PARALLEL-DISTRIBUTED, REVERSE MONTE-CARLO RADIATION IN COUPLED, LARGE EDDY COMBUSTION SIMULATIONS

by

Isaac L. Hunsaker

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Chemical Engineering

The University of Utah

November 2013

The University of Utah Graduate School

STATEMENT OF THESIS APPROVAL

The dissertation of Isaac L. Hunsaker
has been approved by the following supervisory committee members:

David S. Chapman , Chair enter date

_____ _____

Date Approved

Throw this page away and use print out ThesisChildren/SupervisoryApproval.pdf in its place , Member

_____ _____

Date Approved

First M. Last , Member

_____ _____

Date Approved

ABSTRACT

Radiation is the dominant mode of heat transfer in high temperature combustion environments. Radiative heat transfer affects the gas and particle phases, including all the associated combustion chemistry. The radiative properties are in turn affected by the turbulent flow field. This bi-directional coupling of radiation turbulence interactions poses a major challenge in creating parallel-capable, high-fidelity combustion simulations. In this work, a new model was developed in which reciprocal monte-carlo radiation was coupled with a turbulent, large-eddy simulation combustion model. A patch re-composition technique was implemented to allow for scalable parallelism. The combustion model runs in parallel on a decomposed domain. The radiation model runs asynchronously in parallel on a recomposed domain. The recomposed domain is stored on each processor after information sharing of the decomposed domain is handled via the message-passing interface. Verification and validation testing of the new radiation model were favorable. The monte-carlo radiation model that was written for use on CPU-based computers was translated into a GPU-specific language. Strong scaling analyses were performed on the Ember cluster and the Titan cluster for the CPU-radiation model and GPU-radiation model, respectively. The model demonstrated strong scaling to over 1,700 and 16,000 processing cores on Ember and Titan, respectively.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

E = Exact Solution
R = Uniformly distributed random number
L = Length of one side of a cube
ϵ = Statistical error
σ^2 = Statistical variance
Ω = Solid Angle
$Q3$ = 3rd Quadrant of a 2D Cartesian Grid
$Q4$ = 4th Quadrant of a 2D Cartesian Grid
A = Rotation matrix
$\vec{b}$ = x,y,z coordinates rotated into the appropriate orientation
$\vec{x}$ = x,y,z coordinates prior to rotation into the appropriate orientation
θ = Radiometer rotation angle about the y axis
ϕ = Radiometer rotation angle about the x axis
ξ = Radiometer rotation angle about the z axis
θ_v = Radiometer View Angle
θ_r = Ray polar angle
ϕ_r = Ray azimuthal angle
ir = Ray index number
l = A point along a ray
T = Temperature
N = Number of rays traced per cell
q = Radiative flux
I = Radiative Iintensity
κ = Absorption coefficient
$\vec{n}$ = Radiometer normal vector
M = Number of discretized segment lengths of the ray
f = Unity minus the fraction of intensity attenuated by all previous wall reflections
τ = Optical thickness
α = Absorptivity
G = Incident radiation function
σ_s = Scattering coefficient
Φ = Scattering phase function
s = Pre-scattering direction vector
s' = Post-scattering direction vector

SUBSCRIPTS

i = Incident
b = Black body
o = Outgoing
w = Wall
n = Net
r = Ray
cv = Control Volume
sur = Surface

ABBREVIATIONS

TRI = Turbulent Radiation Interactions
DivQ = Radiative Flux Divergence
RMCRT = Reverse Monte Carlo Ray Tracing
DOM = Discrete Ordinates Method
LES = Large Eddy Simulation
GPU = Graphics Processing Unit
CPU = Central Processing Unit
SN4 = 24-direction DOM
SN8 = 80-direction DOM
FSK = Full spectrum k-distribution property model
RHS = Right hand side
RNG = Random number generator
IFRF = International Flame Research Foundation

ACKNOWLEDGMENTS

It is only because of the tremendous amount of help I have received from friends and colleagues that I was able to complete this work. I would first like to thank my advisor, Professor Philip J. Smith, for his support and encouragement. Dr. Smith's energy and optimism created an environment that was condusive to creativity and productivity. His interest and concern for his students' welfare are indicative of his sincerity as a manager and friend.

Dr. Jeremy Thornock was a true mentor to me, giving guidance in areas ranging from Unix tips to writing philosophy. During his stay in the United Kingdom, he still made time to collaborate with me, even at odd hours of the night.

Dr. Todd Harman performed the work that made scaling of RMCRT possible.

Dr. Tony Saad introduced me to the multi-platform program "Inkscape" which greatly enhanced the quality of my graphics. He was always willing to lend an eye to help track down code bugs.

Dr. Charles Reid was active in helping me get up to speed in the world of high performance computing, and even provided a well-documented template for this dissertation.

The work of this dissertation is built upon the work of many others, most notably, the prior graduate student Dr. Xiaoxing (Paula) Sun. Many of the concepts of ray tracing I learned from her thesis and from communication via telephone and email.

Dr. Steven Parker's wisdom in ray tracing efficiency has been invaluable. His insight into parallel ray tracing schemes aided in solving the dilemmas inherent to extreme scaling.

I owe thanks to Dr. Mathieu Francoeur who taught an advanced radiation course in a manner that helped elucidate many complex topics.

Lyubima Simeonova performed the work that allowed for the discretization of spectral-radiation properties.

I would like to thank Professor James Sutherland for his help in ensuring the efficiency of my model as well as his help in implementing the models developed by Ms. Simeonova.

Dr. Sean Smith was always willing to lend an ear when I needed to sound out my thoughts. He was also instrumental in the formulation of statistical analyses for various

tests.

This work would not have been possible without the support of the Center for High Performance Computing at the University of Utah.

This research was made possible by generous grants from the National Nuclear Security Administration under the Advanced Simulation and Computing program through DOE Research Grant # DE-NA0000740 and DE-NT0005015.

Finally, I wish to thank my wife Emily, and our two sons, Jayden, and Aaron for their love and support.

CHAPTER 1

INTRODUCTION

1.1 Literature Review

1.1.1 General Radiation Transport

Radiation is the dominant mode of heat transfer in high temperature combustion environments [2]. Radiative heat transfer in turbulent flames affects the gas and particle phases, including all the associated combustion chemistry. The turbulence-radiation interactions (TRI) have been shown to be of great importance in turbulent flames [3, 4, 5, 6, 7]. Modeling TRI is difficult due to the nonlinear coupling between temperature, species concentrations and radiative intensities [8, 5]. Further, coupling parallel simulations of combustion and radiation poses several numerical challenges. The fluid mechanics of combustion are local phenomena, making them amenable to domain decomposition. Conversely, radiation is a long-distance, and potentially all-to-all phenomenon, creating difficulties for domain decomposition. Further, accurate calculation of radiative transfer requires spatially resolved information regarding the temperature and species composition fields. Traditional modeling of turbulent systems has included Reynolds-averaged Navier Stokes (RANS) simulations. The RANS model provides, at a relatively low computational cost, spatially averaged values of the gas temperature and species fields. However, for highly non-linear physics such as radiation, spatial averaging in this manner may introduce large errors [9]. Alternatively, direct numerical simulation (DNS) fully resolves the power spectrum of eddies, giving access to the full spatial distribution of the pertinent fields. Wu *et al.* [10] and Deshmukh [11] have coupled a monte-carlo ray tracing method to solve the radiative transfer equation in a turbulent reacting flow modeled by DNS. Unfortunately, due to its high computational demand, DNS remains impractical for use in large-scale combustion simulations. In contrast, large eddy simulations (LES) resolve the largest fluid motions, down to the Nyquist limit for a given turbulent field and mesh resolution. Beyond this limit, the less-important smaller eddies are approximated via simpler models. Because combustion turbulence is generally

dominated by large eddies [12], LES gives a better description of the fluid mechanics than RANS, and does so without the computational cost of DNS.

The various levels of accuracy in which thermal radiation has been modeled in combustion simulations have been reviewed by Snegirev [13] and Sacadura [14]. The radiation models cited include the optically-thin approximation [15], the discrete ordinates method [16, 17] the discrete transfer method [18], and the finite volume method [19]. The optically thin model neglects the participation of media (absorption, emission, and scattering), and has been shown to introduce error even in small flames [20]. The remaining methods model radiative emission as energy emanating along a set of pre-defined directions. Such angular discretization suffers from the ray effect [21]. Conversely, monte-carlo techniques that select randomly-distributed rays at each time step have low sensitivity to angular discretization and are applicable regardless of media optical thickness [13]. In his earlier work, Snegirev presents a RANS model of buoyant turbulant diffusion flames coupled with statistical modeling of thermal radiation transfer. Although Snegirev's earlier model used a robust formulation of thermal radiation via the monte-carlo method, his turbulence model suffered from the lack of resolution of the sharply varying fluctuations of temperature and species concentrations that are lost in RANS approximations. More recently, Snegirev coupled monte-carlo radiation with large eddy simulations [22, 23]. These simulations operated on modest meshes of approximately 498,000 control volumes. Other examples of coupled LES monte-carlo radiation models are rare, but include the work of Zhang et al., in which a larger mesh of 4.7 million cells were used [24]. In this emerging field remain several unresolved issues. One such issue is how to deal with increasing mesh sizes that are run on increasingly parallelized high-performance computing systems.

Modern super computers are comprised of hundreds of thousands of computing cores, and are used to run simulations with meshes comprised of billions of computational cells [25, 26, 27]. Strong scaling in massively-parallel computing is difficult to obtain due to load imbalancing and inter-processor communication demands. The strong scalability limit of a code is reached when an increase in the number of parallel processors used on a fixed problem size does not result in a decrease in computational wall time [28]. Numerous examples of parallelized monte-carlo radiation models were investigated by the author, most of which cease to scale beyond 100 processors [9, 24, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43]. An example of a coupled combustion and monte-carlo radiation model that has a scalability limit above 200 processors was not found in the literature.

In this work, a new numerical technique has been developed to perform large eddy

simulations of large-scale combustion flows coupled with a three-dimensional reciprocal monte-carlo ray tracing radiation model. This model has been optimized for use on high-performance computing systems, runs on meshes comprised of 8 million+ cells, and achieves nearly-ideal strong scaling to over 16,000 processors.

As mentioned above, fluid mechanics and most other phenomena in combustion physics are localized phenomena and are readily solved on domain-decomposed meshes. In this work, to represent the long-range effects of radiation, the computational domain is recomposed at the time of each radiation solve. This is accomplished over a message passage interface, through which each processor shares the temperature and radiative-properties fields (absorption coefficients, scattering coefficients, and cell types) with all other processors. This reconstructed domain combined with the mutually exclusive nature of reciprocal monte-carlo rays is amenable to massive parallelism. Radiative properties are calculated via the Hottel and Sarofim method [44]. For efficiency, these calculations are pre-computed and tabulated in narrow increments of temperature and species mixture fraction values.

1.1.2 History of the Monte-Carlo Method

The monte-carlo method was developed by Enrico Fermi, John von Neumann, and Nicholas Metropolis for the Manhattan Project during World War II [45]. The method modeled the behavior of neutrons and involved following the histories of these neutrons during fission. In the early 1960's, Fleck, and later Howell and Perlmutter, applied this method to thermal radiation heat transfer [46, 47, 48, 49, 50]. The monte-carlo method was later adopted and greatly enhanced by the computer graphics community [51, 52, 53]. Since then, the monte-carlo technique has been widely applied to practical problems with participating media [46]. Additionally, this method has been used to produce semi-exact solutions to problems that have no known analytical solution [54].

The monte-carlo method is not without its drawbacks. Because it is a statistical technique, the variance of the error is inversely proportional to the number of rays used in the sampling. The standard deviation is therefore a function of the square root of the number of samples, resulting in slow convergence rates [46]. The monte-carlo method is also computationally expensive when run in serial on a single processor. However, because of the uniqueness of the solutions to each of the rays, reverse monte-carlo ray tracing (RMCRT) is amenable to massive parallelism. In certain cases shown in later sections, this attribute outweighs the low serial efficiency, making RMCRT an attractive option in such scenarios.

1.1.3 Other Numerical Radiation Methods

There is a handful of other numerical models that perform approximations of the RTE. These include the Spherical Harmonics Method, the Discrete Ordinates Method (DOM), the Zonal Method, the Discrete Transfer Method (DTM), and the Finite Volume Method. Each of these models is given a cursory overview in the proceeding paragraphs.

The spherical harmonics method approximates the RTE with a set of simultaneous partial differential equations. This approach was developed by J.H. Jeans in the early 20th century as a way to model stellar radiative-heat transfer [55]. The set of partial differential equations that represent the RTE is significantly simpler than the RTE itself, allowing radiation calculations to be carried out by hand. It was therefore quite popular prior to the advancements of electronic computing. The major downfall of this method is that an increase in the accuracy of the solution comes at the price of higher order partial differential equations, and subsequently much longer computation times [56].

Similar to the spherical harmonics method, the discrete ordinates method (DOM) transforms the RTE into a set of simultaneous partial differential equations. This is accomplished by discretizing the angular domain into a well-defined set of ordinate directions, and integrating along the path lengths. It was first proposed by Chandrasekhar for work in stellar radiation [57], and was later adopted by the neutron transport community [58]. Fiveland and Smith then optimized the DOM for use in general radiative heat transfer applications [59, 60]. Since its inception, this method has been used in many applications including furnaces, diesel engines, and composite materials [61, 62, 63]. Although the discrete ordinates method performs well in serial, an obvious path to GPU-based parallelism is not available [17, 64]. It also suffers from an angular discretization artifact known as the ray effect, which can become particularly pronounced if surface fluxes to small objects are of interest [21]. Further details of the discrete ordinates method can be found in the books by Kourganoff, Davison, and Murray [65, 66, 67].

Another method exists that, unlike the previous two methods, discretizes the domain spatially, rather than angularly. This method is known as the Zonal Method, and was developed by Hottel and Cohen in 1958 [68]. Each sub-volume, or zone, was treated as isothermal, whereby radiative exchange rates between the zones could be computed. An energy balance throughout the domain is then performed to solve for the unknown heat fluxes. Initially the method could handle only non-scattering, gray gases with constant absorption coefficients, but in 1968, Hottel and Sarofim extended the method's capabilities to include isotropically scattering media with non-constant non-gray absorption coefficients

[44].

The discrete transfer method (DTM) was developed by Lockwood and Shaw in 1981 as an attempt to address some of the shortcomings of previous numerical radiation techniques. It was developed specifically for general combustor prediction procedures. The DTM is somewhat of a chimera of several methods and includes features of the zonal, monte carlo, and flux model solution methods. In the words of Lockwood, this method

> "is based on the solving of representatively directed beams of radiation within the enclosure between the known wall boundary conditions and on the subsequent computing of the radiation sources which arise within the finite difference control volumes of the flow procedure due to the passage of the beams. It is fast, exact applicable to complex geometries, and it retains in evidence the physics of the problem by avoiding complex mathematics" [18].

Unfortunately, in the process of simplifying the mathematics, this model has been shown to represent poorly the effects of anisotropic scattering, and similar to its predecessor, the DOM, still falls victim to the ray effect [69, 70, 61, 71, 72]. To address the prevalence of the ray effect in the DTM, Li later modified the model by further discretizing each angular direction into 9 rays with discrete quadrature. This mitigated the ray effect without increasing the number of simultaneous partial differential equations to be solved [73].

More recently, a method has been developed that is catered to unstructured meshes and is designed to accommodate simultaneously for heat conduction, convection and radiation. This method, known as the Control Volume Finite Element Method, was developed by Rousse in 1999. True to its name, this method creates unique, non-overlapping control volumes around each node in the domain. It then performs operations similar to the discrete ordinates method by discretizing the angular domain and solving a set of coupled partial differential equations related to the RTE, which has been modified to account for convection and conduction. This method has similar advantages and disadvantages to the DOM, with the additional advantage of being more amenable to unstructured meshes [70].

1.1.4 Parallel Ray Tracing

Algorithmic parallelism involves dividing tasks among multiple processors to solve simultaneously a given problem [74]. In theory, parallelism can lead to a wall-time speedup that is proportional to the number of processors used in parallel. Ideal speedup occurs when the time spent passing information between processors is negligible compared to the work done by each processor, and when no computers sit idle while others complete their

tasks. The former constraint is met by efficient code writing that ensures that all or most of the information a given processor needs to complete its computations is available to that processor. The latter constraint is met via proper load balancing that distributes the work load equally between processors.

The first attempts to parallelize monte-carlo methods did so on single-instruction multiple-data stream (SIMD) machines. On this architecture, vectors or groups of rays were distributed to the processors. This however, led to poor scalability as it necessitated the termination of all rays before generating a subsequent group. More recent algorithms generate new rays at the onset of termination of a ray to avoid creating idle time amid processors [75].

Load balancing may be accomplished in at least two general ways–dynamic load balancing, and static load balancing. Static load balancing schemes distribute the load only once, at the onset of computation. However, because the computation times of different regions of a domain are problem-dependent and rarely uniform, static load balancing often creates idle time amid processors. Dynamic load balancing begins with an initial load distribution which can then be modified if and when computers complete their original tasks. Heirich and Arvo have noted that when total computational time is of importance, static load balancing is insufficient for parallel ray tracing on massive high-performance computing systems [76].

Strong scaling in massively-parallel computing is difficult to obtain due to load imbalancing and inter-processor communication demands. The strong scalability limit of a code is reached when an increase in the number of parallel processors used on a fixed problem size does not result in a decrease in computational wall time [28]. Some methods to parallelize ray tracing for radiation applications do so by passing between processors rays that have breeched the local grid extents. Wise and Abel of Princeton and Stanford Universities, respectively, have expended considerable efforts on their parallelization strategy of their ray casting scheme for the coupled hydryodynamics radiation code, ENZO. Unfortunately, strong scaling analysis of this algorithm showed no improvement of computational time for parallelism at 70 or more processing cores [29, 30]. This is perhaps due to the large amount of communication that resulted from the passing of rays between processors. Kuiper *et al*. developed a similar parallel ray tracing scheme for computing radiation transport in stellar formations, but to date, have not demonstrated strong scaling beyond 64 processors [31]. In 2009, Gentile successfully scaled ray tracing for radiation calculations to 128 processors. Any further increase in processors resulted

in no further decrease in computational time [43]. Numerous examples of parallelized monte-carlo radiation models were investigated, most of which ceased to scale beyond 100 processors [29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 9, 40, 24, 41, 42, 43]. An example of a coupled combustion and monte-carlo radiation model that has a scalability limit above 200 processors was not found in the literature.

As part of this dissertation, a parallel-capable radiative-flux solver that scales to 16,000 processors has been created. Such scaling is attained by avoiding the Message Passing Interface (MPI) by stitching together a decomposed domain into a global domain for each processor, and utilizing reverse monte-carlo rays that traverse freely throughout the domain until extinction.

The computer graphics community has taken advantage of Graphics Process Units (GPUs) to run parallel ray tracing. The shared memory of the GPU and breakdown of each GPU into hundreds of cores allows for the simultaneous tracing of hundreds of rays per GPU. GPUs are allowing for real-time volume rendering in the graphics community [77]. In 2008, Despres showed that a simplified monte-carlo ray tracing algorithm performed 6 times faster on an nVidia 7600 GS GPU than it did on a Xeon 2.4 GHz CPU. This speedup is further increased for larger numbers of computational cells, indicating that at finer meshes, the GPU will dominate in ray tracing efficiency [78].

Todd Harman and Alan Humphry of the University have translated the RMCRT model described in this thesis into the CUDA language, allowing RMCRT to run on the GPU nodes of the supercomputer, Titan. The CUDA version of RMCRT attained ideal strong scaling to 16,000 cores.

1.1.5 Previous Work in RMCRT By Paula Sun

A former student of our laboratory, Paula Sun made significant contributions to a reverse monte-carlo ray tracing algorithm. She performed an extensive literature review on the topic, consulted a ray tracing expert, and designed a stand-alone radiation solver [45]. Her dissertation outlined the governing equations for a RMCRT scheme for an absorbing, emitting, scattering medium with gray, diffuse/specular boundaries. She also identified several benchmark cases that have proven useful in the verification of the algorithms described in this dissertation. Although her work was not compatible with the Uintah framework and was not designed for parallel performance, it paved the way for this work that has such capabilities.

1.2 Research Objectives

Accurate radiative-heat transfer solving methods that handle complex physics are inherently computationally expensive. The legacy radiation solver, the Discrete Ordinates Method, which is used within the Arches component, is not amenable to GPU applications. It also suffers from the ray effect inherent to fixed-angular discretization techniques.

To address the above issues, the following objective has been proposed and met: Develop a reverse monte-carlo ray tracing scheme that computes the radiative-flux divergence for interior cells, and the net radiative flux for boundary cells of a computational domain. To accurately represent reality, the model incorporates the following physics

- Non-homogeneous, absorbing, emitting media
- Homogeneous, isotropic, scattering media
- Black or gray wall absorption and emission
- Specular wall reflections for arbitrary reflectivities
- Complex domains which may include intrusion features
- Fluxes with arbitrarily sized view angles, orientations, and locations

This final item comprises the "virtual radiometer" feature and allows the user to define the parameters of radiometers such as locations, orientations, and view angles. This will allow the user to perform validation and uncertainty quantification with flux measurements from experimental radiometers.

From the onset of this research project, parallel capability of the radiation solver has been an area of focus. Several parallelism techniques were considered. These include

1. Parallelize by patch domain decomposition with global information, eliminating the need to hand off rays
2. Parallelize by patch domain decomposition with local information only, and hand off rays between the patches
3. Parallelize with a hybrid of (1) and (2). Each processor stores the fine information for a different region of the grid which becomes the focal region for that processor. Each processor is then passed a coarsened version of the rest of the domain, eliminating the need to hand off rays.

The details of these techniques as well as their respective advantages and disadvantages are given in Section 2.1.2.

1.3 Expected significance

High performance computing systems are being equipped with an ever-increasing number of graphics processing units (GPUs) [79]. Examples include the Keeneland Initial Delivery System (KIDS) and the conversion of the DOE Jaguar system to Titan [80, 25]. The radiation calculations, which represent a small fraction of the overall physics involved in a complex fire simulation, consume 20 to 80% of the total simulation time. For a given level of accuracy, this research suggests that RMCRT run on CPU clusters is faster than the DOM (see section 9.1). A GPU-compatible version of RMCRT has demonstrated an additional 4 to 5X improvement over the CPU version [64]. One of the key developments of this research has been the capability to stitch together a patch-decomposed domain into one that is unified. This has eliminated the need to hand off RMCRT rays between processors during run time, reducing inter-processor communication allowing for the parallel scalability demonstrated above.

1.4 Broad description of modeling

An efficient reverse monte-carlo ray tracing algorithm includes a handful of key components. Among them are a fast, reliable random number generator, an efficient ray marching algorithm, and generality to handle phenomena including reflecting rays, non-cubic cells, and scattering. The model ultimately solves for radiative fluxes and radiative-flux divergences, so appropriate equations for those physics are also used. The developed model is one that gives the user the ability to dial-in the desired accuracy/speed through variable numbers of rays and/or mesh resolutions. Another primary objective has been to develop a radiation model that can scale in massive parallelism. The chosen route to parallelism has been to reconstruct a domain-decomposed mesh to allow for the uninterrupted tracing of rays to extinction.

In summary, the modeling undertaken for this research has been two-fold: first, produce a high-fidelity RMCRT algorithm that incorporates the major physics of radiation, and second, create a scheme that allows for efficient massive parallelism of this algorithm.

1.5 Components and Architecture

The Uintah framework was originally designed by the Center for the Simulation of Accidental Fires and Explosions (C-SAFE). This software suite was funded by the Department of Energys Accelerated Strategic Computing Initiatives (ASCI) Academic Strategic Alliance Program (ASAP). Uintah provides a software system in which complex multi-scale, multi-physics chemistry and engineering simulations can be performed. To provide a means

under which a variety of problems can be simulated, Uintah makes use of a component system under which large pieces of software (components) can be implemented independently [81, 82]. To the maximum extent possible, RMCRT was developed generically to allow the model to be used by any Uintah component. Because members of our research group have the greatest experience in the Arches component, validation and verification were performed primarily therein.

The Arches component solves the conservative, finite volume, compressible, low-mach formulation of the Navier-Stokes equation [83]. It was designed to solve the mass, momentum, mixture fraction, and thermal energy governing equations inherent to coupled turbulent reacting flows. Arches is a Large-eddy simulation component, and as such resolves the fluid motion down to the Nyquist limit of the power spectrum. It has relied heavily on the legacy Discrete Ordinates Method (DOM) for the radiative source term solve [84].

Performance profiling indicates that the DOM solve represents the most computationally intensive portion of an Arches combustion simulation. Although DOM can be made to scale [85], it is unknown whether this radiation solver is amenable to GPU computing and embarrassing parallelism. Conversely, RMCRT lends itself to scalable parallelism because the intensities of the rays are mutually exclusive. Therefore, multiple rays can be traced simultaneously from any location in the computational domain [86].

CHAPTER 2

MODEL DESCRIPTION

2.1 Model Description

The robustness of monte-carlo ray tracing to predict radiative fluxes has previously been established by Snegirev and Modest among others [13, 87]. In a massively parallelized framework, where the computational domain is heavily decomposed, traditional forward monte-carlo methods (FM) suffer due to the large number of traced rays that never reach the subdomain of interest handled by a particular processor. Therefore, an emission-based reciprocity method (ERM) similar to that developed by Tesse et al. [88, 89] has been implemented. In this model, optical paths (*i.e.* rays) propagate away from cells whose radiative-source terms are currently being solved, and the emission from the cells along the paths are attenuated in a reciprocal manner back to the origin cells. In this manner, rays are generated only from cells where results are expected [90].

The governing equation for reciprocal monte-carlo ray tracing in nonhomogeneous, participating media was developed by Walters and Buckius [91]. Specifically,

$$I_{i,k} = \int_0^{l_k} I_{b,cv}\kappa(l')exp[-\int_{l'}^{l_k}\kappa(l'')dl'']dl' + I_{o,sur}(T_w)exp[-\int_{l_w}^{l_k}\kappa(l')dl'], \tag{2.1}$$

where $I_{i,k}$ represents the incident intensity at location k, κ represents the absorption coefficient, and l' represents the locations of the segment lengths along a ray.

In a discretized domain, piecewise homogeneity is assumed and Eqn. (2.1) is posed in the following form,

$$I_{i,k} = \sum_{m=1}^{M}\left(I_{b,v}(e^{-\int_{l_m}^{l_k}\kappa(l')dl'} - e^{-\int_{l_{m-1}}^{l_k}\kappa(l')dl'})\right) + I_{o,s}(T_w)e^{-\int_{l_w}^{l_k}\kappa(l')dl'}, \tag{2.2}$$

where M represents the total number of discretized segment lengths of the ray. Thus, at a given location a distance l' away from the starting point of the ray, $\int_{l_m}^{l_k}\kappa(l')dl$ represents the optical thickness of the path from l_k to l' and $\int_{l_{m-1}}^{l_k}\kappa(l')dl$ is the optical thickness of the path from l_k to the previous l'. The intensities from each of the rays of a cell are then weighted according to the solid angle that each ray subtends [45]. Assuming uniform distribution of the rays, each ray subtends $\frac{\Omega}{N}$ steridians, where Ω is 4π Sr. for flow cells, 2π for boundary cells, and N is the number of rays per cell used in the simulation. The radiative flux is then calculated from the intensities of the rays, weighted by the discretized solid angle,

$$q_i = \frac{\Omega}{N}\sum_{r=1}^{N} I_i(ir)cos(\theta(ir)), \tag{2.3}$$

where $I_i(r)$ and $\theta(r)$ represent for a particular ray, the incoming intensity and angle from the cell boundary normal, respectively. The radiative flux divergence of flow cells is calculated as

$$\nabla \cdot q = \kappa(4\pi I_b - \int_{4\pi} I_{in} d\Omega), \tag{2.4}$$

where $\int_{4\pi} I_{in} d\Omega$ is represented by

$$\sum_{r=1}^{N} I_r \frac{4\pi}{N}.$$

The origin locations of the rays are distributed randomly throughout the cell. In this model, the Mersenne Twister random number generator is used to select the origin locations and ray orientations [92]. In Cartesian meshes, randomly distributed ray location generation is trivial, and is accomplished by scaling three random numbers with the length, width, and height of the cell, respectively. Randomly-distributed ray orientation requires more treatment. Rays propagating from boundary surfaces are distributed over a hemisphere as follows.

$$\phi = 2\pi R_1$$

$$\theta = \arccos(R_2)$$

$$\hat{x} = \sin(\theta)\cos(\phi)$$

$$\hat{y} = \sin(\theta)\sin(\phi)$$

$$\hat{z} = \cos(\theta).$$

R_1 and R_2 are random numbers that vary between zero and one, ϕ and θ are the azimuthal and polar angles, respectively, and $\hat{x}$, $\hat{y}$, and $\hat{z}$ are the resulting components of the direction vector in Cartesian coordinates. The above formulation generates rays that are randomly distributed over a hemisphere with a normal vector in the positive z direction. The ray marching model adjusts the ray directions into the proper orientation based on the surface normal of the boundary cell at hand. This is accomplished by changing the order and sign of the three direction components.

For flow cells, no re-orientation of the direction vector is necessary, as the rays are randomly distributed over the full 4π Sr. Direction assignment is as follows,

$$\hat{z} = 2R_1 - 1$$

$$r = \sqrt{1 - z^2}$$

$$\phi = 2\pi R_2$$

$$\hat{x} = r\cos(\phi)$$

$$\hat{y} = r\sin(\phi)$$

Ray marching proceeds in a manner similar to that described by Amanatides and Woo [93]. The location and orientation of a ray are used to calculate the distances to each of the 3 potential exit faces of the cell in which the ray currently resides. The shortest of these three distances is used in determining through which face the ray will pass. This information is then used to calculate the next cell in which the ray will reside. Reflections are allowed to occur on non-black boundary faces. The temperature and emissivity of the boundaries are referenced, and the intensity at the ray-boundary intersection is computed and attenuated to the target location. For non-black surfaces, the ray reflects off the surface, and the subsequently referenced intensities are attenuated both by the total optical thickness and by the absorption of the boundary. Ray marching continues until the optical thickness of a ray exceeds a predetermined threshold value. In general, the threshold is met when $fe^{-\tau} < 0.01$, where τ is the current optical thickness, and f is unity multiplied by one minus the absorptivity of each intersected boundary, $(1 - \alpha_b)$. In other words when less than 1% of the intensity from a location in the domain will reach the target cell, ray tracing of the current ray ceases.

2.1.1 Scope

The scope of this dissertation includes the demonstration of a parallel, reverse monte-carlo ray tracing (RMCRT) model that incorporates the following physics.

- Non-homogeneous, absorbing, emitting media

- Homogeneous, isotropic, scattering media
- Black or gray wall absorption and emission
- Specular wall reflections for arbitrary reflectivities
- Complex domains which may include intrusion features
- Fluxes with arbitrarily sized view angles, orientations, and locations

Each of these items has been discretized and modeled in the RMCRT algorithm. Some detail on the implementation of each of the above items is given below.

To accommodate for item (1), non-homogeneous, absorbing, emitting media, each cell in the domain traces N number of rays from each cell within the patch, and follows those rays throughout the domain until extinction. These rays pick up and attenuate intensity according to the Radiative Transfer Equation (RTE) for an absorbing, emitting and scattering medium, according to the following expression,

$$\frac{\partial I(s,\hat{s})}{\partial s} = \kappa I_b(s) - (\kappa + \sigma_s) I_b + \frac{\sigma_s}{4\pi} \int_{4\pi} I(\hat{s}_i) \Phi(\hat{s}_i, s) d\Omega_i. \tag{2.5}$$

The intensity of the cells through which the rays pass are accumulated and attenuated along their respective paths back to the origin. The intensity contributions from all the rays for a given cell are summed, and scaled by $\frac{\Omega}{N}$, where Ω represents the solid angle, and is generally specified as 4π $Sr.$, a full sphere. This scaled value gives, for a specific cell, the contribution from the intensities from all cells in the entire domain, and is frequently denoted as the incident radiation function, G. This value is then be used to yield the radiative-flux divergence as follows,

$$\nabla \cdot q = \kappa(4\pi I_b - G). \tag{2.6}$$

These equations are solved and discretized for use in a Cartesian mesh.

When item (2), homogeneous, scattering media, is introduced, the ray marching algorithm allows the ray direction to change at any cell boundary within the domain. This involves calculating scattering lengths to determine when a ray will scatter, as well as determining the new ray direction based on the scattering phase function.

The distance between scattering events is determined from the cumulative probability function,

$$l_{\sigma_s} = \frac{1}{\sigma_s} ln \frac{1}{R_{\sigma_s}},$$

where σ_s is the scattering coefficient, and R_{σ_s} is a random number with a range of zero to one.

The azimuthal and polar angles of the new direction are be determined from

$$R_\psi = \frac{\int_0^{\psi'} \int_0^{\pi} \Phi(s \cdot s') sin\theta' d\theta d\psi'}{\int_0^{2\pi} \int_0^{pi} \Phi(s \cdot s') sin\theta' d\theta d\psi'},$$

$$R_\theta = \frac{\int_0^{\theta} \Phi(s \cdot s') sin\theta' d\theta}{\int_0^{\pi} \Phi(s \cdot s') sin\theta' d\theta},$$

where Φ is the scattering phase function and is a function of the particles in the media as well as the original direction vector, s, and the new direction s'. When isotropic scattering is assumed, as in this model, the following formulation is used to compute the components of the post-scattering direction vector.

