AN IMPLEMENTATION OF SURFACE-TO-SURFACE, BLACKBODY RADIATION HEAT TRANSFER IN A MOOSE APPLICATION

A Thesis

by

ANDREW MARK FRANKLIN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Jean C. Ragusa
Committee Members,	Rodolfo Vaghetto
	Bojan Popov
Head of Department,	Michael Nastasi

August 2020

Major Subject: Nuclear Engineering

ABSTRACT

Surface-to-surface, blackbody radiation heat transfer is a fundamental phenomena essential in the evaluation of some heat conduction problems. Representing it in digital simulations is of interest in the nuclear power industry in a variety of contexts. Some of these contexts include, but are not limited to: the analysis and design of micro- and space-reactors, the analysis of high-temperature surfaces present during severe accidents, *etc*. An open source MOOSE application, Phoenix, has been created and made freely available on github. The aim of this application is to provide modeling capabilities for problems that involve surface-to-surface, blackbody radiation heat transfer.

The work conducted for the completion of this thesis and in the construction of Phoenix includes:

- The incorporation of a vetted version of CGAL in the MOOSE application that is LGPL v3+ compliant. This allows for ray tracing capabilities provided by the trusted and broadly used library. This is leveraged for the view factor calculations.
- Development of a unit-test suite of 264 tests within the MOOSE app. This provides confidence in the correctness of the basic view factor calculations as the application is developed.
- Investigation into the appropriateness of quadrature-based integration in view factor calculations for a variety of geometrical configurations. This includes benchmarks of the generated solutions against analytic solutions or solutions reported in trusted literature.
- Integration of view factor calculations with the multi-physics capabilities in MOOSE.
- Implemention of surface-to-surface, blackbody radiation boundary conditions for heat conduction problems.
- Investigation and benchmarks of the capabilities for surface-to-surface, blackbody radiation boundary conditions.

By accomplishing these tasks an extensible and perhaps useful application for simulating surface-to-surface, blackbody radiation heat transfer is be freely available. Suggestions and direction for future work is discussed in the conclusion of the thesis.

DEDICATION

To Mark and Lynne. For their support and inspiration.

The Crux-Odile Hunters. For helping me gain a broader perspective.

Finally, to all who have mentored me

during my time at Texas A&M.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Jean C. Ragusa and Rodolfo Vaghetto of the Department of Nuclear Engineering and Professor Bojan Popov of the Department of Mathematics. The reference solution for the temperature distribution in the parallel, directly opposed plates were generated using ANSYS FLUENT, and provided by Mauricio Tano Retamales of the Department of Nuclear Engineering.

The computational libraries and frameworks directly used include MOOSE [1, 2], CGAL [3] and GMESH [4]. The programming languages use in the creation of this thesis include C++ [5] and Python [6]. These tools and their transitive dependencies are not the work of the student. All other work conducted for the thesis was completed by the student independently.

Funding Sources

Graduate study was supported by graduate research and teaching assistantships from Texas A&M University and support through internships conducted at Idaho National Laboratory.

NOMENCLATURE

CGAL	The Computational Geometry Algorithms Library
MOOSE	Multiphysics Object Oriented Simulation Environment
PETSc	Portable, Extensible Toolkit for Scientific Computation
LWR	Light Water Reactor
RCCS	Reactor Cavity Cooling System
VHTRs	Very High Temperature Reactors
HEX	Used to denote the hexahedral meshing of the parallel, directly opposed plates.
PYR	Used to denote the triangular pyramid meshing of the parallel, directly opposed plates.
PRI	Used to denote the triangular prism meshing of the parallel, directly opposed plates.
API	Application Programming Interface
PJFNK	Preconditioned Jacobian-free Newton-Krylov
GMRES	Generalized Minimal RESidual method
TDD	Test-Driven Development
LGPL	GNU Lesser General Public License

Math Symbols

Ω	A spatial domain of interest. For this thesis, $\Omega \subset \mathbb{R}^3$.
$\partial\Omega$	The surface of Ω.
$\partial\Omega_e^h$	The discretization of $\partial\Omega$ into disjoint and flat elements.
$\vec{r}$	Position vector in Ω.
T	Temperature (K).
ρ	Density (kg/m^3).
c_p	Specific heat capacity (J/K·kg).
k	Thermal conductivity (W/(m·K)).
q'''	Local rate of heat generation (W/m^3).
$\hat{n}$	The local, surface unit-normal.
$\mathcal{G}$	Irradiation power of a surface (W/m^2).
$\mathcal{E}$	Emissive power of a surface (W/m^2).
$\mathcal{J}$	Radiosity (W/m^2).
σ	The Stefan–Boltzmann constant ($5.670374419 \times 10^{-8}$ W/m^2·K^4).
$\mathcal{G}^{\mathrm{b}}$, $\mathcal{E}^{\mathrm{b}}$	Blackbody irradiation and emissive powers, respectively (W /m^2).
I^{b}	The local blackbody intensity (W /(m^2·sr)).
r	Reflection Fraction (i.e. reflectivity).
α	Absorbtion Fraction (i.e. absorptivity).
τ	Transmission Fraction (i.e. transmissivity).
q''	Heat Flux (W/m^2).
i	Test function index.
j	Basis function index.
ψ_i	A test function.

ϕ_j	A basis function.
T_h	The discrete approximation of T.
T_j	Coefficients for T_h.
$\vec{T}$	The vector of T_js.
N_{dofs}	The total number of degrees of freedom.
$K_i(\vec{T})$	The ith element of a kernel's vector.
$R_i(\vec{T})$	The ith element of the residual vector.
$\vec{x}$	A general vector of solution values for a nonlinear problem.
$\delta\vec{x}$	The newton direction.
n	The iteration index.
$\vec{F}$	A general residual vector for a nonlinear problem.
$J_{i,j}$	An element of the jacobian matrix of $\vec{F}$.
e, f	Arbitrary surface indices.
dA	A differential surface area.
$\hat{n}$	The local unit normal of a surface.
$\vec{s}_{e\to f}$	The ray cast from surface location e to location f.
θ	Angle formed between a ray and the unit surface normal.
$dF_{e\to f}$	The infinitesimal view factor of the diffusely emitting surface e directed towards surface f.
$d^2Q_{e\to f}$	The total energy leaving an infinitesimal surface, e that irradiates surface f.
$d\Omega_e$	The solid angle subtended by dA_f when viewed from dA_e.
$\lhd$	The view function.
$\partial\Omega_e$	A finite surface with index e.
A_e	The surface area of the surface e.
$F_{e\to f}$	The finite view factor directed from e to f.

$dI^{\text{irr}}_{e\to f}$	The diffuse energy leaving dA_e that is intercepted by dA_f.
$\langle \cdot \rangle_e$	The surface–averaging operator over element e.
$\mathcal{E}^{\text{b}}_e$	The total energy emitted from the element via blackbody radiation from the surface element e.
$\mathcal{G}^{\text{b}}_e(\vec{r}_e)$	The differential, amount of diffuse energy that irradiates an element.
$\mathcal{G}^{\text{b}}_e$	The total amount of diffuse energy that irradiates the surface element e.
$K_{e\leftrightarrow f}$	The view factor kernel between the two surface locations e and f.
$R_{e\leftrightarrow f}$	The view factor residual between two finite surfaces $\partial\Omega_e$ and $\partial\Omega_f$.
Q_{gen}	Total amount of heat generated in the plate in §5.1.
A	The surface area of the boundary that is emitting radiation in §5.1.
t	The thickness of the plate in §5.1.
$T_{\text{rad, surf}}$	The surface temperature of the face that is emitting radiation in §5.1.

TABLE OF CONTENTS

Page

ABSTRACT ii

DEDICATION iv

CONTRIBUTORS AND FUNDING SOURCES v

NOMENCLATURE vi

TABLE OF CONTENTS x

LIST OF FIGURES xii

LIST OF TABLES xiv

1. INTRODUCTION 1

2. THEORY AND LITERATURE REVIEW 3

    2.1 Overview 3
    2.2 Radiation Heat Transfer and Heat Conduction 4
    2.3 Finite Element Modeling 6
    2.4 Nonlinear Solve 8
    2.5 View Factor Calculation and Computational Geometry 9
        2.5.1 Surface–Averaging Simplification 14
        2.5.2 General View Factor Determination 15
        2.5.3 Determination of View Factors by Approximate Surface Integration 16

3. CODE DEVELOPMENT AND IMPLEMENTATION 18

    3.1 Overview on the Construction of Phoenix 18
    3.2 CGAL and the Computational Geometry Aspects 20
        3.2.1 View Factor Implementation 21
        3.2.2 Occlusion Detection 23
    3.3 Boundary Conditions 23
    3.4 Meshing 26
    3.5 Where to Find the Code and How to Get it Working 26
    3.6 About the Tools Used to Generate the Results 27

4. VIEW FACTOR EXAMPLES AND RESULTS 28

4.1 Parallel and Directly Opposed Plates 28
4.1.1 Mesh Invariance 30
4.1.1.1 Hexahedron – Hexahedron 30
4.1.1.2 Triangular Prism – Hexahedron 31
4.1.1.3 Triangular Pyramid – Hexahedron 32
4.1.1.4 Triangular Prism – Triangular Prism 33
4.1.1.5 Triangular Pyramid – Triangular Prism 34
4.1.1.6 Triangular Pyramid – Triangular Pyramid 35
4.1.2 Mesh Refinement 36
4.1.2.1 Hexahedron – Hexahedron 37
4.1.2.2 Triangular Pyramid – Triangular Pyramid 38
4.1.3 Quadrature Refinement 40
4.2 Ring to Parallel Coaxial Ring 42
4.3 Ring to Parallel Coaxial Ring with a Blocking Coaxial Cylinder 45
4.3.1 Concave Mesh 47
4.3.2 Convex Mesh 49
4.4 W-Tube 51

5. HEAT TRANSFER EXAMPLES AND RESULTS 54

5.1 Radiation Emission from a Single Plate 54
5.2 Parallel and Directly Opposed Plates with Symmetric Heat Generation 57
5.3 Parallel and Directly Opposed Plates with Asymmetric Dirichlet Boundary Condition 62

6. SUMMARY 64

6.1 Further Study 64

REFERENCES 65

LIST OF FIGURES

FIGURE Page

2.1 The boundary of an opaque medium experiencing radiation heat transfer. 5

2.2 An arbitrary geometric configuration between two infinitesimal surfaces e and f. ... 10

4.1 The geometric configuration for identical, parallel, directly opposed rectangles. 28

4.2 The HEX – HEX mesh. 30

4.3 The PRI – HEX mesh. 31

4.4 The PYR – HEX mesh. 32

4.5 The PRI – PRI mesh. 33

4.6 The PYR – PRI mesh. 34

4.7 The PYR – PYR mesh. 35

4.8 The two ends of the mesh refinement spectrum for the HEX – HEX configuration. .. 37

4.9 The mesh refinement results for the HEX – HEX configuration. 38

4.10 The two ends of the mesh refinement spectrum for the PYR – PYR configuration. ... 39

4.11 The mesh refinement results for the PYR – PYR configuration. 40

4.12 The quadrature refinement results. 41

4.13 The geometric configuration for ring to parallel coaxial ring. 42

4.14 The parallel coaxial ring mesh. 43

4.15 The geometric configuration for ring to parallel coaxial ring with a blocking coaxial cylinder. 45

4.16 The concave mesh for the parallel coaxial ring with a blocking coaxial cylinder example. 47

4.17 The convex mesh for the parallel coaxial ring with a blocking coaxial cylinder example. 49

4.18 An example of the W-Tube mesh. 51

5.1 Steady-state temperature profile of surface–averaged radiation emission for the emitting plate example. 56

5.2 Steady–state, line–sampled temperature profiles for the emitting plate example. 57

5.3 Steady-state temperature profiles of the symmetric heating example. 58

5.4 Steady-state, surface temperature profiles of the plate with heat generation in the symmetric heating example. 59

5.5 Plots of the steady–state, line–sampled temperature profiles for the plate with heat generation. 60

5.6 Steady-state, surface temperature profiles of the plate without heat generation in the symmetric heating example. 61

5.7 Plots of the steady–state, line–sampled temperature profiles for the plate without heat generation. 62

5.8 Steady-state temperature profiles of the asymmetric heating case. 63

LIST OF TABLES

TABLE Page

4.1 View factor results for the HEX – PRI mesh. 31

4.2 View factor results for the PYR – HEX mesh. 32

4.3 View factor results for the PYR – PRI mesh. 34

4.4 View factor results for the PYR – PYR mesh. 35

4.5 View factor results for the parallel coaxial ring example. 44

4.6 View factor results of the concave mesh for the parallel coaxial ring with a blocking coaxial cylinder example. 48

4.7 View factor results of the convex mesh for the parallel coaxial ring with a blocking coaxial cylinder example. 50

4.8 View factor results for various W-Tube configurations. 53

1. INTRODUCTION

> Although this task appears very simple, its solution is considerably more knotted than one would expect... the highly laborious computation would fill even the most patient with disgust and drive them away.
>
> *Johann Heinrich Lambert (1760)*

Modeling and simulation of radiation heat transfer is of interest in many applications relevant to the nuclear power industry, but their also exists a seemingly endless number of applications outside of the nuclear industry. A few examples where radiation heat transfer plays an important role includes:

- The gap between the fuel and the cladding in conventional light water reactor (LWR) fuel pins [7],
- The heat transferred from the reactor vessel of Very High Temperature Reactors (VHTRs) to the cooling panel of the Reactor Cavity Cooling System (RCCS) [8, 9],
- the emission of radiation from corium and other high temperature surfaces during severe accident scenarios [10],
- The surface of heat pipes in certain terrestrial and space, nuclear reactors [11, 12, 13, 14],
- Continuous annealing furnaces used for recrystallization annealing and surface treatment in cold-rolled strip manufacturing [15], and
- In a variety of thermal control and thermal protection systems used by spacecraft [16].

The availability of tools to aid in the task of digitally representing surface-to-surface, blackbody radiation heat transfer is critical for progress across multiple domains. Safety, policy and design decisions are often informed by the models that are widely available. Each problem presents unique requirements for its analysis, and only a wide availability of approaches leads to the possibility of addressing all challenges that might arise.

The aim of this thesis is to create a free and open source tool that can be used in the analysis of engineering problems that involve surface-to-surface, blackbody radiation heat transfer. Phoenix, a MOOSE application, was created and is freely available on github. In this application, the Computational Geometry Algorithms Library (CGAL) is leveraged to handle the deterministic computation of view factors. The view factor calculations can be used in the boundary conditions for a heat transfer problem.

This thesis begins by overviewing the theory underlying the functionality provided by Phoenix in §2. Next, the code development process (including the development of the benchmark examples) is discussed in §3. A variety of examples of the view factor calculation are included in this thesis – from the very simple (which can be compared against analytic solutions) to the more complex (which require occlusion detection and only have benchmark numerical approximations for comparison). These benchmark cases can be found in §4. Two cases exercising the surface-to-surface, blackbody radiation boundary conditions are included in §5. The thesis is concluded with suggestions for future work and investigations in §6.

2. THEORY AND LITERATURE REVIEW

> The purpose of computing is insight,
> not numbers.
>
> *Richard Wesley Hamming (1962)*

2.1 Overview

Radiation heat transfer is the exchange of thermal energy between surfaces via photons. All surfaces of materials emit photons as a result of oscillations or transitions of the many electrons that constitute matter. In general, the emitted photons encompass a range of wavelengths, and they are distributed in a variety of directions. Eventually, a photon will escape a local system, or it will be absorbed by another surface within the system [17]. This thesis will focus on the idealized case of modeling and simulating radiation exchange between blackbody surfaces.

The following sections will overview the background theory underlying the MOOSE app, Phoenix. The first three subsections will provide the necessary theoretical background, and the final subsection will narrow-in on the heart of the problem being addressed in this thesis. The first section will review the fundamentals of heat conduction in a solid and radiation heat exchange at a surface (§2.2). This will give the reader a high-level overview of the problem at hand. Next, the basic finite element procedure in the MOOSE style is discussed (§2.3). Following this, the non-linear solve by PJFNK is touched on (§2.4). A brief review will acquaint unfamiliar readers with this critical component for solving surface-to-surface radiation heat transfer problems in MOOSE. Finally, the view factor computations and the discretization of the surface-to-surface, blackbody radiation boundary conditions are described (§2.5). All of this serves as the foundation for the discussion of code development and implementation in §3.

2.2 Radiation Heat Transfer and Heat Conduction

The solution for temperature, T, (K) to heat transfer problems involving heat conduction in a solid material conform to the equation:

$$\rho c_p \frac{\partial T}{\partial t} - \nabla \cdot (k \nabla T) = q''', \quad \forall \vec{r} \in \Omega, \tag{2.1}$$

where ρ is density (kg/m^3), c_p is the specific heat capacity (J/K·kg), k is thermal conductivity (W/(m·K)) and q''' is the local rate of heat generation (W/m^3). The specification of boundary conditions is required in-order to close the problem. A variety of boundary conditions arise in real world problems (e.g. convection, mixed, etc.). Blackbody radiation boundary conditions will be the focus of the discussion, and it will be assumed that the reader is familiar with the standard variety of boundary conditions for the heat diffusion equation. The heat flux on the boundary, $\partial\Omega_{\text{rad}}$, is related to the temperature field by:

$$q''_{\text{rad}} = -k \frac{dT}{d\hat{n}}, \quad \forall \vec{r} \in \partial\Omega_{\text{rad}}, \tag{2.2}$$

where q''_{rad} is the net radiative heat flux, and $\frac{dT}{d\hat{n}}$ is the temperature gradient in the direction of the surface normal $\hat{n}$. The expression for q''_{rad} will be derived below.

Consider a boundary of an opaque medium experiencing radiation heat transfer.

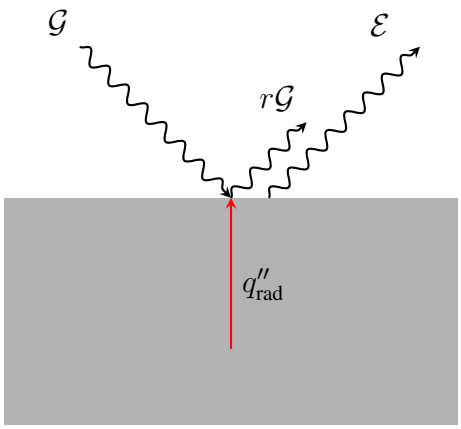


Figure 2.1: The boundary of an opaque medium experiencing radiation heat transfer.

In Figure 2.1, $\mathcal{G}$ is the irradiation incident on the surface and $\mathcal{E}$ is the emissive power of the surface. Incident irradiation can either be reflected, absorbed or transmitted through the surface. The fractions r, α and τ are defined to account for these possibilities. These fractions are referred to as the reflectivity, absorptivity and transmissivity, respectively.

It follows that,

$$r + \alpha + \tau = 1.$$

Hence, $r\mathcal{G}$ represents the incident radiation that is reflected. Finally, q''_{rad} is the net radiative heat flux – the quantity that is relevant for defining the radiation boundary condition. It should be noted that all of the quantities depicted in Figure 2.1 have units of W/m^2.

