Internal Proposal for Implementation of Radiance Cascades to Neutral Particle Transport

Subtitle TBD

Prepared by:
Nathan GLASER
Liam POHLMANN

Prepared for:
INSTITUTE OF REPORTS
CONTRACT NN-NNNN

Principal Investigators:
Prof. Kathryn D. HUFF
Prof. Madicken MUNK

UIUC-ARFC-2025-02

December 15, 2025

ADVANCED REACTORS AND FUEL CYCLES

DEPT. OF NUCLEAR, PLASMA, & RADIOLOGICAL ENGINEERING
UNIVERSITY OF ILLIOIS AT URBANA-CHAMPAIGN

ILLINOIS

Contents

1 Introduction

Neutron transport methods can be loosely classified into two domains: deterministic and Monte Carlo. The latter of these has been dubbed the gold standard for simulations because it directly solves the governing Boltzmann Transport Equation directly through the simulation of distinct particles. Though embarrassingly parallelizable, Monte Carlo methods are notoriously slow to converge and can still yield high uncertainty in areas of low flux. Deterministic methods, alternatively, discretize the phase space to numerically approximate the solution. They are more difficult to mathematically formulate but can produce an acceptable solution in a fraction of the time it takes for Monte Carlo methods.

The field of computer graphics has long been interested in the solution of global transport problems to accurately render scenes in animation and video games. Since the development and release of consumer-available ray tracing hardware, significant effort has been put towards the development of realtime global illumination solution methods. One recent method, Radiance Cascades (RCs), is a novel approach to solving these problems that also promises impressive scalability on modern graphical processing unit (GPU) hardware.

In this proposal, we present a comprehensive discussion and analysis of the theory of RCs to motivate our proposed work. We then outline our implementation plans (and thought process) and describe how we will demonstrate the effectiveness of Radiance Cascades for global neutron transport using selected benchmark problems.

2 Literature Review and Background

Deterministic neutron transport techniques are commonly developed around the expansion of the Boltzmann Transport Equation (BTE)'s scattering source term in solid angle ($\mathbf{\Omega}$) using orthogonal basis functions. The most-widely used expansion is the set of spherical harmonics, which is a complete set of orthogonal functions that form a basis on the surface of a sphere [1]. When treated continuously in angle, the angular flux is then expanded using Legendre polynomials [2], making up the so-called P_N method. While the details of this method are outside the scope of this paper, we note that the P_N method has mostly fallen out of favor because the complexity involved in its formulation and the approximations needed to achieve vacuum boundary conditions, which themselves are defined piecewise-discontinuous in angle. Moreover, the P_N method is capable of producing nonphysical negative particle densities [3].

The Discrete Ordinates (SN) method has remained the standard numerical approach in transport theory in applications ranging from heat transfer [4] to neutronics to astrophysics. Indeed, since its introduction by Chandrasekhar [5] to astrophysics and subsequent application to neutronics by Carlson [6, 7], it has seen continued research, development, and applications at the national laboratories through codes such as Denovo [8] and PARTISN [9], and in the open-source

community with OpenSN [10][1] and OpenMOC [11]. In one form of SN, the BTE is evaluated along selected angles ("ordinates") that make up a quadrature set (usually Gauss-Legendre). Once each direction has been solved for, the results are multiplied by their associated weighting functions and summed together at each spatial point to construct the scalar (angle-integrated) flux. SN methods usually use "sweeps," a colloquial term referring to an iterative scheme in which the solver follows ("sweeps") along the direction of neutron travel, updating the source term until convergence. An excellent introduction to the sweeping algorithm, along with the discrete ordinates method in general, can be found in [2].

Despite its effectiveness, SN struggles with numerical artifacts known as ray effects. These arise because discretizing the angular domain into a finite number of ordinates breaks rotational symmetry [3], causing radiation from a localized source to appear smeared or faceted along discrete directions. Mitigation strategies stretch back to early work by Lathrop [12, 13], yet ray effects remain a central challenge—particularly for large domains, where resolving far-field behavior demands extremely dense quadrature sets. See Figure 1 provided in [14] on page 4. Another industry engaged in solving transport problems is computer graphics, which focuses on rendering light for realistic scene generation. A major subset of this field is video game development, where developers must balance solution accuracy with stringent performance requirements. Unlike scientific simulations, graphics pipelines often demand realtime (near-instantaneous) results, which places tight limits on computational cost. A related application within computer graphics is animation and computer-generated imagery (CGI) in film production, where high-end computational resources allow for more expensive offline rendering. This paper, however, focuses specifically on photon-transport methodologies emerging from the real-time rendering community in gaming.