$z = 2R - 1$

$r = \sqrt{(1 - z^2)}$

$\theta = 2\pi R$

$y = rsin(\theta)$

$x = rcos(\theta)$

For item (3), wall absorption and emission, the intensity of a ray is augmented by the emission and absorption of the boundaries that the ray strikes. When a ray strikes a wall, the intensity that is accumulated at the ray origin is increased by the following amount, $I_{o,sur}(T_w) exp[-\int_{l_w}^{l_k} \kappa(l')dl']$, where $I_{o,sur}$ is the intensity of the wall at the surface location struck by the ray, and the exponential function represents the attenuation of that intensity on its path back to the ray origin. To incorporate these physics into the RMCRT model, Eqn. (2.2) is modified to include this additional term,

$$I_{i,k} = \sum_{m=1}^{M} \left(I_{b,cv}(T_m)(exp[-\int_{lm2}^{l_k} \kappa(l')dl'] - exp[-\int_{lm1}^{l_k} \kappa(l')dl']) \right) + I_{o,sur}(T_w) exp[-\int_{l_w}^{l_k} \kappa(l')dl']. \tag{2.7}$$

For item (4), specular wall reflections for arbitrary reflectivities, several modifications were introduced. First, a subroutine was created that is called at the moment a ray enters a non-flow cell. This new cell is referenced for item (3). Because the location of the ray is now outside of the flow domain, the ray is backed out into the flow region. The next step is

to accurately determine the new direction of the ray. Assuming the domain is one in which the faces of all cells line up with the Cartesian directions, as is the case in Arches, then this step is simplified significantly. The calculations essentially reduce to flipping the sign of the component of the direction vector that corresponds to the boundary face that was struck. See section (3.1.7) for a more detailed derivation.

For item (5), complex domains which may include internal features, an additional test along each step of the ray marching algorithm is implemented. The test is simple and is essentially an $if/else$ statement conditional on the cell type of the current cell. If the cell type corresponds to a flow cell, ray marching continues without interruption. Otherwise, the subroutines for items (3) and (4) are called.

Item (6) has been labeled as the "Virtual Radiometer Model." Its implementation consists of allowing the user to specify in an input file a radiometer location, orientation, and view angle. These parameters would correspond to a true radiometer that was placed in a fire during a combustion experiment. This has allowed validation efforts to be performed between experiments and simulations.

In addition to the 6 items mentioned above, the developed RMCRT model offers versatility in accuracy and speed. The user may select any positive integer number of rays to produce the desired accuracy/speed ratio. Further, the number of rays used for boundary luxes, flux divergences, and virtual radiometer fluxes remain independent. This gives the user the ability to separate the flux solution from the divQ solution. For example, if the user is most interested in the fluxes at a handful of locations or a series of locations that lie in a single plain, as often occurs in validation cases, the user can run with relatively few rays to produce basic radiative effects in the flame, while using extremely fine angular discretization at the locations of interest.

2.1.2 Parallel RMCRT

RMCRT lends itself to massive parallelism because each ray's intensity is independent of all other ray's. Multiple rays are traced simultaneously at a given time step. In theory, one could simultaneously trace as many rays as one has processing cores. For this method to scale, however, inter-processor communication should be low relative to the work of ray tracing. This is challenging, considering that radiation is a globally-coupled phenomenon.

Several parallelism techniques were considered. These include

1. Parallelize by patch domain decomposition with global information, eliminating the need to hand off rays

2. Parallelize by patch domain decomposition with local information only, and hand off rays between the patches

3. Parallelize with a hybrid of (1) and (2). Each processor stores the fine information for a different region of the grid which becomes the focal region for that processor. Each processor is then passed a coarsened version of the rest of the domain, eliminating the need to hand off rays.

Item (1) has the advantage of avoiding the message passing interface for each ray. The disadvantage, however, is that this scheme would require information about the entire domain to be stored on each processor. Specifically, the absorption coefficient, temperature, cell type and scattering coefficients fields, each of which is a double precision variable, of $O(n^3)$ cells, where n is the number of cells in a single direction, would be stored on each processor. This is a strict requirement, and would limit the size of the domain to approximately 350^3 for the current computers.

One advantage of item (1) is that it takes advantage of existing software in the Uintah framework that allows the domain to be decomposed by patches, or subsections of the domain. In this approach, a processor computes the flux divergence of each cell within its patch by tracing N rays from each cell and allowing them to march through the entire domain.

Item (2) is not limited by memory as item (1) is, as it does not store any global information. However, it is suffers from another factor–interprocessor communication. Because radiation is a global phenomenon, physically-accurate rays would travel through the entire domain. Yet for item (2), as soon as a ray leaves a local patch, the information necessary to compute an incoming intensity is not available to the processor that owns the origin cell of the ray. Therefore, the processor would either request the information of the adjacent patch, (and in optically thin domains, the adjacent to the adjacent, and so on), or it would hand off the ray to an adjacent processor. This would amount to the handing off of millions of rays for a single time-step of a typical simulation, and would burden the message passing interface, likely degrading the parallel performance.

Item (3) is somewhat of a hybrid of items (1) and (2) in that the domain is decomposed into patches, the fine information existing only locally, yet the global information still existing on the processor, but in a coarsened state. The simplest case would be a fine focal level that has the same resolution as the rest of the domain, *i.e.* the whole domain is on the same refinement level, leading to the case described in situation (1). A slightly more advanced case is one where all but the local patch are coarsened to a single, coarse

state, leading to a total of two levels. More advanced cases can be imagined where the patches adjacent to the focal patch are coarsened slightly, the ones adjacent to those are coarsened moderately, and the most distal patches are coarsened heavily. Such a multi-level approach allows the information most pertinent to the incoming intensity to be left relatively un-coarsened, yet information that is distal to be less accurate, and therefore less memory intensive. Justification for coarsening the distal information is two fold. First, the cells that are not proximal to a given set of focal cells is separated by an optical thickness that will attenuate the intensity between them, thereby limiting the effect on the focal cells. Second, the distal cells subtend a smaller solid angle than the proximal cells, again limiting the effect on the focal cells. This second effect is demonstrated numerically in that a distal cell will have a far smaller probability of being intersected by a given ray than would a proximal cell of comparable size.

Using item (3), the most pertinent information of the domain is preserved, intra-timestep message passing is avoided, and the distal regions are coarsened to an extent that balances the desired accuracy with the memory constraints of the computer at hand. The radiation solver has been designed to run with a coupled CFD component such as Arches or Wasatch, and as such will usually deal with information that is domain decomposed. To stitch together and coarsen the non-focal region of the domain, the domain information that is scattered between the processors is requested by each processor. Therefore, at the beginning of each time-step, a considerable amount of information passes through the message passing interface. Shy of reverting to the use of (2), which has its own message passing requirements, this "start-up" message passing is unavoidable. Fortunately, the amount of start-up message passing decreases when multiple levels are used, as coarser and coarser versions of the domain are being stitched together.

To optimize run-time efficiency, items (1) and (3) were selected for development. At present, item (1) has been successfully developed and implemented and favorable results have been obtained. Item (3) has also been developed, but results were less favorable.

Both methods involve a mesh reconstruction technique that allows ray generation and propagation to occur on a each processor independently, negating the passing of rays, and minimizing inter-processor communication.

At each radiation solve, the decomposed domain used for parallelism of the combustion model is recomposed and the radiation-specific field variables from each processor are shared with all other processors. Information sharing is accomplished through a message-passing interface.

2.1.3 Adaptive Focus Mesh Refinement

The patch recomposition technique mentioned in section 2.1.2 is viable only so long as each processor has sufficient memory to store the temperature, absorption coefficient, cell type, and scattering coefficient fields for the entire domain. However, for a typical processor commanding 4GB of RAM, segmentation faults begin appearing for domains larger than approximately 350^3 cells. A potential solution to this problem is to discretize the domain such that each processor is handed a subset of the domain, called a patch. For regions outside the patch, the processor is handed a coarsened version of the rest of the domain. This takes advantage of the existing framework common to parallelized LES codes such as the Arches code owned by the Institute for Clean and Secure Energy[94]. To create a mesh with multiple refinement levels, framework and data management adjustments have been made to the ARCHES component. The ray tracing algorithm has also been modified to run on this adaptive-focus mesh (a.k.a. Data Onion). Modifications include adjustment of the step size once a refinement level boundary has been reached, as well as an algorithm to determine the new ray location upon entering a new level. At present, the Data Onion approach is producing unfavorable timing and accuracy results. Further investigation may reveal a flaw in the programming or perhaps the method itself. Details of the implementation are given in section (7.3).

Although the adaptive-focus mesh addresses the issue of global storage of radiation field values, thus avoiding the passing of rays between processors, it does not completely avoid the message passing interface (MPI). Arches and other Uintah components perform parallelism via patch domain decomposition. In this paradigm, an intact version of the entire domain simply does not exist. Portions of the domain are stored amid the various processors, so during the creation a composite mesh, each processor gathers the patches from all other processors. The more time that is spent on passing information between processors, the less efficiently the algorithm will scale. To mitigate the amount of data that is handled on the MPI, I propose aggressive coarsening on regions of the domain that are distal to the focus region. Justification for this is two-fold. First, the physical distance between the distal and focus regions increases the optical thickness. Therefore, any contribution from the distal regions will be attenuated exponentially along that path length, thus decreasing the effect of the distal region on the origin cells. Second, the distal regions subtend a smaller solid angle than do proximal regions, again limiting the impact on the focus cells.

2.1.4 Function Abstraction

Originally, the code that handled ray marching and the updating of intensity along the ray resided within the function that contained the code that determined the directions and locations of rays. However, as development progressed to include solvers for boundary fluxes, imaginary surface fluxes, and virtual radiometer fluxes in addition to the original flux-divergence solver, the code began to become cluttered with a series of "if" statements, and other conditional statements, depending on which physics were being handled at the moment. Because each of these four physical phenomena requires information about intensities along rays, the intensity solver was isolated into its own function that could be called independently from the other solvers. For instance, the virtual radiometer model uses cell-centered rays that have direction vectors distributed across a small solid angle defined by the view angle of the radiometer, while surface fluxes use face-distributed rays that have direction vectors that span the 2π hemisphere of the surface, yet both of these methods ultimately require the intensity integrated along each ray. Therefore, it was most intuitive to have the ray-marching and intensity solver abstracted into its own function that takes as arguments the location and direction of a ray as specified by the solver at hand. In this manner, we also avoid cloning portions of the code into multiple files, which would otherwise necessitate maintaining and updating several independent files every time one wished to make a change to any one of the files. There was a marginal increase in computational time (˜8%) that was introduced when the intensity solver was abstracted into a function, but the end result was cleaner code that is easier to maintain and enhance.

2.1.5 GPU implementation

The author collaborated with Todd Harman and Alan Humphrey of the University of Utah as they developed a GPU version of RMCRT. The radiation model was translated from its original language of C++ into the GPU-specific language, CUDA. This allowed the model to be run on the GPU processors of the super-computing cluster, Titan. Strong scaling was demonstrated up to 16,000 processors. Results and discussion can be found in section (7.2)

CHAPTER 3

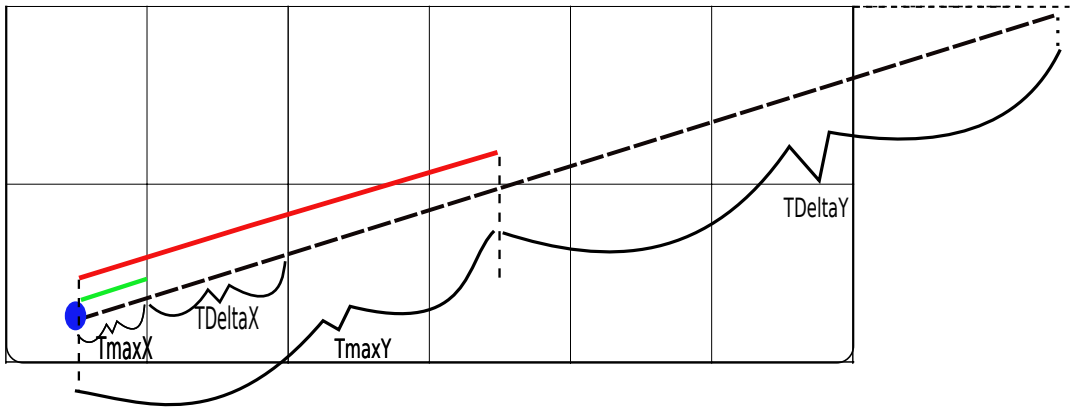
RAY TRACING

3.1 Ray Marching Algorithm

3.1.1 Cubic Cells

This section demonstrates the procedure of the ray marching algorithm in determining the piecewise path taken by a ray as it moves through the computational domain. For simplicity, this description will be carried out in two dimensions, although the same principles are applied in three dimensional cases.

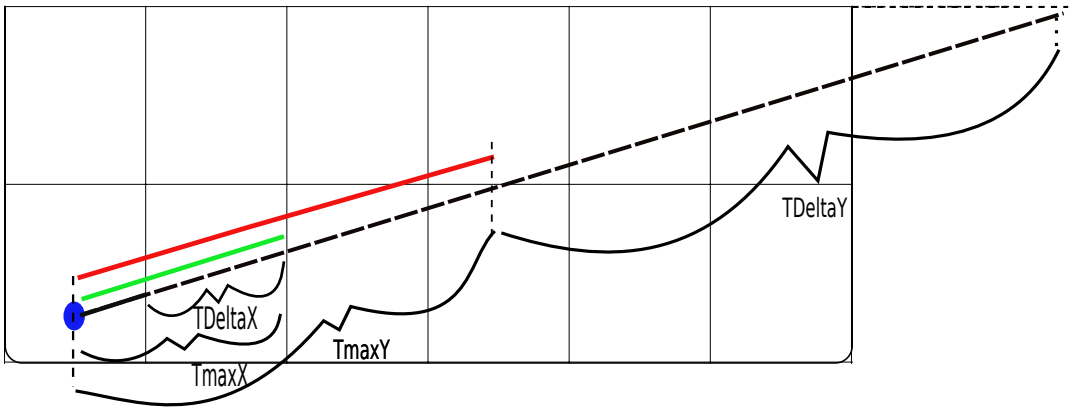
Let us begin with a small computational domain with two rows of cells, and six columns of cells, indexed as (i,j), where i represents the row number starting from the bottom of the domain, and j represents the column number starting from the left of the domain. Let the vertical lines represent x planes, and the horizontal lines represent y planes. A ray is to be traced from cell (1,1) from the ray origin indicated by the blue circle in the below figure. Assume that a ray direction has already been determined, and will be represented by the long dashed line below.



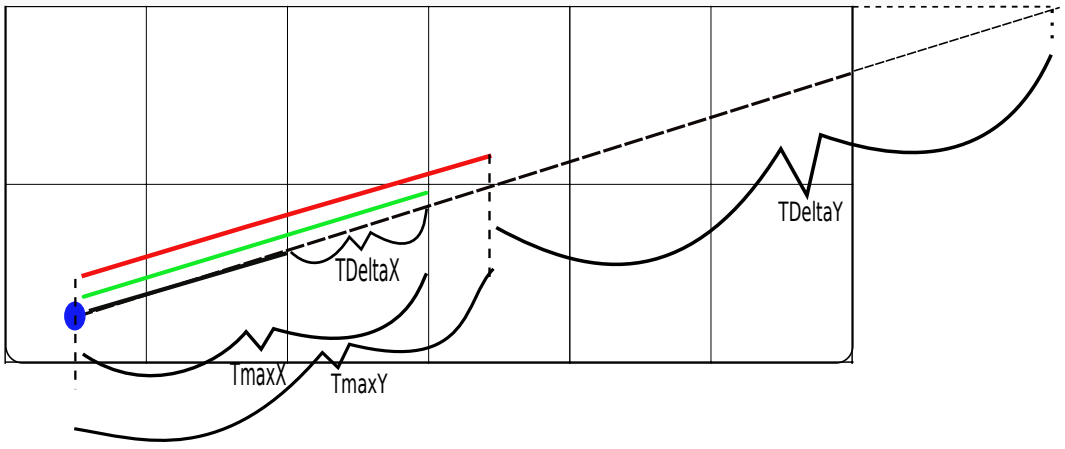
The length of the ray segment from the origin to the first cell wall that is breached by the ray is determined. At this point in the algorithm, it is unknown whether the x plane or a y plane will first be breached by the ray. To determine which plane will be breached (and subsequently determine the next cell that the ray will enter) the distance from the origin to the first x plane, TmaxX (green), is compared to the distance from the origin to the first y plane, TmaxY (red) in the direction of the ray. In two dimensions, this is accomplished by a simple "if/else" statement (additional comparisons are necessary in three dimensions).

Once the shortest of the two distances is determined, the current cell is updated by use of the step variable. In this case, the shortest direction is TmaxX, so the ray steps in the x direction. The distance traveled (in this case the green line above) is stored as disMin, and will be used later in an algorithm that determines ray attenuation. Because the x component of the direction vector is positive, the cell index is incremented by 1 in the x

direction, and the current cell becomes (1,2). The ray has progressed, and the scenario is now represented by the following figure.

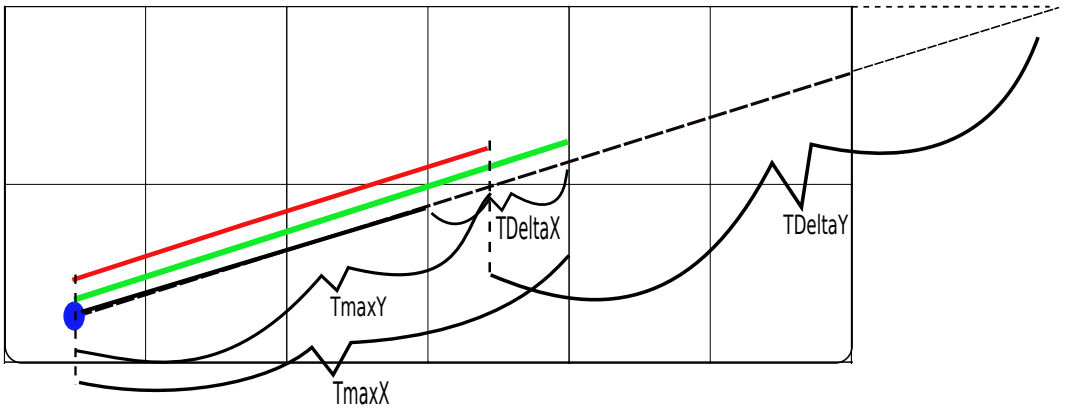


Note that the first segment of the dashed ray has now become solid. The distance from the origin to the first y plane has not changed, so the red line representing TmaxY, remains unchanged. However, the distance from the current location to the next x plane has changed, and its length has been increased by the distance TDeltaX. TDeltaX represents the distance required to traverse one cell length in the x direction. With the updated value for TmaxX, again the green line is compared to the red line. Because TmaxX is still shorter than TmaxY, the ray again steps in the x direction, incrementing i. The current cell then becomes (1,3), and TDeltaX is stored as disMin for later use. The following figure demonstrates the current state of the ray.

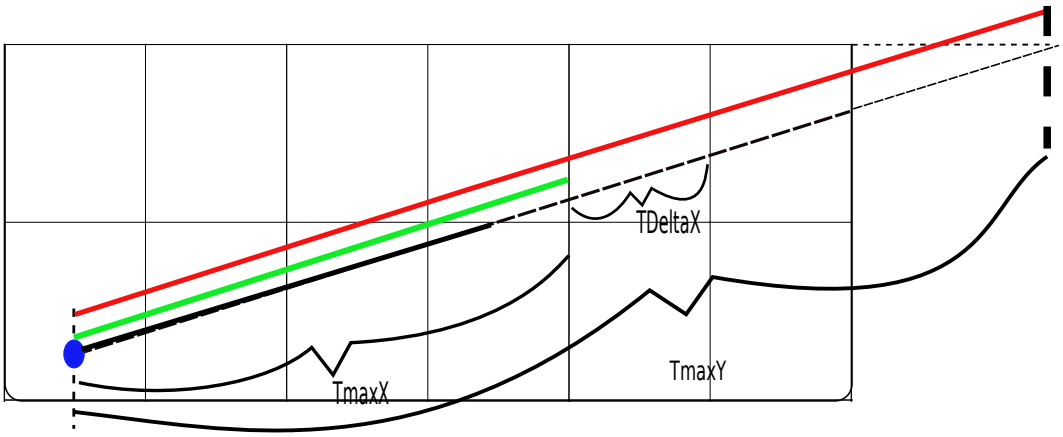


Now, the second segment of the dashed ray has become solid, and TmaxX has been increased by TDeltaX. Note that for a given ray in a uniform mesh, TDeltaX and TDeltaY do not change, as the distance required to traverse a cell in the x or y direction is independent of the current cell. Again TDeltaX is compared to TDeltaY. The green line is still shorter

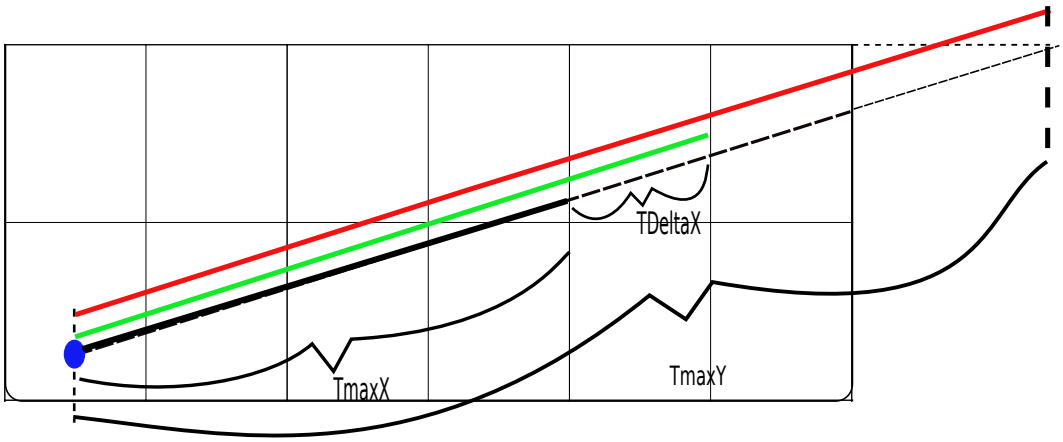
than the red, so the ray again steps in the positive x direction, TDeltaX is stored as disMin, and cell (1,4) is reached as shown below.



The third segment of the ray has now become solid. The value of TmaxX is increased by TDeltaX, and for the first time in this example, TmaxX exceeds the length of TmaxY. Thus, the ray steps in the y direction and TDeltaY is stored as disMin. Because the y component of the direction vector is positive, j is incremented and the ray enters cell (2,4) as shown below.



The fourth segment of the ray is shown as solid, and TmaxY has been increased by the distance TDeltaY. It is visually apparent that TmaxX is much shorter than TmaxY, and therefore the comparison in the algorithm would lead to a subsequent step in the x direction into cell (2,5) as shown below.



The fifth segment has become solid, and TmaxX has been increased by TDeltaX. At this time, the reader should be familiar enough with the algorithm to predict that the next two steps will be in the positive x direction, at which point the ray would either terminate if the wall is black, or reflect based on the reflection algorithm which will be discussed in later sections. Also in later sections, the reader will find a discussion of the attenuation of radiation from each of the cells along the ray path back to the origin, and the importance of disMin will become apparent.

3.1.2 Non-cubic Cells

For domains that contain cells with non-unity aspect ratios, additional considerations become necessary in the ray marching algorithm. Throughout the above described algorithm, distances are handled in units of cell width, which can then be converted to physical units simply by multiplying by the cell size of Dx. However, when Dx is not equal to Dy or Dz, this conversion becomes non-trivial, and requires additional computations. By normalizing the lengths of Dy and Dz by Dx, the number of additional computations is minimized, such that only 6 lines of code require modification. Explanation of this procedure is as follows.

Take the distance Dx to be of unit length. Then Dy and Dz have normalized lengths of $\frac{Dy}{Dx}$ and $\frac{Dz}{Dx}$, respectively. The first section of the algorithm that requires modification is then the determination of ray origins. Previously, for cubic cells, it was assumed that the origin was located at $(i+rand(),\ j+rand(),\ k+rand())$, where $rand()$ represents a function call to the random number generator which returns a random number distributed between 0 and 1. For non-cubic cells, the random numbers for the y and z directions are scaled by $\frac{Dy}{Dx}$ and $\frac{Dz}{Dx}$, such that the origin of a ray becomes $(i+rand(),\ j+\frac{Dy}{Dx}rand(),\ k+\frac{Dz}{Dx},rand())$. In this manner, the origins are randomly distributed throughout the cell, and not simply throughout a cube.

The second portion of the algorithm in need of modification is the determination of the original TMax values. Recall that the initial TMax values represent the distance from the origin to each of the respective x,y, and z planes. For example, recall that for cubic cells, TMaxY is calculated as follows

$$TMaxY = (j + sign[1] - rayLocation[1]) * invDirVector[1] \tag{3.1}$$

,

where $sign[1]$ is a boolean with a value of 1 if the y component of the direction vector is positive, and zero otherwise. $invDirVector[1]$ represents one divided by the y component of the direction vector. For instance, in figure 3.1, the origin is located at 17.343, 8.617. The direction vector has components of 0.7071, and 0.7071, such that the sum of their squares is equal to 1. Because the sign of the y component of the direction vector is positive, one is added to 17 to represent the location of a y breach, and that value is subtracted from the y value of the origin, then multiplied by the inverse of the y direction vector. Implementing equation (3.1) TMaxY is computed as follows

$$TmaxY = (8 + 1 - 8.617) * \frac{1}{.7071} = 0.5416. \tag{3.2}$$

This value is smaller than that of TmaxX which is calculated as follows

$$TmaxX = (17 + 1 - 17.343) * \frac{1}{.7071} = 0.9291 \tag{3.3}$$

For non-cubic cells, however, when a given component of the direction vector is positive, the origin value is subtracted not from $1 + j$, but from $1 * \frac{Dy}{Dx} + j$. When the component of the direction vector is negative, this multiplication of $\frac{Dy}{Dx}$ becomes unnecessary. This is because the negative face value (8 in figure 3.1) is independent of the skewness ratio. To elegantly handle the condition of multiplying by the ratio $\frac{Dy}{Dx}$ the following formulation is used for the more general case of non-cubic cells.

$$TMaxY = (j + sign[1] * \frac{Dy}{Dx} - rayLocation[1]) * invDirVector[1]$$

To illustrate, see figure 3.2 where a cell with $\frac{Dy}{Dx} = 2$ is superimposed onto the same setup as illistrated in figure 3.1. Here, TMaxX is still equal to 0.9291, but TMaxY is solved as follows

$$TMaxY = (8 + sign[1] * \frac{2}{1} - 8.617) * \frac{1}{.7071} = 1.959.$$

Therefore, the ray will not breach the y face during the first step, but will instead breach the x face and enter into the cell to the right. TDeltaY and TDeltaZ are solved in a similar

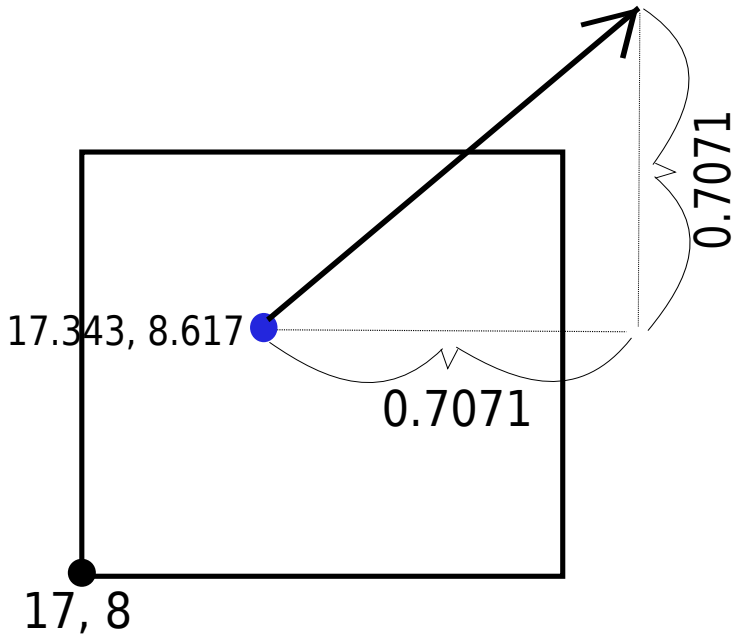


Figure 3.1. First step in a cubic cell

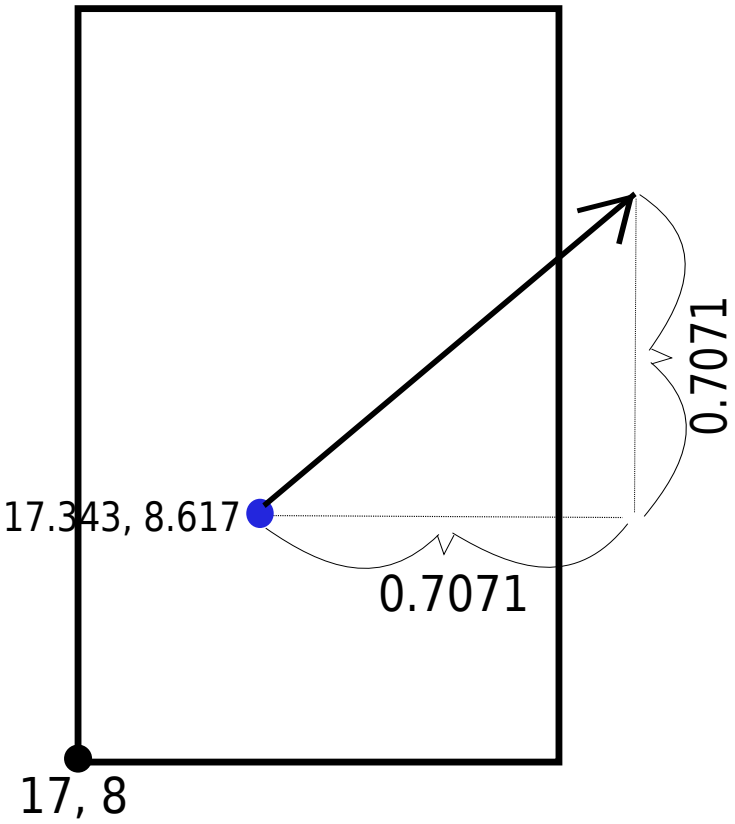


Figure 3.2. First step in a non-cubic cell with $\frac{Dy}{Dx} = 2$

manner, such that for non-cubic cells, the following formula holds.

$$TDeltaY = invDirVector[1] * \frac{Dy}{Dx}. \tag{3.4}$$

3.1.3 Multiple Levels

The adaptive-focus mesh currently suffers from accuracy delinquencies. Nevertheless, it is hoped that this problem is ameliorated soon, and a description of ray marching in such a mesh is given in this section.

To accommodate for memory and message-passing constraints, a mesh with multiple levels is passed to each processor. An example of one such mesh is shown in Fig. (3.3). Although still under testing, the ray tracing algorithm has been modified to accomodate such a mesh. Because the first step of a ray on a new level is handled in a unique fashion, the general case of ray marching in the coarser regions is considered first. Recall from Eqn. (3.4) that the values of TDelta are obtained from the direction vector and the normalized lengths of the cell in the x, y, and z direction. Therefore, for general ray marching in a coarsened domain, these values simply need to be scaled by the respective coarsening ratios in each of the Cartesian directions. T DeltaYc = invDirV ector[1] $*$ Dy $*$ Dyc , Dx Dyf 8 where TDeltaYc represents TDeltaY on the coarser level, Dyc represents the cell width in the y direction on the coarser level, and Dyf represents the cell width in the y direction on the previous (fine) level. Similar expressions may be obtained for the x and z directions. Notice that this scheme can work with any arbitrary number of levels, by simply using the current level as the "coarse" level, and the previous level as the "fine" level.

3.1.3.1 First Step in a new level

The first step in a new level requires special attention because the values of $tMax$ cannot simply be incremented by $tDelta$ of the previous level, nor can it simply be incremented by $tDelta$ of the current level. This is demonstrated by the four different segment lengths through cell L1:01 of Fig. (3.4). The four rays have equivalent direction vectors, but each enters the coarser cell through a different fine cell. With careful attention to the location of the fine cell relative to the coarser cell, the four segment lengths of the rays can be deduced. The cell indices of Fig. (3.4) are written as if the left and bottom edges represent the boundary of the domain. However, even if this is not the case, equivalent indices can be obtained by use of the modulus operator (%). Let *cur* represent a Uintah Vector that contains the non-modultated cell indices of the finer cell after a new level has been reached, but before the indices have been mapped to the new coarser level. Then, to get the cell

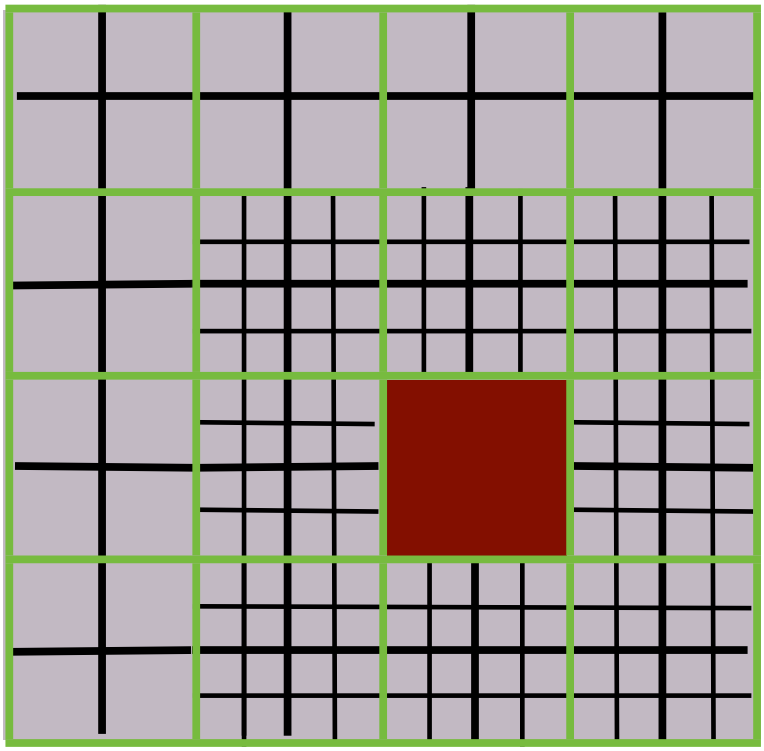


Figure 3.3. Multi-level mesh. The processor owns the fine mesh information indicated by the red square, and is passed coarsened versions of the mesh for regions outside of the local extents.

indices of the fine cell relative to the coarser cell, the following operation is performed: *cur* % CR, where CR is also a Uintah Vector and represents the coarsening ratio between the two levels in each of the three directions. For example, CR in the y direction is given as follows,

$$CR_y = \frac{Dy_c}{Dy_p},$$

where Dy_c represents the current cell spacing in the y direction, and Dy_p represents the cell spacing in the y direction on the previous level.