For an opaque surface, $\tau = 0$ (i.e. $r + \alpha = 1$). Radiosity, $\mathcal{J}$, is commonly described as the rate at which all radiant energy leaves a surface per unit area. For an opaque surface, radiosity can be expressed as:

$$\mathcal{J} = \mathcal{E} + r\mathcal{G}. \tag{2.3}$$

The net radiative heat flux is then:

$$q''_{\text{rad}} = \mathcal{J} - \mathcal{G}. \tag{2.4}$$

The exchange of blackbody radiation between surfaces will be the focus of this thesis, and considerably simplifies the boundary conditions for the heat diffusion equation. Blackbody surfaces absorb all incident radiation, and diffusely emit the maximal amount of radiation for a given temperature. In this scenario, $r = \tau = 0$ and $\alpha = 1$. Hence, $\mathcal{J} = \mathcal{E}$ and

$$q''_{\text{rad}}(\vec{r}) = \mathcal{E}^{\text{b}}(\vec{r}) - \mathcal{G}^{\text{b}}(\vec{r}), \quad \forall \vec{r} \in \partial\Omega_{\text{rad}}. \tag{2.5}$$

The local emission of blackbody radiation is characterized by the Stefan-Boltzmann law:

$$\mathcal{E}^{\text{b}}(\vec{r}) = \sigma T(\vec{r})^4, \tag{2.6}$$

where σ is the Stefan–Boltzmann constant. Determining $\mathcal{G}^{\text{b}}(\vec{r})$ is a bit more complicated because it involves the emission from all other surface locations within the view of a given surface location. The details of this will be discussed in section 2.5. First, the basics of the finite element method will be reviewed within the context of application to the heat diffusion equation.

2.3 Finite Element Modeling

The following section will demonstrate the basics of the finite element procedure applied to Eq. (2.1) in the context of MOOSE. MOOSE is a computational framework that excels at simulating large systems of coupled, nonlinear partial differential equations that are suitable for being approximated in the continuous Galerkin paradigm. For this demonstration, let $\phi_j(\vec{r})$ be that basis functions and $\psi_i(\vec{r})$ be the test functions. In the case of the heat diffusion equation, the approximate solution variable is represented as:

$$T_h(\vec{r}) = \sum_{j=1}^{N_{\text{dofs}}} T_j \phi_j(\vec{r}), \tag{2.7}$$

where N_{dofs} is the total number of degrees of freedom, and T_j are the coefficients of the basis

function. If the basis functions are nodal then $T_j = T_h(\vec{r}_j)$ where $\vec{r}_j$ are the node locations.

It follows that,

$$\nabla T_h(\vec{r}) = \sum_{j=1}^{N_{\text{dofs}}} T_j \nabla \phi_j(\vec{r}). \tag{2.8}$$

Plugging the approximate solution for T (i.e. T_h) into the heat diffusion equation, multiplying by the test function $\psi_i(\vec{r})$, integrating over the entire spatial domain, Ω, and moving all of the terms to the left of the equal sign:

$$\int_\Omega \rho c_p \frac{\partial T_h}{\partial t} \psi_i \, dV - \int_\Omega \nabla \cdot (k \nabla T_h) \, \psi_i \, dV - \int_\Omega q''' \psi_i \, dV = 0. \tag{2.9}$$

Using Gauss' divergence theorem:

$$\int_\Omega \rho c_p \frac{\partial T_h}{\partial t} \psi_i \, dV + \int_\Omega (k \nabla T_h) \cdot \nabla \psi_i \, dV - \int_{\partial\Omega} \psi_i \, (k \nabla T_h) \cdot \hat{n} \, dA - \int_\Omega q''' \psi_i \, dV = 0 \tag{2.10}$$

If ψ_i has local support restricted to an element in $\Omega_e \subset \Omega$ (as it does in MOOSE), then:

$$\int_{\Omega_e} \rho c_p \frac{\partial T_h}{\partial t} \psi_i dV + \int_{\Omega_e} (k \nabla T_h) \cdot \nabla \psi_i dV + \int_{\partial\Omega \cap \partial\Omega_e} -\psi_i \, (k \nabla T_h) \cdot \hat{n} dA + \int_{\Omega_e} -q''' \psi_i dV = 0 \tag{2.11}$$

Each of the terms in the above equation is a so called kernel in MOOSE (analogous to kernels used in image processing). This forms the i^{th} element of the residual vector (assuming Dirichlet boundary conditions are not applied):

$$R_i(\vec{T}) = K_i^{\text{time}}(\vec{T}) + K_i^{\text{cond}}(\vec{T}) + K_i^{\text{bndry}}(\vec{T}) + K_i^{\text{src}}(\vec{T}), \tag{2.12}$$

where $\vec{T}$ is the vector of coefficients for $T_h(\vec{r})$. Each of the integrals in the kernels are evaluated by quadrature (typically a Gauss–Legendre quadrature of a sufficiently high degree in order

to exactly evaluate the integral). The heat diffusion equation is said to be satisfied in a weak sense when $\vec{R}(\vec{T}) = 0$. Each of these kernels is implemented in the "heat_conduction" module of MOOSE with the exception of the radiation boundary condition. The implementation of the radiation boundary condition will be discussed in section 3.3. The next section will overview how the solution vector, $\vec{T}$, is solved for by searching for the solution that satisfies $\vec{R}(\vec{T}) = 0$. For notational convenience, the h from T_h will be dropped for the remainder of the thesis.

2.4 Nonlinear Solve

Discretization of radiation heat transfer problems result in a systems of nonlinear equations to be solved. MOOSE leverages the Preconditioned Jacobian-free Newton-Krylov (PJFNK) solvers provided by PETSc [18, 19, 20] to solve nonlinear multiphysics problems. The advantages of using PJFNK is that the Newton method typically has super-linear convergence in the nonlinear iteration [21, 22], and the Krylov subspace solver (typically GMRES) is efficient and suitable when using appropriate preconditioning to approximately solve the intermediate linear systems of equations [23, 24].

An overview of PJFNK solvers follows. Consider the residual statement for a general nonlinear system of equation:

$$\vec{F}(\vec{x}) = 0, \tag{2.13}$$

where $\vec{x}$, $\vec{F}(\vec{x})$ and 0 are vectors of length N_{dofs}. The elements of the jacobian matrix for the system are:

$$J(\vec{x})_{i,j} = \frac{\partial(F)_i}{\partial(x)_j}(\vec{x}). \tag{2.14}$$

Newton methods (in essence) consists of iteratively updating the iterate solution vector $\vec{x}^n$ in the following fashion:

$$\vec{x}^{n+1} = \vec{x}^n + \delta\vec{x}^n, \tag{2.15}$$

where n is the iteration number, and $\delta\vec{x}^n$ is the so called newton direction. Roughly speaking, the iteration is continued until the residual statement is satisfied to a sufficient degree, or it is terminated if divergence of the solution is detected. The newton direction is determined by solving the following linear system:

$$J(\vec{x}^n)\delta\vec{x}^n = -\vec{F}(\vec{x}^n). \tag{2.16}$$

If the initial guess is sufficiently close to the solution, and this linear system is exactly solved in each iterate then it is provable that Newton's method has q-quadratic convergence [21]. For large, multiphysics problems this is impractical and often an approximation for $\delta\vec{x}^n$ is sufficient for q-superlinear convergence. MOOSE uses preconditioned Krylov methods to rapidly arrive at an approximation for $\delta\vec{x}^n$. A nice feature of Krylov methods is that only matrix-vector products are needed (as opposed to needing the details of the matrix itself). Krylov methods are often called matrix-free if they are implemented in this manner, and this is why PJFNK is called jacobian-free.

Now, the reader should have some understanding of the theory underlying MOOSE. It is now appropriate to discuss the surface–averaging process employeed to approximate the blackbody radiation boundary conditions. This will provide as the inception of how surface-to-surface heat transfer can be incorporated in MOOSE. The next section will first focus on view factor theory as it is a critical component for handling the radiation boundary condition. The details of the surface–averaging process for handling the boundary conditions will then be covered in §2.5.1. The section is finished by over-viewing possible methods for computing view factors in §2.5.2, and then providing the bare essentials of how view factors can be determined by surface integration in §2.5.3.

2.5 View Factor Calculation and Computational Geometry

Consider the geometrical configuration of the two infinitesimal surface in the neighborhoods of $\vec{r}_e$ and $\vec{r}_f$, as shown in Figure 2.2.

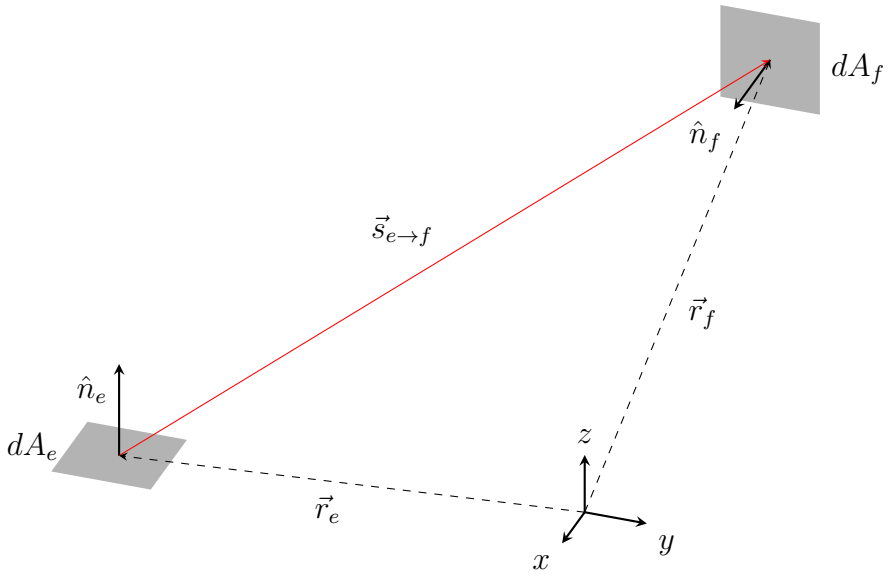


Figure 2.2: An arbitrary geometric configuration between two infinitesimal surfaces e and f.

The dA's and $\hat{n}$'s are the associated infinitesimal surface areas and unit normals. The ray, $\vec{s}_{e\to f}$ is related to the surface locations, $\vec{r}_e$ and $\vec{r}_f$, by:

$$\vec{s}_{e\to f} = \vec{r}_f - \vec{r}_e. \tag{2.17}$$

The cosine of the angles between $\vec{s}_{e\to f}$ and each of the surface normals can be represented as:

$$\cos\theta_e = \frac{\vec{s}_{e\to f}\cdot\hat{n}_e}{\|\vec{s}_{e\to f}\|}, \text{ and} \tag{2.18}$$

$$\cos\theta_f = -\frac{\vec{s}_{e\to f}\cdot\hat{n}_f}{\|\vec{s}_{e\to f}\|}. \tag{2.19}$$

It is advantageous to define the infinitesimal view factor of the diffusely emitting surface e directed towards surface f:

$$dF_{e\to f} \equiv \frac{\text{diffuse energy leaving } dA_e \text{ directly toward and intercepted by } dA_f}{\text{total diffuse energy leaving } dA_e}. \tag{2.20}$$

For black body radiation the denominator is:

$$\mathcal{E}^{\mathrm{b}}(\vec{r}_e)dA_e = \pi I^{\mathrm{b}}(\vec{r}_e)dA_e, \tag{2.21}$$

where $I^{\mathrm{b}}(\vec{r}_e)$ is the local blackbody intensity having units of (W /(m^2 $\cdot$ sr)). The total energy leaving an infinitesimal surface, e that irradiates surface f is:

$$d^2Q_{e\to f} = \mathcal{E}^{\mathrm{b}}(\vec{r}_e)\lhd(\vec{r}_e,\vec{r}_f)\cos\theta_e d\Omega_e dA_e, \tag{2.22}$$

where $d\Omega_e$ is the solid angle subtended by dA_f when viewed from dA_e, and $\lhd(\vec{r}_e,\vec{r}_f)$ is the view function. The solid angle is:

$$d\Omega_e = \frac{\cos\theta_f}{\|\vec{s}_{e\to f}\|^2}dA_f. \tag{2.23}$$

The view function, $\lhd(\vec{r}_e,\vec{r}_f)$, takes the entirety of the surface configuration into account. It must behave such that:

$$\lhd(\vec{r}_e,\vec{r}_f) = \begin{cases} 0 & \text{if the ray, } \vec{s}_{e\to f}\text{, is occluded by a surface} \\ 1 & \text{otherwise} \end{cases}. \tag{2.24}$$

A property of this function is that $\lhd(\vec{r}_e,\vec{r}_f) = \lhd(\vec{r}_f,\vec{r}_e)$ (i.e. the arguments are commutative). Working with expression (2.20):

$$\begin{aligned} dF_{e\to f} &\equiv \frac{\text{diffuse energy leaving } dA_e \text{ directly toward and intercepted by } dA_f}{\text{total diffuse energy leaving } dA_e} \\ &= \frac{d^2Q_{e\to f}}{\mathcal{E}^{\mathrm{b}}(\vec{r}_e)dA_e} \\ dF_{e\to f} &= \lhd(\vec{r}_e,\vec{r}_f)\frac{\cos\theta_e\cos\theta_f}{\pi\|\vec{s}_{e\to f}\|^2}dA_f \end{aligned} \tag{2.25}$$

To be consistent with notation commonly found in finite element literature let a finite surface

be denoted by $\partial\Omega_e$. The surface area of that surface will then be represented as $A_e = |\partial\Omega_e|$. The finite view factor directed from $\partial\Omega_e$ to $\partial\Omega_f$ is arrived at by integrating Eq. (2.25) over the two surfaces:

$$F_{e\to f} = \frac{1}{A_e}\int_{\partial\Omega_e}\int_{\partial\Omega_f} \lhd(\vec{r}_e, \vec{r}_f)\frac{\cos\theta_e \cos\theta_f}{\pi\|\vec{s}_{e\to f}\|^2} dA_f dA_e. \tag{2.26}$$

Before moving on, some important relationships that are used in the process of verifying and implementing the code will be enumerated. In the relationships below, lowercase letters are used to represent finite, disjoint surfaces, and uppercase letters represent sets of those surfaces.

The reciprocity relationship:

$$A_e F_{e\to f} = A_f F_{f\to e}. \tag{2.27a}$$

The partial additive relationship:

$$F_{e\to F} = \sum_{\partial\Omega_f\in\partial\Omega_F} F_{e\to f}. \tag{2.27b}$$

The convex, partial-enclosure constraint:

$$\sum_{\partial\Omega_f\in\partial\Omega_F} F_{e\to f} \leq 1. \tag{2.27c}$$

The complete additively relationship:

$$F_{E\to F} = \frac{1}{A_E}\sum_{\partial\Omega_e\in\partial\Omega_E}\sum_{\partial\Omega_f\in\partial\Omega_F} A_e F_{e\to f}. \tag{2.27d}$$

It should be noted that

$$A_E \equiv \sum_{\partial\Omega_e \in \partial\Omega_E} A_e.$$

Now that the properties of view factors have been reviewed the expression for $\mathcal{G}^{\text{b}}(\vec{r})$ can be discussed. Let $dI^{\text{irr}}_{e\to f}$ be defined as the diffuse energy leaving dA_e that is intercepted by dA_f. The expression for this is:

$$\begin{aligned} dI^{\text{irr}}_{e\to f} &= \mathcal{E}^{\text{b}}(\vec{r}_e) dA_e dF_{dA_e \to dA_f} \\ dI^{\text{irr}}_{e\to f} &= \sigma T(\vec{r}_e) \lhd(\vec{r}_e, \vec{r}_f) \frac{\cos\theta_e \cos\theta_f}{\pi \|\vec{s}_{e\to f}\|^2} dA_e dA_f \end{aligned} \tag{2.28}$$

By swapping e and f, this expression can be integrated over all surfaces to get the total diffuse energy intercepted by dA_e:

$$\mathcal{G}^{\text{b}}(\vec{r}_e) dA_e = \sigma dA_e \int_A T(\vec{r}_f)^4 \lhd(\vec{r}_e, \vec{r}_f) \frac{\cos\theta_e \cos\theta_f}{\pi \|\vec{s}_{e\to f}\|^2} dA_f. \tag{2.29}$$

Finally, we arrive at the expression for the differential amount of diffuse energy intercepted:

$$\mathcal{G}^{\text{b}}(\vec{r}_e) = \sigma \int_A T(\vec{r}_f)^4 \lhd(\vec{r}_e, \vec{r}_f) \frac{\cos\theta_e \cos\theta_f}{\pi \|\vec{s}_{e\to f}\|^2} dA_f. \tag{2.30}$$

The last piece of the puzzle is to relate the temperature profile at the surface to the radiation emission and irradiation. This is done by using Fourier's law of heat conduction:

$$(-k\nabla T(\vec{r})) \cdot \hat{n} = \mathcal{E}^{\text{b}}(\vec{r}) - \mathcal{G}^{\text{b}}(\vec{r}), \quad \forall \vec{r} \in \partial\Omega_{\text{rad}}. \tag{2.31}$$

The next section will discuss how equations (2.6), (2.30) and (2.31) can be leveraged, after applying some simplifying assumptions, to incorporate external, blackbody radiation heat transfer effects in MOOSE.

2.5.1 Surface–Averaging Simplification

The following section will describe the surface–averaging process employed to approximate external, blackbody radiation heat exchange. Consider the scenario where the surface, $\partial\Omega$, has been discretized into disjoint and flat elements, $\partial\Omega_e^h$. Let $\partial\Omega^h$ represent the set of all surface elements, and $\partial\Omega_{\text{rad}}^h$ represent the set of all surface elements experiencing radiation heat exchange. It follows that $\partial\Omega_{\text{rad}}^h \subseteq \partial\Omega^h$.

Consider the element, $\partial\Omega_e^h \in \partial\Omega_{\text{rad}}^h$. Similar to the previous section, $A_e = |\partial\Omega_e^h|$ (i.e. A_e is the area of $\partial\Omega_e^h$). Let the surface–averaging operator be defined by:

$$\langle \cdot \rangle_e \equiv \frac{\int_{\partial\Omega_e^h} \cdot \, dA}{A_e}. \tag{2.32}$$

The total energy emitted from the element via blackbody radiation is:

$$\begin{aligned} \mathcal{E}_e^{\text{b}} &\equiv \int_{\partial\Omega_e^h} \mathcal{E}^{\text{b}}(\vec{r}) dA \qquad (2.33) \\ &= \int_{\partial\Omega_e^h} \sigma T(\vec{r})^4 dA \\ &= \sigma \int_{\partial\Omega_e^h} T(\vec{r})^4 dA \\ \mathcal{E}_e^{\text{b}} &= \sigma A_e \langle T^4 \rangle_e. \qquad (2.34) \end{aligned}$$

The differential, amount of diffuse energy that irradiates the element is:

$$\mathcal{G}_e^{\text{b}}(\vec{r}_e) \equiv \sigma \sum_{\partial\Omega_f \in \partial\Omega_{\text{rad}}^h} \int_{\partial\Omega_f} T(\vec{r}_f)^4 \lhd(\vec{r}_e, \vec{r}_f) \frac{\cos\theta_e \cos\theta_f}{\pi \left\| \vec{s}_{e\to f} \right\|^2} dA_f. \tag{2.35}$$

Naturally, this lead to the definition of the total amount of diffuse energy that irradiates the element:

$$\mathcal{G}_e^{\mathrm{b}} \equiv \int_{\partial\Omega_e} \mathcal{G}_e^{\mathrm{b}}(\vec{r}_e) dA_e \tag{2.36}$$
$$= \sigma \sum_{\partial\Omega_f \in \partial\Omega_{\mathrm{rad}}^h} \int_{\partial\Omega_f} T(\vec{r}_f)^4 \left(\int_{\partial\Omega_e} \measuredangle(\vec{r}_e, \vec{r}_f) \frac{\cos\theta_e \cos\theta_f}{\pi \|\vec{s}_{e\to f}\|^2} dA_e \right) dA_f.$$

Make the simplifying assumption that the temperature profile of every surface element on the radiation boundary is constant (i.e. $T(\vec{r}_e) = \langle T \rangle_e \ \forall \vec{r}_e \in \partial\Omega_e, \ \forall \partial\Omega_e \in \partial\Omega_{\mathrm{rad}}^h$). A consequence of this is that $\langle T^4 \rangle_e = \langle T \rangle_e^4$. The above equations then simplifies as follows:

$$= \sigma \sum_{\partial\Omega_f \in \partial\Omega_{\mathrm{rad}}^h} \langle T \rangle_f^4 \int_{\partial\Omega_f} \int_{\partial\Omega_e} \measuredangle(\vec{r}_e, \vec{r}_f) \frac{\cos\theta_e \cos\theta_f}{\pi \|\vec{s}_{e\to f}\|^2} dA_e dA_f$$
$$= \sigma \sum_{\partial\Omega_f \in \partial\Omega_{\mathrm{rad}}^h} \langle T^4 \rangle_f \int_{\partial\Omega_f} \int_{\partial\Omega_e} \measuredangle(\vec{r}_e, \vec{r}_f) \frac{\cos\theta_e \cos\theta_f}{\pi \|\vec{s}_{e\to f}\|^2} dA_e dA_f$$
$$= \sigma \sum_{\partial\Omega_f \in \partial\Omega_{\mathrm{rad}}^h} \langle T^4 \rangle_f A_f F_{f\to e}$$
$$\mathcal{G}_e^{\mathrm{b}} = \sum_{\partial\Omega_f \in \partial\Omega_{\mathrm{rad}}^h} \mathcal{E}_f^{\mathrm{b}} F_{f\to e} \tag{2.37}$$

It is expected that the surface–averaged approximation of an idealized surface configuration approaches the real configuration in the limit of infinitely many sub-surfaces.