Historically, standard image-rendering pipelines relied on rasterization, in which a 3D schene is projected onto a 2D display . While extremely fast, rasterization does not inherently capture the complex light–matter interactions needed for physically realistic lighting. Ray tracing techniques, alternatively, directly simulate light paths and their interactions with the environment, but until recently was prohibitively expensive for consumer hardware, not to mention unreasonably time-consuming to render each frame. The emergence of ray-tracing-dedicated units within modern graphics processing units (GPUs) has changed this landscape, making real-time ray tracing feasible and establishing it as a new standard in contemporary video games. [15]

Within computer graphics, global illumination (GI) denotes the complete solution of a scene's lighting, including both direct and indirect light transport. This is analogous to global neutron transport problems such as full-core flux calculations. The central argument of this paper is that GI techniques have direct relevance to nuclear-engineering transport problems. The underlying physics is essentially the same – identical for photon transport – and the computer-graphics community has already built a substantial ecosystem around GPU-accelerated ray tracing. This creates an opportunity to take advantage of developments in real-time photon transport and apply them

[1] At the time of this writing, a reference journal article for citations of OpenSN has not been provided.

to scientific transport simulations.

A new method for real-time global illumination called Radiance Cascades (RCs) emerged recently[2] in the paper "Radiance Cascades: A Novel Approach to Calculating Global Illumination" by Alexander Sannikov [16]. Sannikov, a video game developer, proposed a new data structure he termed "radiance cascades" that more effectively calculates and stores radiance field information by decomposing the domain into near- and far-field regions. This idea is based on a simple observation: to resolve the radiance from an emitter, one needs to have higher spatial resolution near the object but higher angular resolution far from it.

Another integral concept of the technique is ray construction. Instead of explicitly casting rays, each probe (within each cascade) is assigned a spherical "shell" region and angular discretization, representing the portion of the domain contributing to that probe's radiance. The thickness of this shell is termed the "radiance interval" and is one of the foundational principles of the method. Thus, once a cascade has evaluated the contributions to each probe within its respective radiance interval, these intervals can be combined into reconstructed rays via interpolation when needed.

Sannikov claims that the method enables, for practical all purposes, infinitely many rays to be constructed at finite computational cost. Indeed, he shows that the total memory required to store an infinite number of cascades is less than a constant, that constant being exactly equal to twice the amount of memory required to store the first cascade. This holds for *both* 2D and 3D problems. As in neutronics methods like the Method of Characteristics and its relative, the The Random Ray Method, scalability is constrained by the memory demands associated with casting a sufficiently dense set of rays to resolve far-field solutions. Although Sannikov [16] presents a promising approach, he notes that storing only the first cascade is effectively equivalent to discretizing the entire domain, which can be prohibitive for sufficiently large domains or dense probe configurations. In practice, the benefits of the technique emerge only after allocating approximately twice the memory of the finest spatial discretization because the incremental memory cost of subsequent cascades decreases exponentially.

The most relevant work to this project proposal is that written by Osborne and Sannikov entitled "Radiance cascades: a novel high-resolution formal solution for multidimensional non-LTE radiative transfer" [14]. In this paper, the authors apply Radiance Cascades to radiative transfer in astrophysics. Similar to deterministic neutron transport methods, the discrete ordinates method has dominated as the method-of-choice, and has likewise struggled with ray effects. Unlike terrestrial neutronics, astrophysics often cannot take advantage of Monte Carlo methods because of massive length scales, and these scales likewise greatly degrade the SN solution far from the source. Achieving accurate solutions therefore requires extremely high angular resolution, but the computational needs scale linearly with the number of angular quadrature points.

Osborne revisits the claim of infinite rays constructed at finite cost. He observes that this asymptotic scaling breaks down before the infinite-ray limit is reached, but emphasizes that, so

[2]This is intentionally vague, as the paper written was not dated nor published in a journal. It is available freely on GitHub.

long as the penumbra criterion[3] holds, "the integration of the radiation field at high spatial and angular resolution [is] far cheaper than could be obtained with any traditional ray-casting approach." Additionally, the computational cost is dependent on the length of the radiance interval, and for higher cascades, these intervals can be considerably longer than those of lower cascades. Lastly, the scaling laws introduced by [16] and [14] are dependent on the so-called "branching factor," which controls the number of rays computed for each cascade. It is mentioned that more complex branching strategies can be applied in 3D to take advantage of the second angular dimension, potentially leading to improved scaling. This hypothesis was not explored further in Osborne's paper.

The most significant contribution to Radiance Cascades in [14] is the so-called bilinear fix. One weakness of RCs as it was introduced in [16] is ringing artifacting. The fix eliminates these numerical errors by correctly interpolating the source contributions from the next highest cascade. An overview of this fix is provided in Section 3.5.

The final relevant paper is entitled "Holographic Radiance Cascades for 2D Global Illumination" [17]. In this work, the authors reformulate the original Radiance Cascades approach to reduce resolution in only a single direction as the cascades progress. This modification both improves shadow resolution by limiting the diffusion inherent to the classical Radiance Cascades formulation and introduces an effective acceleration structure. The results demonstrate particular promise for radiation shielding applications, where accurately resolving shadowed regions is critical.

3 Radiance Cascades

As previously discussed, Radiance Cascades was developed to simulate global illumination in real time for computer-graphics applications. The method is built upon the following principle: near a source, the radiance field varies rapidly in space but only weakly in angle. Far from the source, the radiance field varies slowly in space but rapidly in angle. There are, of course, caveats to this generalization, and the aim of the proposed work is to explore cases where these assumptions hold and deteriorate. In this section, we provide an overview of the theory presented in the papers by Sannikov and Osborne [16, 18], along with preliminary extensions of the method as it pertains to neutron transport.