Notice that in Fig. (3.4), after the y face has been breached, but before *cur* is mapped to the coarser level, the indices of *cur* would become 0,2; 1,2; 2,2; and 3,2. Performing the modulus operation on these values relative to their respective coarsening ratios yields the following values: 0,0; 1,0; 2,0, 3,0. Now, in order to determine the segment length of each of the rays through cell L1:0,1,the value of $tMax[dir]$ is subtracted from $tMax_{prev}$. $tMax_{prev}$ is the length from the ray origin to the level boundary breach, and is equivalent in all 4 rays, and $tMax[dir]$ is equal to the length from the origin of the ray to the location where the ray exits cell L1:0,1, and is different for each ray. The distance of $tMax[dir]$ is a function of the location of the finer cell from which the ray entered the coarser cell. Note that for the ray leaving cell L0:3,1, its $tDeltaX$ value for this first step in the new level is equivalent to $tDeltaX$ of the previous level. Therefore, the $tMaxX$ value correctly describes the next x breach of this ray, and needs no adjustment for this first step in the new level. The remaining cells L0:0,1, L0:1,1, and L0:2,1 however, will require an additional $3tDeltaX$, $2tDeltaX$, and $tDeltaX$, respectively, to be added to the current $tMaxX$ values in order to accurately represent the location at which the next x breach will occur in this new level.

Noting the x component of *cur*, the indices are mapped to the appropriate factor that will be used to become a multiple of $tDeltaX$ in determining the new $tMaxX$ as shown in Table (3.1). This table holds when the component of the direction vector that corresponds to the breached face is positive. When this component of the direction vector is negative, the mapping is trivial, and is shown in Table (3.2). The mapping shown in these two tables is accomplished by the variable *lineup* which is then used to scale $tDelta$ and update $tMax$ as shown in Algorithm (3.5).

The other component (or components, if in three dimensions) follow a similar procedure. If in three dimensions, the component normal to the page in Fig. (3.4) would be handled in an identical fashion to the x component. The y component is also handled identically, given

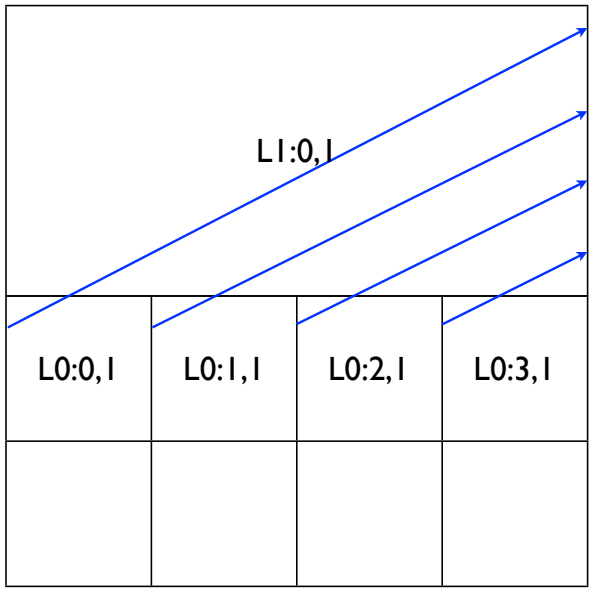


Figure 3.4. The segment length of the first step in a new level is a function of the location of the fine cell of interest relative to the coarser cell.

index	factor
0	3
1	2
2	1
3	0

Table 3.1. For positive components of the direction vector, mapping is shown of the modulus of the index to a factor that will scale the $tDelta$ values.

index	factor
0	0
1	1
2	2
3	3

Table 3.2. For negative components of the direction vector, mapping is shown of the modulus of the index to a factor that will scale the $tDelta$ values.

that the usual incrementation of $tMaxY$ is handled prior to the executions of Algorithm (3.5). The modulus of the component of the index that corresponds to the breached face will always return 0, giving a *lineup* value of 0 for the positive cases and a value of one minus the coarsening ratio for the negative cases. When *lineup* is scaled with $tDelta$ as shown in Algorithm (3.5), then $tMax$ in the first step of the new level will be assigned appropriately such that the subsequent breach in the new level will occur on the wall of the coarser cell, as indicated in Fig. (3.4).

```
for (int ii=0; ii¡3; ii++){
    if (sign[ii]) {
    lineup[ii] = cur[ii] % coarsenRatio[ii] - (coarsenRatio[ii] - 1 );
  }
  else {
    lineup[ii] = cur[ii] % coarsenRatio[ii];
  }
}
tMax += lineup * tDelta[L];
```

Figure 3.5. This algorithm computes the $tMax$ values for the first step in a new level. sign[ii] returns true if that component of the direction vector is positive. At this point in the algorithm, L has not yet been updated, and therefore represents the previous level.

3.1.4 Determining When a Ray Leaves the Domain

At present, RMCRT uses the Arches-specific variable "cellType" to determine when a ray has reached the extent of the flow domain. This approach uses a simple test of the cell type to determine whether the next cell along a ray's path is a flow cell or not. in_domain = (celltype[cur]==-1); //cellType of -1 is flow If so, the ray perpetuates, if not, the ray either reflects or terminates depending on the boundary condition. For component generality, volume fraction could be used as an alternative, so long as at each step, the location of the ray is compared with the extents of the domain. Otherwise, a ray using volume fraction to locate boundaries would attempt to pass through all non-wall boundaries and reference values that do not exist, leading to segmentation violations. A naive approach to comparing the location of the ray with the domain extents, as well as a recommended approach are as follows.

The naive approach to determining when a ray has left the domain extents is to compare each of the three components of the current location with the positive and negative extents of the domain. This would lead to the following six comparisons.

low.x() <= cell.x() &&
low.y() <= cell.y() &&
low.z() <= cell.z() &&
high.x() > cell.x() &&
high.y() > cell.y() &&
high.z() > cell.z().

However, because for each step, the face through which a ray will breach is known, this information can be used to reduce the number of comparisons to test whether or not the ray has left the domain. For instance, if a ray has just breached an x face, only the x component of the location with the x domain extents need be compared. Generalizing this approach, the following is obtained,

low[face] <= cell[face]
&& high[face] > cell[face];

This simple modification saves 5% to 8% of the radiation compute time, and is the recommended approach for use in combination with the volume fraction test.

In addition to implementing the domain-extent test, if volume fraction is to be used to locate boundaries, it is also recommended that an error be thrown if a user attempts to specify anything other than unity absorption coefficient values for inlet, outlet, and pressure boundary conditions. Otherwise, rays would reflect off these gaseous boundaries, leading to non-physical results.

3.1.5 Stopping Criteria

This section outlines the algorithm by which RMCRT determines when to cease the propagation of a ray.

Let τ represent the optical thickness from the origin to the current location of a ray. The optical thickness is defined as follows

$$\tau = \kappa x, \tag{3.5}$$

where x is the distance between the origin and the current point of the ray. Then, by the Beer-Lampert Law,

$$I = I_o e^{-\tau} \tag{3.6}$$

where I_o is the intensity at the current point, and I is the fraction of that intensity that arrives at the origin following attenuation through the medium. A threshold criterion may be set whereby the ray termination point can be determined. For instance, if a threshold value of 0.05 is chosen, the ray is terminated at the location where less than 5% of the initial intensity at a given location would arrive at the origin. Mathematically, this criterion is met when the following relation holds

$$0.05 > e^{-\tau}. \tag{3.7}$$

Throughout most of the model development, the stopping criterion has been left in this form, and has been evaluated for each step of each ray. However, because exponential functions can be expensive to evaluate, an alternative approach was developed and tested. This approach is as follows.

To avoid the evaluation of an exponential function at each step of each ray, a Beta approach was developed wherein the total distance needed to meet the stopping criteria based on the absorption coefficient of the origin cell was calculated. To demonstrate, let T_c represent the threshold criterion. Then Eqn. 3.7 becomes

$$T_c > e^{-\tau}, \tag{3.8}$$

and τ is computed by simply taking the natural log of both sides of Eqn. 3.8 to yield

$$ln(T_c) > -\tau.$$

Dividing both sides by -1 and flipping the direction of inequality gives

$$\tau > -ln(T_c). \tag{3.9}$$

The term on the right of Eqn. 3.9 can then be pre-computed a single time for the whole simulation. This value can be stored as a variable such as $thresh$, and the following condition would be checked at each step, negating an exponential evaluation,

$$\tau > thresh. \tag{3.10}$$

Unfortunately, this approach to avoid an exponential evaluation is valid only when the gradients of the absorption coefficient field are negligible, as it uses an assumption of homogeneity to compute the stopping criterion. In most combustion simulations, this would lead to premature and/or delayed termination of rays. Further, this modification leads to a mere 2% reduction in execution time, which is in the noise of run-to-run variability. For these reasons, the alternative stopping-criterion approach was abandoned.

3.1.6 Intrusion Cells

Intrusion cells are cells that lie within the computational domain, are not flow, inlet, or outlet cells, and do not lie on the domain extents. Examples include solid objects that are placed in the flow of a computational domain and geometric protrusions of the domain boundary. It is often of value to predict radiative fluxes to intrusion cells, such as the effect of placing an object in a fire. Further, the existence of intrusion cells changes the nature of radiative heat transfer throughout the domain. The RMCRT algorithm is capable of computing fluxes to the cells of intrusion objects, as well as handling the interaction between intrusion boundaries and rays from other cells.

3.1.6.1 Fluxes to Intrusion Cells

To compute the fluxes to surfaces in a simulation that may or may not contain intrusion cells, the RMCRT algorithm first tags which flow cells are adjacent to boundaries, and which faces of the cells interface with the flow. Because intrusion can occur anywhere in the domain, a cell iterator that loops through all interior cells has been implemented. Within this cell iterator, a function called has_a_boundary is invoked. This function loops through the six adjacent cells to the cell at hand, and returns "true" if at least one adjacent cell is a boundary cell. In addition, has_a_boundary also returns a vector of enumerated values that corresponds to the faces that are adjacent to a boundary. Currently, has_a_boundary uses the Arches-specific variable"cellType" to determine if an adjacent cell is a flow cell or not. This can be changed for generality, but as is, the algorithm is elegant and efficient as it takes advantage of the integer values of cellType.

int UintahFace[6] = WEST,EAST,SOUTH,NORTH,BOT,TOP; bool hasBoundary = false;

adjacentCell = c; adjacentCell[0] = c[0] - 1; // west

if (celltype[adjacentCell]+1) // cell type of flow is -1, so when cellType+1 isn't false, we know we're at a boundary boundaryFaces.push_back(WEST); hasBoundary = true;

adjacentCell[0] += 2; // east

if (celltype[adjacentCell]+1) boundaryFaces.push_back(EAST); hasBoundary = true;

adjacentCell[0] -= 1; adjacentCell[1] = c[1] - 1; // south

if (celltype[adjacentCell]+1) boundaryFaces.push_back(SOUTH); hasBoundary = true;

adjacentCell[1] += 2; // north

if (celltype[adjacentCell]+1) boundaryFaces.push_back(NORTH); hasBoundary = true;

adjacentCell[1] -= 1; adjacentCell[2] = c[2] - 1; // bottom

if (celltype[adjacentCell]+1) boundaryFaces.push_back(BOT); hasBoundary = true;

adjacentCell[2] += 2; // top

if (celltype[adjacentCell]+1) boundaryFaces.push_back(TOP); hasBoundary = true;

If none of the above return true, then the current cell is not be adjacent to a wall.

At this point, the algorithm loops through the cells that have at least one boundary, then loops through the vector of faces that are boundary faces. Second, for ray tracing to occur in the proper direction, the face value is used to determine the proper orientation of the rays.

3.1.6.2 Ray/Intrusion Boundary Interactions

The inclusion of intrusion cells modifies the behavior of radiative transport primarily in two ways: emission from the intrusion cells to other flow cells; and reflection of radiation off the intrusion boundaries, resulting in a change in ray direction and/or intensity. To account for these phenomena, the RMCRT algorithm tests the cells along a ray path to determine whether the ray has encountered a boundary. This is accomplished by a function call that returns the cell type of the current cell. If the cell type is determined to be a wall, whether of intrusion or domain-boundary, the subroutine that handles wall emission and reflection is invoked.

3.1.6.3 Parallel Implementation

During patch domain decomposition, intrusion objects may be broken into two or more different patches. Currently, the RMCRT algorithm uses a Uintah stencil7 variable type to store the flux values of boundary and intrusion faces. However if a C++ std::map is used, the following modification is used. If a std::map is to be used, the RMCRT algorithm would create a map of maps as a way to identify which which cells on which patches contain intrusion cells. The more general map uses the patch ID as its key, and as the value, another map. This more specific map contains the cell indices and face as the key, and solved fluxes as the values. A Uintah stencil7 variable, conversely, is a field variable and as such requires no patch identification information. Six of the seven variables of the stencil7 (w,e,s,n,b,t) are doubles that store the flux values of each of the faces. The faces that do not interface between boundaries and flow remain initialized to zero. The seventh value (p) is set to zero if none of the 6 faces of a flow cell interface with a boundary and to 1.00000000 if one or more of the faces do. Because the p value of the stencil7 is a double, where only a boolean variable would suffice, some memory is wasted. The benefit of the stencil7 approach is that

the algorithm to assign flux values is simple and efficient. In fact, when placed inside a loop of all boundary and intrusion faces, the assignment is accomplished in a single line as follows boundFlux[origin][face] = sumProjI * 2 *_pi/_nFluxRays, where sumProjI is the sum of the projected intensities of all rays, and _nFluxRays is the number of rays used in the flux ray loop.

3.1.6.4 Efficiency considerations

To improve the efficiency, it was assumed that the intrusion boundaries remain fixed in time. Therefore, once the vector containing the faces of all boundary and intrusion cells has been created, the need to loop through all cells before doing ray tracing is avoided, and a simple vector iterator can be invoked, decreasing the computation time.

3.1.7 Reflections

This section explains how the model represents the physics of specular reflections within a domain with non-black walls. The model is optimized to work with a structured domain composed of rectangular hexahedrons with faces that are always aligned in the Cartesian directions. The properties of this type of mesh greatly reduce the overhead of handling reflections. To illustrate, consider the general form of the equation that determines the post-reflection direction vector $\vec{R}$ given the original, incident direction vector $\vec{\Phi}$ and the surface normal vector $\vec{N}$ [95].

$$\vec{R} = \vec{\Phi} - 2(\vec{N} \cdot \vec{\Phi})\vec{N} \tag{3.11}$$

Equation 3.11 requires the use of three cosine functions, nine multiplications and three subtractions. Clever use of the properties of the Cartesian-oriented mesh reduced this overhead to a single multiplication. The proceeding paragraph explains.

In a Cartesian mesh, the six possible surface normals are as follows.

$$N_{north} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{3.12}$$

$$N_{south} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \tag{3.13}$$

$$N_{east} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{3.14}$$

$$N_{west} = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} \tag{3.15}$$

$$N_{bottom} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.16}$$

$$N_{top} = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \tag{3.17}$$

Let $face$ be a variable that can take on the value of $x, y,\ or\ z$, and represents the cell face that is struck during a reflection. Then the only non-zero component of $\vec{N}$ is N_{face}.

Therefore, the other two components of $\vec{R}$ are equal to the corresponding components of $\vec{\Phi}$. To compute the remaining component, R_{face}, we begin with

$$R_{face} = \Phi_{face} - 2(N \cdot I)N_{face.} \tag{3.18}$$

Recognizing that $N \cdot I = ||O||cos(\theta_i)$, and that the magnitude of Φ is always unity, equation 3.18 may be written as

$$R_{face} = \Phi_{face} - 2cos(\theta_i)N_{face}. \tag{3.19}$$

Because the definition of cosine is the adjacent component divided by the hypotenuse of the triangle formed between $\vec{\Phi}$and $\vec{N}$, we substitute $cos(\theta_i) = \frac{\Phi_{face}}{N_{face}}$. With this substitution, equation 3.19 becomes

$$R_{face} = \Phi_{face} - 2\frac{\Phi_{face}}{N_{face}}N_{face}.$$

Canceling N_{face} yields

$$R_{face} = \Phi_{face} - 2\Phi_{face} = -\Phi_{face}. \tag{3.20}$$

In other words, R_{face} is simply the negative if Φ_{face}. Therefore, to determine the new direction vector following a reflection, one need only change the sign of the component that corresponds to the face that was struck during the reflection. This is illustrated in figure 3.6.

What further adds to the simplicity of reflections in Cartesian grids is that the ray marching scheme is minimally affected by a reflection. Recall from section (3.1) that when the ray location surpasses one of the Tmax values, a step is taken in the corresponding face direction. Conveniently, the sequence in which Tmax values are updated, and thus the sequence in which x, y, and z faces are breached remains unchanged even after reflections (see figure 3.7). This means that the algorithm need not reset the values of Tmax and TDelta following a reflection. The only variable that requires adjustment is the *step* variable. This variable determines whether the ray will step in the positive direction of $face$ or the negative direction of $face$. Because it is a function of the direction vector, when the $face$ component of the direction vector is changed, it too must be changed in a similar manner. Therefore, similar to equation 3.20, $step_{face}$ is assigned to be the negative of its current value.

One caveat of the ray tracing algorithm is that the reflection condition is not triggered until after the ray has stepped outside the domain. Neglecting to account for this would lead to inaccuracies such as reflections occurring within the boundary rather than on the surface. This is remedied by creating a variable that lags behind the current ray location by

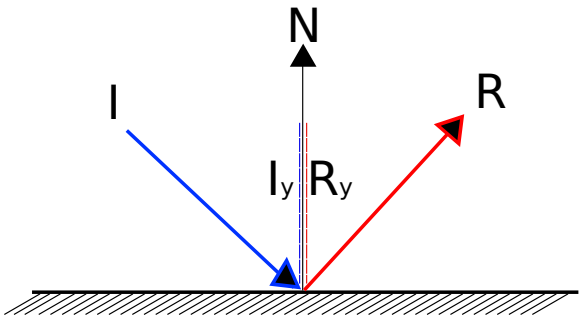


Figure 3.6. Specular reflection about the surface normal, N. Note that $R_y = -I_y$.

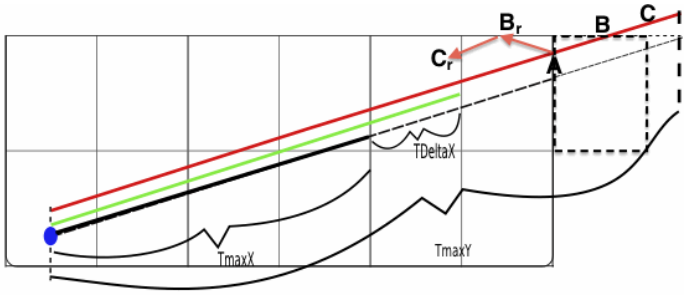


Figure 3.7. Reflection ray marching. This figure demonstrates that the values of Tmax and TDelta need not be adjusted after a reflection. The x and y faces are breached in the same order even after a reflection. For example, after the ray has reached the first non-black boundary, indicated by point A, the following breach occurs at a y face as shown both at point B and point B_r. Subsequently, there is another reflection at point B_r followed by an x breach both at points C and C_r.

one step. Then, upon reaching the reflection condition, the current ray location is assigned to the value of the lagging variable.

The RTE is also minimally affected by reflections. For a moment, assume that in figure 3.7, a reflection does not occur at point A, and that the ray location is currently at point B in figure 3.7. Then the black body intensity from the current cell (in this case the cell with the dotted border) is as follows,

$$I_b = \sigma T_{cur}^4/\pi,$$

where T_{cur} represents the temperature of the current cell. Then following augmentation by absorption and emission along the path back to the origin, the intensity from this cell is

$$I_i = I_b[exp(-\tau_A) - exp(-\tau_B)]. \tag{3.21}$$

Let us now investigate the effect on I_i for the case in figure 3.7 where reflections do occur at the domain boundaries, and the ray is currently at position B_r of figure 3.7. The current cell would then be the cell to the left of the dotted cell, and the intensity that reaches the origin from the segment length of A to B_r is

$$I_i = I_b[exp(-\tau_A) - exp(-\tau_{B_r})]f_s, \tag{3.22}$$

where f_s represents the remaining fraction of spectral intensity following all previous reflections. In this case, $f_s = \rho_A$,where ρ_A is the reflectivity of the boundary at point A. Note the minimal differences between the formula for non-reflected incoming intensity, equation 3.21 and the expression for reflected incoming intensity, equation 3.22.

To demonstrate the procedure of determining incident intensity after a series of reflections, consider the intensity emitted between points C_r and B_r attenuated back to the origin, where $f_s = \rho_{B_r}\rho_A$. This scenario may be viewed in a piece-wise fashion as follows. A small amount of intensity is emitted between points C_r and B_r based only on the optical thickness between the two points, not the running total of the optical thickness back to the origin. This intensity would then be scaled by ρ_{B_r} and then attenuated according to Beer's Law, again using only the optical thickness between to point A and point B_r. This value would then be scaled by ρ_A, and finally attenuated by Beer's law using the optical thickness from the origin to point A. Mathematically,

$$I_i = I_b[(exp(0) - exp(-(\tau_{C_r} - \tau_{B_r})))\rho_{B_r}(exp(0) - exp(-(\tau_{B_r} - \tau_A)))\rho_A exp(-\tau_A). \tag{3.23}$$

However, for any series of consecutive, independent optical thicknesses, *e.g.* τ_1, τ_2, and τ_3, the following relationship holds

$$[exp(-\tau_2) - exp(-\tau_3)]exp(-\tau_1) = exp(-(\tau_1 + \tau_2)) - exp(-(\tau_1 + \tau_3)).$$

Furthermore, by the commutative property of multiplication, ρ_{B_r} and ρ_A of equation 3.23 are factored into the single term, f_s and placed at the end of the equation. This yields

$$I_i = I_b[exp(-\tau_{B_r}) - exp(-\tau_{C_r})]f_s. \tag{3.24}$$

This equation is mathematically equivalent to equation 3.23, but is much more tractable as it does not necessitate the storage of independent optical thicknesses for each ray section between reflections. It requires only the running total of the optical thickness as well as the optical thickness from the prior step.

In RMCRT, equation 3.24 is repeated for each step of each ray to get the total contribution for all rays. In the model, this is accomplished in a ray-marching loop within the ray loop as sumI += sigmaT4OverPi[prevCell] * (expOpticalThick_prev - expOpticalThick) * fs, where sumI is the ongoing sum of all steps of all rays for a given cell at a given time step.

3.1.8 Verification Testing of the Reflection Condition

To assess the accuracy of the model, results were compared against an analytical solution provided in section 13.2 of Modest [87].

Here two infinite parallel black plates are separated a distance L apart. The medium between the plates is gray, with an optical thickness between the plates of τ_L. The solution to the radiative-flux divergence at location τ is given by,

$$\frac{dq}{d\tau}(\tau) = \frac{-2\sigma(T_w^4 - T_m^4)[E_2(\tau) + E_2(\tau_L - \tau)]}{1 + (1/\epsilon - 1)[1 - 2E_3(\tau_L)]},$$

where $\tau = \kappa x$ and $\tau_L = \kappa L$. E_1 and E_2 are exponential integrals, and are given by

$$E_1(x) = \int_0^1 \frac{e^{-x/\mu}}{\mu} d\mu,$$

$$E_2(x) = \int_0^1 e^{-x/\mu} d\mu,$$

where $\mu = cos(\theta)$. In this case, the following values were used in the calculation of the exact solution to the radiative-flux divergence,

$$\tau_L = 1,$$

$$T_w = 1000K,$$

$$T_m = 1500K,$$

$$\epsilon = 1.$$

To create a numerical model of an infinite parallel plates, a cubic domain with black top and bottom surfaces and perfectly reflecting side surfaces was implemented.

The analytical solution at varying values of x that corresponded to the cell centered locations of the numerical solution was computed using a MATLAB script. To compute E_1 and E_2, the script used trapezoidal integration of 10^5 discretization points for $d\mu$. The first nine digits of the obtained values of the exponential integrals remained insensitive to a factor of 10 increase in the discretization, and the first six digits of the exponential integrals matched the six digits tabulated by Modest. The advantage of computing the exponential integrals over using the table is that we are now not limited to the relatively sparse number of values in Modest's table, *i.e.* we computed the exponential integrals at cell centers for any arbitrary resolution without resulting to interpolation between tabulated values. Furthermore, the computed values have a tolerance of 10^{-9} rather than 10^{-6}. The exact solution that results from using the computed exponential integrals is indicated by the blue lines of figures 3.8 and 3.9. The values of the radiative-flux divergence computed using RMCRT are given in green in the same figures.

The L2 error norms were computed using the values at the cell centers of the discretized points along the vertical line between the two plates as well as the values of the exact solution at corresponding locations. The equation used for the L2 norm is given by

$$L2 = \sum_{i=0}^{n_c} \sqrt{(E_i - N_i)^2},$$

where E_i and N_i represent respectively the exact solution and the numerical solution at a given cell center. n_c represents the number of cells between the plates and is given by

$$n_c = \frac{L}{dx},$$

where dx is the width of a cell, and L is the distance between the plates. The L2 norm as a function of ray number, N, converged at the expected order, $O^{1/2}$ for $1 < N < 10^5$. For

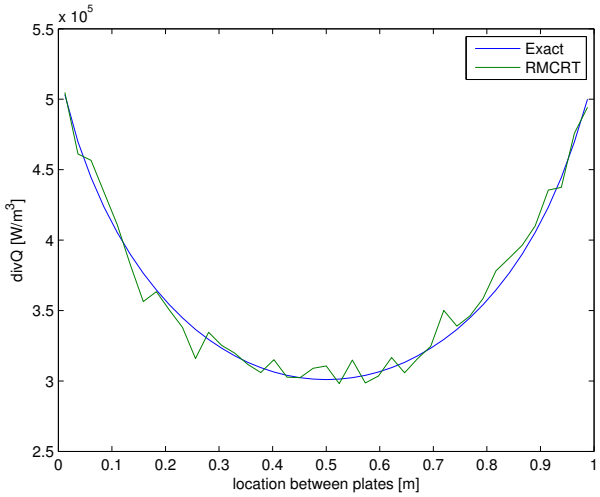


Figure 3.8. Exact solution for the radiative-flux divergence compared to RMCRT with 1000 rays and 41^3 cells for Modest's Benchmark case 13.2.

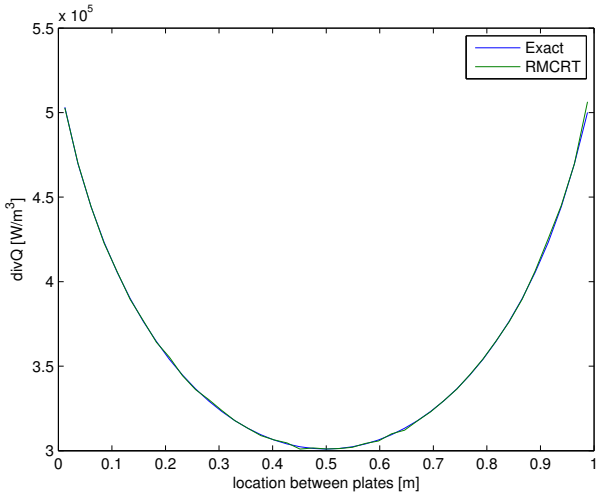


Figure 3.9. Exact solution for the radiative-flux divergence compared to RMCRT with 100,000 rays and 41^3 cells for Modest's Benchmark case 13.2.

values of N greater than 10^5, the convergence rate slowed. This is explained by the finite error from the threshold error and grid resolution error becoming non-negligible for large values of N. The pink and red circles shown in figure 3.10 have larger N and finer grid resolutions, allowing for an analysis of ray error.

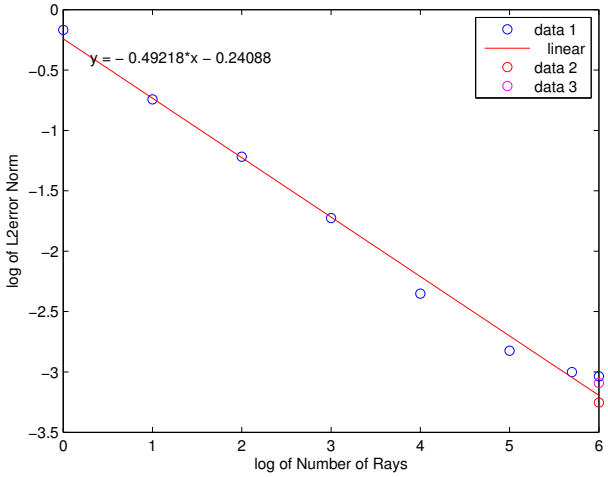


Figure 3.10. Convergence rate of the L2 error norm of RMCRT on benchmark 13.2 as a function of ray number from 1 to one million rays. The blue circles represent the L2 error norms from RMCRT data with a ray threshold of 10^{-3} on a grid of 41^3 cells. The red line is a curve fit of these norms. The pink circle represents the L2 error norm of RMCRT with N=1,000,000 rays and a threshold of 10^{-4}. The red circle represents the L2 error norm of RMCRT with N=1,000,000 and a threshold of 10^{-4}, but on a grid of 101^3 cells.

3.1.9 Scattering

Radiative scattering is the re-direction of radiation within participating media. Three separate phenomena are responsible for this re-direction [87].

1. Reflection off the surface of particles in the media.
2. Refraction of the radiation after passing through a particle.
3. Diffraction of radiation that passes close to particles.

Generally, the physical scale at which combustion simulations take place is significantly larger than the size of the particles in the media. Therefore, in the RMCRT model, the three scattering phenomena are not differentiated and a single scattering coefficient, σ_s is used for a given computational cell.

At present, RMCRT is capable of performing homogeneous scattering. However, when scattering coefficient data is available to the radiation model, a simple adjustment from sigma_s to sigma_s[c] will allow for non-homogeneous scattering.

RMCRT models scattering as follows.

The scattering length, σ_l, is selected as,

$$\sigma_l = -\frac{log(R)}{\sigma_s}, \tag{3.25}$$

where R is a random number that varies between 0 and 1, and σ_s is the scattering coefficient of the current cell at the time the scattering length is calculated, which is either the origin, or the cell at the location where the most recent scattering event has occurred. Concurrently, the current length, l_c is set to zero. As the ray propagates throughout the domain, l_c is updated as follows

$$l_c = l_c + l_s, \tag{3.26}$$

where l_s is the segment length of the current step, and is computed using $disMin$, a variable from the ray marching algorithm, and the cell width in the x direction, Δx, as follows,

$$l_s = \Delta x * disMin. \tag{3.27}$$

l_c is updated as the ray propagates until the following condition is met

$$l_c \geqq \sigma_l. \tag{3.28}$$

Once l_c has met or exceeded the size of σ_l, a scattering event occurs.

At a scattering event, a new direction for the ray is chosen. For isotropic scattering, the direction is chosen in the same manner as the original direction for a non-boundary cell whose rays will subtend a full 4π Sr. This is accomplished as follows.

$$z = 2R - 1 \tag{3.29}$$

$$r = \sqrt{1 - z^2} \tag{3.30}$$

$$\phi = 2\pi R, \tag{3.31}$$

where R is a random number uniformly distributed between 0 and 1, and z, r, and ϕ are the components of the direction vector in polar coordinates. For convenience, these values are converted to Cartesian coordinates, as follows

$$\vec{d} = r\cos\theta,\ r\sin\theta,\ z, \tag{3.32}$$

where $\vec{d}$ is the new direction vector. To avoid division in later computations the inverse of this vector is calculated as follows,

$$\vec{d}_{inv} = \frac{1}{\vec{d}_x},\ \frac{1}{\vec{d}_y},\ \frac{1}{\vec{d}_z}. \tag{3.33}$$

In accordance with the ray marching algorithm, new *step* and *sign* values are assigned. Each of the three components of *step* and *sign* are assigned based on the corresponding components of the new direction vector. If a component of the direction vector is positive, then the corresponding component of *step* and *sign* are assigned the value of 1. Otherwise, the values of -1 and 0, respectively, are assigned. Then, based on the current location of the ray, the inverse direction vector, *sign*, the values of $tMax$ and $tDelta$ are calculated as was demonstrated in section (3.1).

$$tMax_i = cur_i + sign_i \frac{\Delta i}{\Delta x} - location_i, \tag{3.34}$$

$$tDelta_i = \text{abs}(\vec{d}_i)\frac{\Delta i}{\Delta x}, \tag{3.35}$$

where i is the x, y, or z component, and $\frac{\Delta i}{\Delta x}$ is the ratio of the cell width in the ith direction relative to the cell width in the x direction. *step* is then used in the incrementation/ decrementation of *cur*, the variable that represents the indices of the current cell.

Let $face$ represent x, y, or z, and assume that it is updated based on the cell face through which a ray has most recently passed. During some scattering events, there is a

sign change for the $face$ component of *step*, *e.g.* the x component of *step* was -1 prior to a scattering event that occurred on an x cell face, yet +1 after the scattering event. In this circumstance, the ray is scattered back into the cell through which it would have passed had it not scattered. To account for this form of scattering, *cur* is assigned to the previous cell's indices. Note that in other scenarios, *step* may change sign, yet *cur* need not be re-assigned to the previous indices. For instance, if the x component of *step* changes sign at a scattering event that occurs on the y face, then simply the adjustment of *step* and *sign* will account for the correction of the order in which the ray will march from the cells in the domain. In this case, no cell need be referenced more than once as there is only one segment length per cell along the ray's path. In the prior case, however, there will be multiple ray segments within the same cell, requiring *cur* to be referenced more than once for intensity and attenuation calculations. The RMCRT model accounts for this phenomenon.