2.5.2 General View Factor Determination

A variety of methods exist to compute or approximate view factors:

- Direct integration
  - Surface integration. Analytical or numerical.
  - Contour integration. Analytical or numerical.
- Statistical determination (Monte Carlo)

- Special methods
  - View factor algebra.
  - Crossed-strings method.
  - Unit sphere method.
  - Inside sphere method.
- Approximate quadrature methods commonly used in computer graphics
  - The hemicube algorithm.
  - The cubic tetrahedral algorithm.

Details of these methods can be found throughout the literature [25, 26, 27, 28, 29, 30, 31, 32]. The focus of this thesis is to approximate view factors by double surface integration via numerical quadrature. View factor algebra is also leveraged to reduce the number of quadratures that must be conducted. This method has been shown to be accurate for sufficiently resolved discretizations of geometric configurations [33].

Direct integration methods are attractive in certain circumstances because they offer deterministic run-times. The drawback is that analytic evaluation is only possible for simple geometric configurations. Numerical methods must be leveraged to approximate the integrals.

The following sections will describe how the view factors are computed by this method and justify the tools and methods utilized in the computations.

2.5.3 Determination of View Factors by Approximate Surface Integration

The form of the view factor integral in Eq. (2.26) does not lend itself well to computation. One would have to determine the angle between the surface normal and the ray which is unnecessary. An alternative approach is to leverage the properties of dot products expressed in Eq. (2.18) and (2.19). Let the view factor kernel be defined as:

$$K_{e\leftrightarrow f} \equiv -\sphericalangle(\vec{r}_e, \vec{r}_f)\frac{\left(\hat{n}_e \cdot \vec{s}_{e\to f}\right)\left(\hat{n}_f \cdot \vec{s}_{e\to f}\right)}{\pi\|\vec{s}_{e\to f}\|^4}. \tag{2.38}$$

Using this, the infinitesimal view factor from dA_e to dA_f can be expressed in terms of the view factor kernel as:

$$dF_{e \to f} = K_{e \leftrightarrow f} dA_f. \tag{2.39}$$

Let the view factor residual between finite surfaces $\partial\Omega_e$ and $\partial\Omega_f$ be defined as:

$$R_{e \leftrightarrow f} \equiv \int_{\partial\Omega_e} \int_{\partial\Omega_f} K_{e \leftrightarrow f} dA_f dA_e. \tag{2.40}$$

By leveraging view factor algebra, the view factor from $\partial\Omega_e$ to $\partial\Omega_f$ can be computed by:

$$F_{e \to f} = \frac{R_{e \leftrightarrow f}}{A_e}. \tag{2.41}$$

The reciprocal view factor can be computed by a similar formula without having to recompute the residual. In effect, this halves the number of residual computations that must be conducted.

3. CODE DEVELOPMENT AND IMPLEMENTATION

3.1 Overview on the Construction of Phoenix

A smaller scale code was first developed in python to test the evaluation of view factors by quadrature. This code is called "thermal_radiation" and is available at https://github.com/andfranklin/thermal_radiation. The version referenced by this thesis can be accessed with the tag: thesis. Documentation and information on how to install "thermal_radiation" can be built as follows:

1. Having a compatible version of python (e.g. 3.8.1) installed and an Internet connection.
2. Navigate to the "docs" directory.
3. Running the make file: "make html".
4. Open "_build/html/index.html" in a web browser of choice.

After thermal radiation is installed the scripts at the root of the repository can be run. To test that everything was installed properly the following can be run from the root directory: "python tritri.py".

The foundational geometry code can be found in "geometry.py". Numpy [34, 35] is leveraged to compute the unoccluded, differential view factors using double-precision, floating-point numbers. An arbitrary geometric configuration can be constructed out of instances of the "Triangle" class.

Scipy [36] conveniently provides an interface to adaptively integrate functions over multiple variable to an arbitrary degree of accuracy (scipy.integrate.nquad). The root functionality is supported by the Fortran library QUADPACK. Sympy [37] is used to construct the Gauss-Legendre quadrature rules. Each of these libraries have been in wide use for a number of years, and they are generally accepted as reliable. This function is used to integrate the differential unoccluded

view factors to arrive at a reference solution for the view factors of triangles in the function "adaptive_triangle_view_factor".

Alternatively, the closure, "get_fixed_triangle_view_factor", can be used to approximate view factors using a given quadrature rule. A user is expected to provide one of the two quadrature classes defined in "quadrature_2d.py": "TriangleTensorProductGaussLegendre2D" or "TriangleSymmetricalGauss2D". When provided one of the quadrature rules the closure returns a function that can approximate the view factors between two triangles. The resulting function has the same interface as "adaptive_triangle_view_factor".

A good example of how to use these building blocks can be found in "tritri.py". First, the reference solution is computed by adaptive integration. Then, two approximate solutions are computed by using the quadrature rules. This script measures the time of how long it takes each one of the computations to complete. It was useful to develop this script to ensure the consistency between the solutions as the quadrature orders are increased to the highest level. Although this does not prove correctness of the implementation, it does provide some confidence when consistency is observed. A fringe benefit of this script is that one can easily adjust the triangles and other parameters to observe the interplay on the effects of the results.

Playing with this script might also impart some insight into how to improve the mesh in large scale problems so as to obtain accurate solutions. Generally speaking, accuracy decreases when using quadrature rules, and computation time increases when using adaptive integration. Also, the tensor product quadrature rule is generally slower and on the same order of accuracy as the symmetrical quadrature rule.

Other test cases for evaluating the view factor between quadrangles and triangles can be found at the root of the directory. Of particular interest is the file "quadquad.py". Here, the view factor is evaluated between identical, parallel, directly opposed rectangles. Each rectangle is constructed out of two triangles, and the final view factors for the rectangles are computed after the view factors between the triangles are computed. This case is of interest because the analytic solution is easily computed (see §4.1). In this way, both correctness and consistency between the methods can be

evaluated.

After some level of confidence had been gained in using standard quadratures to evaluate view factor computation it was decided to implement the method in a way that it could be used to address more practical problems. C++ was chosen as the language of choice because then the computations could be integrated easily with MOOSE, and because its emphasis on zero cost abstraction. The Computational Geometry Algorithms Library (CGAL) was selected to handle the geometric aspects.

A standalone MOOSE application, Phoenix, was created. A vetted branch of CGAL was created that contains only LGPL portions of the library. This is included in Phoenix as a git submodule. Then, the low-level geometric primitives in CGAL were used to implement and extend the functionality of "thermal_radiation".

A test-driven development (TDD) approach was taken where, ultimately, a unit test suite of 264 tests were implemented using the googletest testing harness. Included in this test suite are tests of view factors, tests of the collision detection primitives and testing of data structures used in the full-scale view factor computations. For the tests of the view factors, the results are compared against solutions generated by "thermal_radiation". It should be noted that identical tests are run for both exact and approximate number types wherever possible (see §3.2 for more information). From there, functionality was incrementally added until surface-element-averaged radiation heat transfer problems could be run.

3.2 CGAL and the Computational Geometry Aspects

CGAL was selected to handle the geometric aspects of the view factor calculations because of its wide use, ease of access, permissive licenses and ability to robustly handle ray tracing while also providing the flexibility to handle multiple number types.

The Computational Geometry Algorithms Library (CGAL) is utilized to compute the ray-tracing [38]. This library was chosen because all of the required algorithms and routines were implemented, the library has a permissive license for the required functionality, and this library is a well tested and broadly used. In the event that there is a bug it is likely that it would be found

quickly and fixed.

One of the nice features that CGAL provides is that it is trivial to switch between "exact" and standard floating-point computation paradigms. This provides users with the option to choose between precise computation of the connecting rays between elements, or one can sacrifice some precision (which may or may not be catastrophic) to gain faster run-times and lower memory profiles.

From the software developer's perspective, a nice feature of CGAL is that it leverages template meta-programming to generalize the algorithms to be independent of the underlying number type. This leads to code that needs to be written only once, and will work for multiple number types after template specialization. Sometimes this is referred to as compile-time polymorphism. Ultimately, the template meta-programming style results in compact code that is less likely to have bugs, or if there is a bug it is easier to fix across all template cases. It also allows for an effective method for managing all possible combinations of user options. Many of the available compilers claim to generate machine code from templates with run-time and space efficiency that matches handwritten code [39].

The major drawback is that the code tends to becomes more abstract. It can be difficult for those who do not have experience with template programming to understand what the code is doing. From a design standpoint, functionality and correctness outweighed this drawback.

3.2.1 View Factor Implementation

The view factor residual can be analytically computed for arbitrary convex polygons [40], but it is not appropriate when occlusion occurs. It was decided that it would be advantageous to use quadrature rules provided by libMesh to approximate the residual integration [1, 2, 41].

Quadrature rules are used to approximate integrals by using finite summations:

$$\int_R f(x)dx \approx \sum_{i=1}^{n} f(x_i) \cdot w_i. \tag{3.1}$$

Here, x_i are quadrature points that lie within a reference spatial domain R, and w_i are the quadra-

ture weights. For a quadrature rule to be valid in view factor calculations, every x_i must strictly be within the reference spatial domain and every $w_i > 0$. There are many different quadrature rules and they each have their merits. libMesh provides may quadrature rules for quadrilaterals and triangles.

To apply a quadrature rule to arbitrary domains, A, a mapping, m, must be defined between R and A. The change of integration domain follows:

$$\int_A u(x)dx = \int_R u(m(y))\,|m'(y)|\,dy. \tag{3.2}$$

It should be noted that quadrature rules where, $R \subset \mathbb{R}^2$, are of primary interest for this thesis. The meshes of interest are embedded in $\mathbb{R}^3$. This implies that the mapping for each surface element will be of the form, $m : \mathbb{R}^2 \to \mathbb{R}^3$.

The view factor residuals are then approximated by the double summation:

$$R_{e\leftrightarrow f} = \sum_{(\vec{x},w)_e \in \mathrm{Q}_{\partial\Omega_e}} \sum_{(\vec{x},w)_f \in \mathrm{Q}_{\partial\Omega_f}} K(m_{\partial\Omega_e}(\vec{x}_e), m_{\partial\Omega_f}(\vec{x}_f)) w_e |m'_{\partial\Omega_e}(\vec{x}_e)| w_f |m'_{\partial\Omega_f}(\vec{x}_f)|, \tag{3.3}$$

where

- Q's are the sets of quadrature points and weights (i.e. $(\vec{x}, w)$) for a give quadrature rule,
- m's are the mappings from the reference element domains to the actual surface elements' domains,
- m''s are the respective jacobians of the mappings, and
- K is the discrete view factor kernel.

The discrete view factor kernel is:

$$K(\vec{r}_e, \vec{r}_f) = -\sphericalangle(\vec{r}_e, \vec{r}_f) \frac{\left(\hat{n}_e \cdot \vec{s}_{e\to f}\right)\left(\hat{n}_f \cdot \vec{s}_{e\to f}\right)}{\pi \left\|\vec{s}_{e\to f}\right\|^4}, \tag{3.4}$$

and $\vec{s}_{e\to f}$ is computed from $\vec{r}_e$ and $\vec{r}_f$ using equation 2.17.

3.2.2 Occlusion Detection

There are currently three implementations of $\lhd(\vec{r}_e, \vec{r}_f)$ in Phoenix:

- "NONE" – returns 1 if the surfaces are facing each other, and 0 otherwise. This does not satisfy property (2.24), but it does minimize the computation time.
- "BRUTE_FORCE_WITHOUT_BBOX" – If the surfaces of interest are facing each other then it searches through every other surface element and does collision detection of the ray and that test surface. It stops early if a collision is detected.
- "BRUTE_FORCE_WITH_BBOX" – Similarly searches through every surface element, but it first does a collision detection between the ray and the surface's bounding box, and then does the actual collision detection of the ray and the surface. The hope is that there is some speed-up in the computation because bounding box collision detection is quicker than that of a triangle or quadrangle. This is especially true if the "exact" number system is used.

Having multiple options allows for users to select an appropriate option for their problem. If some *a priori* knowledge about the geometric configuration is known a user might opt for no collision detection or the bounding box method. It is up to the user to select an option that could result in correct answers. To account for occlusion, and if uncertain of which option is appropriate, the brute force option should be selected.

3.3 Boundary Conditions

To allow for the full generality of mixed boundary conditions, $K_i^{\text{bndry}}(\vec{T})$, is actually:

$$K_i^{\text{bndry}}(\vec{T}) = K_i^{\text{cond}}(\vec{T}) + K_i^{\text{conv}}(\vec{T}) + K_i^{\text{rad}}(\vec{T}). \tag{3.5}$$

If one of the modes of heat transfer is not active then one could think of respective kernel returning zero. In MOOSE the kernel is simply not activated. The kernels for conduction and

convection boundary conditions ($K_i^{\text{cond}}(\vec{T})$ and $K_i^{\text{conv}}(\vec{T})$, respectively) are included in MOOSE. The theory for theses kernels will not be covered in this thesis.

Focusing on the boundary condition kernel for radiation:

$$\begin{aligned} K_i^{\text{rad}}(\vec{T}) &\equiv \int_{\partial\Omega_e} -\psi_i \left(k\nabla T_h\right)\cdot\hat{n}\, dA \\ &= \int_{\partial\Omega_e} \psi_i \left(\mathcal{E}_e^{\text{b}}(\vec{r}) - \mathcal{G}_e^{\text{b}}(\vec{r})\right)\, dA. \end{aligned} \tag{3.6}$$

Since it is assumed that each surface is locally isothermal:

$$\mathcal{E}_e^{\text{b}}(\vec{r}) = \frac{\mathcal{E}_e^{\text{b}}}{A_e} \quad \forall \vec{r} \in \partial\Omega_e, \text{ and} \tag{3.7}$$

$$\mathcal{G}_e^{\text{b}}(\vec{r}) = \frac{\mathcal{G}_e^{\text{b}}}{A_e} \quad \forall \vec{r} \in \partial\Omega_e. \tag{3.8}$$

The term in the parentheses within the right hand side of the surface integral can then be expanded as:

$$\begin{aligned} \mathcal{E}_e^{\text{b}}(\vec{r}) - \mathcal{G}_e^{\text{b}}(\vec{r}) &= \frac{\mathcal{E}_e^{\text{b}}}{A_e} - \frac{\mathcal{G}_e^{\text{b}}}{A_e} \\ &= \frac{\mathcal{E}_e^{\text{b}}}{A_e} - \frac{1}{A_e}\sum_{\partial\Omega_f \in \partial\Omega_{\text{rad}}^h} \mathcal{E}_f^{\text{b}} F_{f\to e} \\ &= \sigma\langle T^4\rangle_e - \frac{1}{A_e}\sum_{\partial\Omega_f \in \partial\Omega_{\text{rad}}^h} \sigma A_f \langle T^4\rangle_f F_{f\to e}. \end{aligned} \tag{3.9}$$

Using relationship (2.27a):

$$= \sigma \langle T^4 \rangle_e - \frac{1}{A_e} \sum_{\partial\Omega_f \in \partial\Omega^h_{\text{rad}}} \sigma \langle T^4 \rangle_f A_e F_{e \to f}$$

$$\mathcal{E}^{\text{b}}_e(\vec{r}) - \mathcal{G}^{\text{b}}_e(\vec{r}) = \sigma \left(\langle T^4 \rangle_e - \sum_{\partial\Omega_f \in \partial\Omega^h_{\text{rad}}} \langle T^4 \rangle_f F_{e \to f} \right). \tag{3.10}$$

Substituting the expansion (3.10) into Eq. (3.6):

$$K^{\text{rad}}_i(\vec{T}) = \sigma \left(\langle T^4 \rangle_e - \sum_{\partial\Omega_f \in \partial\Omega^h_{\text{rad}}} \langle T^4 \rangle_f F_{e \to f} \right) \int_{\partial\Omega_e} \psi_i \, dA. \tag{3.11}$$

This is how the radiation boundary conditions for the heat diffusion equation are handled. The view factors, $F_{e \to f}$, are computed before the simulation begins. The $\langle T^4 \rangle_e$'s are computed at the beginning of the nonlinear iteration before the residual vector is computed, but after $\vec{T}$ is updated.

The jacobian of this kernel is determined by taking the derivative with respect to the T_j's:

$$J^{\text{rad}}_{i,j}(\vec{T}) = \frac{\partial K^{\text{rad}}_i}{\partial T_j}(\vec{T}). \tag{3.12}$$

An initial evaluation will yield:

$$J^{\text{rad}}_{i,j}(\vec{T}) = \left(\frac{\partial \langle T^4 \rangle_e}{\partial T_j} - \sum_{\partial\Omega_f \in \partial\Omega^h_{\text{rad}}} \frac{\partial \langle T^4 \rangle_f}{\partial T_j} F_{e \to f} \right) \sigma \int_{\partial\Omega_e} \psi_i \, dA. \tag{3.13}$$

From Equations 2.7 and 2.32, $\langle T^4 \rangle_e$ expanded is:

$$\langle T^4 \rangle_e = \frac{1}{A_e} \int_{\partial\Omega^h_e} \left(\sum_{k=1}^{N_{\text{dofs}}} T_k \phi_k(\vec{r}) \right)^4 dA. \tag{3.14}$$

Taking the partial derivative of this:

$$\frac{\partial \langle T^4 \rangle_f}{\partial T_j} = \frac{4}{A_e} \int_{\partial\Omega^h_e} T(\vec{r})^3 \left(\sum_{k=1}^{N_{\text{dofs}}} \frac{\partial T_k}{\partial T_j} \phi_k(\vec{r}) \right) dA. \tag{3.15}$$

This can be simplified even further by recognizing that $\frac{\partial T_k}{\partial T_j} = \delta_{kj}$, where δ_{kj} is the Kronecker delta. The final expression is:

$$\frac{\partial \langle T^4 \rangle_f}{\partial T_j} = \frac{4}{A_e} \int_{\partial \Omega_e^h} T(\vec{r})^3 \phi_j(\vec{r}) dA. \tag{3.16}$$

Like the $\langle T^4 \rangle_e$'s, these values are computed at the beginning of the nonlinear iteration before the residual vector is computed, but after $\vec{T}$ is updated. Once these values are computed, determining $J_{i,j}^{\text{rad}}(\vec{T})$ is done by using those values in the evaluation of equation 3.13.

