of two things. One, we do ray tracing (sometimes referred to as Monte Carlo simulations. These can be extremely accurate, but super computationally expensive. Essentially the equivalent of a direct numerical solution. Two, we simplify the the equations. We make assumptions. And so on. This is potentially less accurate, but more practical. By simplifying the fundamental governing equations, we create models that apply to various situations, systems, and geometries (i.e. black body assumptions, directional assumptions, etc.). There are models developed for CFD and specific combustion systems. These are the models that we've put into practice here in such a way that they're easy to apply to various systems. Modular organization for this purpose.

So far there isn't an easy way to access and use radiation models. There are simple ones in Cantera (check this, there might not be any), but they make too many assumptions or don't work well for combustion or what have you. RadLib is a radiation library of models that you can apply to any simulation type, and we'd like to add it to Cantera, too. Basically, there isn't a reliable way to do radiation calculations in CFD simulations without coding the models yourself, which is extra hard because they're complicated and hard. So we've done it for you and put them in a library that easy for anyone to use. [1]

Models we're using:

- planck mean (optically thin?)
- two flux model (not sure if we even have this)
- weighted sum of grey gases (CITE Bordbar and others)
- SLW model (CITE Solovjov and others)

Discuss pros and cons of each model for combustion simulations? You still have to pick the right one for your simulation and situation. That discussion might fit better in software description section. See example section for comparisons of models.

2. Software description

2.1. Model descriptions

You need two separate (but very related) models for any given case: one to get radiation absorption coefficients and one for the solution to the governing equations. The first can happen independently of anything else, but the second requires the first. You can sort of mix and match models, but usually we don't, so it's not presented that way in the software.

2.1.1. Planck Mean

2.1.2. WSGG

2.1.3. RCSLW

2.2. Software Architecture

Object-oriented C++ class library. Examples folder contains sample driver scripts for using the library.

Python version somewhere? Corresponding examples?

3. Illustrative Examples

Simple comparisons of each of these models as implemented to some ray-tracing results (equivalent to a direct numerical solution) to establish validity. Illustrates when to use various models and why you might choose one over another. Plots and discussion go here.

4. Impact

As far as I'm aware, there isn't an easy way to incorporate radiative heat transfer into a CFD simulation. Most cases use optically thin assumption (which works in some cases, but not in others) or neglect radiation entirely (also applicable sometimes, but not always). Unfortunately, this means that when you simulation some configuration in which radiation might be important to the overall heat transfer, you won't get accurate simulation results. Furthermore, if your study is about something else entirely, your radiation model (or lack thereof) becomes a source of error that may be very difficult to separate from other sources of error in your simulation study (i.e. soot modeling studies). As of right now, if you want any detailed radiation treatment, you have to code it yourself, which is difficult and requires external validation.

RadLib can make researchers' lives easier by providing a library of prevalidated radiation models that can be switched out with no difficulty. Additional models can be added easily using the provided modular framework. Researchers can even use Radlib as a tool for comparing models. No need to code multiple different complex radiation models yourself just to decide which one works best for your simulation. And no need to puzzle through the literature to figure our which model(s) might be best for your simulation, either; instead, you can test them yourself. It saves tons of time and effort that researchers can now put toward results rather than code or model development.

By putting all these models side by side in a modular framework, RadLib also provides a structure on which new or altered radiation models can be tested against existing ones. Maybe more comparative studies can be done, especially for more complex simulation cases. We plan to add radiation models as appropriate to RadLib, too.

RadLib opens up new horizons for CFD simulations, especially in combustion cases. It is designed such that it can be easily incorporated into existing research codes. Our group plans to use it with ODT [CITE OTHER SOFTWAREX PAPER HERE?] alongside soot model library (in development) to study late-flame phenomena such as soot oxidation, flame extinction, and soot-flame breakthrough. These topics in particular are difficult because they require accurate simulation data over a relatively long computational time in addition to good models for both soot chemistry (which is an active research area) and radiation heat transfer (which is the difficult part that RadLib can solve). Using RadLib for such studies allows us to separate and quantify sources of error that may occur due to various models, which is super difficult if you only have one model to work with or it hasn't been validated well.

RadLib can also be used outside of combustion CFD research for anything involving radiative heat transfer. Potential research areas that could benefit include atmospheric and climate sciences, interstellar phenomena, improving efficiency of energy-producing processes, safety in chemical plant design, etc. We aren't experts in these areas, but radiative heat transfer is a universal phenomena that applies to any system or process that involves heat transfer.

This software has not been used outside of the current research group, and it is not used in any commercial settings at this time.

5. Conclusions

Set out the conclusion of this original software publication.

6. Conflict of Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgements

This work was supported in part by the funding from the National Science Foundation under Grant Number CBET-1403403 and [INSERT SALLY FUNDING SOURCE HERE?].

Special thanks to Vladimir P. Solovjov of Brigham Young University for his insights and assistance with model development.

References

[1] D. O. Lignell, V. B. Lansinger, J. Medina, M. Klein, A. R. Kerstein, H. Schmidt, M. Fistler, M. Oevermann, One-dimensional turbulence modeling for cylindrical and spherical flows: model formulation and application, Theoretical and Computational Fluid Dynamics 32 (4) (2018) 495–520. doi:10.1007/s00162-018-0465-1.

Current executable software version

Ancillary data table required for sub version of the executable software: (x.1, x.2 etc.) kindly replace examples in right column with the correct information about your executables, and leave the left column as it is.

Nr.	**(Executable) software metadata description**	**Please fill in this column**
S1	Current software version	TODO 2.1
S2	Permanent link to executables of this version	TODO For example: *https ://github.com/combogenomics/DuctApe/releases/tag/DuctApe* − 0.16.4
S3	Legal Software License	MIT license (MIT)
S4	Computing platforms/Operating Systems	Linux, OS X, Microsoft Windows
S5	Installation requirements & dependencies	TODO
S6	If available, link to user manual - if formally published include a reference to the publication in the reference list	TODO For example: *http ://mozart.github.io/documentation/*
S7	Support email for questions	davidlignell@byu.edu

Table 2: Software metadata (optional)