For the implemented numerical method of scattering, scattering events always occur on cell faces. Then, at a scattering event, one of the three $tMax$ values will be zero since the ray location will lie on a cell face. The corresponding $tDelta$ is added to this $tMax$ value to avoid erroneously stepping immediately in the $face$ direction.

After each scattering event, l_c is reset to zero, and a new scattering length is chosen according to Eqn. (3.25). This procedure continues until the ray is terminated according to the threshold value.

3.1.9.1 Verification

The selected scattering verification case was described by Siegel [96]. Briefly, this case includes a 1m unit cube with cold, infinite parallel plates on the top and bottom, and cold, mirror sides. The cube is filled with absorbing, emitting, scattering media at T = 64.7K. Analytical solutions were given for the surface fluxes at the top and bottom plates at varying optical thickness, and varying scattering albedo, ω.

3.1.9.2 Properties of pulverized coal particles

In the Arches combustion simulations, a common particle in the media is pulverized coal particles. To accurately calculate a scattering coefficient for participating media, the complex index of refraction of the particle is used as an input parameter. Experimental studies in the field suggest that a value of $1.8 - 2i$, where i is the square root of -1, can be used to obtain effective radiative properties of pulverized coal properties in a radiative field where the dominant wavelengths are on the order of 10μ m [97]. The RMCRT model is

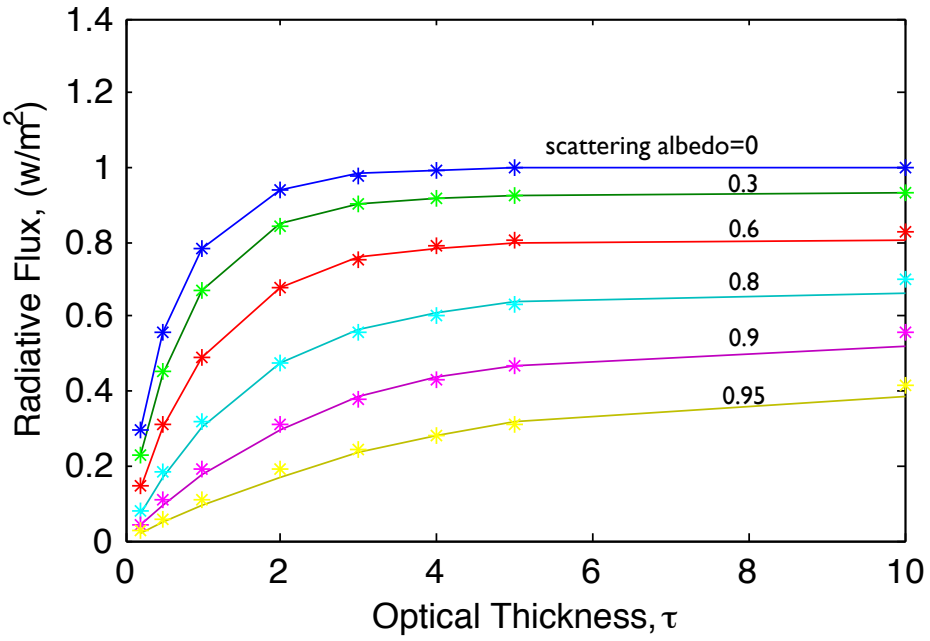


Figure 3.11. Computed and analytical surface fluxes along the bottom plate of the case described by Siegel at varying optical thicknesses, and scattering albedo.

set up to receive this input value, which will be used when the non-homogeneous scattering properties become available.

CHAPTER 4

FLUX CALCULATIONS

4.1 Explanation of Boundary Fluxes

To compute the boundary fluxes, a similar methodology as was implemented for the virtual radiometer model, and the 6 Flux method was used. In RMCRT, rays are generated over a hemisphere, rotated into the the appropriate hemisphere for the boundary at hand, traced as usual, then weighted by the cosine of the polar angle from the surface normal. The details of each of these steps are given below.

4.2 Generating Rays on a Hemisphere.

Because a hemisphere is symmetric about the surface normal, the azimuthal, ϕ, is assigned simply as $2\pi R_1$, where R_1 is a random number between 0 and 1, and thereby achieving the appropriate range of ϕ of 0 to 2π. For the polar angle, θ, because the area of a given ring of the hemisphere is a function of the polar angle, our random number is scaled by the arccosine in order to achieve equidistribution of rays throughout the solid angle,

$$\theta = acos(R_2).$$

4.3 Rotating Rays Onto a Hemisphere

When a ray direction has been selected, initially, the ray will be oriented in the positive z direction, as if it were originating from the top face of a cell. This direction is adjusted to lie within the appropriate hemisphere for the face at hand. For a structured Cartesian mesh, all of the surface normals of the cells are aligned in the coordinate directions. This greatly simplifies the rotation of the rays as it negates the necessity of using a rotation matrix, as was done for virtual radiometers with arbitrary orientations. To re-orient a rays into a new direction such that the rays originate from the face at hand, a simple rearrangement of the vector indices was implemented. This adjustment takes place as follows, where $face$ is an enumeration with the following order: E,W,N,S,T,B. Notice that this enumeration is slightly different than the face enumeration that is passed in from a call to the Uintah type "face" iterator, which has the order: W,E,S,N,B,T. A simple array called RayFace, with values [1,0,3,2,5,4] is used to ameliorate the problem, as the RayFace[Uintah face] will return the proper faces. With the proper face enumeration, the direction is reassigned onto the face at hand, and the sign of one of the components may be reversed as well, if the current face is E,N, or T, as shown in Tab. (4.1). Numerically, this appears as

```
Vector tmpry = directionVector;
directionVector[0] = tmpry[indexOrder[0]] * signOrder[0];
directionVector[1] = tmpry[indexOrder[1]] * signOrder[1];
```

directionVector[2] = tmpry[indexOrder[2]] * signOrder[2];

One may note that for any face, the ray direction will always point toward the inside of the cell, placing the first segment length of a ray through the origin cell. This is because the operation that loops through the cells in the domain to identify which cells have boundary faces, loops through the interior cells. One could imagine a scenario where the algorithm would loop through the "extra" or boundary cells and identify which of those have faces that are adjacent to the flow cells. In this scenario, several modifications to the algorithm would be necessary. First, the positive and negative faces would need to be reversed, as a west boundary face would need to have rays placed on its east face in order to determine the flux at the actual interface between flow cells and boundary cells. Second, the hemisphere would then be on the outside of the cell face as opposed to the inside, as rays should not be traced through boundary material. This would lead to the third adjustment that would need to be made, which affects the intensity solver of the ray tracer. Namely, the first cell being referenced for temperature and absorption coefficient would need to be on a lag, so as to not reference the origin cell for the first segment length, since the ray would not pass through the origin cell at all, but would begin at its face and continue outward. For these reasons, I have chosen to have RMCRT loop through the interior cells to find those with boundary faces as opposed to looping through exterior cells. Therefore, flux rays are generated at the boundary of the domain and are oriented to point inward in the origin cell.

4.4 Shifting the Rays To a Cell Face

Similar to adjusting the ray location from a default hemisphere, ray origins on a plane are generated on a default surface, and are adjusted onto the proper face. By default, points that represent the ray origins are generated on the S face (see Fig. (4.1)), which are then moved onto the appropriate face by reordering the indices and applying a shift value if the face of interest is E,N, or T. This method holds for non-cubic cells as well, given that the unity shift value is scaled by the ratio of Dy to Dx for the y direction, and Dz to Dx for the z direction. Numerically, this appears as

Vector tmpry = location;

location[0] = tmpry[indexOrder[0]] + shift[0];

location[1] = tmpry[indexOrder[1]] + shift[1] * DyDxRatio;

location[2] = tmpry[indexOrder[2]] + shift[2] * DzDxRatio;

face	new direction index order
0	2,1,0
1	2,1,0
2	0,2,1
3	0,2,1
4	0,1,2
5	0,1,2
face	new direction sign
0	-1,1,1
1	1,1,1
2	1,-1,1
3	1,1,1
4	1,1,-1
5	1,1,1
face	new location index order
0	1,0,2
1	1,0,2
2	0,1,2
3	0,1,2
4	0,2,1
5	0,2,1
face	new location shift
0	1,0,0
1	0,0,0
2	0,1,0
3	0,0,0
4	0,0,1
5	0,0,0

Table 4.1. Reordering of indices for adjustment of ray direction and origin location as a function of cell face. Also shown are the values that allow for location shift and direction sign change.

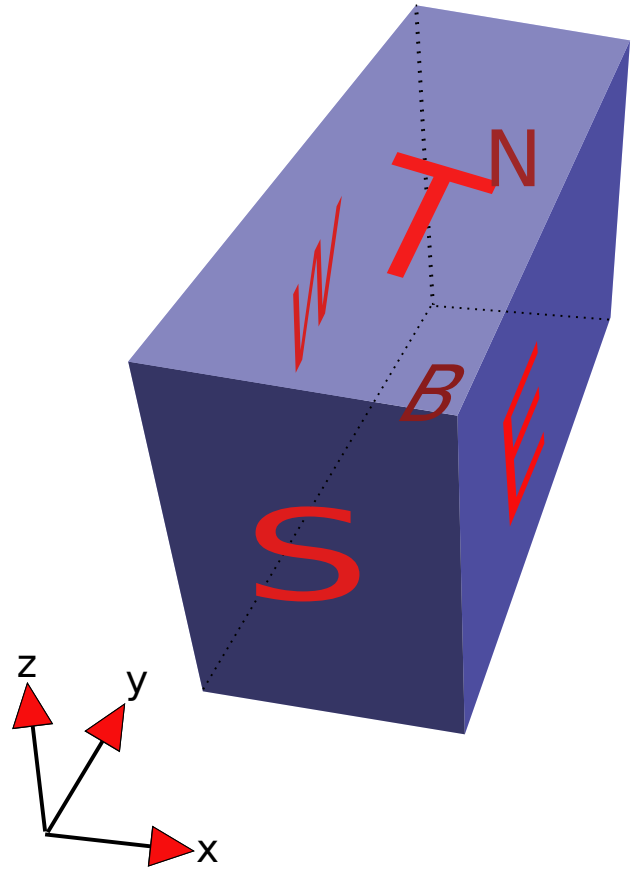


Figure 4.1. A hexahedron with its 6 faces labeled.

4.5 Flux Ray Tracing and Weighting of Rays

Once a location and direction have been specified for a given ray, ray marching, and the update of intensity are handled in the same manner as is done for the flux divergence solver, and the virtual radiometer solver. To avoid code redundancy, the ray marching and intensity solver have been abstracted into an independent method called "updateSumI." Because this method returns a running total of intensity for a given cell, and because the boundary flux solver weights rays according to unique values of $\cos(\theta)$, the intensity of each ray must be known. To allow for this, the current total of intensity is subtracted from the previous total intensity, to yield a unique intensity for the ray, which is then weighted by the cosine of the polar angle, to give the flux contribution from that ray. The flux from all rays are summed and weighted by the solid angle that each subtends, to yield an incident flux for the face at hand, as follows

$$q = \frac{N}{2\pi} \sum_{ir=1}^{N} I_i(r) cos(\theta(r)),$$

where $I_i(r)$ and $\theta(r)$ are the incident intensity and polar angle, respectively, for a given ray, and $\frac{N}{2\pi}$ is the solid angle that each ray subtends. Notice that the solid angle is assumed constant, given the equi-distribution for a large number of rays, and is therefore removed from the summation, improving numerical efficiency.

4.6 Ray Convergence Analysis

Verification testing was performed on the boundary flux calculations. The benchmark case is the Burns and Christon case which has been used in prior verification for the flux divergence results, but also contains flux results for the same cubic, trilinear case with cold black walls. Agreement between computed results and the Burns converged results was obtained. An increase in the number of rays led to a decrease in the L1 error norm at the expected convergence of $\frac{1}{2}$ order. See figures (4.2,4.34.44.5).

To ensure RMCRT did not have a bias in one or more of the Cartesian directions, the L1 error norms from the 2 center lines of each of the 6 faces of the unit cube of the Burns case were analyzed. Four of these twelve center lines are found by varying the x values from zero to one. Similarly, there are 4 lines in the y, and z directions. Figure (4.6) demonstrates the lack of a bias, and therefore invariability of the L1 error norm as a function of direction, which is as expected for this symmetric case.

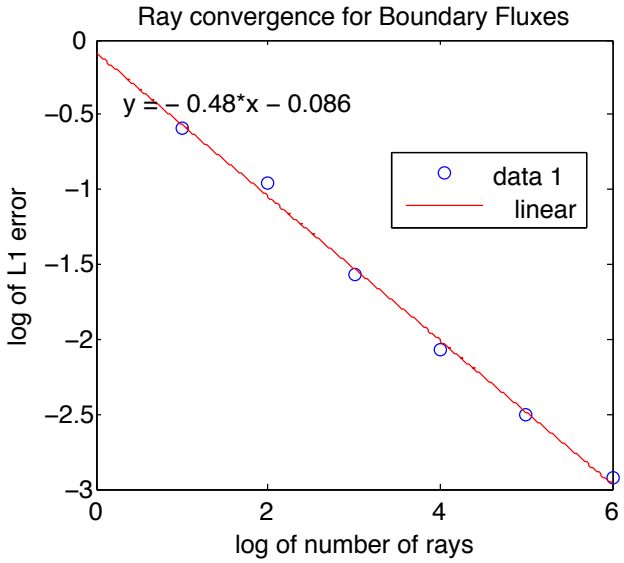


Figure 4.2. Ray convergence for Boundary Fluxes.

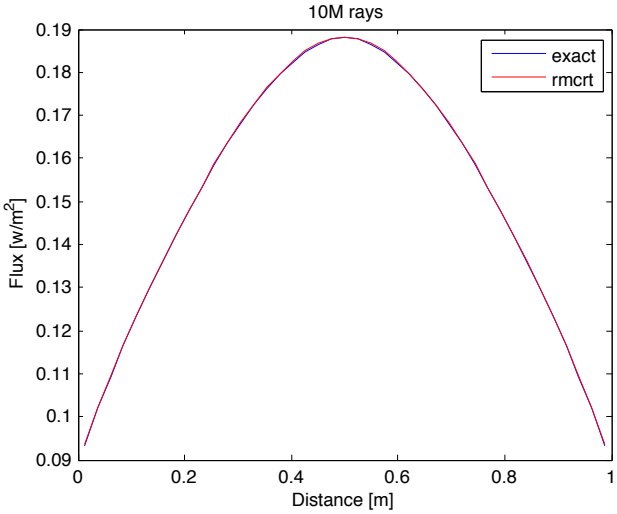


Figure 4.3. RMCRT vs. Burns' converged solution at 10M rays.

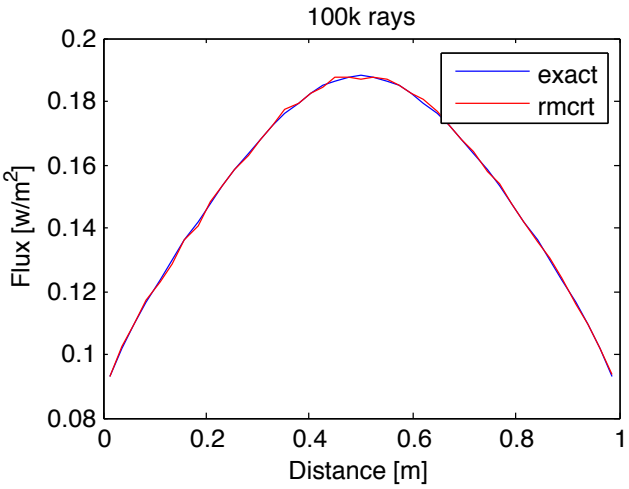


Figure 4.4. RMCRT vs. Burns' converged solution at 100k rays.

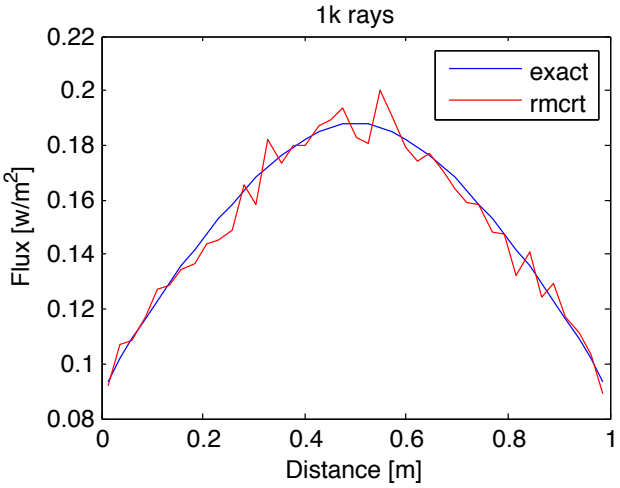


Figure 4.5. RMCRT vs. Burns' converged solution at 1,000 rays.

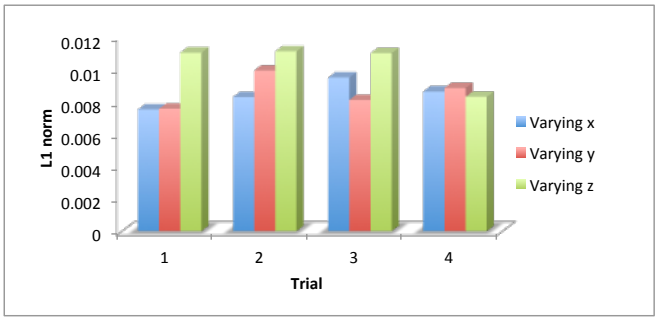


Figure 4.6. Invariability of the L1 error norm of the fluxes as a function of direction for the symmetric Burns and Christon case.

4.7 Storage

The Uintah framework is set up in such a way that the variable types that are used for storage in the data warehouse allocate memory sufficient to store a value for each cell in the domain. In general, not all flow cells in the computational domain are adjacent to a boundary, and therefore need not contain a value for a boundary flux. Therefore the use of Uintah data types ties up memory that is never used. It was hypothesized that a map variable could be used for only the locations where values for the boundary flux exist, thereby avoiding needless memory allocation. To test this, a std::map variable was implemented where values for the boundary flux of cells that contain a boundary, such as walls and intrusion. This map was labeled "cellToValuesMap." The key to the map was a std::vector comprised of the cell index and enumerated face value, and the mapped value was the flux. To handle the complexity of multiple patches in a domain, a larger map was created that houses the patchID as the key, and the corresponding cellToValuesMap as the value. The time required to generate these maps was comparable to the time required to loop through the cells in a patch and assign only a face value. Similarly, the time required to reference the map and solve for the fluxes on these mapped surfaces was comparable to the time required to solve the fluxes using CCVariables. These results are shown in Tab. (4.2).

However, because the use of these maps increased the complexity of the algorithm, particularly for use in multiple levels, and no increase in execution time was attained, the use of maps was abandoned and a uintah stencil7 variable was selected for ease of programming and code maintenance.

	std::map	CCVar
Create	0.22 sec	0.14 sec
Run	212.22 sec	212.29 sec

Table 4.2. Time comparison to create and run std::map vs Uintah's CC Variables.

4.8 Volumetric Integral Vs Surface Integral

The RMCRT algorithm is coupled with the combustion simulation via the enthalpy solver. It computes the radiative flux in W/m^2 and the radiative flux divergence in W/m^3 for boundaries and flow cells, respectively. By the Gauss Divergence Theorem, two methods exist to compute the radiative flux divergence, specifically, a volumetric method, and a surface method,

$$\int_V \nabla \cdot q dV = \oint_S q \cdot \vec{n} dS, \tag{4.1}$$

where V is the volume of the cell, S is the surface of the cell, and $\vec{n}$ is the surface normal for a given face of the cell. Due to a lack of data in the literature, prior to this research, it was unclear which of the two approaches would give the highest accuracy/cost ratio. Therefore, both methods were developed and tested.

The Volumetric Method

The volumetric method involves solving for $\nabla \cdot q$ using Eqn. 4.2 [87].

$$\nabla \cdot q = \kappa(4\pi I_{out} - \int_\Omega I_{in} d\Omega). \tag{4.2}$$

The second term on the right hand side of Eqn. 4.2 is solved readily given the temperature of the origin cell, and the first term is solved by selecting ray origins that are randomly distributed throughout the volume of the cell from which rays are traced as explained in section (3.1).

The Surface Method

The surface approach to solving for $\nabla \cdot q$ via a surface integral over the fluxes on the six faces of a Cartesian hexahedron.

Because RMCRT traces intensity histories throughout the domain, we make the following substitution for q and $q \cdot \vec{n}$ in Eqn. 4.1,

$$q = \int_\Omega I cos\theta d\Omega$$

$$q \cdot \vec{n} = \int_\Omega I cos^2\theta d\Omega, \tag{4.3}$$

where θ is the angle between a given direction of intensity and a surface normal. Eqn. 4.3 is discretized as follows

$$q \cdot \vec{n} = \frac{2\pi}{N} \sum_{r=1}^{N} (I cos^2\theta),$$

where $N$is the number of rays traced per face. Substituting this into Eq. 4.1, and discretizing the surface integral into the sum over the positive and negative sides of each of the six faces of a hexahedron yields

$$\int_V \nabla \cdot q = \frac{2\pi}{N} \sum_{f=1}^{12} \left(\sum_{r=1}^{N} (I cos^2\theta) dS(f) \right), \tag{4.4}$$

where dS is a function of the current face, f, and is equal to either $\Delta x \Delta y$, $\Delta x \Delta z$, or $\Delta z \Delta y$, where Δx, Δy, and Δz represent respectively the length, width and height of the cell. The justification for integrating over 12 faces is that for each of the six faces, a *net* flux is solved. Algorithmically, this is accomplished by appropriately weighting the intensities of each face by either +1 or -1 as shown in table 4.3.

The face normals are ordered as shown in Fig. 4.7.

Notice that Eq. 4.4 gives units of watts. If we assume homogeneous properties within a cell, then to obtain the flux divergence, which has units of $\frac{W}{m^3}$ we need only divide by volume to obtain

$$\int_V \nabla \cdot q = \frac{\sum_{f=1}^{12} \left(\sum_{r=1}^{N} (I cos^2\theta) dS(f) \right)}{\Delta x \Delta y \Delta z}. \tag{4.5}$$

Recognizing that dS for any given face is the volume divided by either Δx, Δy, or Δz, we introduce a new variable, $D_{x,y,z}$ that represents the length of the cell that is normal to the current face. With this substitution, Eqn. 4.5 becomes

$$\int_V \nabla \cdot q = \sum_{f=1}^{12} \frac{\sum_{r=1}^{N} (I cos^2\theta)}{D_{x,y,z}(f)}. \tag{4.6}$$

Equation 4.6 gives reasonable results, but suffers from sensitivity to $D_{x,y,z}$, in that the error of the solution is inversely proportional to the cell length. This is because in the limit that $D_{x,y,z}$ approaches zero, the incident flux of each negative face should respectively equal the outgoing flux of each positive face, and vice versa. However, in practice, this will never be the case, because of the error inherent to a finite population of randomly distributed rays. So, for a fixed number of rays per face, and a decreasing cell size, the ray error remains constant while the analytical solution to the terms $q^-_{inc} - q^+_{out}$ approaches zero. Therefore, the error norms at the limit of infinitesimal cell lengths approaches infinity. This is demonstrated in Fig. 4.8.

face	sign
0	-1
1	1
2	1
3	-1
4	-1
5	1
6	1
7	-1
8	-1
9	1
10	1
11	-1

Table 4.3. Signs of each of the incident and outgoing faces of a cube.

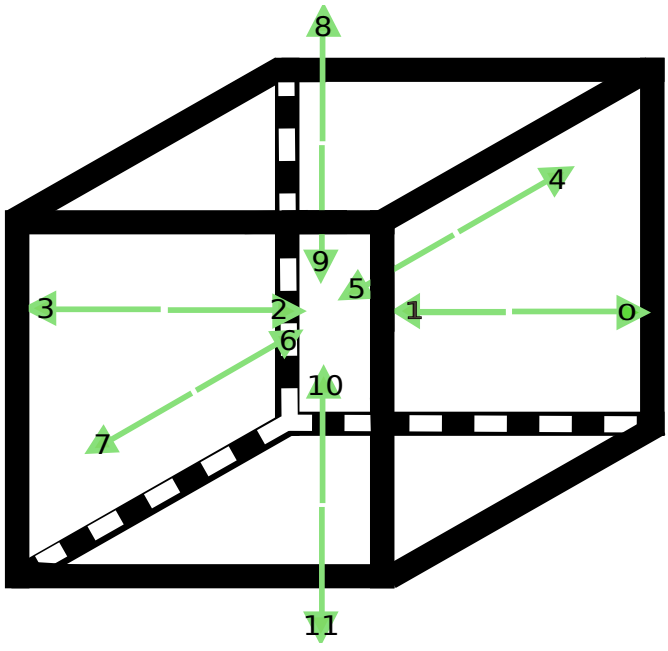


Figure 4.7. Ordering of the 12 face normals of a hexahedron.

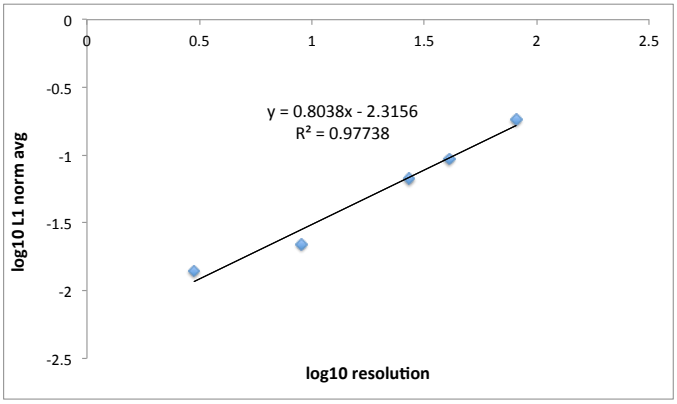


Figure 4.8. Grid convergence of the 12-flux method of computing the flux divergence. Note the positive slope indicating growing error with finer mesh resolutions.

The dependence on the cell size of the 12-flux method was avoided by implementing a clever method suggested by Dr. Philip Smith. This is accomplished by recognizing that the 6 outgoing fluxes can be computed not only by ray tracing, but by the emissive power of the cell itself. To demonstrate, consider the definition of a radiative-flux divergence as the sum of the net fluxes of the faces of a cell,

$$\nabla \cdot q = \frac{q_E + q_W}{\Delta x} + \frac{q_N + q_S}{\Delta y} + \frac{q_T + q_B}{\Delta z},$$

where a positive value for a net flux represents a positive flux out of the cell. Each of the net fluxes is then be broken into its outgoing and incident fluxes as follows.

$$\nabla \cdot q = \frac{(q^E_{out} - q^E_{inc}) + (q^W_{out} - q^W_{inc})}{\Delta x} + \frac{(q^N_{out} - q^N_{inc}) + (q^S_{out} - q^S_{inc})}{\Delta y} + \frac{(q^T_{out} - q^T_{inc}) + (q^B_{out} - q^B_{inc})}{\Delta z}. \tag{4.7}$$

In this form, we immediately see that we can separate out the outgoing fluxes and represent them by the emissive power of the cell, *i.e.*

$$q^E_{out} + q^W_{out} + q^N_{out} + q^S_{out} + q^T_{out} + q^B_{out} = \frac{4\pi\sigma T^4}{\pi} \tag{4.8}$$

Before we can make this substitution, we transform Eqn. 4.7 into a form that doesn't have unique denominators. This is accomplished by assuming that the optical thickness of the media is not sufficiently high to absorb all incident radiation in a single cell width. Indeed, we must account only for the portion of the radiation that is absorbed within the cell. With this consideration, Eqn. 4.7 becomes

$$\nabla \cdot q = \kappa\Delta x \frac{(q^E_{out} - q^E_{inc}) + (q^W_{out} - q^W_{inc})}{\Delta x} + \kappa\Delta y \frac{(q^N_{out} - q^N_{inc}) + (q^S_{out} - q^S_{inc})}{\Delta y} + \kappa\Delta z \frac{(q^T_{out} - q^T_{inc}) + (q^B_{out} - q^B_{inc})}{\Delta z}, \tag{4.9}$$

which simplifies to

$$\nabla \cdot q = \kappa(q^E_{out} - q^E_{inc} + q^W_{out} - q^W_{inc} + q^N_{out} - q^N_{inc} + q^S_{out} - q^S_{inc} + q^T_{out} - q^T_{inc} + q^B_{out} - q^B_{inc}).$$

With the denominator now eliminated, we can now make the substitution proposed in Eqn. 4.8 and obtain

$$\nabla \cdot q = \kappa(-q^E_{inc} - q^W_{inc} - q^N_{inc} - q^S_{inc} - q^T_{inc} - q^B_{inc}) + \frac{\kappa 4\pi\sigma T^4}{\pi}.$$

In discrete form, this becomes similar to Eqn. 4.6, but with a loop over only 6 faces and an extra term which represents the other 6. Specifically,

$$\int_V \nabla \cdot q = \frac{2\pi}{N}\kappa \sum_{f=1}^{6} \left(\sum_{r=1}^{N} I cos^2\theta \right) + 4\kappa\sigma T^4, \tag{4.10}$$

where the summation over 6 faces includes faces 0, 3, 4, 7, 8, and 11 of table 4.3. Notice that we are no longer taking the difference between two fluxes that are equivalent in the limit of infinitesimal cell sizes. We therefore avoid the increase in error that occurred in the 12-flux method at finer resolutions, with the added benefit that we perform ray tracing on only half the number of faces. This 6-flux method produces results comparable to those of the volumetric method, and converges with grid resolution at an approximately first order rate (see Fig. 4.9). The down-side of this method is that to obtain accuracy comparable to that of the volumetric method, it requires 6X the number of rays per cell, which leads to an increase in the computation time by the same factor. The benefit of this method is that in the process of computing the radiative-flux-divergence, the flux for each face is computed, essentially for free. This value could be labeled and stored for later use such as determining the radiative fluxes through any arbitrary surface of the domain. Nevertheless, the objective of most simulations rarely necessitates the computation of every flux to every cell face in the domain. For this reason, a factor of 6 increase in the computation time could not be justified, and the volumetric method was selected for the RMCRT divQ calculations. Naturally, RMCRT flux calculations are performed in a surface manner.

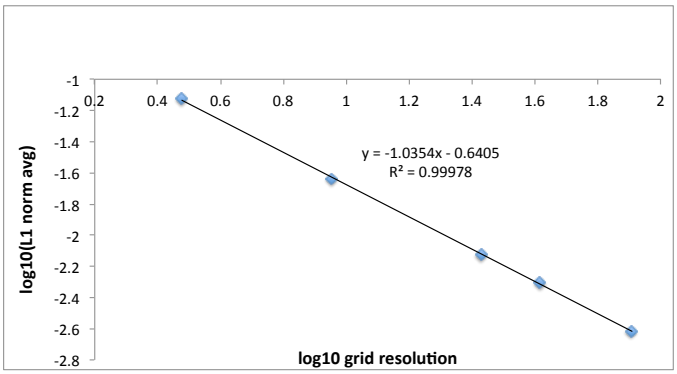


Figure 4.9. Grid convergence analysis of the 6-Flux method of benchmark1. Grids of size 3^3, 9^3, 27^3,41^3, and 81^3 were analyzed. The L1 error norm decreases with mesh refinement at an approximately first order rate.

CHAPTER 5

COUPLING THROUGH THE ENTHALPY EQUATION

RMCRT is fully coupled with the thermal energy balance via the radiative source term. At the onset of a simulation, initial conditions including temperature and absorption coefficient fields are referenced by RMCRT. The radiative source terms produced by RMCRT are subsequently referenced by the enthalpy equation, which, in the Arches component for example, reads,

$$\int_V \frac{\partial \rho h}{\partial t} dV + \oint_S \rho \mathbf{u} h \cdot d\mathbf{S} = \oint_S k \nabla h \cdot d\mathbf{S} - \oint_S q \cdot d\mathbf{S} \,, \tag{5.1}$$

for surfaces, and

$$\int_V \frac{\partial \rho h}{\partial t} dV + \oint_S \rho \mathbf{u} h \cdot d\mathbf{S} = \oint_S k \nabla h \cdot d\mathbf{S} - \int_V \nabla \cdot q, \tag{5.2}$$

for flow cells, where h is the sum of the chemical plus sensible enthalpy, q is the radiative flux, k is a diffusion coefficient used with a Fourier's law form of the conduction term, and the pressure term is neglected [98]. The temperature of each cell is then updated based on the enthalpy change in the given timestep and the heat capacity of the cell. During the next radiation solve, RMCRT references the updated temperatures and temperature-dependent absorption coefficients.

CHAPTER 6

RADIATION PROPERTIES

6.1 Spectral Properties

A method to generate spectral data for gas and particle absorption coefficients and particle scattering coefficients was developed by Lyubima Simeonova during her Master's thesis at the University of Utah [99]. This full spectrum k-distribution (FSK) property model was implemented into the ARCHES component. Radiation is assumed to be controlled by motions on the resolved scale, and no sub-grid radiation model is taken into account. A similar assumption was made by Goncalves dos Santos, *et al.* [9]. Radiation properties oscillate sharply as a function of wavelength. The FSK model reorders the property values into a smoothly-varying g-space, where the cumulative k-distribution, g, is a non-dimensional, Planck-function-weighted, reordered wavenumber. The general derivation of the FSK method is given by Modest [87], and the specific implementation in the ARCHES component is explained in [99].

To instantiate Lyubima's particle class, a complex index of refraction is required by the constructor. The value of 1.85 -0.25i, which is an average complex index of refraction for anthracite, bituminous and subbituminous coal types was used adams1993computational. Lyubima's particle model returns scattering and absorption efficiencies multiplied by πr^2 as a function of temperature and particle size. So it returns the absorption and scattering coefficients divided by the particle number density, and thus has units of $m^2/particle$. Let us distinguish these returned values as "almost-coefficients."