3.4 Meshing

All meshing was done using the open source tool Gmsh [4]. The python Application Programming Interface (API) was used to parametrically define and automate the construction of the meshes used in the examples. Gmsh does not allow for multi-block meshes, so geometric configurations which require multiple blocks were constructed by separate scripts. MOOSE's mesh generation system is extensively used to read the Gmsh meshes into memory, specify the boundaries of those meshes, combine the meshes into a common mesh, and specify the boundaries as separate blocks.

A critical feature for the analysis of radiation heat transfer is specifying the boundaries of the mesh for which view factors are to be calculated and will have radiation boundary conditions. Multiple generators that aid in the specification of boundaries are included in MOOSE by default ("SideSetsFromNormalsGenerator", "SideSetsFromPointsGenerator", etc.). To accommodate some of the example cases presented in this thesis the "SideSetsPoolFloodGenerator" was created and included in Phoenix. This new mesh generator adds new sidesets starting at the specified point containing all connected element faces that are a part of the specified pool sideset. It is analogous to the paint bucket in Microsoft Paint.

3.5 Where to Find the Code and How to Get it Working

The code can be found at and cloned from "https://github.com/andfranklin/Phoenix". Like all MOOSE applications, the version control system, git, is used to manage and maintain the code.

The “README.md” file at the root of the directory contains instructions on how to clone and compile the code. All scripts and input files for the examples presented in this thesis are included in the “problems” directory. It should be noted that python and Gmsh must be installed in order to run the meshing scripts.

3.6 About the Tools Used to Generate the Results

The results presented in this thesis were generated on 2014 MacBook Pro with a 2.5 GHz Quad-Core Intel Core i7 CPU and 16 GB of RAM running macOS Catalina 10.15.3. Python 3.8.1 and Gmsh 4.5 were installed and used to create the meshes. The version of Phoenix used to generate the results presented in this thesis can be accessed through the commit tag “thesis”.

4.1 Parallel and Directly Opposed Plates

A variety of test cases were run of parallel and directly opposed plates. This helps verify that the unoccluded view factor calculations are correct for both triangular and quadrangular surface elements. Consider the geometric configuration in Figure 4.1.

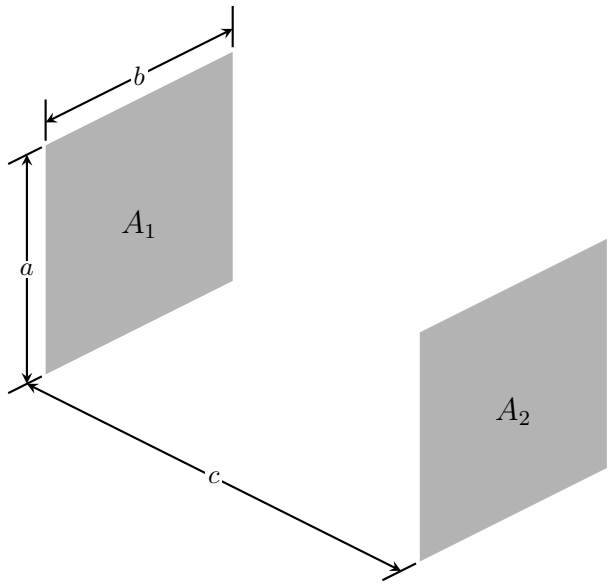


Figure 4.1: The geometric configuration for identical, parallel, directly opposed rectangles.

The analytic solution for the view factors is given by [42, 32]:

$$F_{A_1 \to A_2} = \frac{2}{\pi XY}\left[\ln\left(\frac{(1+X^2)(1+Y^2)}{1+X^2+Y^2}\right)^{1/2} + X\sqrt{1+Y^2}\arctan\left(\frac{X}{\sqrt{1+Y^2}}\right) + Y\sqrt{1+X^2}\arctan\left(\frac{Y}{\sqrt{1+X^2}}\right) - X\arctan X - Y\arctan Y\right], \quad (4.1)$$

where $X \equiv a/c$ and $Y \equiv b/c$. It was arbitrarily chosen that the interacting plates will be of

dimension 10 m $\times$ 10 m $\times$ 1 m, and are separated by 9 m. The analytic solution for the view factor of one plate to the other (rounded to 16 digits) is 0.2285656684427082. All results presented in the following subsections were generated by using standard double floating point numbers for all geometric computations and not using occlusion detection (as it is know *a priori* that occlusion does not occur in the configuration).

The following six subsections present figures of the meshes used in the calculations, the resulting view factors, and the respective errors of the calculations. Fourth-order Gauss quadrature was used to approximate the double surface integrals. For the hexahedral meshes (HEX) each element is 1 m $\times$ 1 m $\times$ 1 m. This results in 100 surface element on each surface of interest. For the triangular prism meshes (PRI), 244 surface elements are present on each surface of interest. This results in an average surface area of 0.40984 m^2. The triangular pyramid meshes (PYR) have 246 surface elements on each surface of interest. The average surface area of each element is 0.40650 m^2.

The subsections following the base calculations very the quadrature rule, quadrature order and the mesh refinement level. The hexahedron-hexahedron configuration was used for this case. This section concludes with a brief summary of the results presented.

4.1.1 Mesh Invariance

4.1.1.1 Hexahedron – Hexahedron

Figure 4.2 presents the HEX – HEX mesh for the identical, parallel, directly opposed rectangle case.

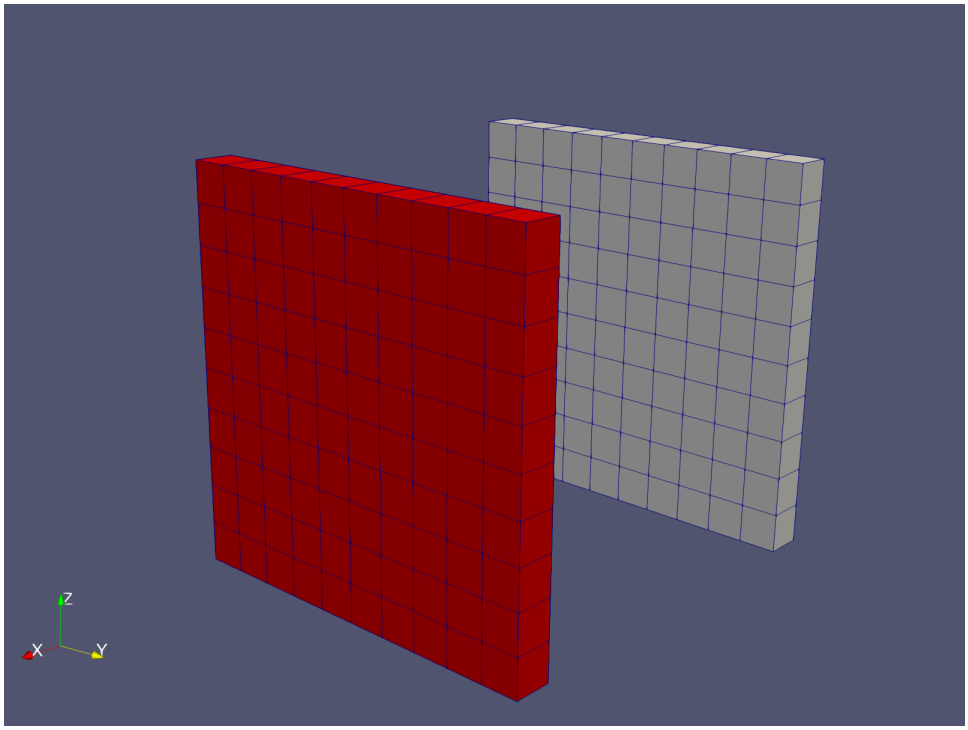


Figure 4.2: The HEX – HEX mesh.[1,2]

As expected, the view factor from each plate to the other is exactly the same. The computed view factor rounded to 16 digits is 0.2285656685503451. The absolute error of this view factor is 1.07637×10^{-10}. The relative error is $4.70923 \times 10^{-8}\%$.

[1]To some, it may appear that the plates are closer than 9 m. This is a byproduct of visualizing the geometric configuration with linear, perspective projection. Readers that are still doubtful are encouraged to explore the freely available code (see §3.5).

[2]Strictly speaking, only surfaces need to be meshed in order for view factors to be computed. It was necessary to use volumetric meshes for the view factor benchmarks because of how Gmsh, the mesh generator system in MOOSE and the EXODUS II data file system currently work. Using volumetric meshes does not significantly impact view factor computation performance. Under the hood, a surface mesh is constructed with CGAL from a provided volumetric mesh. The surface mesh is then used to efficiently conduct all view factor computations.

4.1.1.2 Triangular Prism – Hexahedron

Figure 4.3 presents the PRI – HEX mesh for the identical, parallel, directly opposed rectangle case.

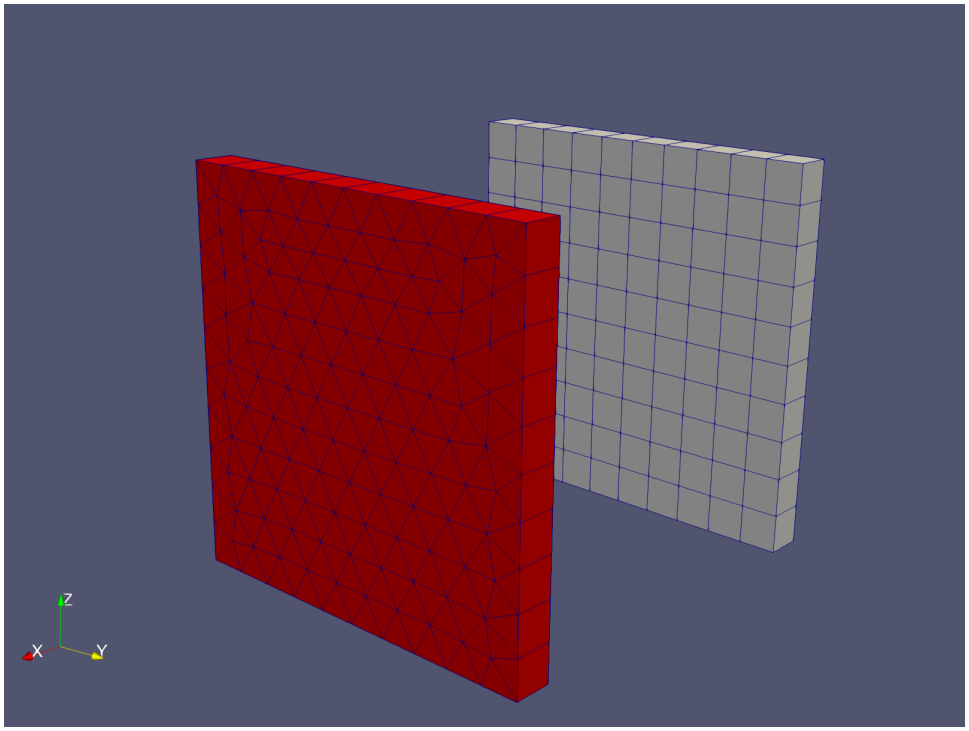


Figure 4.3: The PRI – HEX mesh.

The view factor from the HEX mesh to the PRI mesh ($F_{\text{HEX}\to\text{PRI}}$), and the complementary view factor ($F_{\text{PRI}\to\text{HEX}}$) are slightly different. They differ from each other by 1.38777×10^{-15}. Table 4.1 summarizes the computed view factors for the PRI – HEX mesh.

View Factor	Value	Absolute Error	Relative Error
$F_{\text{HEX}\to\text{PRI}}$	0.2285656685024703	5.97620×10^{-11}	2.61465×10^{-8} %
$F_{\text{PRI}\to\text{HEX}}$	0.2285656685024717	5.97633×10^{-11}	2.61471×10^{-8} %

Table 4.1: View factor results for the HEX – PRI mesh.

4.1.1.3 Triangular Pyramid – Hexahedron

Figure 4.4 presents the PYR – HEX mesh for the identical, parallel, directly opposed rectangle case.

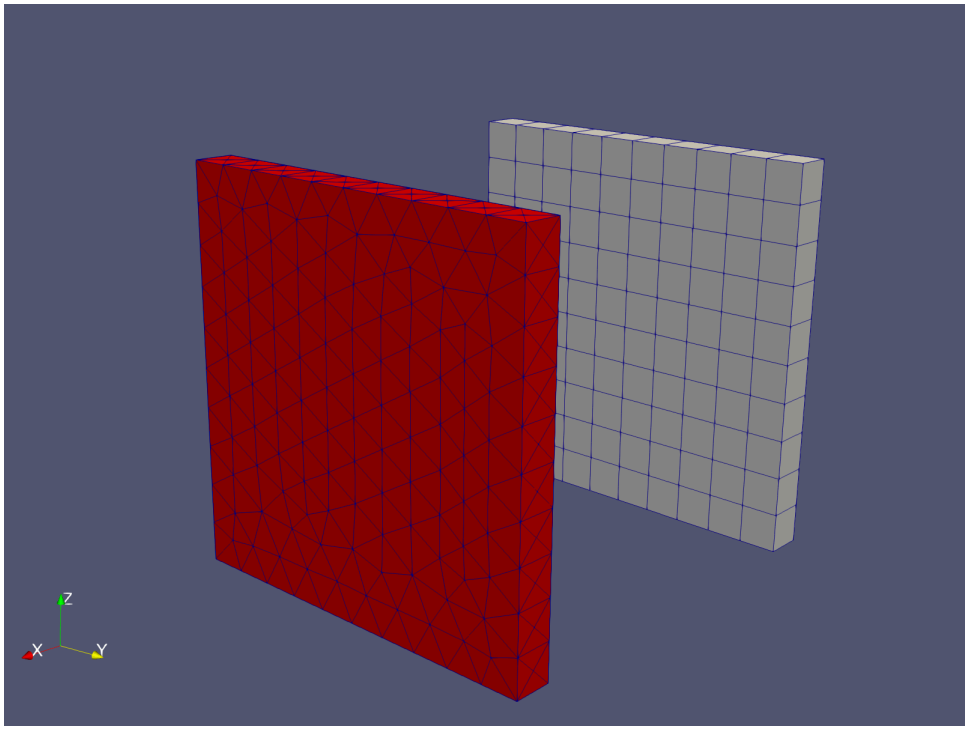


Figure 4.4: The PYR – HEX mesh.

The view factor from the HEX mesh to the PYR mesh differ from each other by 1.94289×10^{-16}. Table 4.2 summarizes the computed view factors for the PYR – HEX mesh.

View Factor	Value	Absolute Error	Relative Error
$F_{\text{HEX}\to\text{PYR}}$	0.2285656685058535	6.31451×10^{-11}	2.76266×10^{-8} %
$F_{\text{PYR}\to\text{HEX}}$	0.2285656685058532	6.31449×10^{-11}	2.76266×10^{-8} %

Table 4.2: View factor results for the PYR – HEX mesh.

4.1.1.4 Triangular Prism – Triangular Prism

Figure 4.5 presents the PRI – PRI mesh for the identical, parallel, directly opposed rectangle case.

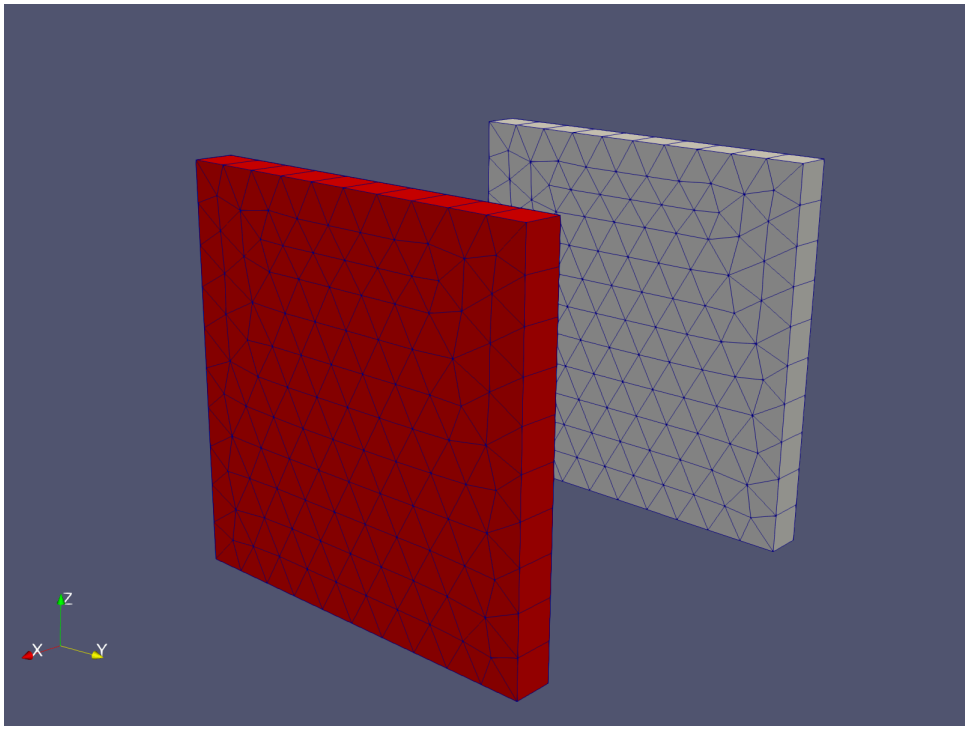


Figure 4.5: The PRI – PRI mesh.

The view factor from each plate to the other is exactly the same. The computed view factor rounded to 16 digits is 0.2285656684545930. The absolute error of this view factor is 1.18846×10^{-11}. The relative error is $5.19968 \times 10^{-9}\%$.

4.1.1.5 Triangular Pyramid – Triangular Prism

Figure 4.6 presents the PYR – PRI mesh for the identical, parallel, directly opposed rectangle case.

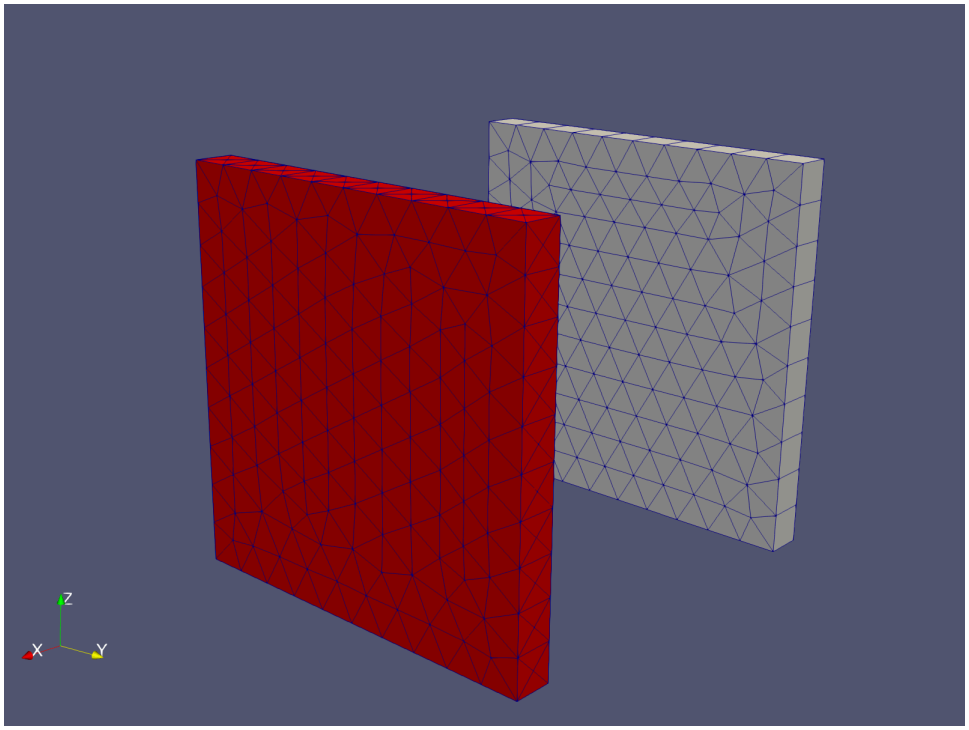


Figure 4.6: The PYR – PRI mesh.

The view factor from the PYR mesh to the PRI mesh differ from each other by 7.77156×10^{-16}. Table 4.3 summarizes the computed view factors for the PYR – PRI mesh.

View Factor	Value	Absolute Error	Relative Error
$F_{\text{PRI}\to\text{PYR}}$	0.2285656684579763	1.52679×10^{-11}	6.67989×10^{-9} %
$F_{\text{PYR}\to\text{PRI}}$	0.2285656684579755	1.52671×10^{-11}	6.67955×10^{-9} %

Table 4.3: View factor results for the PYR – PRI mesh.

4.1.1.6 Triangular Pyramid – Triangular Pyramid

Figure 4.7 presents the PYR – PYR mesh for the identical, parallel, directly opposed rectangle case.

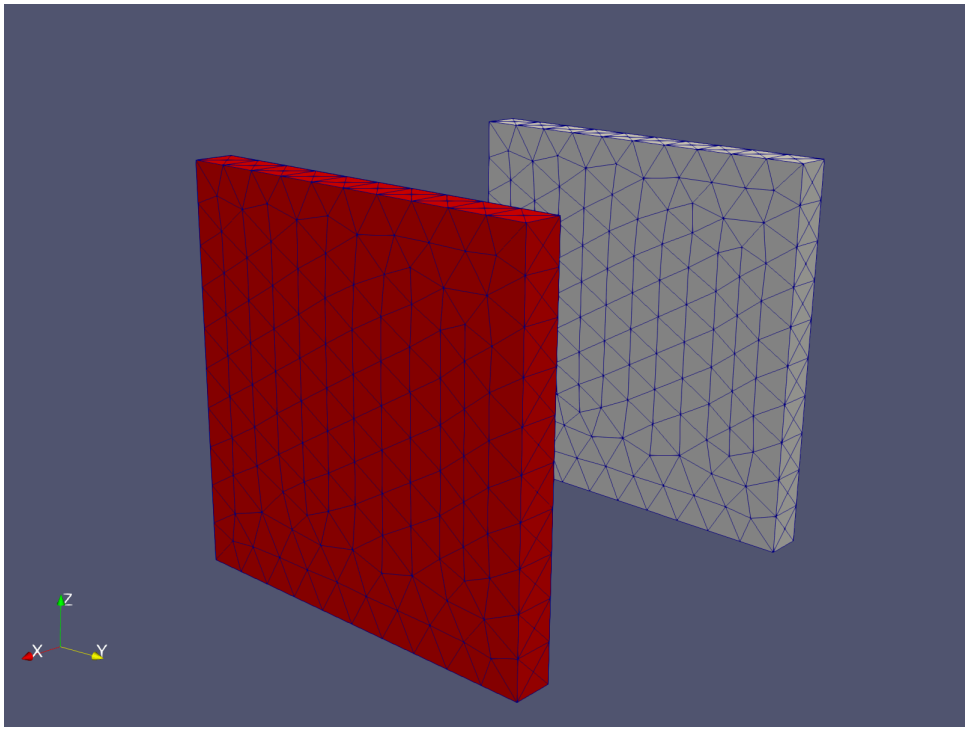


Figure 4.7: The PYR – PYR mesh.

The view factor from one plate to the other, and it's complementary view factor differ from each other by 2.19269×10^{-15}. This is due to the asymmetry in the mesh. Table 4.4 summarizes the computed view factors for the PYR – PYR mesh.

View Factor	Value	Absolute Error	Relative Error
$F_{A \to B}$	0.2285656684594533	1.67450×10^{-11}	7.32611×10^{-9} %
$F_{B \to A}$	0.2285656684594511	1.67428×10^{-11}	7.32515×10^{-9} %

Table 4.4: View factor results for the PYR – PYR mesh.

4.1.2 Mesh Refinement

The effect of mesh refinement on the the approximate view factor between surfaces is explored in the following two subsections. The first subsection explores the HEX – HEX mesh, and the second subsection explores the PYR – PYR mesh. The library, libMesh, provides a variety of quadrature rules. A few quadrature rules were selected to conduct the mesh refinement study. The selected view factors include:

- "GAUSS" – The standard tensor–product, Gauss–Legendre quadrature rules are used when integrating over quadrangles. The well–known Dunavant rules are used when integrating over triangles.

- "GRID" – Consists of quadrature points on a uniform grid, with order + 1 points on an edge. Unlike most libMesh quadrature rules, it does not reduce the integration error exponentially on smooth functions as the quadrature order is increased. It reduces the error quadratically. However, this error reduction is more reliable on non-smooth functions than it is for other quadrature rules. This quadrature type may be useful for integrating functions which have discontinuities on scales smaller than your element size – like when occlusion occurs and must be handled.

- "MONOMIAL" – These are alternate quadrature rules on tensor–product elements which can be useful when integrating monomial finite element bases. This class provides quadrature rules which are more efficient than tensor–product rules when they are available, and falls back on Gaussian quadrature rules otherwise.

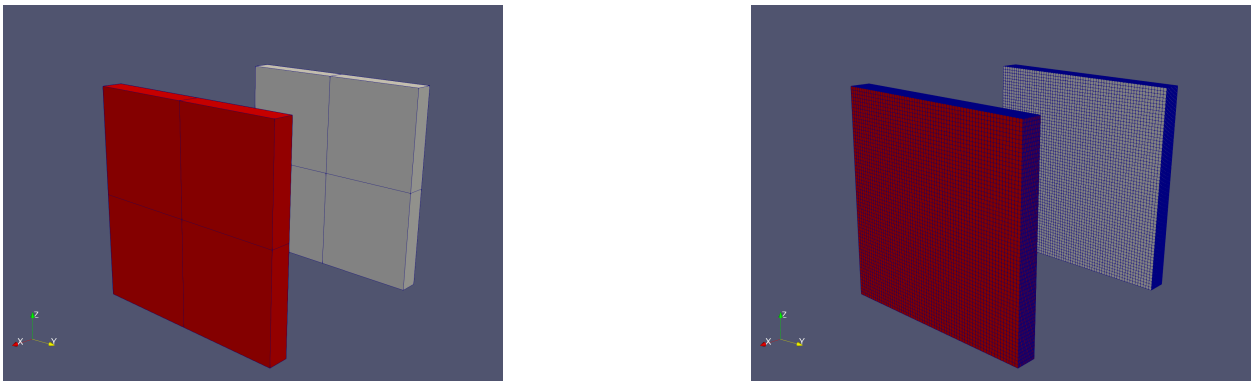
- "SIMPSON" – Simpson quadrature. This is the same thing as Newton–Cotes quadrature with three points. Simpson's rule can integrate polynomials of degree three exactly.

- "TRAP" – Trapezoidal quadrature. Sometimes also known as Newton–Cotes quadrature with two points. These rules sample at the corners and will integrate linear functions exactly.

More information about these quadrature rules can be found in the libMesh documentation and code [41]. The fourth-order quadrature rule was selected for each quadrature rule type. Floating point numbers were used in all aspects of the geometry computations.

4.1.2.1 Hexahedron – Hexahedron

The coarsest mesh consists of 4 elements in each plate – implying an elemental surface area of 25 m^2. Each level of refinement splits a hexahedral element into eight sub-elements. The highest level of refinement consists of 4,096 surface elements on each surface. This implies an elemental surface area of 0.0244140625 m^2 for the highest level of refinement. Figure 4.8 displays the two extremes of the meshes used.



(a) The coarse mesh.

(b) The fine mesh.

Figure 4.8: The two ends of the mesh refinement spectrum for the HEX – HEX configuration.

Figure 4.9 displays the absolute error versus the mesh refinement level for the fourth-order version of five different quadrature rules.

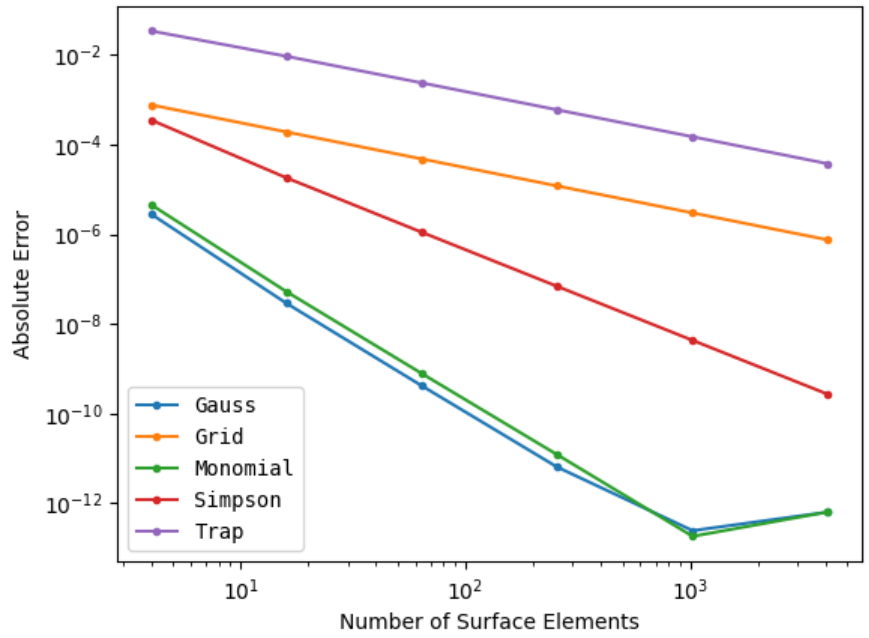


Figure 4.9: The mesh refinement results for the HEX – HEX configuration.

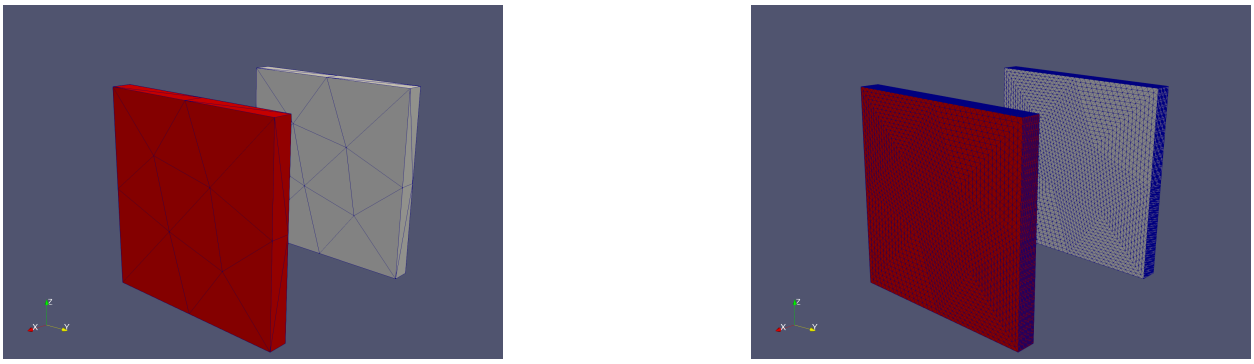
The "TRAP" quadrature rule has the slowest rate of convergence. The "GRID" quadrature rule fared only slightly better than the "TRAP" rule, and with roughly the same rate of convergence. The "SIMPSON" rule has an intermediate convergence rate, but it is still less than that of "GAUSS" and the "MONOMIAL" quadrature rules.

The "GAUSS" and "MONOMIAL" quadrature rules fared the best, and roughly have the same convergence rate. After the fourth refinement level round-off error accumulates and the absolute errors of these two quadrature rules stops converging. Because of their superior convergence rate either the "GAUSS" or "MONOMIAL" quadrature rules are preferred when calculating view factors between quadrangular surface elements.

4.1.2.2 Triangular Pyramid – Triangular Pyramid

The coarsest mesh consists of 14 elements on the surface of each plate – implying an average elemental surface area of 7.142857 m^2. Each level of refinement splits a tetrahedron element

into eight sub-elements. At the highest level of refinement there are 3,584 surface elements on each surface. This implies an elemental surface area of 0.0279017857 m^2 for the highest level of refinement. Figure 4.10 displays the two extremes of the meshes used.



(a) The coarse mesh.

(b) The fine mesh.

Figure 4.10: The two ends of the mesh refinement spectrum for the PYR – PYR configuration.

Figure 4.11 displays the absolute error versus the mesh refinement level for the fourth-order version of five different quadrature rules.

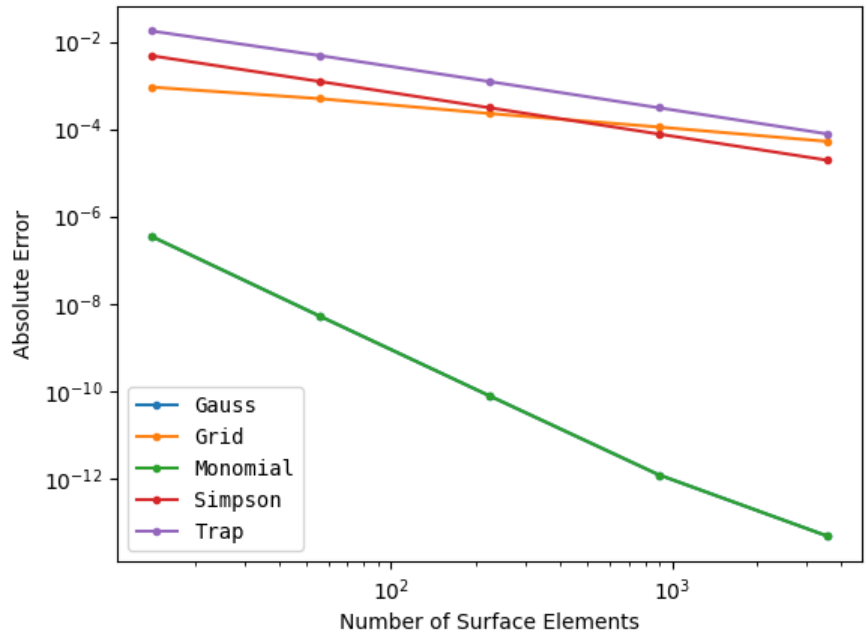
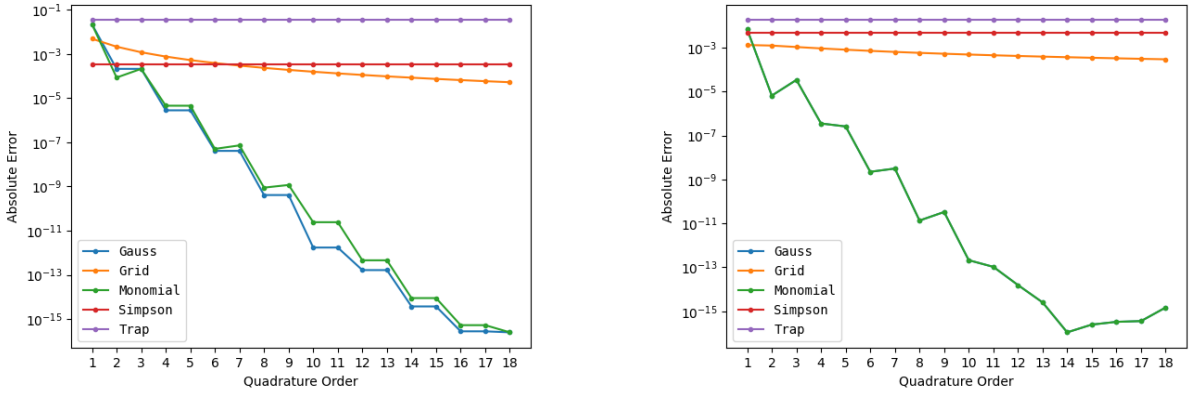


Figure 4.11: The mesh refinement results for the PYR – PYR configuration.

It should be noted that the "MONOMIAL" quadrature rule falls back to the "GAUSS" quadrature rule for triangular surface elements. Hence, the "GAUSS" and "MONOMIAL" lines are identical. Similar to the case of quadrangular surface elements, either the "GAUSS" or "MONOMIAL" quadrature rules are preferred when calculating view factors between triangular surface elements.

4.1.3 Quadrature Refinement

A quadrature refinement study was conducted on the coarsest meshes (4 surface elements for the HEX – HEX configuration, and 14 surface elements for the PYR – PYR configuration). Figure 4.12 presents the absolute error of the view factor versus the quadrature refinement level for both the HEX – HEX and PYR – PYR meshes.



(a) The HEX – HEX mesh.

(b) The PYR – PYR mesh.

Figure 4.12: The quadrature refinement results.

As expected, “SIMPSON” and “TRAP” are simply flat lines as the quadrature order cannot be increased. The “GRID” quadrature rule only gradually decreases the absolute error as the quadrature order is increased. Again, for both configurations “GAUSS” or “MONOMIAL” quadrature rules exhibit the best convergence rate. For these rules, every other quadrature order reduces the absolute error by about two orders of magnitude.

4.2 Ring to Parallel Coaxial Ring

Consider the geometric configuration in Figure 4.13.

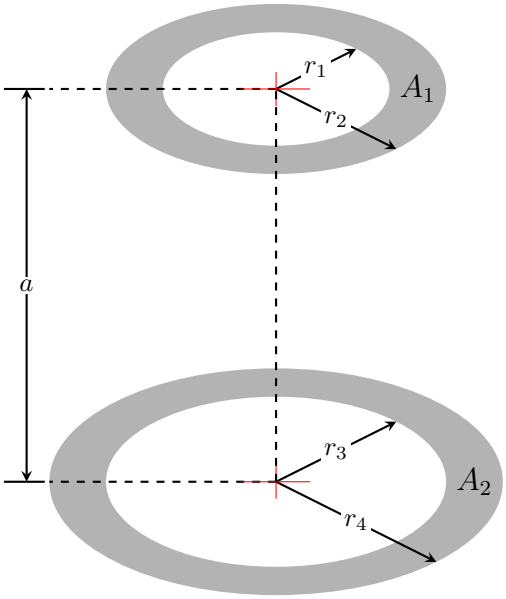


Figure 4.13: The geometric configuration for ring to parallel coaxial ring.