Given a temperature and the average particle size within a cell, Lyubima's particle model is queried to obtain the almost-absorption coefficients. This value is then be multiplied by the particle number density of the cell, to obtain the particle absorption coefficient.

Notice that the only consideration regarding the composition of the coal particle takes place via the complex index of refraction. Currently, objects of Lyubima's class are being instantiated with a single value for the refractive index of unburned coal. Ideally, a value that represents an index of refraction for a mixture of coal and ash should be used. Brad Adams' work shows that for a particle comprised of ash and coal, the absorption efficiencies can be interpolated linearly, *e.g.* efficiency_interp = efficiency_ash + mass_frac_coal * (efficiency_coal - efficiency_ash) [100].

To avoid instantiating Lyubima's class at each radiation solve, yet resolve the varying complex index of refraction, a proposed approach is to generate two tables for all the almost-coefficients: one generated with the complex index of refraction of pure coal, and one with the complex index of refraction of pure ash. For ash, the gray average given by Modest of 1.5-0.02i [87], or the gray value calculated below using Goodwin's data, 1.505 -

$0.1185i$ [101] can be used. This will allow interpolation between the two tables based on composition.

There exists another caveat in the formulation of particle absorption coefficients. The complex indices of refraction for coal and ash are functions of wavelength. Lyubima's code takes as an input a value that is accurate only at a specific wavelength, then generates properties over a wide spectral range. In the range of radiative frequencies present in coal combustion, the real part of the index of refraction is relatively constant, but the complex part varies over 5 orders of magnitude [101]. This is bad news for the assumption that a constant value of the index of refraction can be used to generate a table of efficiencies that span a wide range of frequencies. It is proposed that Lyubima's model be updated to accept a vector of complex indices of refraction to be used in the properties calculations.

6.1.0.3 Digitization of Experimental Property Data

Using plot digitization software, the data from Goodwin's flyash were estimated as shown in Fig. (6.1) and Fig. (6.2). This data was then imported into Matlab where trapezoidal integration was performed over the specified wavelengths. With this integration, a grey-value approximation for the refractive index was calculated as follows

$$\frac{\int_{\lambda_{min}}^{\lambda_{max}} n \partial\lambda}{\lambda_{max} - \lambda_{min}}, \tag{6.1}$$

$$\frac{\int_{\lambda_{min}}^{\lambda_{max}} k \partial\lambda}{\lambda_{max} - \lambda_{min}}, \tag{6.2}$$

where n is the real component of the complex index of refraction, and k is the imaginary component

If an update to Lyubima's code is made to allow for a vector of refractive indices, the digitized forms of the spectrally resolved refractive index data could be used as input into the FSK property calculator to achieve higher-accuracy spectral radiative properties.

There was some confusion pertaining to the units of the absorption coefficients generated by Lyubima's property models. Parts of the documentation give units of $\mathrm{m}-1$ and others give units of $\mathrm{cm}-1$. For this reason, the verification and validation studies in this research were completed using the Hottel properties, though RMCRT is capable of running with either property method.

After the completion of this research, the confusion regarding units was ameliorated, and it appears that Lyubima's models are ready for production coupling with RMCRT.

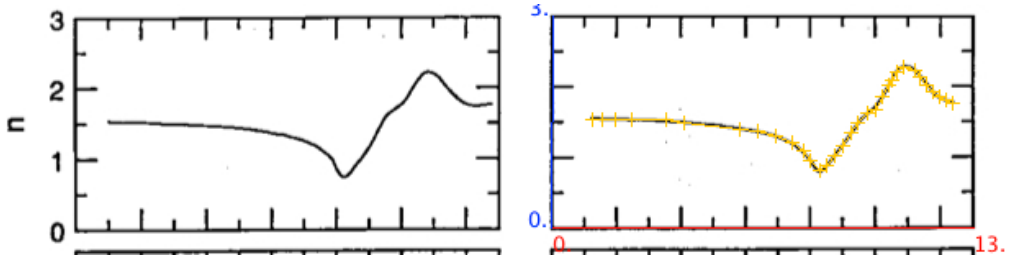


Figure 6.1. Digitized data of the real component of the refractive index of ash with 5.47 percent hematite.

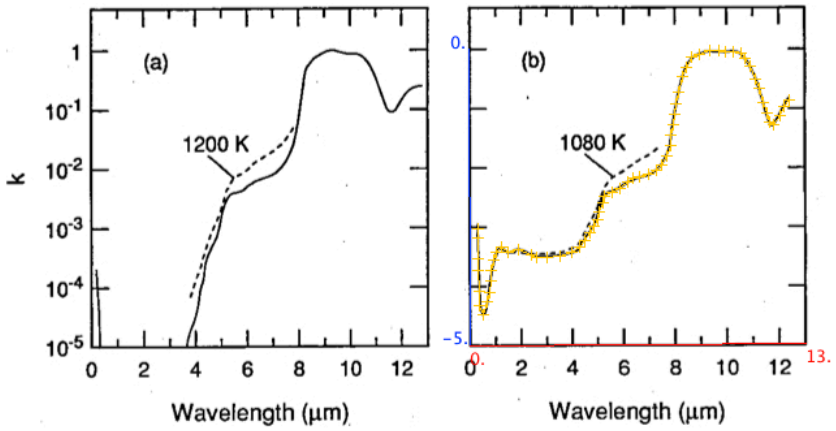


Figure 6.2. Digitized data of the imaginary component of the refractive index of ash with 5.47 percent hematite.

6.2 Hottel Properties

The Hottel-Sarofim property model was linked to the RMCRT code via the RadPropertyCalculator.cc file. The interface requires the passing of CO_2, H_2O, and soot volume fractions as a vector, and subsequently passes these values to a Fortran subroutine that computes the absorption coefficients. To ensure that the interface was written properly, verification testing was performed. The methane_fire__NEW.ups input file was selected for use in comparing absorption coefficient values produced from the RMCRT interface and those produced by the legacy interface to the Hottel properties via the Discrete Ordinates Method. At timestep 1, as expected, the absorption coefficients (abskg) from both cases was identical. At timestep 10, the two differed on average (L1 norm) by 2.0e-6, and at timestep 100 the L1 norm was 0.0012 (See figures 6.3 6.4 6.5 6.6).

As expected, each method produced slightly different divQ values, leading to different domain temperatures. The difference in the absorption coefficient values at subsequent timesteps are expected as a result of the table lookup at these differing temperatures.

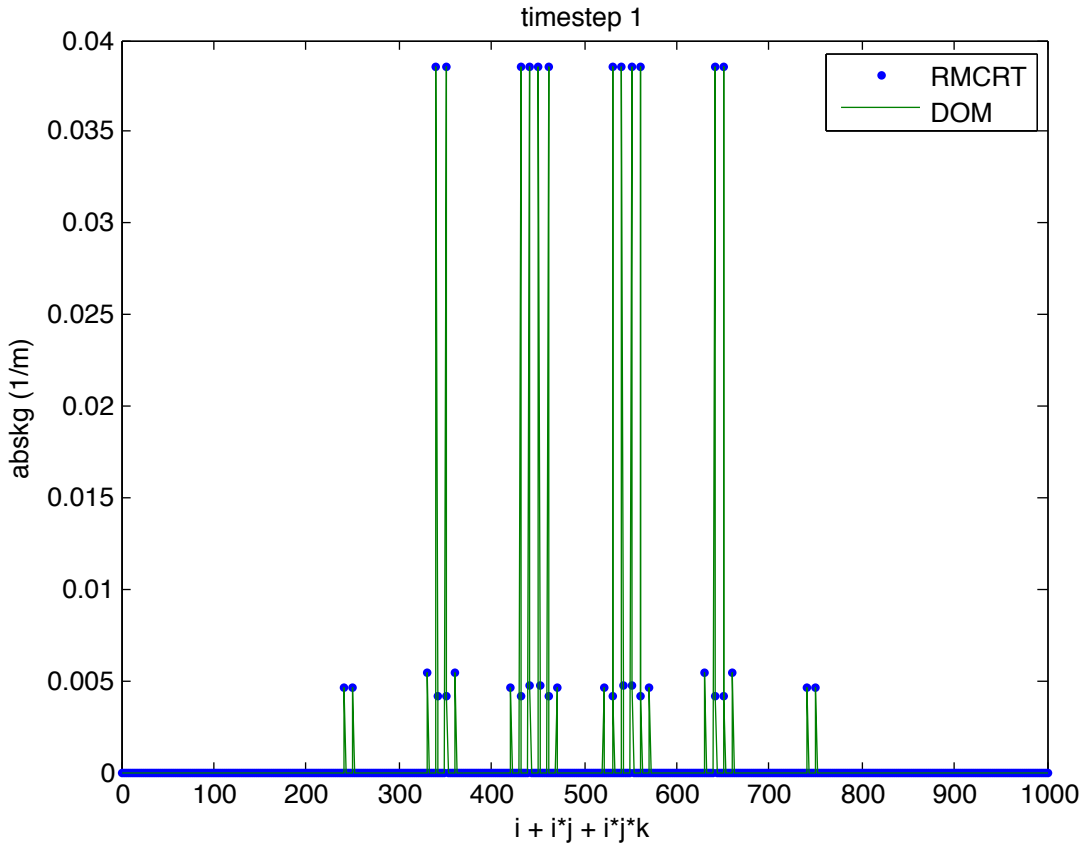


Figure 6.3. Hottel-Sarofim absorption coefficients produced through the RMCRT interface(blue) and DOM interface (green) at timestep 1.

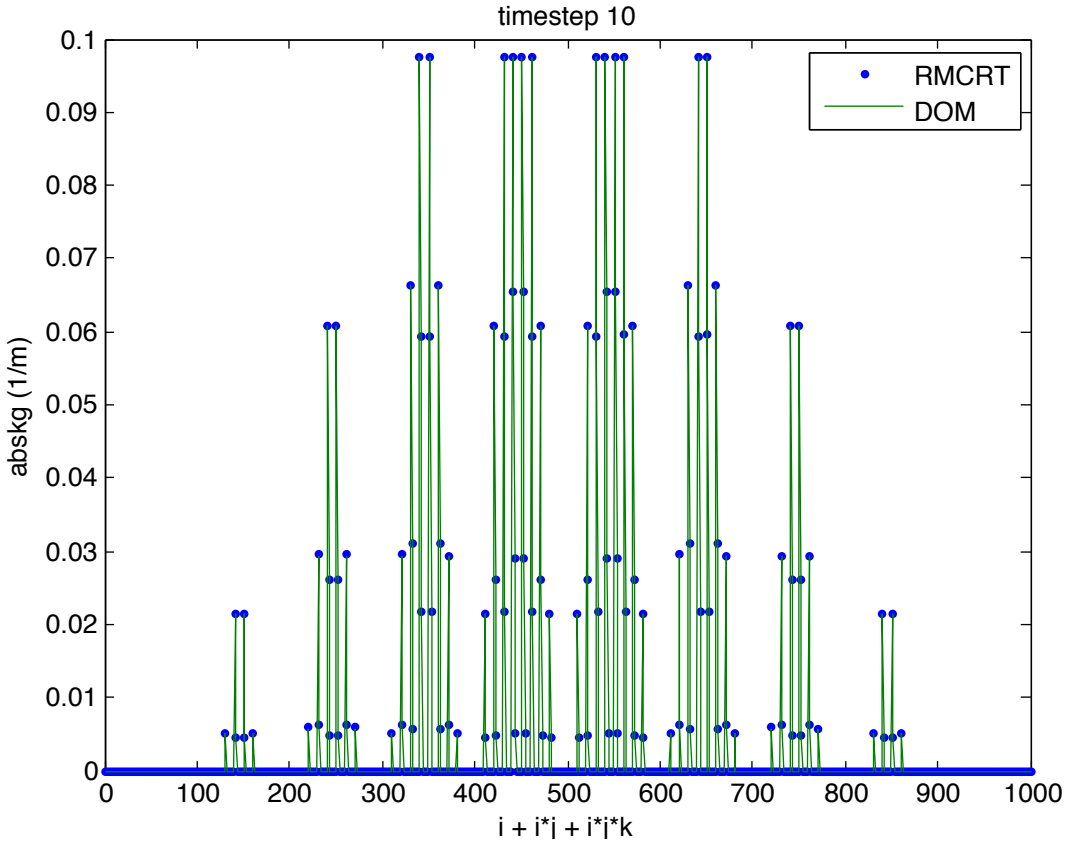


Figure 6.4. Hottel-Sarofim absorption coefficients produced through the RMCRT interface(blue) and DOM interface (green) at timestep 10.

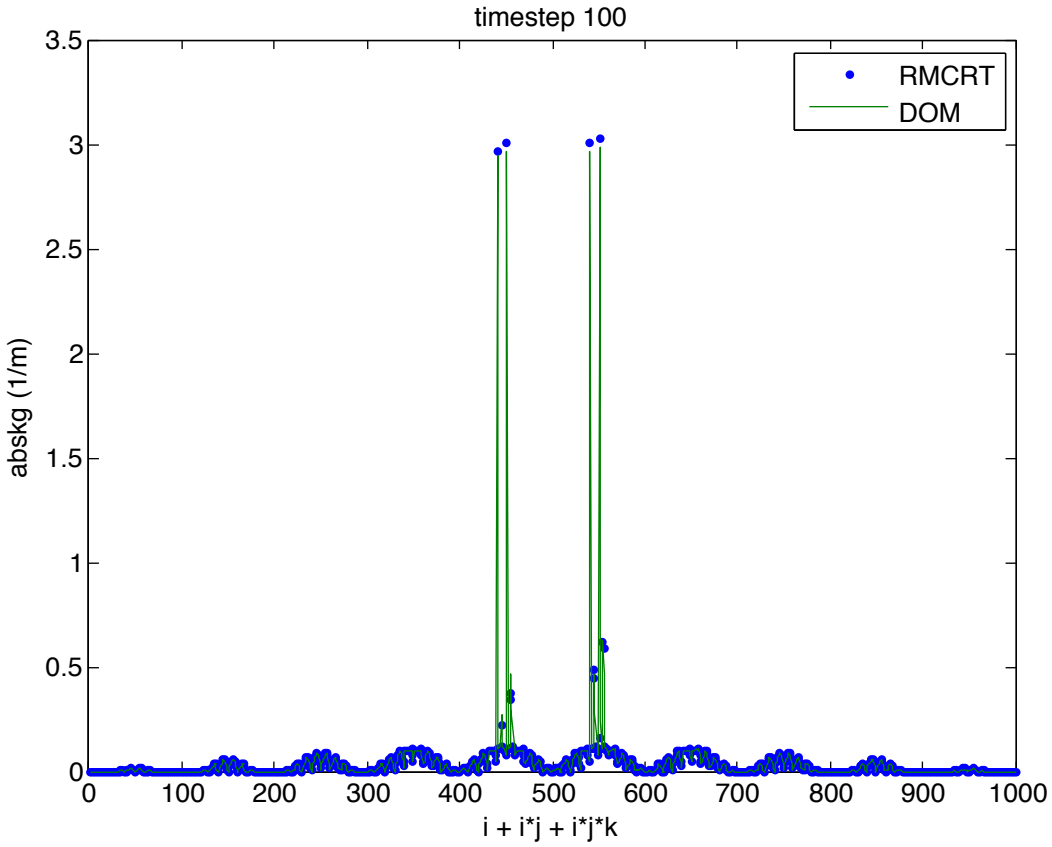


Figure 6.5. Hottel-Sarofim absorption coefficients produced through the RMCRT interface and DOM interface at timestep 100.

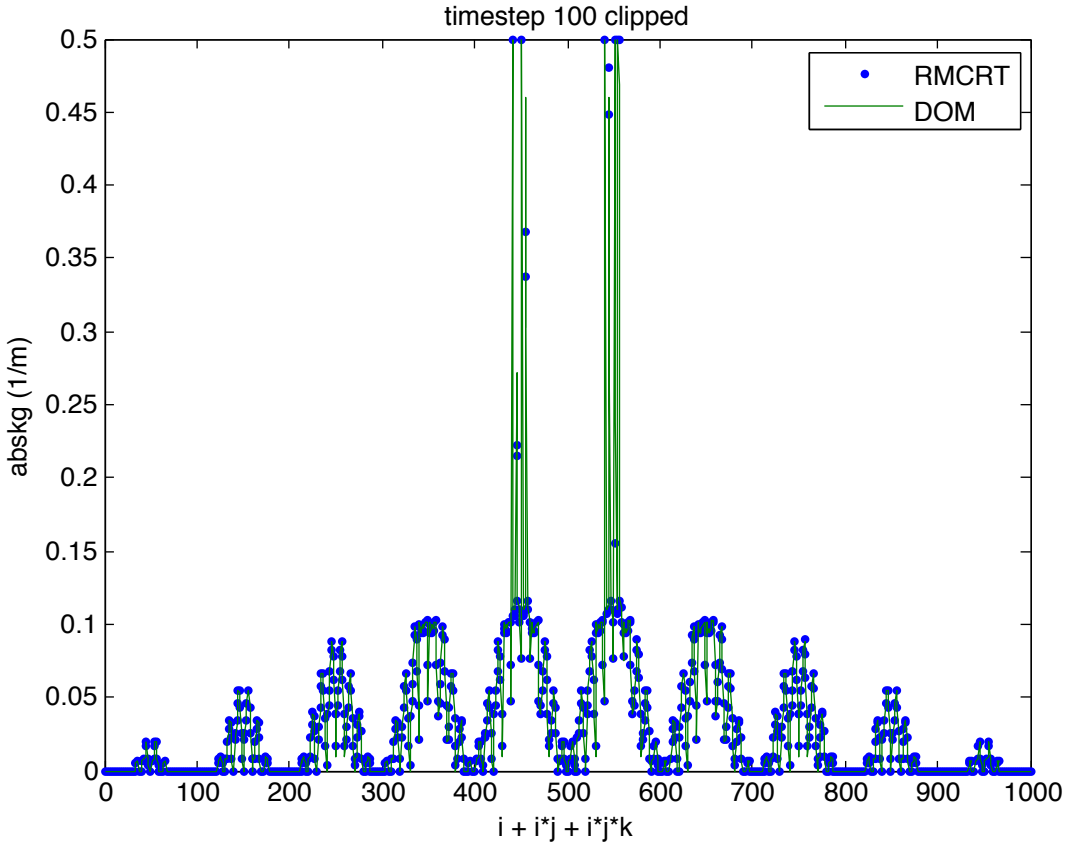


Figure 6.6. Hottel-Sarofim absorption coefficients produced through the RMCRT interface and DOM interface at timestep 100, with values clipped at 0.5 to allowing viewing of the smaller values.

CHAPTER 7

PARALLEL CONSIDERATIONS

7.1 Turbulent Radiation Interactions

Coupling parallel simulations of combustion and radiation poses several numerical challenges. The fluid mechanics of combustion are local phenomena, making them amenable to domain decomposition. Conversely, radiation is a long-distance, and potentially all-to-all phenomenon, creating difficulties for domain decomposition. Further, accurate calculation of radiative transfer requires spatially resolved information regarding the temperature and species composition fields. Traditional modeling of turbulent systems has included Reynolds-averaged Navier Stokes (RANS) simulations. The RANS model provides, at a relatively low computational cost, spatially averaged values of the gas temperature and species fields. However, for highly non-linear physics such as radiation, spatial averaging in this manner may introduce large errors [9]. Alternatively, direct numerical simulation (DNS) fully resolves the power spectrum of eddies, giving access to the full spatial distribution of the pertinent fields. Wu et al. [10] and Deshmukh [11] have coupled a monte-carlo ray tracing method to solve the radiative transfer equation in a turbulent reacting flow modeled by DNS. Unfortunately, due to its high computational demand, DNS remains impractical for use in large-scale combustion simulations. In contrast, large eddy simulations (LES) resolve the largest fluid motions, down to the Nyquist limit for a given turbulent field and mesh resolution. Beyond this limit, the less-important smaller eddies are approximated via simpler models. Because combustion turbulence is generally dominated by large eddies [12], LES gives a better description of the fluid mechanics than RANS, and does so without the computational cost of DNS.

The various levels of accuracy in which thermal radiation has been modeled in combustion simulations has been reviewed by Snegirev [13] and Sacadura [14]. The radiation models cited include the optically-thin approximation [15], the discrete ordinates method [16, 17] the discrete transfer method [18], and the finite volume method [19]. The optically thin model neglects the participation of media (absorption, emission, and scattering), and has been shown to introduce error even in small flames [20]. The remaining methods model radiative emission as energy emanating along a set of pre-defined directions. Such angular discretization suffers from the ray effect [21]. Conversely, monte-carlo techniques that select randomly-distributed rays at each time step have low sensitivity to angular discretization and are applicable regardless of media optical thickness [13]. In his earlier work, Snegirev presented a RANS model of buoyant turbulant diffusion flames coupled with statistical modeling of thermal radiation transfer. Although Snegirev's earlier model used a robust formulation of thermal radiation via the monte-carlo method, his turbulence model suffered

from the lack of resolution of the sharply varying fluctuations of temperature and species concentrations that are lost in RANS approximations. More recently, Snegirev coupled monte-carlo radiation with large eddy simulations [22, 23]. These simulations operated on modest meshes of approximately 498,000 control volumes. Other examples of coupled LES monte-carlo radiation models are rare, but include the work of Zhang et al., in which a larger mesh of 4.7 million cells were used [24]. In this emerging field remain several unresolved issues. One such issue is how to deal with increasing mesh sizes that are run on increasingly parallelized super computers.

Modern super high-performance computing systems are comprised of hundreds of thousands of computing cores, and are used to run simulations with meshes comprised of billions of computational cells [25, 26, 27]. Strong scaling in massively-parallel computing is difficult to obtain due to load imbalancing and inter-processor communication demands. The strong scalability limit of a code is reached when an increase in the number of parallel processors used on a fixed problem size does not result in a decrease in computational wall time [28]. Numerous examples of parallelized monte-carlo radiation models were investigated, most of which cease to scale beyond 100 processors [9, 24, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43]. An example of a coupled combustion and monte-carlo radiation model that has a scalability limit above 200 processors was not found in the literature.

In this work, a new numerical technique has been developed to perform large eddy simulations of large-scale combustion flows coupled with a three-dimensional reciprocal monte-carlo ray tracing radiation model. This model has been optimized for use on high-performance computing systems and achieves nearly-ideal strong scaling to over 16,000 processors.

As mentioned above, fluid mechanics and most other phenomena in combustion physics are localized phenomena and amenable on domain-decomposed meshes. In this work, to represent the long-range effects of radiation, the computational domain is recomposed at the time of each radiation solve. This is accomplished over a message passage interface, through which each processor shares the temperature and radiative-properties fields (absorption coefficients, scattering coefficients, temperatures, and cell types) with all other processors. This reconstructed domain combined with the mutually exclusive nature of reciprocal monte-carlo rays is amenable to massive parallelism. Radiative properties are calculated via the Hottel and Sarofim method. For efficiency, these calculations are pre-computed and tabulated in narrow increments of temperature and species mixture fraction values.

7.2 Parallelism and Load Balancing

Ideal parallel speedup occurs when the time spent passing information between processors is negligible compared to the work done by each processor, when all portions of the algorithm are parallelizable based on Amdahl's law (see appendix) and if no computers sit idle while others complete their tasks. The first constraint is met by efficient code writing that ensures that all or most of the information a given processor needs to complete its computations is available to that processor.

The middle constraint is a function of the model. The RMCRT model has a minimal non-parallelizable portion comprised primarily of initialization routines. The bulk of the algorithm is ray tracing, which as explained in Section 7.1, is parallelizable due to the mutual exclusivity of the rays.

The final constraint is met via proper load balancing that distributes the work load equally between processors. Load balancing may be accomplished in two general ways– dynamic load balancing, and static load balancing. Static load balancing schemes distribute the load only once, at the onset of computation. However, because the computation times of different regions of a domain are problem and time-dependent and rarely uniform, static load balancing often creates idle time amid processors. Dynamic load balancing begins with an initial load distribution which can then be modified if and when computers complete their original tasks. Heirich and Arvo have noted that when total computational time is of importance, static load balancing is insufficient for parallel ray tracing on massive high-performance computing systems. [76].

The Uintah Framework incorporates dynamic load balancing via a scheduler and data warehouse [64]. The scheduler container that stores simulation variables is the data warehouse. The data warehouse stores simulation variables and serves as a dictionary wherein variable names and patch IDs are mapped to memory addresses. The load balancer generates a patch distributions, and the scheduler creates sets of tasks [102]. In this manner, processors are assigned approximately equal loads, reducing idle processor time.

Approaches of other authors to parallelize ray tracing for radiation applications focused on passing between processors rays that have breeched the local grid extents. Wise and Abel of Princeton and Stanford Universities, respectively, expended considerable efforts on their parallelization strategy of their ray casting scheme for the coupled hydryodynamics radiation code, ENZO. Unfortunately, strong scaling analysis of this algorithm showed no improvement of computational time for parallelism at 70 or more processing cores [29, 30]. This was perhaps due to the large amount of communication that results from the passing

of rays between processors. Kuiper *et al*. developed a similar parallel ray tracing scheme for computing radiation transport in stellar formations, but to date, have not demonstrated strong scaling beyond 64 processors [31]. In 2009, Gentile successfully scaled ray tracing for radiation calculations to 128 processors. Any further increase in processors resulted in no further decrease in computational time [43]. Numerous other authors have developed similar parallel ray tracing strategies for radiation calculations, and have achieved scaling in the range of 8 to 72 processors [29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 9, 40, 24, 41, 42, 43]. If there are others outside our research group who have achieved scaling beyond 128 processors for ray tracing of radiation calculations in combustion simulations, it is not obvious in the literature.

As part of this dissertation, a parallel-capable radiative-flux solver that scales to 1,728 CPUs has been created (see figure 7.2). Such scaling is attained by avoiding the Message Passing Interface (MPI) by stitching together a decomposed domain into a global domain for each processor, and utilizing reverse monte-carlo rays that traverse freely throughout the domain until extinction.

Todd Harman and Alan Humphry of the University have translated the RMCRT code described in this thesis into the CUDA language, allowing RMCRT to run on the Graphics Processing Unit (GPU) nodes of the supercomputer, Titan. The shared memory of the GPUs and breakdown of each GPU into hundreds of cores allowed for the simultaneous tracing of hundreds of rays per GPU [64]. The CUDA version of RMCRT attained ideal strong scaling to 16,000 processing cores (see figure 7.1).

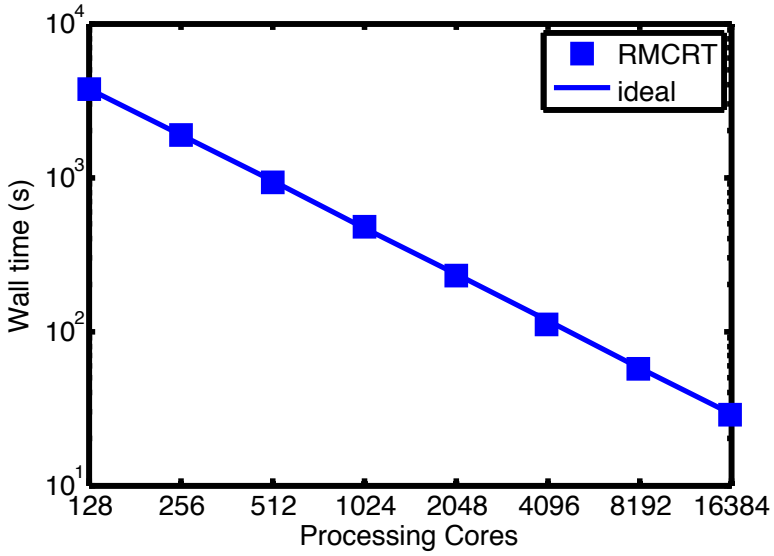


Figure 7.1. Strong scaling of the reciprocal monte carlo radiation model performed on the Titan GPU cluster.

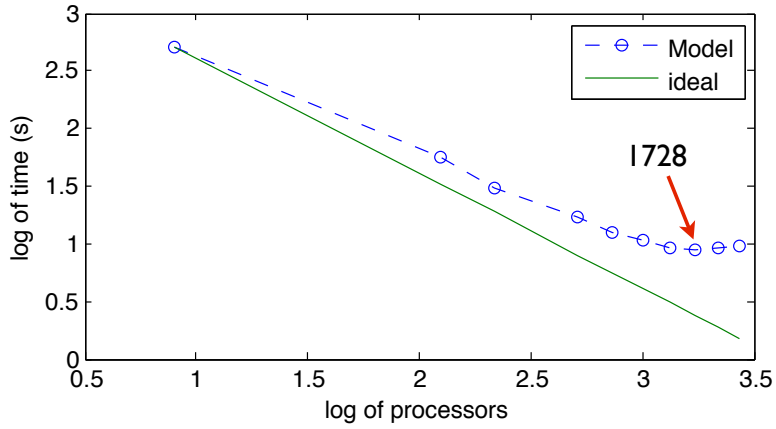


Figure 7.2. Strong scaling analysis of RMCRT on 8 to 1728 processors using 100 rays per cell on a domain of 150^3. on the Titan GPU cluster.

7.3 Adaptive Focus Mesh Refinement

Memory constraints begin to restrict the size of the domain at approximately 300^3 cells. A potential solution to this problem involves a composite mesh that allows each processor to be handed a fully resolved version of a subset of the domain and a coarsened version of the remainder of the domain. This approach has been coined Adaptive Focus Mesh Refinement, or the Data Onion (see figure7.3).

The adaptive-focus mesh addresses the issue of global storage of radiation field values, thus avoiding the passing of rays between processors. Use of the message passing interface (MPI) is minimal, and is limited to the patch re-composition operation. Arches and other Uintah components perform parallelism via patch domain decomposition. In this paradigm, an intact version of the entire domain simply does not exist. Portions of the domain are stored amid the various processors, so to create a composite mesh each processor gathers the patches from all other processors. The more time that is spent on passing information between processors, the less efficiently the algorithm will scale. To mitigate the amount of data that is handled on the MPI, I propose aggressive coarsening on regions of the domain that are distal to the focus region. Justification for this is two-fold. First, the physical distance between the distal and focus regions increases the optical thickness. Therefore, any contribution from the distal regions will be attenuated exponentially along that path length, thus decreasing the effect of the distal region on the origin cells. Second, the distal regions subtend a smaller solid angle than do proximal regions, again limiting the impact on the focus cells.

At present, the Data Onion approach is producing unfavorable accuracy results. Perhaps due to a bug in the code, accuracy when using the data onion is poor (see figure 7.4). Further investigation may reveal a flaw in the programming or perhaps the method itself.

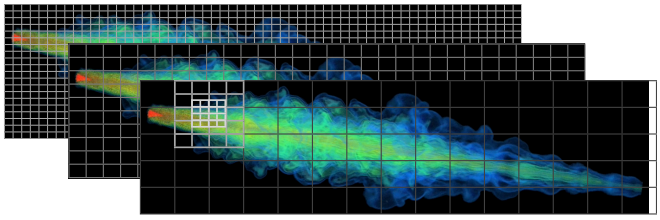


Figure 7.3. Fine CFD mesh on which the fluid/particle equations are solved (left). Single level, asynchronous mesh at a coarser level for the radiation physics (center). Multi-level, adaptive-focus mesh, a.k.a Data Onion (right).

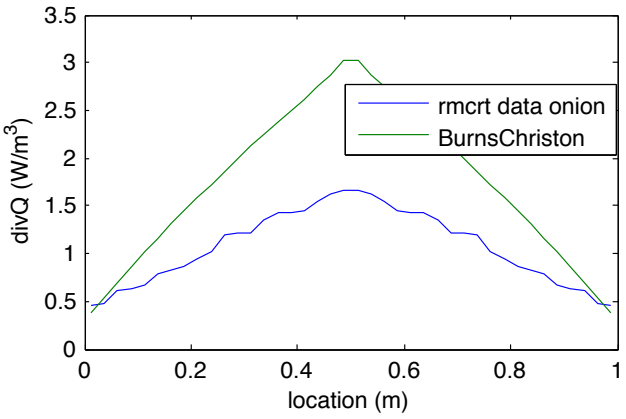


Figure 7.4. RMCRT with Adaptive Focus Mesh Refinement compared with the Burns and Christon numerical solution.

CHAPTER 8

VIRTUAL RADIOMETER MODEL

8.1 Virtual Radiometer Model

There exists a general need to compare radiative fluxes from experimental radiometers with fluxes computed in Thermal/Fluid simulations. Unfortunately, typical numerical simulation suites lack the ability to predict fluxes to objects with small view angles thus preventing validation of simulation results. A new model has been developed that allows users to specify arbitrary view angles, orientations, and locations of multiple radiometers, and receive as the output, high-accuracy radiative fluxes to these radiometers. This virtual radiometer model incorporates a reverse monte-carlo ray tracing algorithm adapted to meet these user specifications and runs on both unstructured and structured meshes. Verification testing of the model demonstrated the expected order of convergence. Validation testing showed good agreement between calculated fluxes from the model and measured fluxes from radiometers used in propellant fires.

NOMENCLATURE

E = Exact Solution

R = Uniformly distributed random number

q_{eff} = Effective emissive power

L = Length of one side of a cube

ϵ = Statistical error

σ^2 = Statistical variance

C = Computed Solution

F = View factor

Ω = Solid Angle

$Q3$ = 3rd Quadrant of a 2D Cartesian Grid

$Q4$ = 4th Quadrant of a 2D Cartesian Grid

A = Rotation matrix

$\vec{b}$ = x,y,z coordinates rotated into the appropriate orientation

$\vec{x}$ = x,y,z coordinates prior to rotation into the appropriate orientation

θ = Radiometer rotation angle about the y axis

ϕ = Radiometer rotation angle about the x axis

ξ = Radiometer rotation angle about the z axis

θ_v = Radiometer View Angle

θ_r = Ray polar angle

ϕ_r = Ray azimuthal angle

ir = Ray index number

l = A point along a ray

T = Temperature

N = Number of rays traced per radiometer

q = Radiative flux

I_{in} = Incoming intensity

κ = Absorption coefficient

I_{out} = Outgoing Intensity

n = Radiometer normal vector

Subscripts

i = Incident

cv = Control Volume

b = Black body

o = Outgoing

w = Wall

n = Net

8.2 Introduction

The ability to validate data from experiments with that of simulations, and *vice versa* is useful in quantifying uncertainty and in improving measurement and numerical techniques. Unfortunately, due to the lack of a viable virtual radiometer model, the fire science community has had difficulty reconciling the data from the instruments that measure radiative flux and the output of models used in fire simulations [103]. The narrow view angles inherent to many experimental radiometers, as well as the variability of location and orientation of these radiometers make them particularly difficult to model. This article elucidates the necessary steps to create a numerical model to represent the physics associated with radiative fluxes to surfaces with arbitrary view angles and orientations, and describes the benefits of using a modified reverse monte-carlo ray tracing scheme. The paper begins with a discussion of the difficulties in modelling an experimental radiometer. A brief analysis of the governing equations for radiation in participating media is then given. We then delve into the additional calculations necessary to accommodate for features such as ray marching in unstructured meshes, and how to generate equi-distributed random numbers on arbitrary solid angles. Results of the model's ability to closely match known solutions of benchmark cases and measured values from experiments are given.