The analytic solution for the view factor is given by [42, 32]:

$$F_{A_1 \to A_2} = \frac{1}{2\left(R_2^2 - 1\right)} \left[\left(\left(R_2^2 + R_3^2 + H^2\right)^2 - \left(2R_3R_2\right)^2 \right)^{1/2} - \left(\left(R_2^2 + R_4^2 + H^2\right)^2 - \left(2R_2R_4\right)^2 \right)^{1/2} + \left(\left(1 + R_4^2 + H^2\right)^2 - \left(2R_4\right)^2 \right)^{1/2} - \left(\left(1 + R_3^2 + H^2\right)^2 - \left(2R_3\right)^2 \right)^{1/2} \right], \quad (4.2)$$

where $H \equiv a/r_1$, $R_2 \equiv r_2/r_1$, $R_3 \equiv r_3/r_1$ and $R_4 \equiv r_4/r_1$. For the sake of verification an arbitrary set of parameters for the configuration was selected. The parameters that were selected are: $r_1 = r_3 = 2$ cm, $r_2 = 6$ cm, $r_4 = 4$ cm and $a = 15$ cm. The analytic solution (rounded to 16 digits) for the view factor of the small ring to the large ring is 0.1131618692082199. A mesh

was generated out of triangular pyramid elements for this geometric configuration. Figure 4.14 displays the mesh generated for this example case.

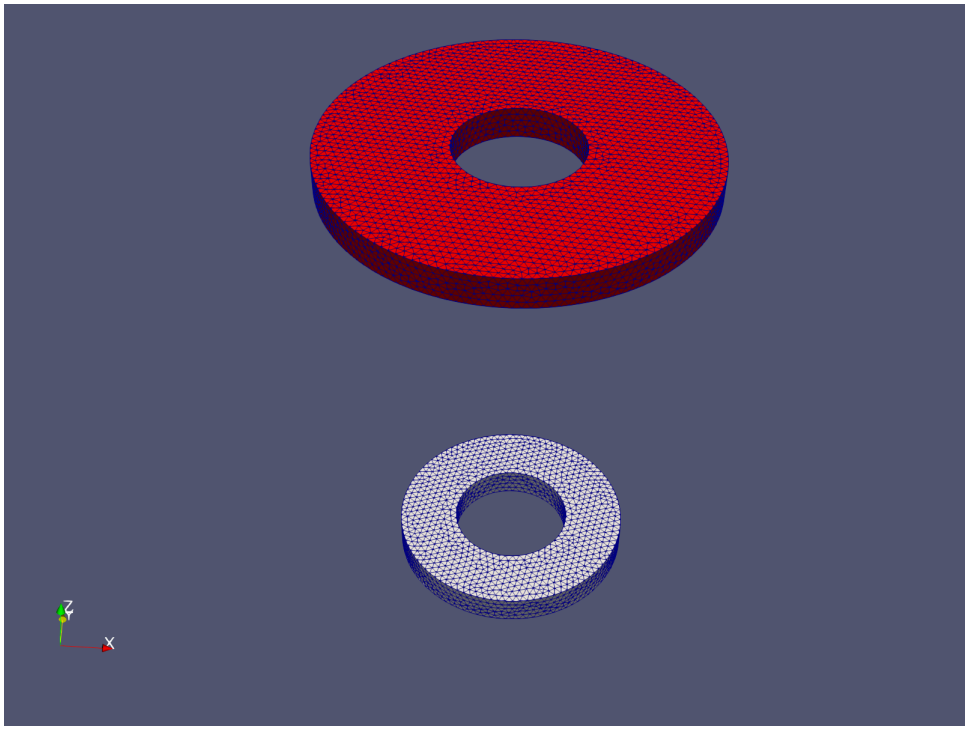


Figure 4.14: The parallel coaxial ring mesh.

A mesh size of 0.25 cm was specified for both rings. The smaller ring has 1,478 surface elements on the surface of interest – resulting in an average element surface area of 0.02550 cm^2. The larger ring has 3,860 surface elements on the surface of interest – resulting in an average element surface area of 0.02604 cm^2.

Table 4.5 presents the errors of the computed view factors from the small ring to the large ring for all number type combinations. The fourth order Gauss quadrature rule was used for each computation. The "NONE" occlusion detection was used because it is known that this configuration does not involve occlusion.

As expected, the solutions are consistent for this simple geometry irrespective of the underlying number types that are used. It should be noted that the run time significantly increases when exact

Base N. Type	**Collision N. Type**	**Quad. N. Type**	**Abs. Error**	**Rel. Error**
Apprx.	Apprx.	Apprx.	1.4447×10^{-5}	1.276×10^{-2} %
Exact	Apprx.	Apprx.	1.4447×10^{-5}	1.276×10^{-2} %
Apprx.	Exact	Apprx.	1.4447×10^{-5}	1.276×10^{-2} %
Exact	Exact	Apprx.	1.4447×10^{-5}	1.276×10^{-2} %
Apprx.	Apprx.	Exact	1.4447×10^{-5}	1.276×10^{-2} %
Exact	Apprx.	Exact	1.4447×10^{-5}	1.276×10^{-2} %
Apprx.	Exact	Exact	1.4447×10^{-5}	1.276×10^{-2} %
Exact	Exact	Exact	1.4447×10^{-5}	1.276×10^{-2} %

Table 4.5: View factor results for the parallel coaxial ring example.

is selected for the quadrature number type. It is highly recommended that one should use the approximate number type if it is sufficient for the application, otherwise the computation will be slow.

4.3 Ring to Parallel Coaxial Ring with a Blocking Coaxial Cylinder

Consider the geometric configuration in Figure 4.15.

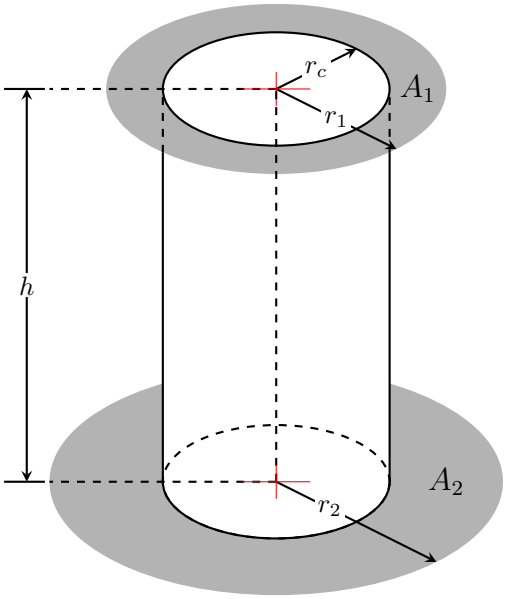


Figure 4.15: The geometric configuration for ring to parallel coaxial ring with a blocking coaxial cylinder.

The analytic solution for the view factor is given by [42, 32]:

$$
\begin{aligned}
F_{A_1 \to A_2} = \frac{1}{\pi A} \Bigg[& \frac{A}{2} \arccos \frac{R_c}{R_2} + \frac{B}{2} \arccos \frac{R_c}{R_1} + 2R_c \left(\arctan Y - \arctan A^{1/2} - \arctan B^{1/2} \right) \\
& - \left[\left(1 + C^2\right) \left(1 + D^2\right) \right]^{1/2} \arctan \left[\frac{\left(1 + C^2\right) \left(Y^2 - D^2\right)}{\left(1 + D^2\right) \left(C^2 - Y^2\right)} \right]^{1/2} \\
& + \left[\left(1 + (R_1 + R_c)^2\right) \left(1 + (R_1 - R_c)^2\right) \right]^{1/2} \arctan \left[\frac{\left(1 + (R_1 + R_c)^2\right) (R_1 - R_c)}{\left(1 + (R_1 - R_c)^2\right) (R_1 + R_c)} \right]^{1/2} \\
& + \left[\left(1 + (R_2 + R_c)^2\right) \left(1 + (R_2 - R_c)^2\right) \right]^{1/2} \arctan \left[\frac{\left(1 + (R_2 + R_c)^2\right) (R_2 - R_c)}{\left(1 + (R_2 - R_c)^2\right) (R_2 + R_c)} \right]^{1/2} \Bigg]
\end{aligned} \tag{4.3}
$$

where $R_1 \equiv r_1/h$, $R_2 \equiv r_2/h$, $R_c \equiv r_c/h$ and $A \equiv R_1^2 - R_c^2$, $B \equiv R_2^2 - R_c^2$, $C \equiv R_2 + R_1$, $D \equiv R_2 - R_1$ and $Y \equiv A^{1/2} + B^{1/2}$.

The following two subsections present the view factor results for a geometric configuration of this type. Dimensions of r_1 = 6 cm, r_2 = 4 cm, r_c = 2 cm and h = 15 cm are used. The exact solution of the view factor from the small ring to the large ring (rounded to 16 digits) is 0.0725555875115452.

These dimensions were purposefully selected to match the dimensions in the previous section. The following subsections present the view factor results computed on two meshes. The first mesh consists of the geometric configuration discretized by tetrahedron. It should be noted that the Delaunay algorithm in Gmsh does not preserve the convexity of the cylinder. The second mesh is discretized by hexahedra, and is meshed in a way that preserves the convexity of the occluding cylinder.

4.3.1 Concave Mesh

Figure 4.16 is a visualization of the configuration discretized by tetrahedron.

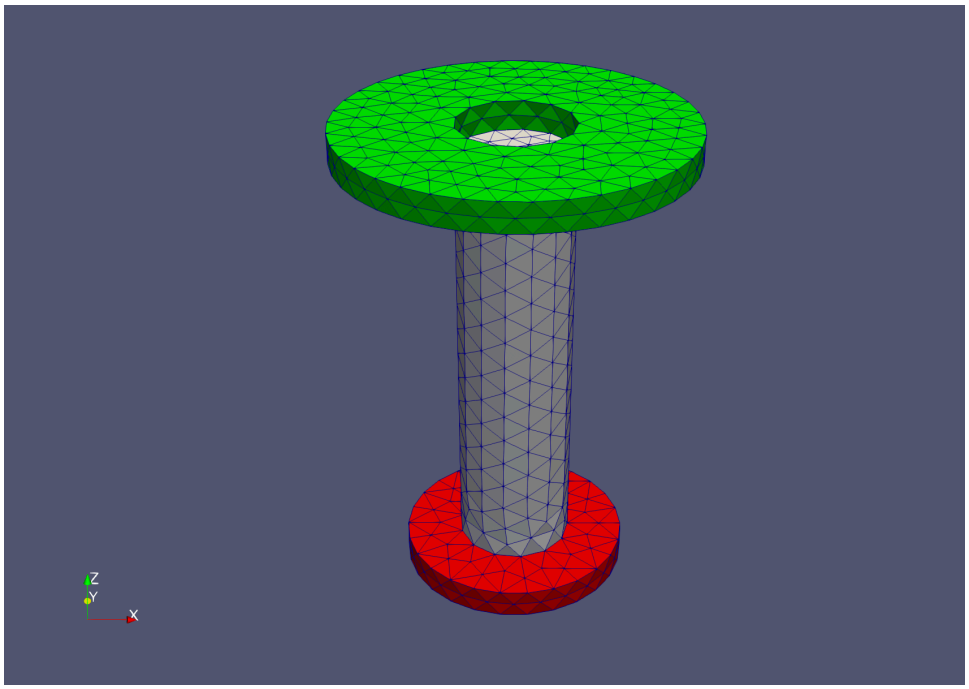


Figure 4.16: The concave mesh for the parallel coaxial ring with a blocking coaxial cylinder example.

A mesh size of 1 cm was used to discretize each entity. The cylinder's surface consists of 472 triangular elements resulting in an average surface area of 0.399355 cm^2. The smaller ring's surface consists of 91 triangular elements resulting in an average surface area of 0.414276 cm^2. The larger ring's surface consists of 261 triangular elements resulting in an average surface area of 0.385176 cm^2. Table 4.6 presents some view factor results computed for this mesh.

Occlusion Detection	$\mathbf{F}_{\text{SMALL}\to\text{LARGE}}$	**Abs. Error**	$\mathbf{F}_{\text{CYL}\to\text{CYL}}$	**Runtime**
None	1.1336×10^{-1}	4.08×10^{-2}	1.7868×10^{-4}	0.59s
Brute Force w/ BBox	7.2418×10^{-2}	1.37×10^{-4}	1.8190×10^{-4}	79.95s
Brute Force w/o BBox	7.2418×10^{-2}	1.37×10^{-4}	1.8190×10^{-4}	109.15 s

Table 4.6: View factor results of the concave mesh for the parallel coaxial ring with a blocking coaxial cylinder example.

As expected, selecting to not use occlusion detection results in the wrong answer that is more consistent with the result without the occluding cylinder. Using occlusion detection results in an answer that is close to the analytic solution. The view factor of the cylinder to itself is not zero because of the concavity of the cylinder's mesh. For this particular configuration, using bounding boxes in the occlusion detection resulted in a faster runttime than not using them.

4.3.2 Convex Mesh

Figure 4.17 is a visualization of the mesh discretized by hexahedra.

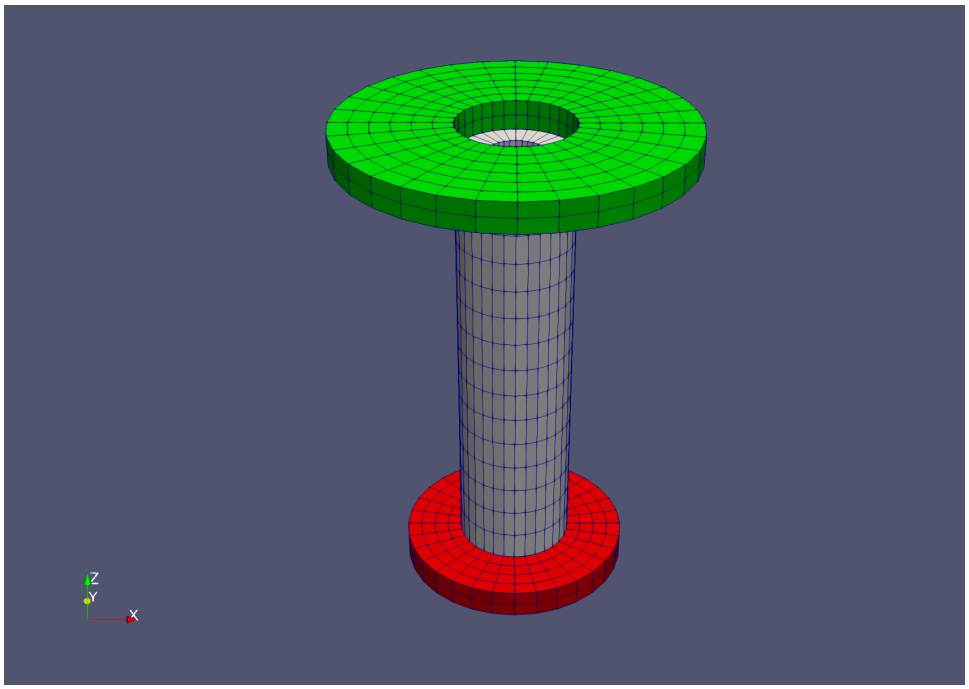


Figure 4.17: The convex mesh for the parallel coaxial ring with a blocking coaxial cylinder example.

The cylinder's surface consists of 480 quadrangular elements resulting in an average surface area of 0.392699 cm^2. The smaller ring's surface consists of 128 quadrangular elements resulting in an average surface area of 0.294524 cm^2. The larger ring's surface consists of 192 quadrangular elements resulting in an average surface area of 0.523599 cm^2. Table 4.7 presents some view factor results computed for this mesh.

Occlusion Detection	$\mathbf{F}_{\text{SMALL}\rightarrow\text{LARGE}}$	**Abs. Error**	$\mathbf{F}_{\text{CYL}\rightarrow\text{CYL}}$	**Runtime**
None	1.1258×10^{-1}	4.08×10^{-2}	0.0	0.70 s
Brute Force w/ BBox	7.2092×10^{-2}	4.64×10^{-4}	0.0	160.97 s
Brute Force w/o BBox	7.2092×10^{-2}	4.64×10^{-4}	0.0	239.34 s

Table 4.7: View factor results of the convex mesh for the parallel coaxial ring with a blocking coaxial cylinder example.

The results are consistent with those presented in the previous subsection, but the view factor of the cylinder to itself is now zero. The runtimes are approximately double the runtimes in the previous subsection because quadrangels are represented as two triangles at the lower level of abstraction in the collision detection routine. Hence, twice as many collision detection computations are conducted when calculating the view factor between each surface.

4.4 W-Tube

This final section presents a benchmark of the view factor computation on a geometry of interest for real applications. The original work was conducted by He et al. [15] as a means for exploring the design space for a component used in continuous annealing furnaces. The results presented in the report were computed using the monte carlo method. Figure 4.18 is an example mesh used in the benchmark calculations conducted with Phoenix.

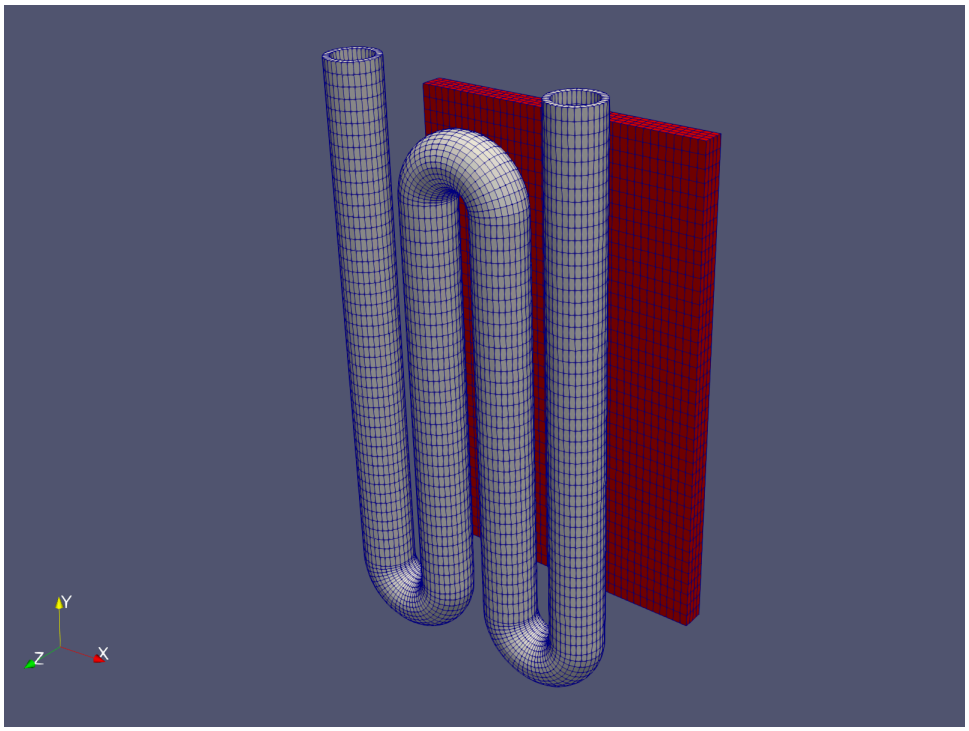


Figure 4.18: An example of the W-Tube mesh.

All geometric configurations of the mesh were programatically generated with the python API of Gmsh. In each case, the same geometry of the W-tube is used, and only the rectangular plane's configuration is varied. Parameters for the geometric configuration of the W-tube can be found in the original report. For the rectangular plane, the parameter a is it's width, b is it's height and s is the perpendicular distance from the rectangular plane (i.e. the cold-rolled strip) to the xOy plane

(i.e. the symmetry plane of W-tube parallel to the strip). More information can be found in the original report.

A discretization similar to the one presented in the original report was used to represent the W-tube. The plane is meshed by regular quadrangles – 40 in each direction. Approximate number types were used, and no occlusion detection was used since it is known *a priori* that occlusion does not occur in this configuration. Table 4.8 presents a comparison of the results generated by Phoenix and those presented in the original report.

All of the results are in close agreement with the exception of case number 8. At this time it is not certain why there is a disagreement in the results of this one case. One possibility is that the results computed in the original report did not have a statistical tolerance that was sufficiently strict. Another is that there was a transcription error when generating the report. An unlikely possibility is that there is an undiscovered error in Phoenix, but the benchmarks presented in the previous section suggest that other possibilities for this discrepancy should be explored first if it is of significant concern.

No.	s, mm	a, mm	b, mm	$F^{\text{ref}}_{0\to W}$	$F^{\text{calc}}_{0\to W}$	Rel. Error	$F^{\text{ref}}_{W\to 0}$	$F^{\text{calc}}_{W\to 0}$	Rel. Error
1	450	1320	900	0.5474	0.5494	0.357%	0.1029	0.1035	0.578%
2	450	1320	950	0.5465	0.5487	0.408%	0.1085	0.1091	0.572%
3	450	1320	1000	0.5457	0.548	0.430%	0.114	0.1147	0.632%
4	450	1320	1050	0.5452	0.5473	0.386%	0.1196	0.1203	0.580%
5	450	1320	1100	0.5434	0.5465	0.569%	0.1249	0.1258	0.748%
6	450	1320	1150	0.5446	0.5456	0.184%	0.1308	0.1313	0.414%
7	450	1320	1200	0.5455	0.5446	0.158%	0.1368	0.1368	0.007%
8	450	1320	1250	0.5427	0.5436	0.163%	0.1308	0.1422	8.741%
9	450	1320	1300	0.5425	0.5424	0.012%	0.1474	0.1476	0.143%
10	450	1320	1350	0.5407	0.5412	0.090%	0.1525	0.1529	0.285%
11	450	1320	1400	0.5368	0.5398	0.563%	0.157	0.1582	0.764%
12	450	1320	1450	0.5383	0.5383	0.008%	0.1631	0.1634	0.184%
13	450	1320	1500	0.5373	0.5367	0.106%	0.1684	0.1685	0.076%
14	450	1320	1550	0.5342	0.535	0.146%	0.173	0.1736	0.334%
15	450	1320	1600	0.5342	0.5331	0.211%	0.1786	0.1785	0.034%
16	450	1320	1650	0.5298	0.531	0.229%	0.1826	0.1834	0.441%
17	450	1320	1700	0.5315	0.5288	0.512%	0.1888	0.1882	0.334%
18	450	1320	1750	0.5257	0.5264	0.127%	0.1922	0.1928	0.323%
19	450	1320	1800	0.5217	0.5238	0.397%	0.1962	0.1974	0.587%
20	450	1320	1850	0.5203	0.521	0.131%	0.2011	0.2018	0.325%
21	450	1320	1900	0.5168	0.518	0.231%	0.2052	0.206	0.398%
22	450	1320	1950	0.5137	0.5148	0.215%	0.2093	0.2101	0.400%
23	450	1320	2000	0.5099	0.5114	0.297%	0.2131	0.2141	0.472%
24	450	1320	2050	0.5066	0.5078	0.242%	0.217	0.2179	0.423%
25	450	1320	2100	0.5027	0.504	0.267%	0.2205	0.2216	0.485%
26	450	1370	2100	0.4952	0.4982	0.597%	0.2255	0.2273	0.788%
27	450	1420	2100	0.4889	0.4921	0.651%	0.2307	0.2327	0.867%
28	450	1470	2100	0.4831	0.4859	0.571%	0.2361	0.2378	0.740%
29	450	1520	2100	0.4771	0.4795	0.506%	0.241	0.2427	0.716%
30	350	1320	2100	0.576	0.5773	0.226%	0.2527	0.2538	0.425%
31	400	1320	2100	0.538	0.5388	0.154%	0.2361	0.2369	0.322%
32	450	1320	2100	0.5027	0.504	0.267%	0.2205	0.2216	0.485%
33	500	1320	2100	0.4704	0.4725	0.444%	0.2064	0.2077	0.630%
34	550	1320	2100	0.4415	0.4437	0.491%	0.1937	0.195	0.686%

Table 4.8: View factor results for various W-Tube configurations.

5. HEAT TRANSFER EXAMPLES AND RESULTS

To demonstrate the heat transfer capabilities of Phoenix a few benchmark cases were generated. First, two examples of a single plate are presented and used to benchmark the radiation emission boundary conditions. Next, two examples of the parallel and directly opposed plates (see §4.1) are presented. These examples are presented to demonstrate and benchmark the capabilities of the area–averaged, blackbody radiation emission and irradiation boundary conditions. It should be noted that material properties of $k = 1$ W/m $\cdot$ K, $\rho = 1$ kg/m^3 and $c_p = 1$ J/kg $\cdot$ K were used for every entity in all of the examples presented in this chapter. All plates are discretized into 40 $\times$ 40 $\times$ 4 elements. All of the details of each simulation can be found in the Phoenix repository (see §3.5 and checkout the tag "thesis").

5.1 Radiation Emission from a Single Plate

Consider a 10 m $\times$ 10 m $\times$ 1 m plate with a uniform heat generation of 10,000 W/m^3. Adiabatic boundary conditions [3] are applied to each side of the plate except for the 10 m $\times$ 10 m face at $z = 1$ m. On that face, blackbody radiation is being emitted in accordance with Stefan–Boltzmann law. One can arrive at the exact surface temperature of the emitting surface by the following an energy balance argument. The total amount of heat generated in the plate is

$$Q_{\text{gen}} = Atq''', \tag{5.1}$$

where A is the surface area of the boundary that is emitting radiation and t is the thickness of the plate (in this problem $t = 1$ m). The total amount of energy emitted from the plate via radiation is derived from Stefan-Boltzmann law:

[3] To clarify, adiabatic boundary conditions means that $q'' \equiv (-k\nabla T) \cdot \hat{n} = 0$. Notice that from Eq. (2.11) and Eq. (2.12), $K_i^{\text{bndry}}(\vec{T}) \equiv \int_{\partial\Omega_e} \psi_i \, (-k\nabla T) \cdot \hat{n} \, dA$. Hence, $K_i^{\text{bndry}}(\vec{T}) = 0$ for adiabatic boundary conditions. Adiabatic boundary conditions can be specified in MOOSE in at least two ways: 1) using the "NeumannBC" with a value of zero, or 2) not specifying a boundary condition on the boundaries that are adiabatic. For simplicity, the second approach was taken for implementing the heat conduction examples.

$$Q_{\text{rad}} = A\sigma T^4_{\text{rad, surf}}, \tag{5.2}$$

where $T_{\text{rad, surf}}$ is the surface temperature of the face that is emitting radiation. Equating (5.1) and (5.2), one arrives at the expression for the surface temperature:

$$T_{\text{rad, surf}} = \left(\frac{tq'''}{\sigma}\right)^{1/4}. \tag{5.3}$$

For the example at hand, the surface temperature is 648.0329 K.

This value can be used to derive the analytic solution for the temperature profile in the plate. Since all of the 10 m $\times$ 1 m boundaries are adiabatic, and we are seeking the steady–state solution, Eq. (2.1) reduces to the differential equation:

$$-\frac{\partial^2 T}{\partial z^2} = q'''. \tag{5.4}$$

Integrating twice over z, and applying the boundary conditions $\frac{\partial T}{\partial z}\big|_{z=0} = 0$ and $T(z = 1) = T_{\text{rad, surf}}$, we arrive at the analytic solution for this example:

$$T(x, y, z) = T_{\text{rad, surf}} + \frac{q'''}{2}\left(1 - z^2\right), \tag{5.5}$$

where $0 \leq x \leq 10$ m, $0 \leq y \leq 10$ m, $0 \leq z \leq 1$ m. From this, the peak temperature in the plate is computed to be: $T(x, y, z = 0) = 5,648.0329$ K.

Two simulations of this problem were run in Phoenix: 1) representing the radiation boundary condition locally by the Stefan–Boltzmann law, and 2) representing the radiation boundary condition by the surface–averaged T^4, and then using Stefan–Boltzmann law to determine the approximate emitted radiation (see §2.5.1). It is expected that the solutions for each simulation will be identical because the temperature profile on the emitting surface should be constant. The purpose of running these simulations is to ensure that the implementations are correct for the surface–averaging "UserObject" and the surface–averaged radiation emission boundary condition.

Figure 5.1 is a visualization of the temperature profile generated by simulation 2.

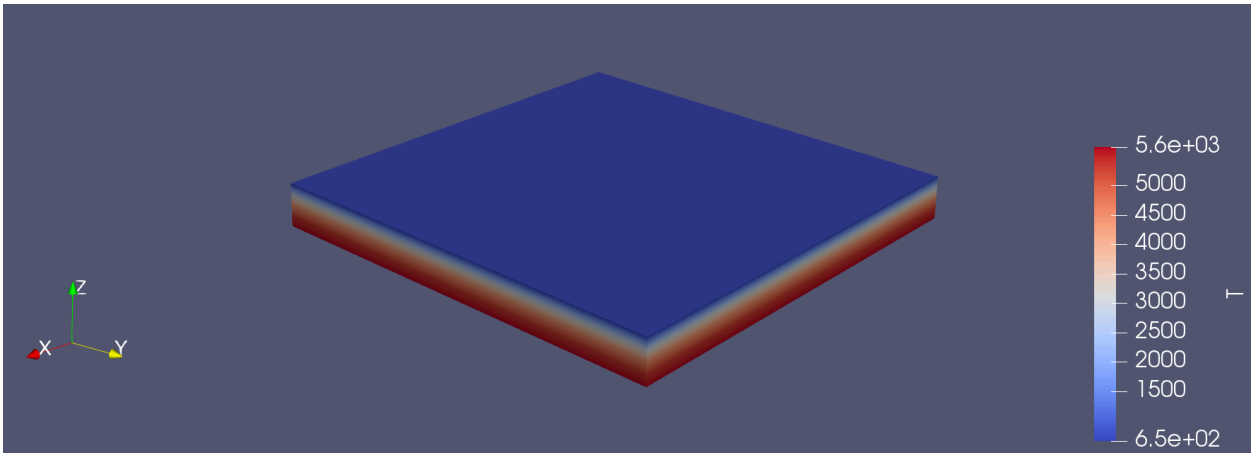


Figure 5.1: Steady-state temperature profile of surface–averaged radiation emission for the emitting plate example.

Visually, the maximum and the minimum temperatures are in agreement with those given by the analytic solution, and the temperature profile is identical with the temperature profile generated by simulation 1. Figure 5.2 is a plot of the temperature profiles sampled along the line centered at $x = 5$ m and $y = 5$ m, and spanning $0 \leq z \leq 1$ m.

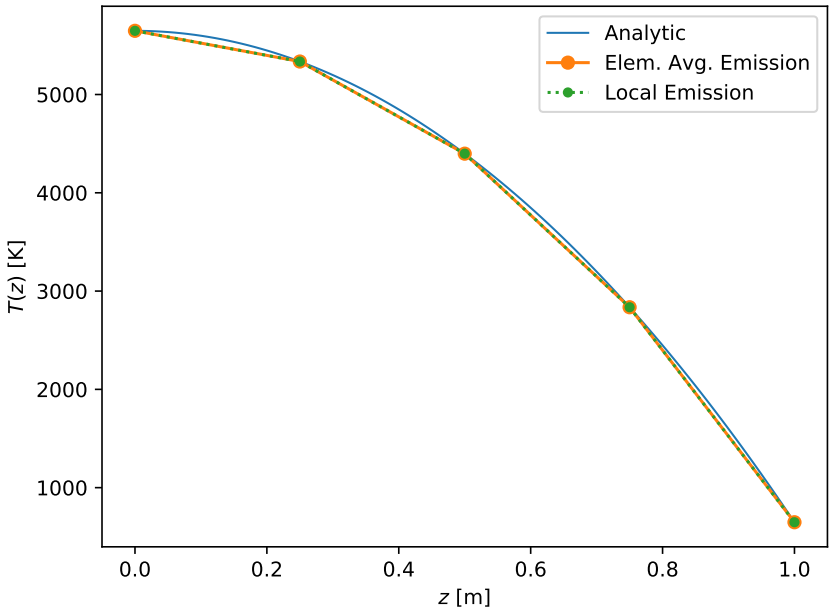


Figure 5.2: Steady–state, line–sampled temperature profiles for the emitting plate example.

As expected, the simulated solutions are in good agreement with the analytic solution, even with the plate being coarsely discretized. Now that a base–line has been established for the emission boundary condition both the emission and irradiation boundary conditions will be exercised and benchmarked in the remaining sections of this chapter.

5.2 Parallel and Directly Opposed Plates with Symmetric Heat Generation

The first example problem consists of adiabatic boundary conditions applied to all boundaries that are not communicating via radiation heat exchange. The right boundary of the left plate and the left boundary of the right plate are communicating. The plate on the left has a uniform heat generation of 10,000 W/m^3. The plate on the right does not have heat generation.

A reference solution was generated with ANSYS FLUENT v.19.02. In that simulation, each plate was discretized with 100 $\times$ 100 elements in yz planes. The following section will present the solutions generated by Phoenix and ANSYS FLUENT to benchmark the radiation boundary conditions implemented in Phoenix.

Figure 5.3 presents the steady–state temperature distributions of both plates.

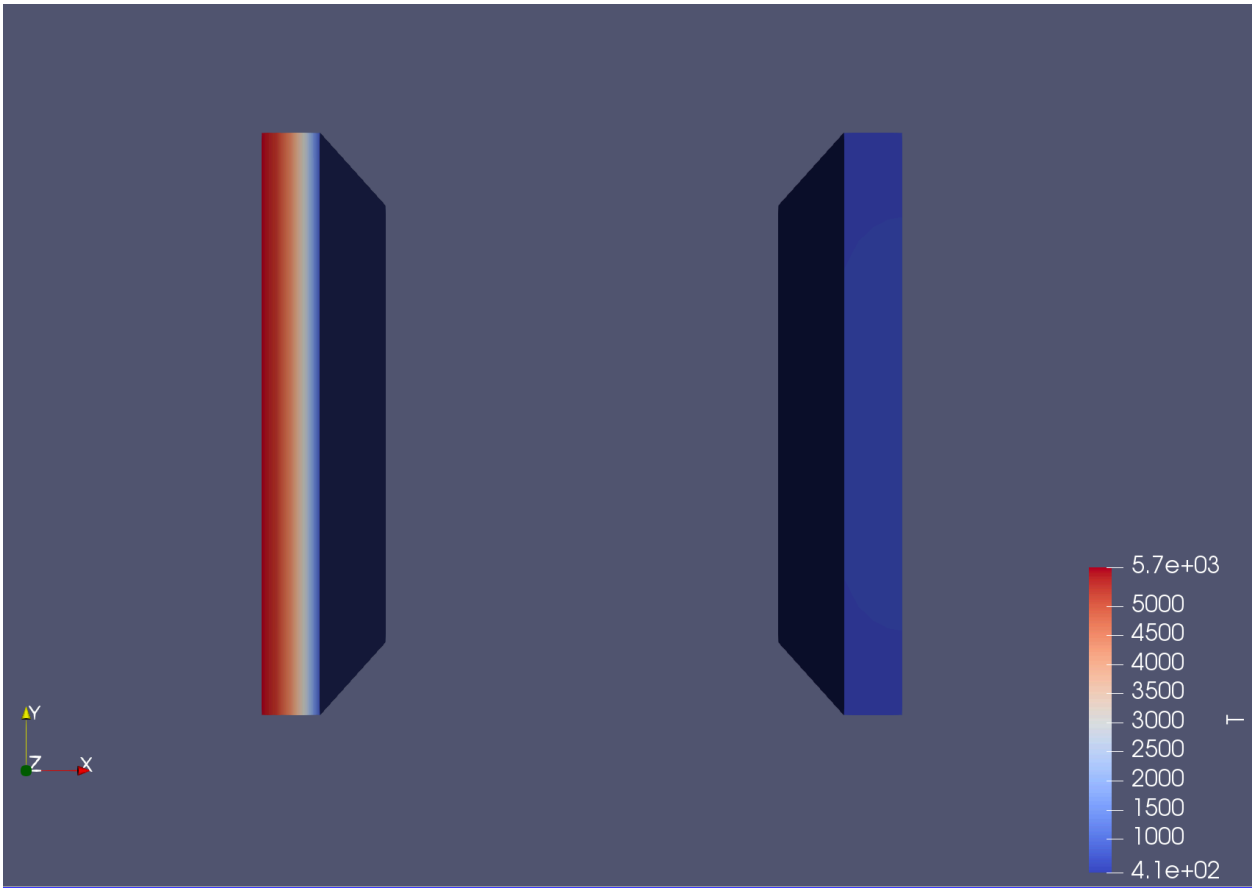
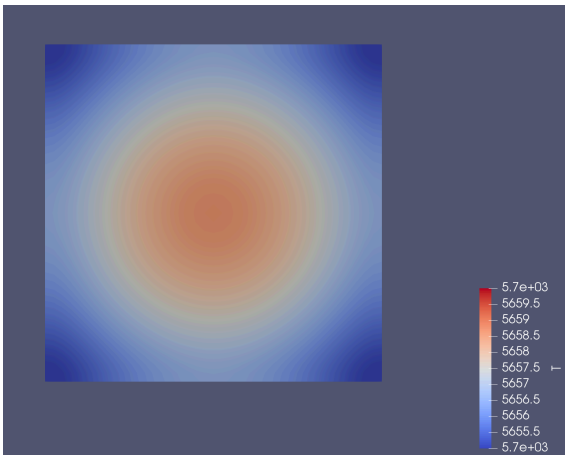
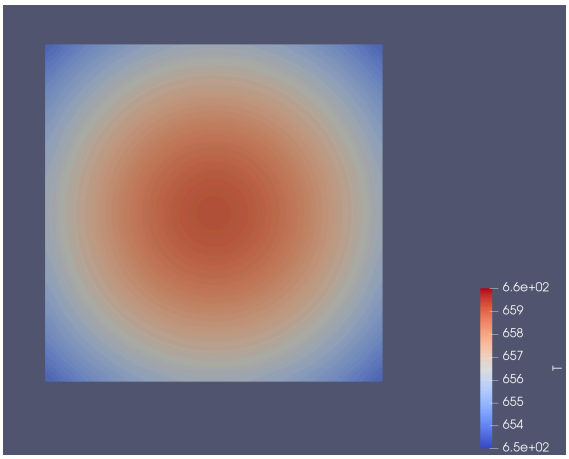


Figure 5.3: Steady-state temperature profiles of the symmetric heating example.

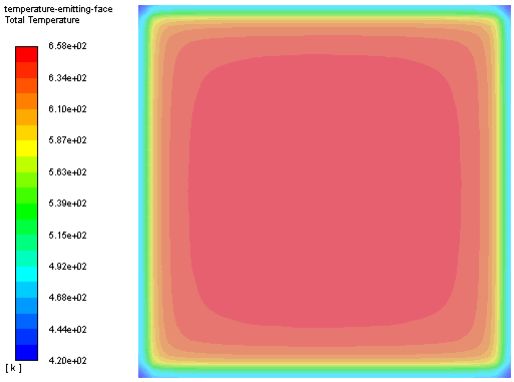
Figure 5.4 shows the temperature profiles of the 10 m $\times$ 10 m surfaces of the plate with heat generation.