8.3 Difficulty of modelling radiometers within a computational framework

Radiometers used by experimentalists come in a wide variety of configurations. To accommodate for this variability, a virtual radiometer model must allow the user to specify the configuration. The primary parameters include the radiometer view angle, location, and orientation. See Fig. (8.1) for a simple schematic of a radiometer.

Traditional radiation models, such as the Discrete Ordinates Method, discretize the spatial domain without regard to the view angle, orientation, or location of a radiometer [17]. This can lead to drastic inaccuracies when predicting fluxes to radiometers, which tend to have small cross sections [103]. This problem, known as the ray effect, is exacerbated for radiometers with small view angles. For example, the radiometer represented by the cone at location (a) in Fig. (8.2) is located between two rays, and therefore would register an under-predicted flux. Conversely, the radiometer at location (b) should reject the incoming ray that is outside its view angle. However, most radiation models cannot account for this, and as a result, would over-predict the incident flux. An example of the ray effect from a fire simulation is demonstrated in Fig. (8.3). In the flame front of this image, the fuzzy light streaks are not a feature of the fire, but rather an artifact of the ray effect.

To overcome the ray effect and create a model that properly restricts incoming intensities to those within the bounds of a radiometer's view angle, we created a modified reverse monte-carlo ray tracing scheme. The details, including the appropriate governing equations and considerations for arbitrary view angles and orientations, are given in the following sections.

8.4 Governing equations of radiation in participating media

The governing equation for reverse monte-carlo ray tracing in nonhomogeneous, participating media was first developed by Walters and Buckius [91]. Specifically,

$$I_{i,k} = \int_0^{l_k} I_{b,cv}\kappa(l')exp[-\int_{l'}^{l_k}\kappa(l'')dl'']dl' + I_{o,sur}(T_w)exp[-\int_{l_w}^{l_k}\kappa(l')dl'], \tag{8.1}$$

where $I_{i,k}$ represents the incident intensity at location k, κ represents the absorption coefficient, and l represents the locations of the segment lengths along a ray.

In a discretized domain, we assume piecewise homogeneity and pose Eqn. (8.1) in the following form,

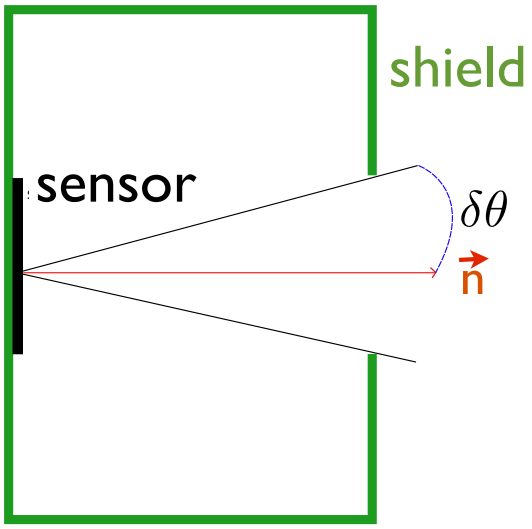


Figure 8.1. simple schematic indicating $\delta\theta$. the view angle of the radiometer is $2\delta\theta$.

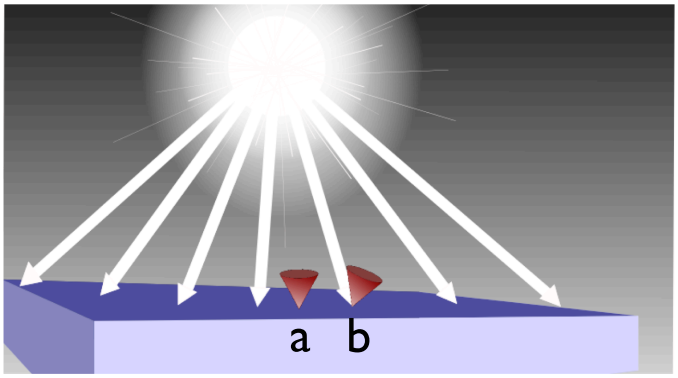


Figure 8.2. With current radiation solvers,the radiometer at location "a" would register an under-predicted flux whereas the radiometer at location "b" would over-predict the incident flux.

$$I_{i,k} = \sum_{m=1}^{M} \left(I_{b,v}(e^{-\int_{lm2}^{l_k} \kappa(l')dl'} - e^{-\int_{lm1}^{l_k} \kappa(l')dl'}) \right) + I_{o,s}(T_w) e^{-\int_{l_w}^{l_k} \kappa(l')dl'}, \tag{8.2}$$

where M represents the full path length of a ray comprised of a series of smaller segments denoted by m. The intensities from each of the rays of a radiometer are then weighted according to the solid angle that each ray subtends [45]. Assuming uniform distribution of the rays, each ray will subtend $\frac{\Omega}{N}$ steridians, where Ω is the solid angle of the radiometer

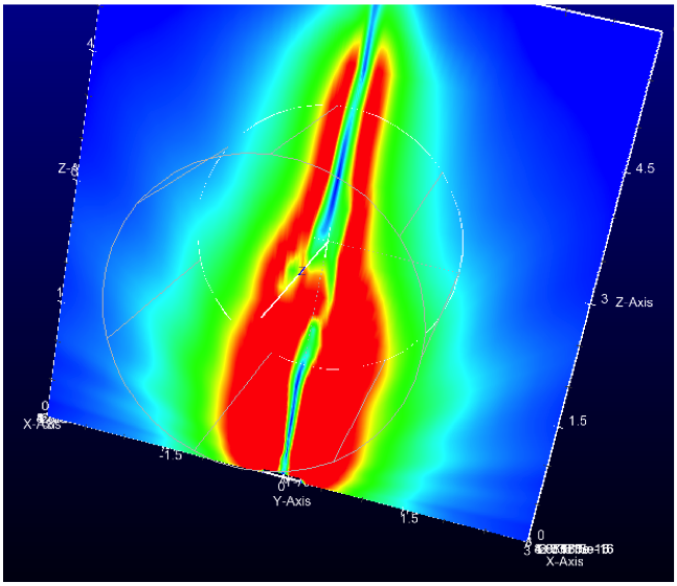


Figure 8.3. The ray effect is visible in this cutaway that shows the spatially varying radiative flux of a simulation of a propellant fire.

and N is the number of rays used in the simulation. The radiative flux is then calculated from the intensities of the rays, weighted by the discretized solid angle,

$$q_i = \frac{\Omega}{N} \sum_{r=1}^{N} I_i(ir) cos(\theta(ir)), \tag{8.3}$$

where $I_i(ir)$ and $\theta(ir)$ represent for a particular ray, the incoming intensity and angle from the radiometer normal, respectively.

Some of the radiometers used in experimental measurements are calibrated against a black body that subtends the entire field of view of the radiometer. These radiometers therefore report a value that is an effective emissive power,

$$q_{eff} = \frac{q_i}{(1 - cos(\theta_v/2))cos(\theta/2)}. \tag{8.4}$$

To accommodate, the output of the model computes both q_i and q_{eff}.

Note: Experimental radiometers measure a net flux, $q_n = q_i - q_o$. Traditionally in radiation texts, $q_n = q_o - q_i$. In the fire sciences, however, because $q_o << q_i$ it is generally accepted to report a positive flux *to* a radiometer, neglecting q_o. Justification for this is increased when the radiometers are liquid cooled, decreasing q_o as is the case with the experimental radiometers mentioned in the Validation section.

8.5 User-specified view angle and orientation

Perhaps the most novel feature of this algorithm is the capability to handle, at run-time, a user-specified orientation and view angle for the radiometer. The implementation of these two features is described in the following two sub-sections.

8.5.1 User-specified view angle

The view angle will define a solid angle about which rays must be generated and uniformly distributed. To accomplish this, two random numbers must be generated to span this two dimensional surface. The naive approach would be to generate two random numbers between 0 and 1, and simply scale them by a factor of 2π for the azimuthal angle ϕ, and a factor of θ_v for the polar angle, where $\theta_v = V/2$, where V is the view angle of the radiometer. Unfortunately, this will lead to non-uniformly distributed rays. To demonstrate, consider a solid angle of 4π $sr.$, a sphere. If one were to attempt to generate a series of rays according to the naive approach, the azimuthal and polar angles of the rays would be described as follows,

$$\phi_r = 2\pi R_1 \tag{8.5}$$

$$\theta_r = \pi R_2, \tag{8.6}$$

where R_1 and R_2 are two independent random numbers uniformly distributed between 0 and 1. This approach would lead to the distribution shown in (a) of Fig. (8.4) [104].

The clustering around the poles is a result of the surface area of spherical objects not being proportional to $\delta\theta$. To account for this, we must be able to produce a random number within the range of $cos(\pi)$ to $cos(0)$ (evaluated as -1 to 1), then take the arccosine of that number. Figure (8.5) demonstrates why this is necessary.

Notice in Fig. (8.5) that the range of random numbers that produces a value of θ between 0 and $\pi/4$ is 0.707 to 1. This range of possible random numbers is less than half the size of that which produces a value of θ between $\pi/2$ and $\pi/4$, specifically 0 to 0.707. This is consistent with the fact that the surface area on a sphere between $\theta = \pi/4$ to $\pi/2$ is more than double that of of the surface on a sphere between $\theta = 0$ to $\pi/4$. Scaling a random number in this manner correctly causes the probability of picking a point within a given $\delta\theta$ to be proportional to the area that $\delta\theta$ subtends in the unit sphere. This is consistent with the equation for an infinitesimal solid angle given by

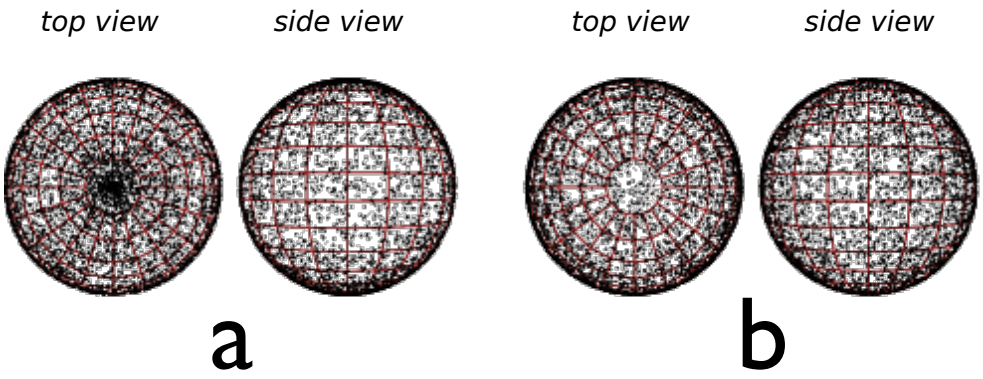


Figure 8.4. Random numbers that vary uniformly with the polar angle produce incorrectly distributed points clustered near the poles as shown in (a). Random numbers appropriately weighted by the polar angle produce points that are correctly distributed (b).

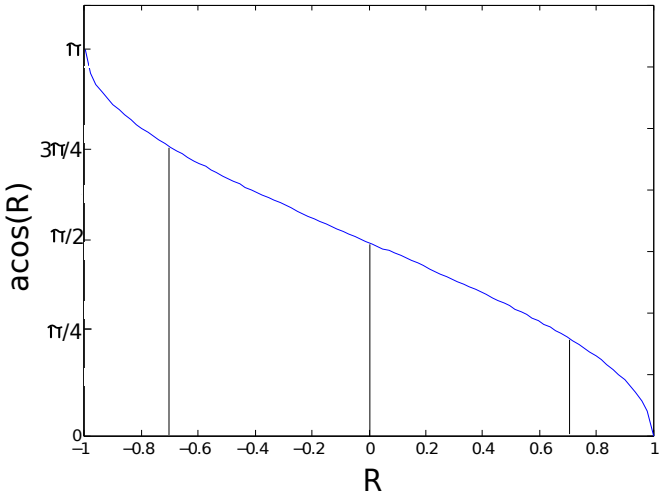


Figure 8.5. Distributing random points on a sphere requires the scaling by arccosine of the random number R, where $R = 2R_2 - 1$, such that R has a range of -1 to 1.

$$d\Omega = sin(\theta)d\phi d\theta = -d\phi(cos(\theta)). \tag{8.7}$$

With this correct implementation of picking random points on a sphere, the azimuthal and polar angles are assigned as,

$$\phi_r = 2\pi R_1 \tag{8.8}$$

$$\theta_r = acos(2R_2 - 1), \tag{8.9}$$

where $(2R_2 - 1)$ yields a random number uniformly distributed between -1 and 1. Similarly, to generate random points on a hemisphere, the polar angle for a given ray would be assigned as $acos(R_2)$ where R_2 has a range of 0 to 1 and corresponds to polar angles between 0 and $\pi/2$. Then, to generate points on an arbitrary solid angle that the user-specified view angle subtends, one needs to ensure that the random numbers have a range of $cos(-\theta_r)$ to $cos(\theta_r)$. This is accomplished by setting the range equal to

$$range = 1 - cos(\theta_v). \tag{8.10}$$

Then, to generate rays within the user-defined solid angle, the azimuthal and polar angles for a given ray are assigned as

$$\phi_r = 2\pi R_1 \tag{8.11}$$

$$\theta_r = acos\left(cos(\theta_v) + range * R_2\right). \tag{8.12}$$

These generated points, with the location of the radiometer as their origin, define the direction vectors of the rays which are then traced through the domain.

8.5.2 User-specified orientation

From a developer's standpoint, it would be most convenient if the orientation of radiometers used in experiments were always aligned in the same direction. This, however, is not the case. To handle the variability in the radiometer orientation, the virtual radiometer model must be able to take as an input, any user-defined direction normal vector, and adjust the orientation of the rays accordingly. To accomplish this, a matrix that transforms any Cartesian point about a vector comprised of rotation angles is employed. This matrix, $[A]$, is defined as

$$\begin{bmatrix} cos\theta cos\xi & -cos\phi sin\xi + sin\phi sin\theta cos\xi & sin\phi sin\xi + cos\phi sin\theta cos\xi \\ cos\theta sin\xi & cos\phi cos\xi + sin\phi sin\theta sin\xi & -sin\phi cos\xi + cos\phi sin\theta sin\xi \\ -sin\theta & sin\phi cos\theta & cos\phi cos\theta \end{bmatrix} \tag{8.13}$$

where ϕ, θ, and ξ represent the counter-clockwise rotation angles about the x,y, and z axes, respectively. To relieve the user of the burden of determining these three rotation angles, the virtual radiometer model takes as an input, the normal vector of the radiometer and computes these rotations automatically. This is accomplished as follows,

$$\theta = acos\left(\frac{n_z}{\sqrt{n_x^2 + n_y^2 + n_z^2}}\right) \tag{8.14}$$

$$\xi = acos\left(\frac{n_x}{\sqrt{n_x^2 + n_y^2}}\right), \tag{8.15}$$

where n_x, n_y, and n_z represent the components of the vector normal of the radiometer. Note that there will never be a need to calculate a rotation about the x axis. All possible rotations can be accomplished using the other two, while fixing ϕ at 0. Due to the constraints of arccosine, the value of ξ must be adjusted if n_x and n_y are in the 3rd or 4th quadrants of the xy plane. Specifically,

$$if(n_x, n_y) \in Q_3 \ : \ \xi = \pi/2 + \xi_{calculated} \tag{8.16}$$

$$if(n_x,n_y) \in Q_4 \ : \ \xi = 2\pi - \xi_{calculated} \tag{8.17}$$

At this point, the rotation angles are applied to the direction vectors of the rays via the matrix multiplication of

$$[A]\vec{x} = \vec{b}, \tag{8.18}$$

where $\vec{x}$ is the pre-rotated direction vector of a ray in Cartesian coordinates, and $\vec{b}$ is the resulting direction vector.

8.6 Ray marching in unstructured meshes

When performing ray tracing on structured meshes, the marching algorithm (the algorithm that determines which cell will be referenced and in what order) is relatively simple due to the geometric relations that exist between the orientation of the ray and the surface normals of the cells (see section 3.1). Optimization of this algorithm also becomes trivial as there are only 6 possible face normal vectors, which are further reduced to three once the signs of the components of the ray direction vector are known [93]. However, when the mesh in use is unstructured, the geometric relationships between the ray orientation and the surface normals of the element faces become complicated, and may vary greatly from step to step. The tracing of rays in an unstructured mesh therefore requires the implementation of one of several possible methods, two of which are described as follows. The first involves walking the node connectivity by calculating intersections between the current location of the ray and the faces of the elements. The second involves selecting points *a priori* along the defined ray direction, and querying the elements to find which element owns those coordinates. While the first method will undoubtedly result in a more efficient algorithm, the development burden is much greater than that of the second as it requires information regarding the elements' neighbors, node extents, face normal vectors, etc. The second approach, involves simple element search routines and is much less developmentally intensive. This approach includes useful speedups such as overlaying the mesh with a coarse, structured grid. This requires the initial search to find not the element that contains a given set of parametric coordinates, but rather the "bucket" that contains these coordinates. Once this bucket has been identified, only the elements within the bucket are queried, rather than

all the elements in the entire domain. For these reasons, the second approach was selected to handle ray marching in unstructured meshes.

8.7 Verification and validation

We performed a series of tests to ensure that our algorithms accurately represented the model, and that the model accurately represented reality. These test and the subsequent results are outlined in the following subsections.

8.7.1 Verification of ray distribution

To verify that the randomly generated rays were uniformly distributed over the solid angle of the radiometer, and that Eqn. (8.3) was implemented correctly, the model was used to solve for the view factor between a circular disc and an infinitesimal surface as shown in Fig. (8.6). For a disc of radius r separated from the infinitesimal area by a distance h, the analytical solution as given by [87] for the view factor is

$$F = \frac{1}{(h/r)^2 + 1}. \tag{8.19}$$

The L2 error norm, defined as $\frac{\sqrt{(E-C)^2}}{E}$, where E is the exact solution and C is the computed solution, was shown to decrease with an increase in the number of rays such that at 100,000 rays, the L2 error norm was approximately 0.001. Because the distribution of the rays is of importance particularly for wide angle gauges, this L2 error norm is printed to the log file when values of the view angle exceed atan(0.5), or approximately 51°. The ray error can then be used as a metric to assess the confidence in other computed values such as the radiative-heat flux. This verification test does not exercise the participating media portion of the algorithm, so the 3D case described by Burns and Christon [1] was employed to verify these features as explained in the following section.

8.7.2 Verification of participating media physics

Exact solutions to radiation problems involving non-homogeneous, non-isothermal, emitting, absorbing media are difficult to come by [41, 46]. The integro-differential Radiative Transport Equation becomes prohibitively complex for these situations, limiting the availability of analytical solutions. There are, however, a handful of well documented semi-exact solutions for participating-media radiation problems. One such solution was given by Burns and Christon in 1997 [1]. This case is described by a three dimensional cube with cold black walls, with an absorption coefficient, κ, that varies according to following the trilinear function,

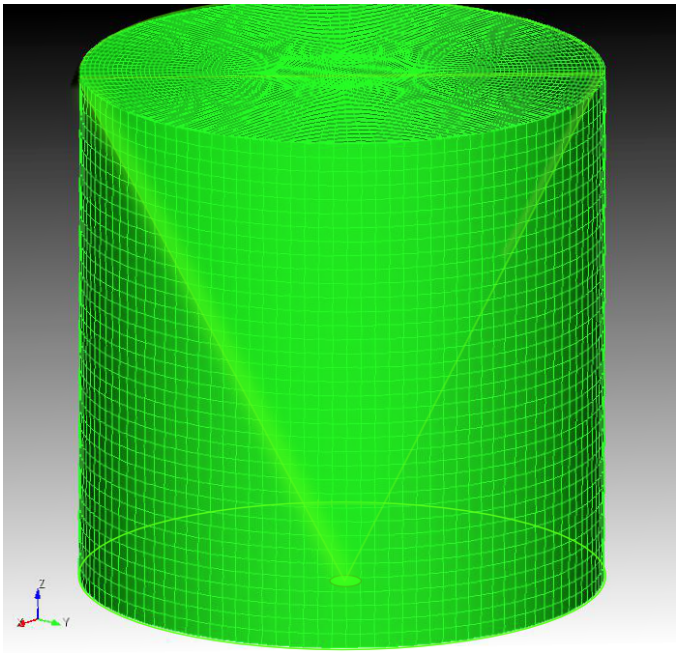


Figure 8.6. Schematic representation of the view factor of a circular disk as viewed by a point at the bottom of a cylinder.

$$\kappa L = 0.9\left(1 - 2 \mid \frac{x}{L} \mid\right)\left(1 - 2 \mid \frac{y}{L} \mid\right)\left(1 - 2 \mid \frac{z}{L} \mid\right) + 0.1 \tag{8.20}$$

where L is the length of the cube, and x, y, and z are the distances from the center of the Cartesian domain. A visualization of this absorption coefficient is given in Fig. (8.7). The semi-exact solution was computed by Burns and Christon via the Discrete Ordinates Method on a 40^3 domain. The number of ordinate directions was increased until the radiative flux divergence, $\nabla \cdot q$, converged to 10^{-5}.

The intensities computed by the virtual radiometer model were used to compute the radiative flux divergence at various locations in the domain for which a solution was given by Burns and Christon. The radiative flux divergence is described in terms of intensity as follows

$$\nabla \cdot q = \kappa \left(4\pi I_{b,out} - \int_{\Omega} I_{in} d\Omega\right), \tag{8.21}$$

or in discretized form as

$$\nabla \cdot q = \kappa(4\pi I_{b,out} - \sum_{ir=0}^{N} \left(I_{in}(ir)\frac{4\pi}{N}\right), \tag{8.22}$$

where I_{in} is described by Eqn. (8.2) and N is the number of rays per radiometer. Although this case solves for the flux divergence rather than a surface flux, because the expression is a function of the incoming intensity, this case exercises the participating media functionality of the algorithm. The resulting flux divergences at the specified locations were in excellent agreement with the solutions of Burns and Christon and are illustrated in Fig. (8.8). Using 10,000 rays per radiometer an L2 error norm of 0.00305 was obtained.

8.7.3 Verification of ray convergence

Convergence of the solution to the incident flux as a function of the number of rays used in the simulation is demonstrated by Fig. (8.9). Here, the virtual radiometer model was run on data produced by an 18" aluminum propellant fire. The resulting fluxes produced from varying ray numbers were compared to the converged values as computed by a simulation with 1.4 million rays. The expected rate of convergence is calculated by recognizing that due to the statistical nature of the random rays, the variance of the solution is proportional to the inverse of the number of rays,

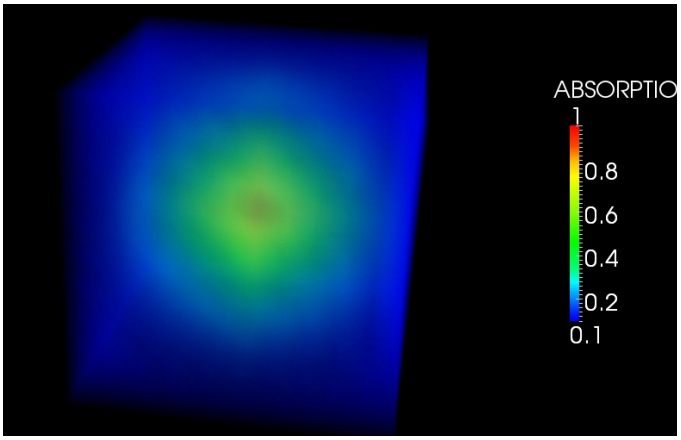


Figure 8.7. absorption coefficient as specified in the case described by Burns and Christon [1].

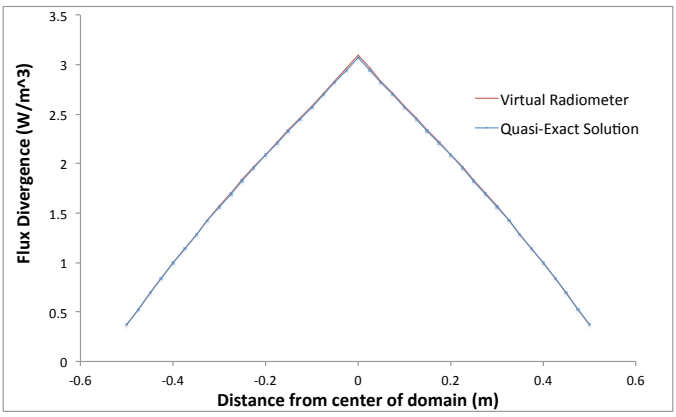


Figure 8.8. Results of the participating media physics verification test. The virtual radiometer model demonstrates excellent agreement with the numerically-exact solution.

$$\sigma^2 \propto \frac{1}{N}. \tag{8.23}$$

The error, which is proportional to the square root of the variance is therefore

$$\epsilon \propto N^{-\frac{1}{2}}. \tag{8.24}$$

Taking the log of both sides of Eqn. (8.24) and moving the exponent to the beginning of the RHS yields

$$log(\epsilon) \propto -\frac{1}{2} log(N). \tag{8.25}$$

The expected slope is therefore $-\frac{1}{2}$ which is within 4% of the calculated slope of ray convergence of -0.5173.

As a second ray convergence test, we then used the solution of Burns and Christon, rather than our own converged solution, to compute the relative error. The flux divergences given by the virtual radiometer model at 41 spatial locations were compared with this solution. An L2 error norm of these points was calculated for each ray number (see Fig. (8.10)). Note in Fig. (8.10) that as the number of rays was increased, the difference between the two solutions didn't converge to zero. This is perhaps due to the fact that the quasi-exact solution has a non-zero error which will result in an offset in the convergence. In fact, it is quite possible that at 100,000 rays, the solution produced by the virtual radiometer model is more correct than the quasi-exact solution.

8.7.4 Validation

We performed validation testing against experimental data to compare the results of the virtual radiometer model with physical reality. Experimental data was supplied by a previously performed test of Sandia National Laboratories [105]. During this testing, a series of aluminum propellant blocks was allowed to combust to completion. Each test consisted of varying sized propellant blocks with radiometers and spectrometers placed in varying arrangements. A simulation model was constructed to match the parameters of one of these cases, the 18" propellant block upward burn case, and the model was analyzed using Fuego [106]. The results of the Fuego simulation were used as input for the virtual radiometer model which then calculated radiative fluxes at the same locations of the radiometers in the experimental setup. The results are summarized in Fig. (8.11) which demonstrates the agreement between experimental and simulation data.

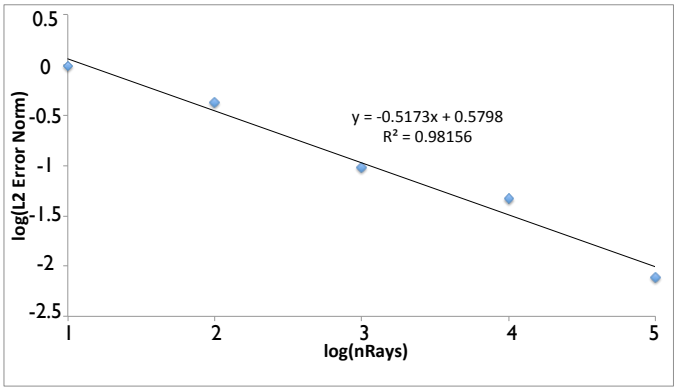


Figure 8.9. L2 error norm as a function of ray number, using the converged solution of 1.4M rays to compute relative error. The solution converged at the expected order of approximately $-\frac{1}{2}$.

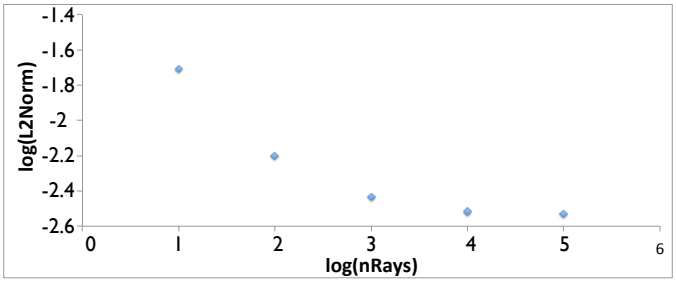


Figure 8.10. Convergence of the virtual radiometer results relative to the quasi-exact solution [1].

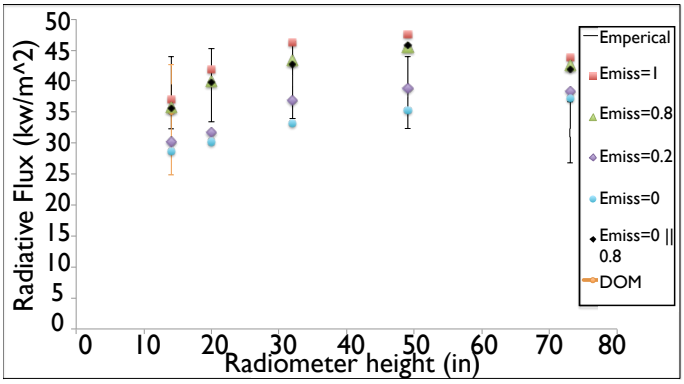


Figure 8.11. Validation results indicate good agreement between the experimental results (black bars) and model results at varying emissivities (dots).

8.8 Efficiency considerations

Several efforts have been made to ensure that the virtual radiometer model runs efficiently. The choice of the random number generator (RNG) is one such effort. The RNG we selected is the Mersenne Twister algorithm, which produces high-fidelity equi-distributed random numbers [92]. We found that this RNG produced random numbers at a rate of $O(10^7)/s$ on our 2.3 GHz processors [107].

The virtual radiometer algorithm also avoids division, and removes constants outside of summation loops. This is demonstrated in the calculation of the flux from the incident intensities, where if we assume uniformly distributed rays, $d\Omega$ is constant and equal to $\frac{\Omega}{N}$. Therefore

$$q_i = \frac{\Omega}{N} \sum_{r=1}^{N} I_i(r) cos(\theta(r)), \tag{8.26}$$

which, for a case with 5 virtual radiometers of 100,000 rays each, requires 500,000 fewer divisions and multiplications than the following mathematically-equivalent expression

$$q_i = \sum_{r=1}^{N} I_i(r) cos(\theta(r)) \frac{\Omega}{N}. \tag{8.27}$$

8.8.1 Differences between Uintah Radiometer and Sandia Radiometer

The virtual radiometer model that has been developed for use in Uintah's RMCRT is very similar to that of the Sandia model, with a few exceptions. The Sandia radiometers trace rays that have pre-defined segment lengths. Therefore, the points at which field values are referenced are independent of the mesh. An element search algorithm is used to determine in which element a particular segment point lies, and the field values are interpolated linearly to this location. In Uintah RMCRT, however, because of the use of structured meshes, the segment lengths are a function of the mesh, and the cell center values along the ray's path are referenced readily. It would be possible to use an interpolation scheme in Uintah to interpolate values to more exact locations along the ray, but to date, this has not been implemented. The other major difference is that the Sandia algorithm allows for independently oriented radiometers, each with an independent view angle. The Utah code handles multiple radiometers as well, but is currently limited to homogeneous normal vectors and view angles. The Uintah virtual radiometer model does have a few advantages over that of Sandia in that it can handle the additional physics of reflecting walls. Perhaps the greatest computational advantage is that the Uintah model is parallel capable, whereas the Sandia model runs only in serial.

For cases that do not have wall emission or reflection, such as Burns' benchmark case, results from the two models are very similar, as indicated in Fig. (8.12)

There is very little change in Fig. (8.12) for the 1 cm and 1mm segment length cases because the Sandia radiometer uses linear interpolation to get the field values at the segment points, and this benchmark uses linearly varying absorption coefficients. Therefore, each ray segment references an analytical value, so the only difference between the two length scales is how many points are referenced. The Uintah radiometer uses cell centered values, so an increase in grid refinement leads to more accurate references at each point, as well as more points to represent the rays. For these reasons, the Uintah radiometer converges as the grid resolution is enhanced as indicated in Fig. (8.13) The increase in L1 norm from the 201^3 case to the 301^3 case is a result of the randomness of the ray error, which even for 100,000 rays is still non-negligible at fine resolutions.