(a) Adiabatic Surface (Phoenix)



(b) Radiation Surface (Phoenix)



(c) Radiation Surface (ANSYS FLUENT)

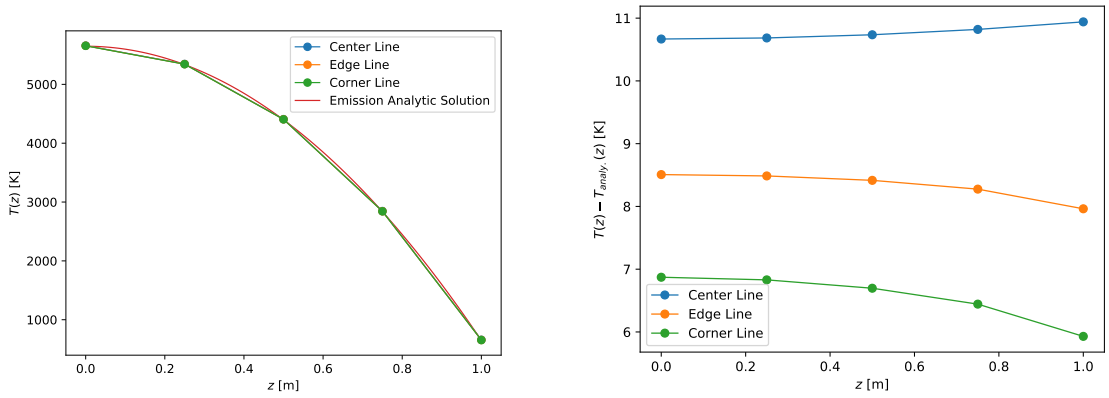
Figure 5.4: Steady-state, surface temperature profiles of the plate with heat generation in the symmetric heating example.

For the simulation conducted with Phoenix, one will notice that the temperature range of the adiabatic surface is 5,655 – 5,660 K and the temperature range of the radiation surface is 653 – 660 K. These temperature ranges are higher than the surface temperatures for the example in the previous section (5,648.0329 K and 648.0329 K, respectively). This makes sense because a small portion of the energy that is emitted from the plate eventually returns to it by being absorbed and re–emitted by the other plate. Ultimately, this increases the temperature of the plate with heat generation. The nonuniform temperature profiles, which exhibit fourfold symmetry, are a result of

the interaction with the other plate and the geometric configuration.

One will notice that the temperatures reported in the solution generated by ANSYS FLUENT (figure 5.4c) have a range of 658 – 420 K. At this time it is uncertain why the temperatures in this simulation are like this, but it is safe to say that they are incorrect.

Figure 5.5 displays plots generated by line–sampling the temperature within the plate with heat generation. These plots were generated from the Phoenix simulation. Three locations on the plate's face were selected to take the line–samples: the center, the middle of an edge and along the corner.



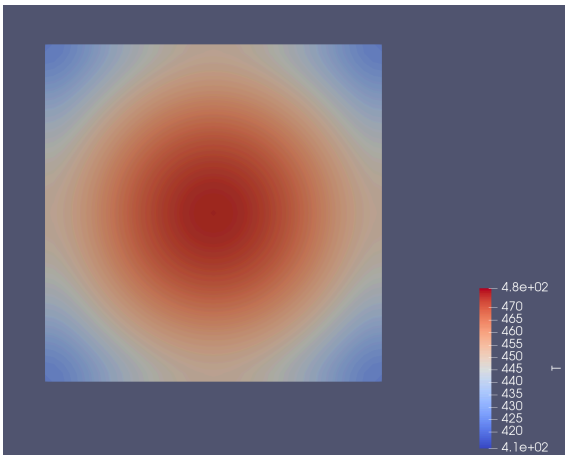
(a) Temperature Profiles

(b) Temperature Difference Profiles

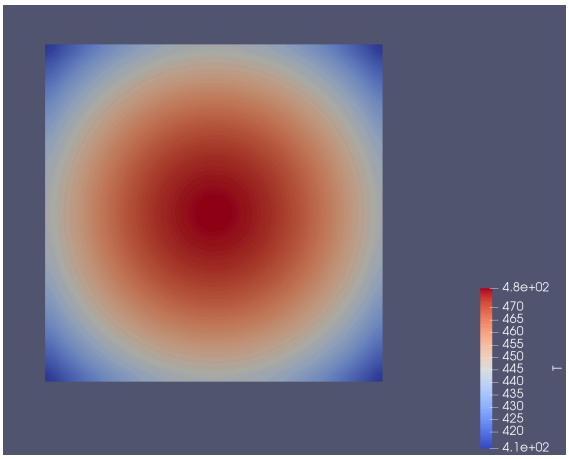
Figure 5.5: Plots of the steady–state, line–sampled temperature profiles for the plate with heat generation.

One will notice that in Fig. 5.5a each of the line–samples coincide very closely with the analytic solution for the plate that emits radiation (§5.1). The difference between the line–sampled temperatures and the analytic temperature for the plate that emits radiation is shown in figure 5.5b. All of the line–sampled temperatures are a few degrees higher than the analytic solution for the radiation emission plate.

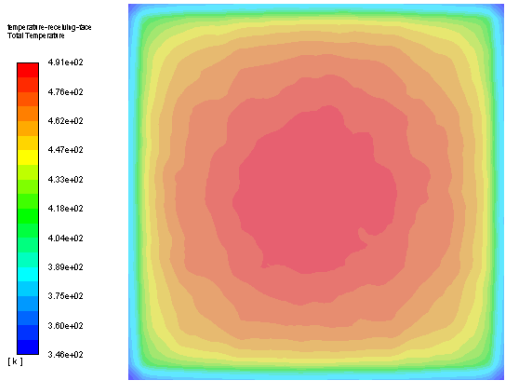
Figure 5.6 shows the surface temperature profiles of the plate without heat generation.



(a) Adiabatic Surface (Phoenix)



(b) Radiation Surface (Phoenix)



(c) Radiation Surface (ANSYS FLUENT)

Figure 5.6: Steady-state, surface temperature profiles of the plate without heat generation in the symmetric heating example.

The temperature range for the results generated with Phoenix are 410 – 480 K (a 70 K difference). The range for the results generated with ANSYS FLUENT are 346 – 491 K (a 145 K difference). The Phoenix results exhibit fourfold symmetry, and are visually smooth. The ANSYS FLUENT results do not exhibit fourfold symmetry and are not visually smooth. At this time it is not certain why these discrepancies exist. It is uncertain to what extent this is an artifact of visualizing the results, or if this stems from one or more issues with the simulation conducted with ANSYS FLUENT. Regardless, the inconsistency of the ANSYS FLUENT results for the plate

with the heat generation with the plate that only emits radiation suggests that one or more issues lie with the ANSYS FLUENT simulation. Figure 5.7 displays a plot generated by line–sampling the temperature in the plate without heat generation.

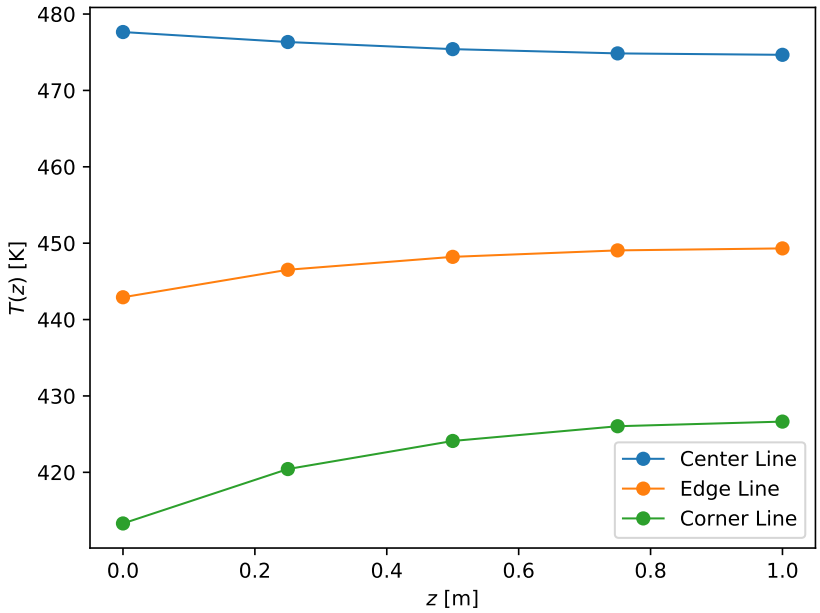


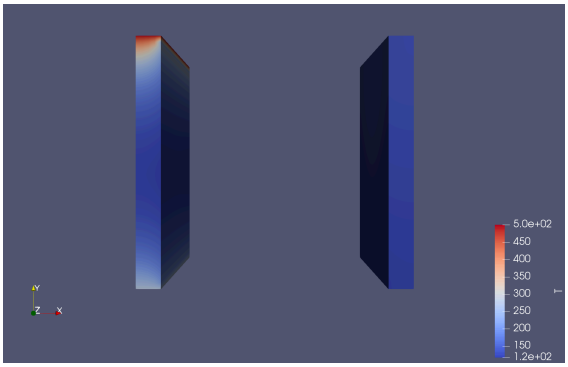
Figure 5.7: Plots of the steady–state, line–sampled temperature profiles for the plate without heat generation.

The largest temperature gradient transverse to the radiation surface occurs in the corner of the face (13.3358 K). The difference in temperature between the center of the plate and the corner is 64.3408 K.

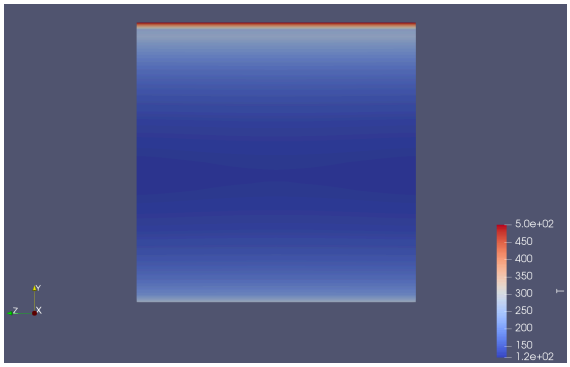
5.3 Parallel and Directly Opposed Plates with Asymmetric Dirichlet Boundary Condition

As in the first example, the second example consists of adiabatic boundary conditions applied to all boundaries that are not communicating via radiation heat exchange. Dirichlet boundary conditions are applied at the top (500 K) and bottom (273.15 K) of the left plate. Both plates do not have heat generation. The geometric configuration and its discretization are identical to

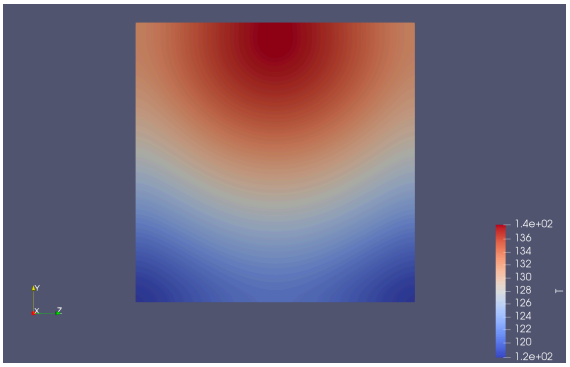
that in the previous example. Figure 5.8 presents visualizations of the steady-state temperature distribution.



(a) Overview



(b) Emitting plate's radiation surface



(c) Receiving plate's radiation surface

Figure 5.8: Steady-state temperature profiles of the asymmetric heating case.

As expected, the temperature on the surface of the emitting plate drops below the temperatures along the edges. The temperature always remains positive (agreeing with what on would expect), and the minimum temperature of nearly 120 K occurs near the middle of the plate. On the receiving plate's surface, the maximum temperature of 140 K occurs along the middle of the top edge. The minimum temperatures of 120 K occur in each of the bottom corners.

6. SUMMARY

In summary, surface-to-surface, blackbody radiation heat transfer has been implemented in the MOOSE app, Phoenix. The double, surface integral for the view factors are approximated by quadrature. CGAL is leveraged to determine occlusion detection – this allows users to use floating point numbers or exact computation. A unit-test suit of 264 tests is included with phoenix to ensure that the view factor computations continue to work. A variety of example cases were simulated and analyzed to ensure that the view factor calculations are correct. Finally, four test cases exercising the radiation boundary conditions were ran and the resulting temperature profiles were analyzed. Potentially significant discrepancies were found between the results generated by Phoenix, and those generated by ANSYS FLUENT.

6.1 Further Study

Opportunities exist to extend and improve the functionality available within Phoenix. It would be possible to include more methods to compute view factors. Another possibility would be to improve the efficiency of the view factor computation by parallelizing the quadrature computation. Clustering of view factors could also be incorporated [43]. It might also be worthwhile to extend the boundary conditions to account for not only blackbody surfaces, but also gray-diffuse surfaces. Additional studies could be conducted to verify, validate and compare Phoenix.

REFERENCES

[1] C. J. Permann, D. R. Gaston, D. Andrs, R. W. Carlsen, F. Kong, A. D. Lindsay, J. M. Miller, J. W. Peterson, A. E. Slaughter, R. H. Stogner, and R. C. Martineau, "Moose: Enabling massively parallel multiphysics simulation," 2019.

[2] D. R. Gaston, C. J. Permann, J. W. Peterson, A. E. Slaughter, D. Andrš, Y. Wang, M. P. Short, D. M. Perez, M. R. Tonks, J. Ortensi, L. Zou, and R. C. Martineau, "Physics-based multiscale coupling for full core nuclear reactor simulation," *Annals of Nuclear Energy*, vol. 84, pp. 45–54, 2015.

[3] The CGAL Project, *CGAL User and Reference Manual*. CGAL Editorial Board, 5.0.1 ed., 2020.

[4] C. Geuzaine1 and J.-F. Remacle, "Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities," *International Journal for Numerical Methods in Engineering*, vol. 79, pp. 1309–1331, 2009.

[5] ISO, *ISO/IEC 14882:1998: Programming languages — C++*. Sept. 1998.

[6] G. van Rossum, "Python tutorial," May 1995.

[7] N. E. Todreas and M. S. Kazimi, *Nuclear Systems*, vol. 1. CRC Press, 2nd ed., 2012.

[8] R. Vaghetto and Y. Hassan, "Experimental investigation of a scaled water-cooled reactor cavity cooling system," *Nuclear Technology*, vol. 187, pp. 282–293, 2014.

[9] H. J. V. Antwerpen and G. P. Greyvenstein, "Evaluation of a detailed radiation heat transfer model in a high temperature reactor systems simulation model," *Nuclear Engineering and Design*, vol. 238, pp. 2985–2994, November 2008.

[10] L. Humphries, B. Beeny, F. Gelbard, D. Louie, and J. Phillips, *MELCOR Computer Code Manuals Vol. 2: Reference Manual*. No. SAND2017-0876 O, January 2017.

[11] B. Zohuri, *Heat Pipe Applications in Fission Driven Nuclear Power Plants*. Springer, 2019.

[12] R. S. Reid, "Heat pipe transient response approximation," Tech. Rep. LA-UR-01-5895, Los Alamos National Laboratory, 2002.

[13] J. J. Martin and R. S. Reid, "Sodium based heat pipe modules for space reactor concepts: Stainless steel safe-100 core," tech. rep., 2004.

[14] A. Faghri, *Heat Pipe Science and Technology*. Global Digital Press, 2nd ed., 2016.

[15] F. He, J. Shi, L. Zhou, W. Li, and X. Li, "Monte carlo calculation of view factors between some complex surfaces: Rectangular plane and parallel cylinder, rectangular plane and torus, especially cold-rolled strip and w-shaped radiant tube in continuous annealing furnace," *International Journal of Thermal Sciences*, vol. 134, pp. 465–474, December 2018.

[16] S. L. Rickman, "Overview and introduction to passive thermal control and thermal protection," December.

[17] T. L. Bergman, A. S. Lavine, F. P. Incropera, and D. P. Dewitt, *Fundamentals of Heat and Mass Transfer*. Wiley, 7th ed., 2011.

[18] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, "PETSc Web page." https://www.mcs.anl.gov/petsc, 2019.

[19] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, "PETSc users manual," Tech. Rep. ANL-95/11 - Revision 3.13, Argonne National Laboratory, 2020.

[20] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, "Efficient management of parallelism in object oriented numerical software libraries," in *Modern Software Tools in Scientific Computing* (E. Arge, A. M. Bruaset, and H. P. Langtangen, eds.), pp. 163–202, Birkhäuser Press, 1997.

[21] C. T. Kelly, *Solving Nonlinear Equations with Newton's Method*. Society for Industrial and Applied Mathematics, 2003.

[22] C. T. Kelly, *Iterative Methods for Linear and Nonlinear Equations*. Society for Industrial and Applied Mathematics, 1995.

[23] Y. Saad, *Iterative Methods for Sparse Linear Systems*. 2nd ed., 2000.

[24] D. A. Knoll and D. E. Keyes, "Jacobian-free newtonkrylov methods: a survey of approaches and applications," *Journal of Computational Physics*, vol. 197, pp. 357–397, 2004.

[25] S. L. Rickman, "Form factors, grey bodies and radiation conductances (radks)." Web, August 2012.

[26] S. C. Francisco, A. M. Raimundo, A. R. Gaspar, A. V. M. Oliveira, and D. A. Quintela, "Calculation of view factors for complex geometries using stokes theorem," *Journal of Building Performance Simulation*, vol. 7:3, pp. 203–216, 2014.

[27] C. K. Krishnaprakas, "View-factor evaluation by quadrature over triangles," *Journal of Thermophysics and Heat Transfer*, vol. 12:1, pp. 118–120, January 1998.

[28] J. A. Clark and M. E. Korybalski, "Algebraic methods for the calculation of radiation exchange in an enclosure," *Wärme- und Stoffübertragung*, vol. 7, pp. 31–44, 1974.

[29] M. F. Modest, *Radiative Heat Transfer*. Academic Press, 3rd ed., 2013.

[30] Z. S. Spakovszky, "16. unified: Thermodynamics and propulsion."

[31] I. Ashdown, *Radiosity: A Programmer's Perspective*. John Wiley & Sons, Inc., 1994.

[32] J. R. Howell, M. P. Mengüç, and R. Siegel, *Thermal Radiation Heat Transfer*. CRC Press, 6th ed., 2016.

[33] C. K. Krishnaprakas, "View-factor evaluation by quadrature over triangles," *Journal of Thermophysics*, vol. 12, pp. 118–120, December 2018.

[34] T. E. Oliphant, "A guide to numpy," *USA: Trelgol Publishing*, 2006.

[35] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: A structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, pp. 22–30, 2011.

[36] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[37] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz, "Sympy: symbolic computing in python," *PeerJ Computer Science*, vol. 3, p. e103, Jan. 2017.

[38] The CGAL Project, *CGAL User and Reference Manual*. CGAL Editorial Board, 5.0.1 ed., 2020.

[39] B. Stroustrup, *The C++ Programming Language*. Addison-Wesley, 4th ed., 2013.

[40] P. Schröder and P. Hanrahan, "On the form factor between two polygons," *Computer Graphics Proceedings (SIGGRAPH '93 Conference)*, pp. 163–164, 1993.

[41] B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey, "libMesh: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations," *Engineering with Computers*, vol. 22, no. 3–4, pp. 237–254, 2006. https://doi.org/10.1007/s00366-006-0049-3.

[42] J. Howell, "A catalog of radiation heat transfer configuration factors."

[43] "Ansys fluent 12.0 user's guide."