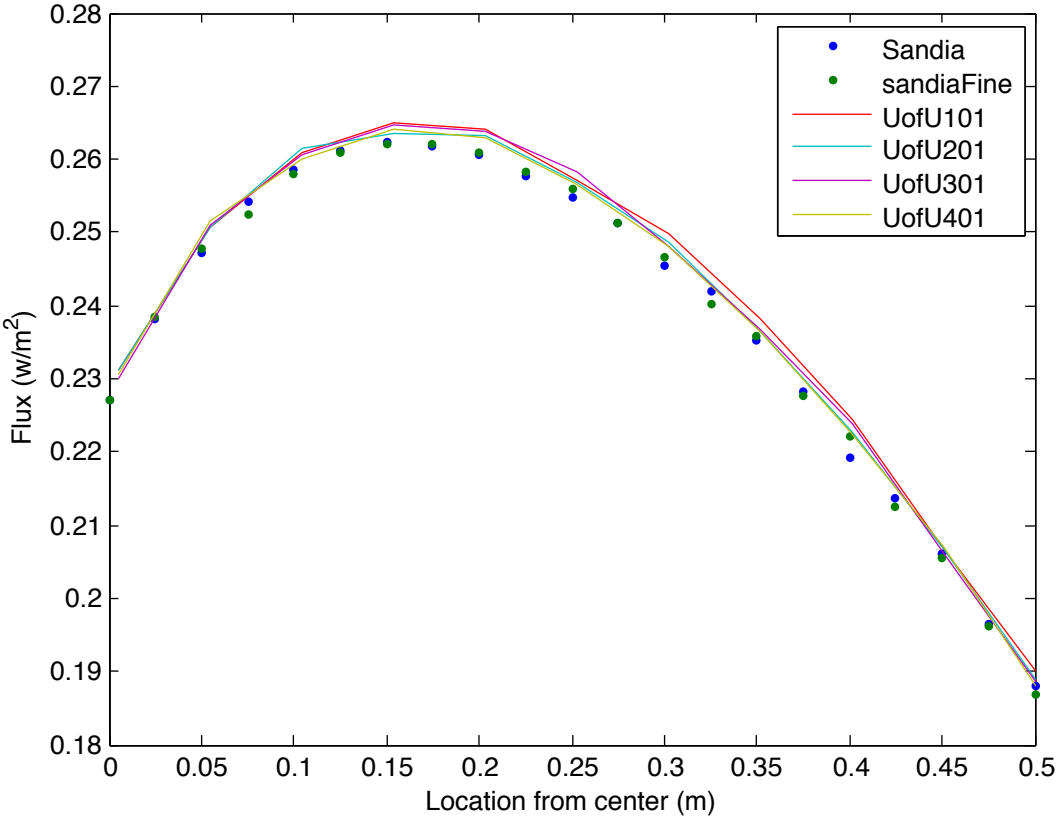


Figure 8.12. Radiative-flux divergence along a center-line of Burn's benchmark case. Two runs were made using the Sandia virtual radiometer using a segment length of 1 cm (Sandia) and 1mm (Sandia Fine). Four runs were made using the Arches radiometer using grid resolutions of 101^3, 201^3, 301^3, and 401^3.

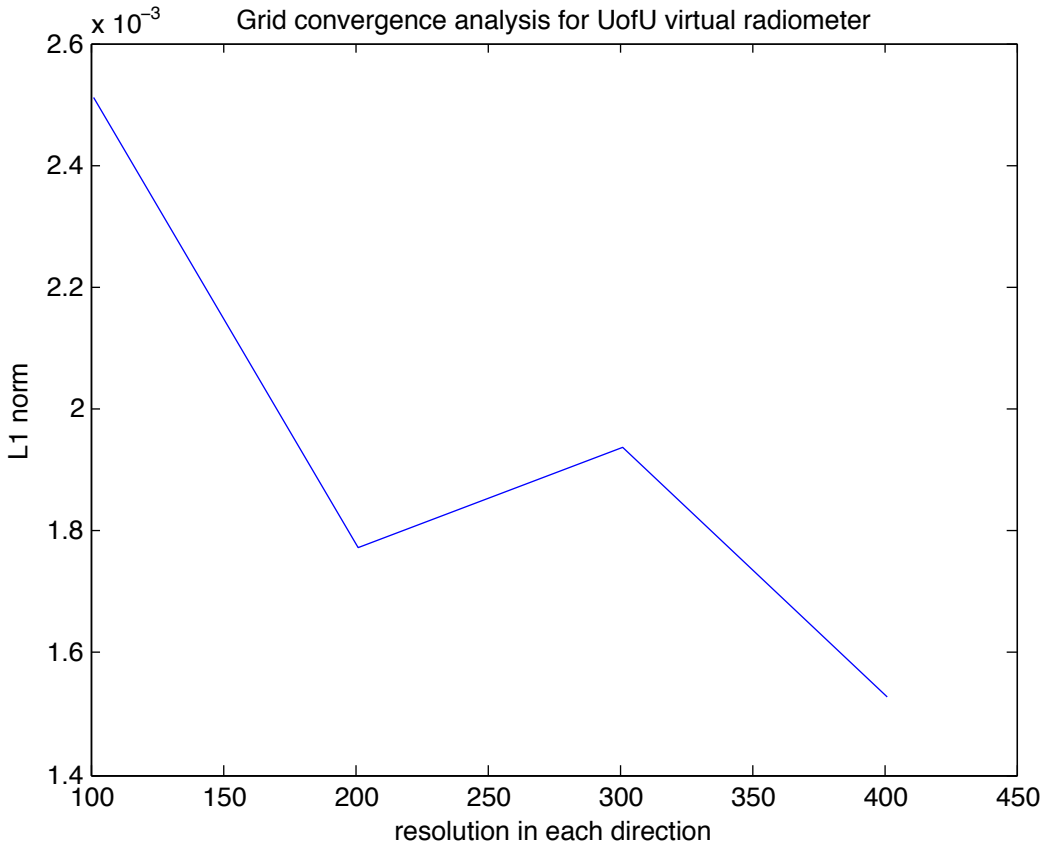


Figure 8.13. Grid convergence analysis for U of U virtual radiometer.

CHAPTER 9

IFRF CASE STUDY

9.1 IFRF case study

The International Flame Research Foundation (IFRF) has shared with the Institute for Clean and Secure Energy (ICSE), valuable data pertaining to combustion experiments in large, gas-fired boilers. Using these data, ICSE has developed Large Eddy Simulation models to represent IFRF experiments. One such case is a representation of a methane-fired boiler. The boiler is a cylinder of 4.1m in length and 1m in diameter. The burner is comprised of a primary fuel annulus and a secondary oxygen-stream annulus. The simulation was to run to a pseudo-steady state of the fluid dynamics, which required approximately 70,000 timesteps. Verification testing of RMCRT was then performed by comparing radiative flux q, and radiative flux divergence $\nabla \cdot q$, values as computed by RMCRT and DOM at the pseudo-steady-state timestep. The agreement between the two radiation methods is demonstrated in Fig. (9.1).

Figure (9.1) was obtained by setting the wall emissivities to 0.5, for both DOM and RMCRT, while restricting RMCRT from performing any reflections. This is un-physical, as some of the radiation that strikes the wall is reflected for walls with an emissivity of less than one. The reflectivity, ρ, is generally calculated as

$$\rho(T) = 1 - \alpha(T),$$

which is accurate so long as the surfaces are opaque, ϵ'_λ is independent of wavelength and direction (gray and diffuse) and the source is gray and diffuse with source temperature T_s, and ϵ'_λ is independent of T, or T_s is equal to the wall temperature [87]. With these assumptions, for the case where the emissivity is 0.5, ρ is calculated to also be 0.5. If RMCRT is performed with this non-zero wall reflectivity, then $\nabla \cdot q$ is calculated as shown in the solid line of Fig. (9.2). Note the approximately -200,000 W/m^2 shift when reflections are taken into account. This is because the radiative flux divergence is calculated as

$$\nabla \cdot q = \kappa(4\pi I_{out} - \int_\Omega I_{in} d\Omega),$$

and therefore, an increase in I_{in} leads to a decrease in $\nabla \cdot q$. If reflections are prohibited in RMCRT, the incident intensity into a given cell is decreased as the reverse-ray stops prematurely. Put another way, the radiosity of the domain walls, R defined as the sum of the emissive power and the reflected radiative flux, would be reduced to simply the emissive power. The decreased radiosity of the walls is simply a result of the model, not a physical phenomenon. Interestingly, if the IFRF domain with non-zero emissive power were to be simulated in a model that cannot perform reflections, such as the current version of DOM,

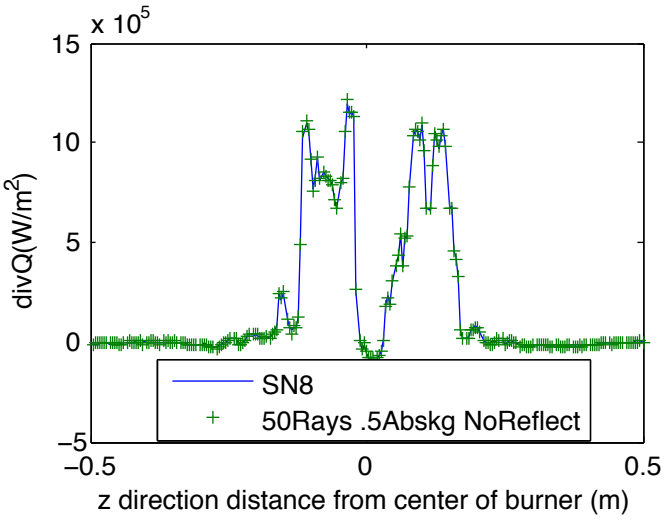


Figure 9.1. Radiative flux divergence from RMCRT (+) and DOM (line) on a z-line through an x-y slice in the center of the domain of an IFRF case. RMCRT used 50 rays per cell, DOM used SN8 in this case.

better results can be obtained simply by assuming black walls, as is demonstrated by the dashed line of Fig. (9.2). Specifying an emissivity of one increases the radiosity of the wall, and shifts the solution approximately 40 W/m^2 toward that of the case where reflections are taken into account.

9.1.1 Fluxes

Figure (fig:ReflectVsNoReflect) shows the comparison between RMCRT and DOM for the radiative flux. Because of the directionality of radiative fluxes, more rays are required to reach a converged solution compared to that required for radiative flux divergences.

To reduce the stochastic noise of the solution, a box filter was applied to the solution to obtain the results shown in figure (9.3).

The L1 error norms of the RMCRT solutions are reduced by approximately 50% by using a simple, one-dimensional box filter with a width of 5. The filtered results of 100 ray-RMCRT have L1 error norms that are 40% lower than those of SN4. RMCRT with 100 rays required approximately the same amount of processor time.

9.1.2 Timing and accuracy

The RMCRT simulations with 10 rays per flow cell, and 1000 rays per boundary cell required a total of 1033 seconds per timestep. This time includes the time to stitch together the full computational domain by doing the all-to-all transfer over MPI, and filter the results. This compares to 2949 seconds per timestep for SN8 (see table (9.1).

9.2 IFRF f85y4 case study

Another IFRF boiler was selected as a case study. The IFRF f85y4 oxy1B case described in [108] was modeled in the Arches component. A copy of the the ups input file is given in the appendix.

The configuration of the boiler is shown in figure (9.5). The geometry of the burner is shown in figure (9.6).

A subsection of the boiler was discretized into 200^3 cells and modeled with a coupled RMCRT Arches simulation as well as a coupled DOM Arches simulation. Both cases were run to pseudo-steady state. The simulations were run on over 1500 processors for tens of thousands of timesteps. Unfortunately, after the simulations were competed, it was discovered temperature boundary condition on the z+ face was not set correctly, leading to inaccurate results. Time did not permit an additional run with the corrected boundary condition. Although comparison with the f85y4 experimental data was futile, valuable

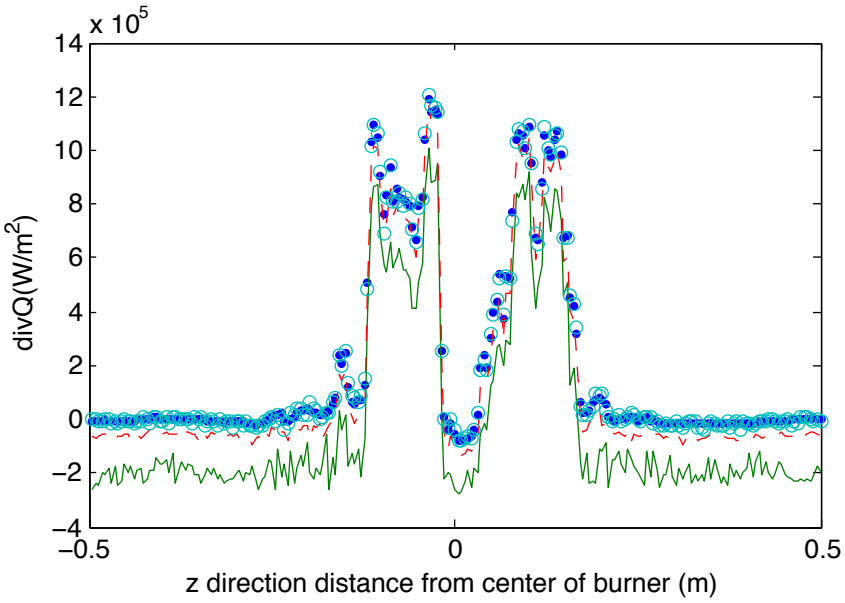


Figure 9.2. Radiative flux divergence for several RMCRT cases and a DOM SN8 case (hollow circle) on a z-line through an x-y slice in the center of the domain of an IFRF case. The three RMCRT cases are respectively, 1 ray per cell with reflections and an emissivity of 0.5 (line) 10 rays per cell with an emissivity of 0.5 without reflections (dot) and 10 rays with black walls (dash).

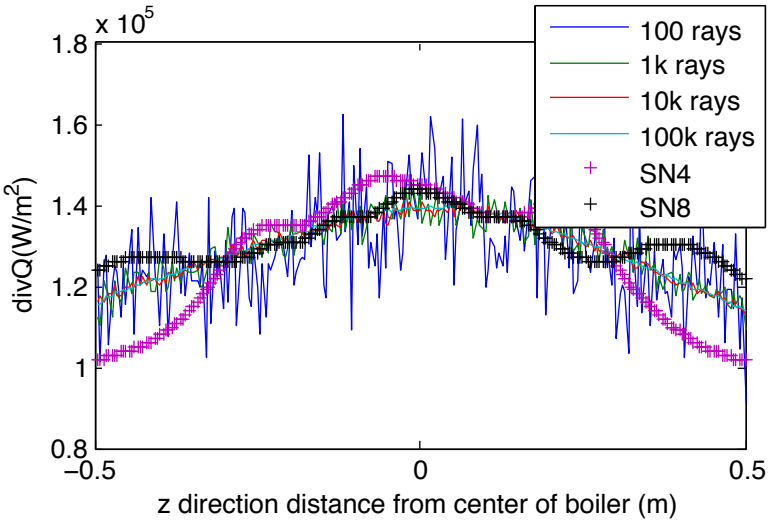


Figure 9.3. Radiative flux as calculated by RMCRT (lines) vs. DOM (+) for varying positions along the z direction of a center-line through the boiler.

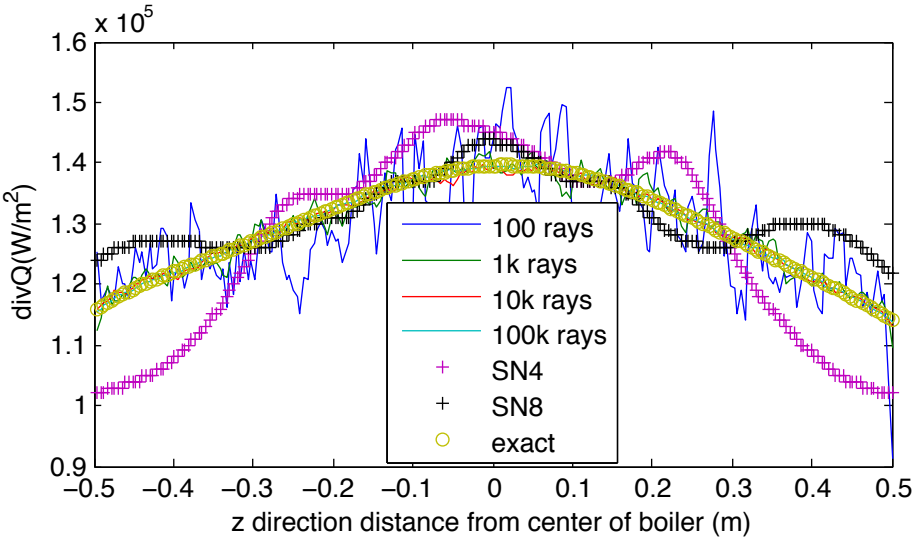


Figure 9.4. Filtered solution of the radiative flux as calculated by RMCRT(lines) vs. DOM(+) for varying positions along the z direction of a center-line through the boiler.

	Computational Time (s)	divQ L1 error norm	q L1 error norm
Filtered RMCRT	1033	1.07	0.0311
DOM SN8	2949	2.05	0.0271

Table 9.1. Timing and accuracy of RMCRT and DOM SN8

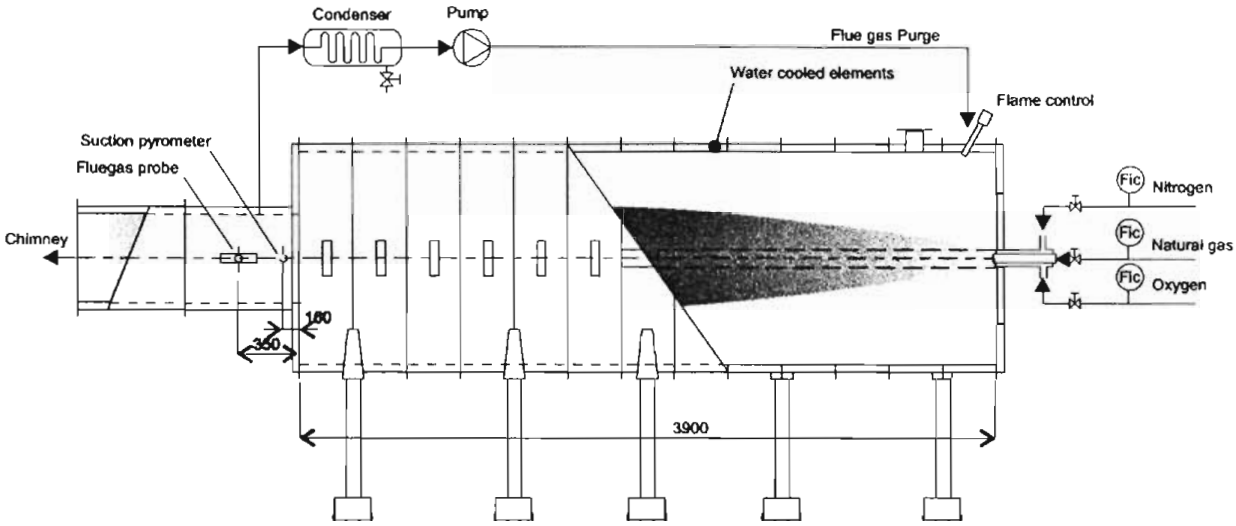


Figure 9.5. Boiler configuration of the IFRF f85y4 oxyflam 1 case.

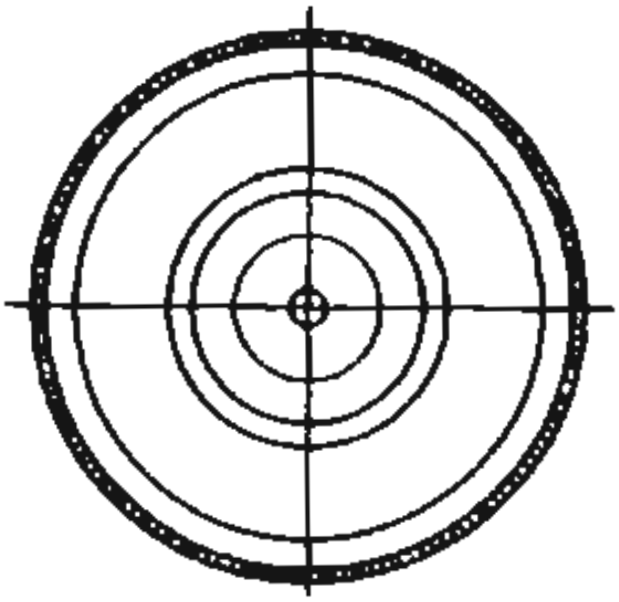


Figure 9.6. Burner geometry of the IFRF f85y4 oxyflam 1B case.

information was still gleaned from these simulations. Perhaps of most importance is that RMCRT demonstrated robustness in handling the coupling with the property models as well as stability to many thousands of timesteps.

9.3 Filtering

Due to the stochastic nature of RMCRT, there is an inherent noisiness to the q and divQ results. This noise can be reduced by increasing the number of rays per cell, but this comes at a linear cost with a 1/2 order benefit. This research indicates that a simple filtering operation can reduce the noise by approximately 50% with negligible cost.

In addition to increasing the accuracy of results, filtering has also been shown to be important for efficiency in the pressure-solve portions of combustion simulations. When divQ results include sharp gradients, the temperature and pressure gradients of subsequent timesteps become steep, leading to stiff equations that require more computation time. For these reasons, filtering of RMCRT results is highly recommended. For this research, filter operations were performed post-process. A method that performs box-filter operations in real time has been developed and is ready for further testing.

CHAPTER 10

SUMMARY AND CONCLUSIONS

A high-accuracy, massively parallel reciprocal monte-carlo ray tracing radiation model was developed and bi-directionally coupled with a turbulent, reacting-flow model in the Uintah framework.

This model computes the radiative-flux divergence for interior cells, and the net radiative flux for boundary cells of a computational domain. To accurately represent reality, the model incorporates the following physics

- Non-homogeneous, absorbing, emitting media
- Homogeneous, isotropic, scattering media
- Black or gray wall absorption and emission
- Specular wall reflections for arbitrary reflectivities
- Complex domains which may include intrusion features
- Fluxes with arbitrarily sized view angles, orientations, and locations

Verification testing of the new radiation model was performed against a series of analytical and numerical solutions. Validation testing was performed against data from an aluminium propellant fire experiment performed by Sandia National Labs. Verification and validation testing results were favorable.

The monte-carlo radiation model that was written for use on CPU-based computers was translated into a GPU-specific language. Strong scaling analyses were performed on the Ember cluster and the Titan cluster for the CPU-radiation model and GPU-radiation model, respectively. The model demonstrated strong scaling to over 1,700 and 16,000 processing cores on Ember and Titan, respectively.

The developed RMCRT model is also quite versatile. Unlike the discrete ordinates method, RMCRT allows separation of the flux solutions from the divQ solutions. For example, if high-accuracy fluxes are needed in only a handful of locations, or a series of locations that lie in a single plain, the user can run with relatively few rays to produce basic radiative effects in the flame, then run with extremely fine angular discretization at the locations of interest. The model handles Cartesian, cubic or non-cubic cells in cubic or non-cubic domains with or without intrusion. RMCRT was validated against experimental data and indicated good agreement (see section 8.1).

RMCRT is amenable to coupling with non-gray properties such as the FSK model, which is currently being incorporated into the Arches and Wasatch components.

APPENDIX A

AMDAHL'S LAW

Algorithm execution time can be decomposed into two portions, the time required to perform the non-parallelizeable portion, x and a the time required to perform the parallelizeable portion, y. Therefore the speedup can be written mathematically as

$$speedup = \frac{x + y}{x + y/N_p}.$$

This expression is known as Amdahl's Law [28]. In ideal cases, where all portions of an algorithm are parallelizeable, x becomes zero and the speedup for any number of processors becomes N_p. However, for realistic algorithms, as the number of processors becomes large, y/N_p becomes negligible and the speedup approaches the limit of $(x + y)/x$.

APPENDIX B

BENCHMARK CASES

Each of the benchmark cases has gradients in the solution of $\nabla \cdot q$ and all but one of the cases has spatial gradients in at least one field property. The four benchmark cases are as follows:

Benchmark 1: Burns and Christon. This case is described by Burns and Christon (Burns 1997). The case describes a three dimensional cube 1m in length, with a constant temperature and a tri-linearly varying absorption coefficient. At the center of the domain, the absorption coefficient is $1m^{-1}$

Benchmark2: Michael Modest black parallel plates. This case is described by Modest in section 13 of his Radiation Heat Transfer book (Modest 2003). A gray, non-scattering medium with refractive index n=1 is contained between two infinitely parallel, gray plates. The medium is isothermal at temperature T_m, with constant absorption coefficient, κ. The two plates are both isothermal at temperature T_w, have the same gray-diffuse emissivity ϵ, and are spaced a distance L apart. For this problem, a known analytical solution exists for the radiative heat flux as well as the radiative flux divergence. The analytical solution for the flux divergence is as follows

$$\frac{dq}{d\tau} = \sigma(T_w^4 - T_m^4) * \frac{-2[E_2(\tau) + E_2(\tau_L - \tau)]}{1 + (1/\epsilon - 1)[1 - 2E_3(\tau_L)]},$$

where E_2 and E_3 are exponential functions, defined as follows

$$E_n(\tau) = \int_0^1 \mu^{n-2} e^{-\tau/\mu} d\mu,$$

and μ is defined as $cos(\theta)$.

See Modest figure 13.2 for the general behavior of the exponential functions. This case was used with the following values: $\epsilon = 1$, $\tau_L = 1$, $T_w = 1000K$, $T_m = 1500K$, and $L = 1$m. To model the infinite parallel plates, the four side walls were specified with emissivities of 0,

such that they had no emissive contribution, and they allowed the rays to reflect specularly. The two remaining walls had an emissivity of 1.

We found that this case required a greater number of rays to achieve comparable accuracy to the Burns and Christon benchmark. This is due to the fact that a ray striking a reflective wall gives a drastically different result than a ray striking the black plate. Therefore the angular discritization, which is a function of ray number, plays a more important role.

Benchmark 3: Modified Burns and Christon. This case is has the same properties of the Burns and Christon case with one modification–rather than a constant temperature throughout the domain, the temperature was set as follows

$$T = 1000 absk g$$

where $absk g$ is the absorption coefficient.

Benchmark 4: Modified Modest. This case is similar to the Michael Modest case, but with the following two changes.

$$absk g(z) = (\frac{z}{L})^2,$$

$$T = 1000 + 1000 absk g,$$

where z is the vertical coordinate, and L is the length of the domain. Therefore the absorption coefficient changes from 0 to 1 quadratically, and temperature varies from 1000K to 2000K quadratically.

With sufficient ray numbers, however, very good agreement between the numerical and analytical solutions is achievable.

APPENDIX C

NOTE TO THE USER

C.1 Source Weight

In cases other than benchmarks, the source weight should be set to -1. This is due to a difference in definitions between DOM and RMCRT. RMCRT defines divQ as $4\pi(I_{out} - G)$, as indicated in [**?**], whereas DOM defines divQ as the negative of this. To allow for the models to be used interchangeably, the divQ source weight in RMCRT should be set to -1.

Random Rays

In cases other than benchmarks, the seed in the RMCRT block should be set as random. This allows RMCRT to run as quickly as possible. Non-random seeds are used in benchmark cases where repeatable results are necessary.

APPENDIX D

IFRF F85Y4 OXYFLAM1B UPS INPUT FILE

The following input file was used to run the f85y4 oxyflame 1-B simulation.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- @version: -->
<Uintah_specification>
  <Meta>
    <title>RMCRT. Near-burner region of the IFRF f85y4 water-cooled furnace with medium-
momentum (type B) burner</title>
  </Meta>
  <SimulationComponent type="arches"/>
  <Solver type="hypre"/>
  <Time>
    <maxTime>1.0</maxTime>
    <initTime>0.0</initTime>
    <delt_min>0.0000001</delt_min>
    <delt_max>0.01</delt_max>
    <timestep_multiplier>0.2</timestep_multiplier>
  </Time>
  <DataArchiver>
    <filebase>Neg1Outletz.uda</filebase>
    <outputTimestepInterval>100</outputTimestepInterval>
    <save label="cellType"/>
    <save label="totalKineticEnergy"/>
    <save label="pressurePS"/>
    <save label="CCVelocity"/>
    <save label="viscosityCTS"/>
    <save label="div_q"/>
    <save label="scalar_var"/>
    <save label="mixture_fraction"/>
    <save label="mixture_fraction_2"/>
    <save label="heat_loss"/>
    <save label="density"/>
    <save label="temperature" table_lookup="true"/>
    <save label="C*H4" table_lookup="true"/>
    <save label="CO2" table_lookup="true"/>
    <save label="velocityDivergence"/>
    <save label="continuityResidual"/>
    <save label="abskg"/>
    <save label="soot"/>
    <save label="boundFlux"/>
    <save label="VRFlux"/>
    <checkpoint cycle="2" timestepInterval="500"/>
    <compression>gzip</compression>
  </DataArchiver>
  <Grid>
    <Level>
      <Box label="1">
        <lower>[-0.15,-0.15,0.0 ]</lower>
        <upper>[ 0.15, 0.15,0.3]</upper>
        <resolution>[200,200,200]</resolution>
        <extraCells>[1,1,1]</extraCells>
```

```
    <patches>[12,12,12]</patches>
   </Box>
   <periodic>[0, 0, 0]</periodic>
  </Level>
  <BoundaryConditions>
   <Face side="x-">
    <BCType id="0" label="x- pressure" var="OutletBC">
     <value>0.0</value>
    </BCType>
    <BCType id="0" var="Neumann" label="mixture_fraction">
     <value>0.0</value>
    </BCType>
    <BCType id="0" var="Neumann" label="mixture_fraction_2">
     <value>0.0</value>
    </BCType>
    <BCType id="0" var="Neumann" label="heat_loss">
     <value>0.0</value>
    </BCType>
    <BCType id="0" var="Neumann" label="enthalpy">
     <value>0.0</value>
    </BCType>
    <!-- for RMCRT-->
    <BCType id="all" label="temperature" var="Neumann">
     <value> 0</value>
    </BCType>
    <BCType id="all" label="abskg" var="Dirichlet">
     <value> 1</value>
    </BCType>
   </Face>
   <Face side="x+">
    <BCType id="0" label="x+ pressure" var="OutletBC">
     <value>0.0</value>
    </BCType>
    <BCType id="0" var="Neumann" label="mixture_fraction">
     <value>0.0</value>
    </BCType>
    <BCType id="0" var="Neumann" label="mixture_fraction_2">
     <value>0.0</value>
    </BCType>
    <BCType id="0" var="Neumann" label="heat_loss">
     <value>0.0</value>
    </BCType>
    <BCType id="0" var="Neumann" label="enthalpy">
     <value>0.0</value>
    </BCType>
    <!-- for RMCRT-->
    <BCType id="all" label="temperature" var="Neumann">
     <value> 0</value>
    </BCType>
```

```
  <BCType id="all" label="abskg" var="Dirichlet">
    <value> 1</value>
  </BCType>
 </Face>
 <Face side="y-">
  <BCType id="0" label="y- pressure" var="OutletBC">
    <value>0.0</value>
  </BCType>
  <BCType id="0" var="Neumann" label="mixture_fraction">
    <value>0.0</value>
  </BCType>
  <BCType id="0" var="Neumann" label="mixture_fraction_2">
    <value>0.0</value>
  </BCType>
  <BCType id="0" var="Neumann" label="heat_loss">
    <value>0.0</value>
  </BCType>
  <BCType id="0" var="Neumann" label="enthalpy">
    <value>0.0</value>
  </BCType>
  <!-- for RMCRT-->
  <BCType id="all" label="temperature" var="Neumann">
    <value> 0</value>
  </BCType>
  <BCType id="all" label="abskg" var="Dirichlet">
    <value> 1</value>
  </BCType>
 </Face>
 <Face side="y+">
  <BCType id="0" label="y+ pressure" var="OutletBC">
    <value>0.0</value>
  </BCType>
  <BCType id="0" var="Neumann" label="mixture_fraction">
    <value>0.0</value>
  </BCType>
  <BCType id="0" var="Neumann" label="mixture_fraction_2">
    <value>0.0</value>
  </BCType>
  <BCType id="0" var="Neumann" label="heat_loss">
    <value>0.0</value>
  </BCType>
  <BCType id="0" var="Neumann" label="enthalpy">
    <value>0.0</value>
  </BCType>
  <!-- for RMCRT-->
  <BCType id="all" label="temperature" var="Neumann">
    <value> 0</value>
  </BCType>
  <BCType id="all" label="abskg" var="Dirichlet">
```

```
          <value> 1</value>
        </BCType>
      </Face>
      <!-- z- Face Wall-->
      <Face side="z-" name="zMinus wall">
        <!-- Momentum BCs -->
        <BCType id="all" label="z- wall" var="WallBC">
          <value>0.0</value>
        </BCType>
        <BCType id="0" var="Neumann" label="mixture_fraction">
          <value>0.0</value>
        </BCType>
        <BCType id="0" var="Neumann" label="mixture_fraction_2">
          <value>0.0</value>
        </BCType>
        <BCType id="0" var="Neumann" label="heat_loss">
          <value>0.0</value>
        </BCType>
        <BCType id="0" var="Neumann" label="enthalpy">
          <value>0.0</value>
        </BCType>
        <!-- for RMCRT-->
        <BCType id="all" label="temperature" var="Dirichlet">
          <value> 400</value>
        </BCType>
        <BCType id="all" label="abskg" var="Dirichlet">
          <value> 1</value>
        </BCType>
      </Face>
      <!--__________________________________-->
      <!--  Annular Ring -->
      <Face annulus="z-" origin="0.0 0.0 0.0" inner_radius="0.0165" outer_radius="0.0225"
name="annulus">
        <!-- Momentum BCs -->
        <BCType id="all" label="inlet" var="VelocityInlet">
          <value>[0.0,0.0,77.1]</value>
        </BCType>
        <BCType id="0" var="Dirichlet" label="mixture_fraction">
          <!--ILH correct?  -->
          <value>0.0</value>
        </BCType>
        <BCType id="0" var="Dirichlet" label="mixture_fraction_2">
          <value>0.0</value>
        </BCType>
        <BCType id="0" var="Dirichlet" label="heat_loss">
          <value>0.0</value>
        </BCType>
        <BCType id="0" var="Neumann" label="enthalpy">
          <value>0.0</value>
```

```
  </BCType>
  <!-- for RMCRT-->
  <BCType id="all" label="temperature" var="Dirichlet">
    <value> 298</value>
  </BCType>
  <BCType id="all" label="abskg" var="Dirichlet">
    <value> 1</value>
  </BCType>
 </Face>
 <!--__________________________________ -->
 <!-- Primary Flow Inlet -->
 <Face circle="z-" origin="0.0 0.0 0.0" radius="0.0105" name="primary">
  <!-- Momentum BCs -->
  <BCType id="all" label="inlet" var="VelocityInlet">
    <value>[0.0,0.0,80.1]</value>
  </BCType>
  <BCType id="0" var="Dirichlet" label="mixture_fraction">
    <!--ILH correct?  -->
    <value>0.0</value>
  </BCType>
  <BCType id="0" var="Dirichlet" label="mixture_fraction_2">
    <value>1.0</value>
  </BCType>
  <BCType id="0" var="Dirichlet" label="heat_loss">
    <value>0.0</value>
  </BCType>
  <BCType id="0" var="Neumann" label="enthalpy">
    <value>0.0</value>
  </BCType>
  <!-- for RMCRT-->
  <BCType id="all" label="temperature" var="Dirichlet">
    <value> 298</value>
  </BCType>
  <BCType id="all" label="abskg" var="Dirichlet">
    <value> 1</value>
  </BCType>
 </Face>
 <Face side="z+">
  <BCType id="0" var="OutletBC" label="the outlet">
    <value>0.0</value>
  </BCType>
  <BCType id="0" var="Neumann" label="mixture_fraction">
    <value>0.0</value>
  </BCType>
  <BCType id="0" var="Neumann" label="mixture_fraction_2">
    <value>0.0</value>
  </BCType>
  <BCType id="0" var="Neumann" label="heat_loss">
    <value>0.0</value>
```

```
      </BCType>
      <BCType id="0" var="Neumann" label="enthalpy">
        <value>0.0</value>
      </BCType>
      <!-- for RMCRT-->
      <BCType id="all" label="temperature" var="Neumann">
        <value> 0</value>
      </BCType>
      <BCType id="all" label="abskg" var="Dirichlet">
        <value> 1</value>
      </BCType>
    </Face>
  </BoundaryConditions>
 </Grid>
 <PhysicalConstants>
  <gravity>[-9.8,0,0]</gravity>
  <reference_point>[-1,-1,-1]</reference_point>
  <viscosity>0.000020</viscosity>
 </PhysicalConstants>
 <CFD>
  <!-- ARCHES specification -->
  <ARCHES>
   <EfficiencyCalculator>
     <calculator type="combustion_efficiency" label="comb_eff">
       <mixture_fraction mf_label_1="mixture_fraction" mf_label_2="mixture_fraction_2"
N="2"/>
       <phi_label>mixture_fraction_2</phi_label>
       <phi_at_feq1>1.0</phi_at_feq1>
     </calculator>
   </EfficiencyCalculator>
   <TimeIntegrator>
     <ExplicitIntegrator order="second"/>
   </TimeIntegrator>
   <TransportEqns>
     <Eqn label="mixture_fraction" type="CCscalar">
      <doDiff>true</doDiff>
      <doConv>true</doConv>
      <conv_scheme>upwind</conv_scheme>
      <determines_properties/>
      <initialization type="constant">
        <constant>0.0</constant>
      </initialization>
      <Clipping>
        <low>0.0</low>
        <high>1.0</high>
        <tol>1e-10</tol>
      </Clipping>
      <src label="pos_source">
        <weight>1</weight>
```

```
  </src>
   <turbulentPrandtlNumber>0.40000000000000002</turbulentPrandtlNumber>
   <scaling_const>1</scaling_const>
 </Eqn>
 <Eqn label="mixture_fraction_2" type="CCscalar">
   <doDiff>true</doDiff>
   <doConv>true</doConv>
   <conv_scheme>upwind</conv_scheme>
   <determines_properties/>
   <initialization type="constant">
     <constant>0.0</constant>
   </initialization>
   <Clipping>
     <low>0.0</low>
     <high>1.0</high>
     <tol>1e-10</tol>
   </Clipping>
   <src label="pos_source">
     <weight>-1.0</weight>
  </src>
   <turbulentPrandtlNumber>0.40000000000000002</turbulentPrandtlNumber>
   <scaling_const>1</scaling_const>
 </Eqn>
 <Eqn label="enthalpy" type="CCscalar">
   <doDiff>true</doDiff>
   <doConv>true</doConv>
   <conv_scheme>upwind</conv_scheme>
   <determines_properties/>
   <initialization type="tabulated">
     <depend_varname>adiabaticenthalpy</depend_varname>
   </initialization>
   <src label="div_q">
     <weight>-1</weight>
  </src>
   <turbulentPrandtlNumber>0.40000000000000002</turbulentPrandtlNumber>
   <scaling_const>1</scaling_const>
 </Eqn>
 <Sources>
   <src label="pos_source" type="westbrook_dryer">
     <A>4e9</A>
     <E_R>24358</E_R>
     <X>1</X>
     <Y>4</Y>
     <m>-0.3</m>
     <n>1.3</n>
      <stoich_fuel_O2_massratio>0.25</stoich_fuel_O2_massratio>
     <fuel_mass_fraction>1.0</fuel_mass_fraction>
     <cstar_fraction_label>C*H4</cstar_fraction_label>
     <fp_label>mixture_fraction_2</fp_label>
```

```
  <eta_label>mixture_fraction</eta_label>
 <flammability_limit>
   <const_diluent>0.0</const_diluent>
   <lower slope="0.046" intercept="0.015"/>
   <upper slope="-0.286" intercept="0.089"/>
 </flammability_limit>
 <hot_spot>
   <geom_object>
     <difference label="pilot">
      <cylinder>
        <bottom>[0,0,0.01]</bottom>
        <top>[0,0,0.013]</top>
        <radius>0.018</radius>
         <cylinder_end>false</cylinder_end>
         <axisymmetric_end>false</axisymmetric_end>
         <axisymmetric_side>false</axisymmetric_side>
      </cylinder>
      <cylinder>
        <bottom>[0,0,0.01]</bottom>
        <top>[0,0,0.013]</top>
        <radius>0.012</radius>
         <cylinder_end>false</cylinder_end>
         <axisymmetric_end>false</axisymmetric_end>
         <axisymmetric_side>false</axisymmetric_side>
      </cylinder>
     </difference>
   </geom_object>
   <start_time>0.0</start_time>
   <stop_time>1.0</stop_time>
   <temperature>2000</temperature>
 </hot_spot>
  <temperature_label>temperature</temperature_label>
  <density_label>density</density_label>
  <o2_label>O2</o2_label>
</src>
 <!--       <src label="div_q" type="do_radiation">
  <calc_frequency>10</calc_frequency>
  <calc_on_all_RKsteps>false</calc_on_all_RKsteps>
  <soot_label>soot</soot_label>
  <DORadiationModel>
   <opl>3.0</opl>
   <LinearSolver type="hypre">
     <res_tol>1.0e-10</res_tol>
     <ksptype>gmres</ksptype>
     <pctype>jacobi</pctype>
     <max_iter>275</max_iter>
   </LinearSolver>
  </DORadiationModel>
</src>
```

```
-->
        <src label="div_q" type="rmcrt_radiation">
          <calc_frequency>10</calc_frequency>
          <RMCRT>
            <randomSeed>        true      </randomSeed>
            <nDivQRays>         16        </nDivQRays>
            <Threshold>         0.05      </Threshold>
            <StefanBoltzmann>   5.67051e-8 </StefanBoltzmann>
            <solveBoundaryFlux> true      </solveBoundaryFlux>
            <nFluxRays>         100       </nFluxRays>
            <allowReflect>      false     </allowReflect>
            <property_calculator type="hottel_sarofim">
            </property_calculator>
            <ignore_BC_bulletproofing> false </ignore_BC_bulletproofing>
            <benchmark>0</benchmark>
            <CCRays>false</CCRays>
            <VirtRadiometer>true</VirtRadiometer>
            <VRViewAngle>180</VRViewAngle>
            <VROrientation>[-1, 0, 0]</VROrientation>
            <VRLocationsMin>[0, 100, 0]</VRLocationsMin>
            <VRLocationsMax>[199, 100, 199]</VRLocationsMax>
            <nRadRays>1000</nRadRays>
            <sigmaScat>0</sigmaScat>
            <abskgBench4>1</abskgBench4>
            <solveDivQ>true</solveDivQ>
            <applyFilter>false</applyFilter>
          </RMCRT>
          <calc_on_all_RKsteps>false</calc_on_all_RKsteps>
          <abskp_label>abskp</abskp_label>
          <psize_label>length</psize_label>
          <ptemperature_label>temperature</ptemperature_label>
        </src>
      </Sources>
    </TransportEqns>
    <Turbulence model="compdynamicprocedure">
      <variance_coefficient>0.1</variance_coefficient>
      <turbulentPrandtlNumber>0.4</turbulentPrandtlNumber>
      <dynamicScalarModel>false</dynamicScalarModel>
     <filter_cs_squared>false</filter_cs_squared>
    </Turbulence>
    <Properties>
      <ClassicTable>
        <inputfile>oxyflam_gas.mix</inputfile>
        <rcce eta_label="mixture_fraction" fp_label="mixture_fraction_2"
hl_label="heat_loss"/>
        <cold_flow>false</cold_flow>
        <temperature_label_name>temperature</temperature_label_name>
      </ClassicTable>
     <filter_drhodt>false</filter_drhodt>
```

```
 <first_order_drhodt>true</first_order_drhodt>
  <inverse_density_average>false</inverse_density_average>
  <mixture_fraction_label>scalarSP</mixture_fraction_label>
</Properties>
<PropertyModels>
  <model type="heat_loss" label="heat_loss">
    <initialization type="constant">
      <constant>0.0</constant>
    </initialization>
    <enthalpy_label>enthalpy</enthalpy_label>
    <sensible_enthalpy_label>sensibleenthalpy</sensible_enthalpy_label>
    <adiabatic_enthalpy_label>adiabaticenthalpy</adiabatic_enthalpy_label>
    <use_Ha_lookup>false</use_Ha_lookup>
  </model>
  <model type="empirical_soot" label="soot">
    <initialization type="constant">
      <constant>0.0</constant>
    </initialization>
    <carbon_content_fuel>0.75</carbon_content_fuel>
    <carbon_content_ox>0.0</carbon_content_ox>
    <E_st>0.08</E_st>
    <temperature_label>temperature</temperature_label>
    <mixture_fraction_label>mixture_fraction</mixture_fraction_label>
    <density_label>density</density_label>
    <absorption_label>absorpIN</absorption_label>
    <soot_density>1950</soot_density>
    <E_cr>1</E_cr>
    <E_inf>2</E_inf>
    <C1>0.10000000000000001</C1>
  </model>
  <model type="scalsim_variance" label="scalar_var">
    <initialization type="constant">
      <constant>0.0</constant>
    </initialization>
    <mixture_fraction_label>mixture_fraction</mixture_fraction_label>
    <density_label>density</density_label>
    <variance_coefficient>0.14</variance_coefficient>
    <!-- Warning: not a good default value -->
  </model>
</PropertyModels>
<BoundaryConditions>
  <suppress_corner_recirculation/>
  <use_new_bcs/>
  <wall_csmag>0</wall_csmag>
</BoundaryConditions>
<ExplicitSolver>
  <initial_dt>0.05</initial_dt>
  <variable_dt>true</variable_dt>
  <PressureSolver>
```

```
        <Parameters>
          <tolerance>     1.0e-10</tolerance>
          <solver>        cg    </solver>
          <preconditioner>pfmg  </preconditioner>
          <maxiterations> 75    </maxiterations>
          <setupFrequency>1     </setupFrequency>
          <npre>1</npre>
          <npost>1</npost>
          <skip>0</skip>
          <jump>0</jump>
          <logging>0</logging>
          <relax_type>1</relax_type>
        </Parameters>
         <normalize_pressure>false</normalize_pressure>
         <do_only_last_projection>false</do_only_last_projection>
       </PressureSolver>
       <MomentumSolver>
         <convection_scheme>upwind</convection_scheme>
        <filter_divergence_constraint>false</filter_divergence_constraint>
       </MomentumSolver>
       <scalarUnderflowCheck>false</scalarUnderflowCheck>
       <extraProjection>false</extraProjection>
       <turbModelCalcFreq>1</turbModelCalcFreq>
       <turbModelCalcForAllRKSteps>true</turbModelCalcForAllRKSteps>
       <restartOnNegativeDensityGuess>false</restartOnNegativeDensityGuess>
       <NoisyDensityGuess>false</NoisyDensityGuess>
       <kineticEnergy_fromFC>false</kineticEnergy_fromFC>
       <maxDensityLag>0</maxDensityLag>
     </ExplicitSolver>
     <turnonMixedModel>false</turnonMixedModel>
     <recompileTaskgraph>false</recompileTaskgraph>
   </ARCHES>
  </CFD>
</Uintah_specification>
```

REFERENCES

[1] Burns, S., and Christon, M., 1997. "Spatial domain-based parallelism in large-scale, participating-media, radiative transport applications". *Numerical Heat Transfer, Part B: Fundamentals,* ***31***(4), pp. 401–421.

[2] Tieszen, S., 2001. "On the fluid mechanics of fires 1". *Annual review of fluid mechanics,* ***33***(1), pp. 67–92.

[3] Modest, M. F., 2005. "Multiscale modeling of turbulence, radiation, and combustion interactions in turbulent flames". *Int. J. Multiscale Comput. Eng,* ***3***(1), p. 85.

[4] Hall, R. J., and Vranos, A., 1994. "Efficient calculations of gas radiation from turbulent flames". *International journal of heat and mass transfer,* ***37***(17), pp. 2745–2750.

[5] Wang, A., Modest, M. F., Haworth, D. C., and Wang, L., 2008. "Monte carlo simulation of radiative heat transfer and turbulence interactions in methane/air jet flames". *Journal of Quantitative Spectroscopy and Radiative Transfer,* ***109***(2), pp. 269–279.

[6] Song, T., and Viskanta, R., 1987. "Interaction of radiation with turbulence-application to a combustion system". *J. Thermophy. Heat Transfer;(United States),* ***1***.

[7] Soufiani, A., Mignon, P., and Taine, J., 1990. "Radiation-turbulence interaction in channel flows of infrared active gases". In Proceedings of the ninth international heat transfer conference, Vol. 6, pp. 403–408.

[8] Coelho, P., 2007. "Numerical simulation of the interaction between turbulence and radiation in reactive flows". *Progress in energy and combustion science,* ***33***(4), pp. 311–383.

[9] Gonçalves dos Santos, R., Lecanu, M., Ducruix, S., Gicquel, O., Iacona, E., and Veynante, D., 2008. "Coupled large eddy simulations of turbulent combustion and radiative heat transfer". *Combustion and Flame,* ***152***(3), pp. 387–400.

[10] Wu, Y., Haworth, D., Modest, M., and Cuenot, B., 2005. "Direct numerical simulation of turbulence/radiation interaction in premixed combustion systems". *Proceedings of the Combustion Institute,* ***30***(1), pp. 639–646.

[11] Deshmukh, K., Haworth, D., and Modest, M., 2007. "Direct numerical simulation of turbulence–radiation interactions in homogeneous nonpremixed combustion systems". *Proceedings of the Combustion Institute,* ***31***(1), pp. 1641–1648.

[12] Coats, C., 1996. "Coherent structures in combustion". *Progress in Energy and Combustion Science,* ***22***(5), pp. 427–509.

[13] Snegirev, A. Y., 2004. "Statistical modeling of thermal radiation transfer in buoyant turbulent diffusion flames". *Combustion and flame,* ***136***(1), pp. 51–71.

[14] Sacadura, J.-F., 2005. "Radiative heat transfer in fire safety science". *Journal of Quantitative Spectroscopy and Radiative Transfer,* ***93***(1), pp. 5–24.

[15] Siegel, R., 2001. *Thermal radiation heat transfer*. Taylor & Francis.

[16] Smith, P., Thornock, J., Hinckley, D., and Hradisky, M., 2011. "Large eddy simulation of industrial flares". In Proceedings of the 2011 companion on High Performance Computing Networking, Storage and Analysis Companion, ACM, pp. 137–138.

[17] Krishnamoorthy, G., 2006. "Predicting radiative heat transfer in parallel computations of combustion". PhD thesis, THE UNIVERSITY OF UTAH.

[18] Lockwood, F., and Shah, N., 1981. "A new radiation solution method for incorporation in general combustion prediction procedures". In Symposium (International) on Combustion, Vol. 18, Elsevier, pp. 1405–1414.

[19] Raithby, G., and Chui, E., 1990. "A finite-volume method for predicting a radiant heat transfer in enclosures with participating media". *Journal of Heat Transfer (Transactions of the ASME (American Society of Mechanical Engineers), Series C);(United States),* ***112***(2).

[20] Coelho, P., Teerling, O., and Roekaerts, D., 2003. "Spectral radiative effects and turbulence/radiation interaction in a non-luminous turbulent jet diffusion flame". *Combustion and Flame,* ***133***(1), pp. 75–91.

[21] Chai, J., Lee, H., and Patankar, S., 1993. "Ray effect and false scattering in the discrete ordinates method". *Numerical Heat Transfer, Part B Fundamentals,* ***24***(4), pp. 373–389.

[22] Snegirev, A. Y., and Isaev, S., 2007. "Turbulent combustion and thermal radiation in a massive fire". *Advanced Combustion and Aerothermal Technologies*, pp. 197–209.

[23] Snegirev, A. Y. "Large-eddy simulations of buoyant turbulent diffusion flames exposed to crosswinds and circulating flows". In European Combustion Meeting.

[24] Zhang, J., Gicquel, O., Veynante, D., and Taine, J., 2009. "Monte carlo method of radiative transfer applied to a turbulent flame modeling with les". *Comptes Rendus Mécanique,* ***337***(6-7), pp. 539–549.

[25] for Computational Sciences, N. C., 2012. Jaguar. On the WWW, March. URL http://www.nccs.gov/computing-resources/jaguar.

[26] Kindratenko, V., and Trancoso, P., 2011. "Trends in high-performance computing". *Computing in Science & Engineering,* ***13***(3), pp. 92–95.

[27] Stowell, M. L., Fasenfest, B. J., and White, D. A., 2008. "Investigation of radar propagation in buildings: a 10-billion element cartesian-mesh fetd simulation". *Antennas and Propagation, IEEE Transactions on,* ***56***(8), pp. 2241–2250.

[28] Martin, W., Majumdar, A., Rathkopf, J., and Litvin, M., 1993. Experiences with different parallel programming paradigms for monte carlo particle transport leads to a portable toolkit for parallel monte carlo. Tech. rep., Lawrence Livermore National Lab., CA (United States).

[29] Wise, J., and Abel, T., 2010. "enzo+ moray: radiation hydrodynamics adaptive mesh refinement simulations with adaptive ray tracing". *Monthly Notices of the Royal Astronomical Society.*

[30] Abel, T., and Wandelt, B., 2002. "Adaptive ray tracing for radiative transfer around point sources". *Monthly Notices of the Royal Astronomical Society,* ***330***(3), pp. L53–L56.

[31] Kuiper, R., Klahr, H., Dullemond, C., Kley, W., and Henning, T., 2010. "Fast and accurate frequency-dependent radiation transport for hydrodynamics simulations in massive star formation". *Arxiv preprint arXiv:1001.3301.*

[32] Marakis, J., Chamico, J., Brenner, G., and Durst, F., 2001. "Parallel ray tracing for radiative heat transfer: Application in a distributed computing environment". *International Journal of Numerical Methods for Heat &# 38; Fluid Flow,* ***11***(7), pp. 663–681.

[33] Cleveland, M., 2011. "Radiative heat transfer in combustion applications: parallel efficiencies of two gas models, turbulent radiation interactions in particulate laden flows, and coarse mesh finite difference acceleration for improved temporal accuracy".

[34] Marrs, R., Moss, W., and Whitlock, B., 2007. "Thermal radiation from nuclear detonations in urban environments". *Livermore, CA: Lawrence Livermore National Laboratory Report UCRLTR-231593.*

[35] Veljkovic, I., and Plassmann, P., 2005. "Scalable photon monte carlo algorithms and software for the solution of radiative heat transfer problems". *High Performance Computing and Communications*, pp. 928–937.

[36] Wang, A., 2007. "Investigation of turbulence–radiation interactions in". PhD thesis, The Pennsylvania State University.

[37] Mehta, R., 2008. *Detailed modeling of soot formation and turbulence-radiation interactions in turbulent jet flames.* ProQuest.

[38] Rijkhorst, E., Plewa, T., Dubey, A., and Mellema, G., 2005. "Hybrid characteristics: 3d radiative transfer for parallel adaptive mesh refinement hydrodynamics". *Arxiv preprint astro-ph/0505213.*

[39] Viswanath, K., Veljkovic, I., and Plassmann, P., 2008. "Parallel load balancing heuristics for radiative heat transfer calculations".

[40] Sawetprawichkul, A., Hsu, P., and Mitra, K., 2002. "Parallel computing of three-dimensional monte carlo simulation of transient radiative transfer in participating media". In Proceedings of the Eighth American Institute of Aeronautics and Astronautics/American Society of Mechanical Engineers Joint Thermophysics and Heat Transfer Conference, St Louis, MI, pp. 1–10.

[41] Farmer, J., and Howell, J., 1998. "Comparison of Monte Carlo strategies for radiative transfer in participating media". *Advances in heat transfer,* ***31***, pp. 333–429.

[42] Wendt, K. A., Drut, J. E., and Lähde, T. A., 2011. "Toward large-scale hybrid monte carlo simulations of the hubbard model on graphics processing units". *Computer Physics Communications,* ***182***(8), pp. 1651–1656.

[43] Gentile, N., 2009. "Implicit monte carlo radiation transport in multi-physics simulations". *American Nuclear Society.*

[44] Hottel, H., and Cohen, E., 1968. *Radiative transfer.* McGraw Hill.

[45] Sun, X., 2009. "Reverse Monte Carlo ray-tracing for radiative heat transfer in combustion systems".

[46] Howell, J., 1998. "The Monte Carlo method in radiative heat transfer". *TRANSACTIONS-AMERICAN SOCIETY OF MECHANICAL ENGINEERS JOURNAL OF HEAT TRANSFER,* ***120***, pp. 547–560.

[47] Howell, J., and Perlmutter, M., 1964. "Radiant Transfer Through a Gray Gas Between Concentric Cylinders Using Monte Carlo". *Journal of Heat Transfer,* ***86***, pp. 169–179.

[48] Howell, J., and Perlmutter, M., 1964. "Monte Carlo solution of thermal transfer in a nongrey nonisothermal gas with temperature dependent properties". *AlChE Jouranl,* ***10***(4), pp. 562–567.

[49] Fleck Jr, J., 1961. The calculation of nonlinear radiation transport by a monte carlo method. Tech. rep., Lawrence Radiation Lab., Univ. of California, Livermore.

[50] Fleck, J., 1961. "The calculation of nonlinear radiation transport by a monte carlo method: statistical physics". *Methods in Computational Physics,* ***1***, pp. 43–65.

[51] Lee, M., Redner, R., and Uselton, S., 1985. "Statistically optimized sampling for distributed ray tracing". In Proceedings of the 12th annual conference on Computer graphics and interactive techniques, ACM, pp. 61–68.

[52] Pegoraro, V., Wald, I., and Parker, S., 2008. "Sequential Monte Carlo Adaptation in Low-Anisotropy Participating Media". In Computer Graphics Forum, Vol. 27, Wiley Online Library, pp. 1097–1104.

[53] Pegoraro, V., Brownlee, C., Shirley, P., and Parker, S., 2008. "Towards interactive global illumination effects via sequential Monte Carlo adaptation". In Interactive Ray Tracing, 2008. RT 2008. IEEE Symposium on, IEEE, pp. 107–114.

[54] Jensen, K., Ripoll, J., Wray, A., Joseph, D., and El Hafi, M., 2007. "On various modeling approaches to radiative heat transfer in pool fires". *Combustion and flame,* ***148***(4), pp. 263–279.

[55] Jeans, J., 1917. "Stars, gaseous, radiative transfer of energy". *Monthly Notices of the Royal Astronomical Society,* ***78***, pp. 28–36.

[56] Mengüç, M., and Viskanta, R., 1985. "Radiative transfer in three-dimensional rectangular enclosures containing inhomogeneous, anisotropically scattering media". *Journal of Quantitative Spectroscopy and Radiative Transfer,* ***33***(6), pp. 533–549.

[57] Chandrasekhar, S., 1960. *Radiative transfer.* Dover Pubns.

[58] Antal, M., and Lee, C., 1976. "Charged particle mass and energy transport in a thermonuclear plasma". *Journal of Computational Physics,* ***20***(3), pp. 298–312.

[59] Jamaluddin, A., and Smith, P., 1988. "Predicting radiative transfer in axisymmetric cylindrical enclosures using the discrete ordinates method". *Combustion science and technology,* ***62***(4-6), pp. 173–186.

[60] Fiveland, W., 1984. "Discrete-ordinates solutions of the radiative transport equation for rectangular enclosures". *Journal of Heat Transfer,* ***106***, p. 699.

[61] Selcuk, N., and Kayakol, N., 1995. "Evaluations of discrete transfer model for radiative transfer in combustors". *ICHMT DIGITAL LIBRARY ONLINE,* ***7***.

[62] Abraham, J., and Magi, V., 1997. "Application of the discrete ordinates method to compute radiant heat loss in a diesel engine". *Numerical Heat Transfer, Part A Applications,* ***31***(6), pp. 597–610.

[63] Liou, B., and Wu, C., 1996. "Composite discrete-ordinate solutions for radiative transfer in a two-layer medium with fresnel interfaces". *Numerical Heat Transfer, Part A Applications,* ***30***(7), pp. 739–751.

[64] Alan Humphrey, Qingyu Meng, M. B. T. H., 2012. "Radiation modeling using the uintah heterogeneous cpu/gpu runtime system".

[65] Kourganoff, V., 1963. *Basic methods in transfer problems.* Dover Publications.

[66] Kourganoff, V., 1958. *Neutron Transport Theory.* Oxford University Press.

[67] Murray, R., 1957. *Nuclear Reactor Physics.* Prentice Hall.

[68] Hottel, H., and Cohen, E., 1958. "Radiant heat exchange in a gas-filled enclosure: Allowance for nonuniformity of gas temperature". *AIChE Journal,* ***4***(1), pp. 3–14.

[69] Carvalho, M., Farias, T., and Fontes, P., 1991. "Predicting radiative heat transfer in absorbing, emitting, and scattering media using the discrete transfer method". *Fundamentals of radiation heat transfer,* ***160***, pp. 17–26.

[70] Rousse, D., 2000. "Numerical predictions of two-dimensional conduction, convection, and radiation heat transfer. i. formulation". *International journal of thermal sciences,* ***39***(3), pp. 315–331.

[71] Meng, F., McKenty, F., and Camarero, R., 1993. "Radiative heat transfer by the discrete transfer method using an unstructured mesh". *ASME-PUBLICATIONS-HTD,* ***244***, pp. 55–55.

[72] Feldheim, V., and Lybaert, P., 2004. "Solution of radiative heat transfer problems with the discrete transfer method applied to triangular meshes". *Journal of computational and applied mathematics,* ***168***(1), pp. 179–190.

[73] LI, H., and ZHOU, H., 2003. "Luji-dong, zheng chu-guang (school of energy and power engineering, huazhong university of science & technology, wuhan 430074, china); an improved discrete transfer method for radiative heat transfer in furnaces [j]". *Proceedings of the Csee,* ***4***.

[74] Almasi, G., 1986. "Research in highly parallel computer systems". *IEEE Electro Technology Review,* ***2***.

[75] Govaerts, Y., and Verstraete, M., 2002. "Raytran: A Monte Carlo ray-tracing model to compute light scattering in three-dimensional heterogeneous media". *Geoscience and Remote Sensing, IEEE Transactions on,* ***36***(2), pp. 493–505.

[76] Heirich, A., and Arvo, J., 1998. "A competitive analysis of load balancing strategies for parallel ray tracing". *The Journal of Supercomputing,* ***12***(1), pp. 57–68.

[77] dos Santos, A., Teixeira, J., de Farias, T., Teichrieb, V., and Kelner, J., 2009. "kd-tree traversal implementations for ray tracing on massive multiprocessors: A comparative study". In Computer Architecture and High Performance Computing, 2009. SBAC-PAD'09. 21st International Symposium on, IEEE, pp. 41–48.

[78] Despres, P., Rinkel, J., Hasegawa, B., and Prevrhal, S., 2008. "Stream processors: a new platform for monte carlo calculations". In Journal of Physics: Conference Series, Vol. 102, IOP Publishing, p. 012007.

[79] Spafford, K., Meredith, J. S., and Vetter., J. S., 2011. "Quantifying numa and contention effects in multi-gpu systems". *Proceedings of the Fourth Workshop on General Purpose Processing on Graphics Processing Units, GPGPU*(11), p. 17.

[80] J.S. Vetter, R. Glassbrook, J. D. K. S. B. L. S. M. J. M. J. R. P. R. K. S., and Yalamanchili, S., 2013. Keeneland web page. On the WWW, March. URL http://keeneland.gatech.edu/.

[81] J. D. de St. Germain, J. McCorquodale, S. G. P., and Johnson, C. R., 2000. "Uintah: A massively parallel problem solving environment". *Ninth IEEE International Symposium on High Performance and Distributed Computing*, p. 3341.

[82] Berzins, M., 2012. "Status of release of the uintah computational framework. technical report uusci-2012-001". *Scientific Computing and Imaging Institute.*

[83] P. J. Smith, R.Rawat, J. S. S. K. S. B., and Violi, A., 2003. "Large eddy simulation of accidental fires using massively parallel computers". *18th AIAA Computational Fluid Dynamics Conference.*

[84] J.Spinti, J. Thornock, E. E. P. S., and Sarofim, A., 2008. "Heat transfer to objects in pool fires, in transport phenomena in fires". *Transport Phenomena in Fires, Southampton.*

[85] J. Schmidt, J. Thornock, J. S., and Berzins, M., 2012. "Large scale parallel solution of incompressible flow problems using uintah and hypre. technical report uusci-2012-002". *Scientific Computing and Imaging Institute.*

[86] I. Hunsaker, J. Thornock, T. H. P. J. S., 2013. "Massively-parallelized reciprocal monte-carlo ray tracing for radiative transfer coupled with turbulent les combustion simulations". *8th US National Combustion Meeting.*

[87] Modest, M., 2003. *Radiative heat transfer.* Academic Pr.

[88] Tessé, L., Dupoirieux, F., and Taine, J., 2004. "Monte carlo modeling of radiative transfer in a turbulent sooty flame". *International journal of heat and mass transfer,* ***47***(3), pp. 555–572.

[89] Tessé, L., Dupoirieux, F., Zamuner, B., and Taine, J., 2002. "Radiative transfer in real gases using reciprocal and forward monte carlo methods and a correlated-k approach". *International Journal of Heat and Mass Transfer,* ***45***(13), pp. 2797–2814.

[90] Zhang, Y., Gicquel, O., and Taine, J., 2012. "Optimized emission-based reciprocity monte carlo method to speed up computation in complex systems". *International Journal of Heat and Mass Transfer.*

[91] Walters, D., and Buckius, R., 1992. "Rigorous development for radiation heat transfer in nonhomogeneous absorbing, emitting and scattering media". *International Journal of Heat and Mass Transfer,* ***35***(12), pp. 3323–3333.

[92] Matsumoto, M., and Nishimura, T., 1998. "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator". *ACM Transactions on Modeling and Computer Simulation (TOMACS),* ***8***(1), pp. 3–30.

[93] Amanatides, J., and Woo, A., 1987. "A fast voxel traversal algorithm for ray tracing". In Eurographics, Vol. 87, Citeseer, p. 10.

[94] Parker, S., Guilkey, J., and Harman, T., 2006. "A component-based parallel infrastructure for the simulation of fluid–structure interaction". *Engineering with Computers,* ***22***(3), pp. 277–292.

[95] Glassner, A., 1991. "An introduction to ray tracing".

[96] Siegel, R., 1987. "Transient radiative cooling of a droplet-filled layer". *ASME Journal of Heat Transfer,* ***109***(159-164).

[97] Mengucc, M., Manickavasagam, S., and D'sa, D., 1994. "Determination of radiative properties of pulverized coal particles from experiments". *Fuel,* ***73***(4), pp. 613–625.

[98] Spinti, J., Smith, P. J., Thornock, J. N., and Borodai, S., 2007. "Validation of les based predictions of heat flux to objects in transportation fuel fires". In The 2007 Annual Meeting.

[99] Simeonova Nathan, L., 2012. "Numerical implementation of models for radiative properties of molecular gases and particulate media in combustion applications". PhD thesis, THE UNIVERSITY OF UTAH.

[100] Adams, B. R., 1993. "Computational evaluation of mechanisms affecting radiation in gas-and coal-fired industrial furnaces". PhD thesis, Department of Mechanical Engineering, University of Utah.

[101] Goodwin, D., and Mitchner, M., 1989. "Flyash radiative properties and effects on radiative heat transfer in coal-fired systems". *International Journal of Heat and Mass Transfer,* ***32***(4), pp. 627–638.

[102] Qingyu. Meng, M. B., and Schmidt, J., 2011. "Using hybrid parallelism to improve memory use in the uintah framework.". *Proceedings of the 2011 TeraGrid Conference.*

[103] Ricks, A. T., 2010. Requirements for radiometer model in Fuego/Syrinx. Technical report, Sandia National Laboratories, Albuquerque, NM, February.

[104] Weisstein, G., 2011. Sphere point picking. On the WWW, March. URL http://mathworld.wolfram.com/SpherePointPicking.html+.

[105] Figueroa, V., 2002. Jpl-nasa propellant fire test series. Tech. rep., Sandia National Laboratories, Albuquerque.

[106] Luketa, A., R. V. D. S. G. D. S. M., and Figueroa, V. "Validation and uncertainty quantification of fuego simulations of calorimeter heating in a wind-driven hydrocarbon pool fire".

[107] I. Hunsaker, T. Harman, J. T., and Smith, P. J., 2011. "Efficient parallelization of rmcrt for large scale les combustion simulations". *AIAA International Conference*.

[108] N. Lallemant, J. D., and Weber, R., 1997. "Analysis of the experimental data collected during the oxyflam 1 and oxyflam 2 experiments". *International Flame Research Foundation*.