3.1 Initial Definitions

Because this paper focuses on nuclear applications of Radiance Cascades, we briefly define certain terms commonly used in the technique. These definitions are intentionally loose, only meant as a primer to have a framework to understand the subsequent theory. Formal definitions, in nuclear engineering context, will be presented as the theory is developed.

[3] More on this term in Section 3.2.

The underlying observation that birthed the idea of RCs is the *penumbra condition*. The major result of this observation is that the angular and spatial discretization required to resolve near- and far-field radiation contributions is not constant between regimes. The concept of the penumbra condition is formalized in Section 3.2.

There are three main concepts in Radiance Cascades:

1. radiance intervals
2. probes
3. cascades

A radiance interval is the contribution of incoming radiation at some point $\mathbf{r}_0$ from a spherical shell centered at $\mathbf{r}_0$. The range of the radiance interval is defined as the difference between the outer and inner radii of the shell. Note that in 2-D, the spherical shell is collapsed into a circular shell with the range definition remaining the same. Radiance intervals are formalized for the Linear Boltzmann Equation (LBE) in Section 3.3, as well as how to obtain the total incoming radiation from all radiance intervals.

A "probe" in RCs is synonymous to a detector in nuclear engineering. It is a simulation structure that has a position in space, and it computes and stores a radiance interval about it's position. To compute the intervals, the probe discretizes intervals into a set of rays, each with a unique direction that is normal to the spherical shell's surface. Note, however, that probes are "point" structures that do not have associated volume or surface area (like a detector would). In this way, probes are analogous to nodes in finite element analysis in that they are points in space that contain more information that just their location. More on this in Section 3.4.

Cascades are a collection of probes that are equidistant from one another and contain the same angular discretizations. Each probe in a cascade solves for a radiance interval of the same size (range), but that interval is centered on the probe's own location. In other words, all probes within a given cascade behave identically, the only thing that differs is where each probe is placed. In RCs, cascades are arranged hierarchically: the lowest cascade provides the finest spatial resolution (the most probes) but uses the coarsest angular resolution, while higher cascades contain progressively fewer probes but have increasingly finer angular discretization. The method solves the radiance field by computing each radiance interval independently. Then, the solved radiance intervals are successively merged downward from the highest cascade until the lowest cascade is reached. Once the probes in the lowest cascade have been encoded with the solutions from the higher cascades, the continuous radiation field is obtained by interpolating between probes. Cascades and merging are discussed more in Section 3.4.

3.2 Penumbra Criterion

The relationship between the distance from a source and the required spatial and angular resolution to resolve the source is the "penumbra criterion." It is formally defined by Sannikov [16] using

a 2-D planar problem with a light source and perfectly opaque absorber. The goal of the problem is to determine the angular and spatial resolution required to accurately capture the penumbra of the shadow cast by the source at some distance past the absorber.

This can be illustrated by considering a detector placed in the penumbra. When the detector is near the source and absorber, moving the detector laterally will expose a significant additional amount of the source to the detector. Inversely, if the detector is far, the lateral step required to yield the same change in the exposure of the source is much greater. Therefore the number of lateral steps required to resolve the source is related to the distance from it, so as the distance increases, the lateral step size (Δ_s) can also be increased. Next, the solid angle of the source and absorber, when the detector is close, is large, and so a small angular shift in the direction of the detector face will not yield a large change in the perceived solid angle. Conversely, far from the source and absorber, the solid angle is small, and so a small angular shift can result in a large change in the perceived solid angle. Thus, the number of angular steps required to resolve the source is inversely related to the distance, and so as the distance increases, the angular step size (Δ_ω) must decrease.

Compressing these observations yields the penumbra condition [16, 14]:

1. Near-field radiance contributions from the light source vary with high spatial frequency and low angular frequency.

2. Far-field radiance contributions from the light source vary with low spatial frequency and high angular frequency.

The penumbra criterion can be represented mathematically as [16, 14]:

$$\begin{aligned} \Delta_s &< f(D) \propto D, \\ \Delta_\omega &< g(D) \propto 1/D, \end{aligned} \tag{1}$$

such that $f(D)$ and $g(D)$ are linear functions of the distance D from a source. It should be noted that these observations still hold for the case in which the source is far from the absorber, but for that case the angular step Δ_ω scales super-linearly with $1/D$ [14].

RCs depends on the hypothesis that the penumbra condition holds generally throughout the problem domain for all sources. There are trivial exceptions to this hypothesis, namely a point source or a small source embedded in a perfectly absorbing medium. In general however, the penumbra condition holds well for most purposes, and RCs are formulated to encode most extreme cases [16].

3.3 Radiance Intervals

This section introduces and derives the concept of radiance intervals in a neutron transport framework. The foundational property of the radiation field that allows for RCs is the notion of splitting incoming radiation rays into distinct intervals that can be solved for separately (and iteratively

combined). These intervals are called "radiance intervals," by convention from Sannikov [16]. A schematic of radiance intervals and their influence of the solution on a probe can be found in Figure 1.

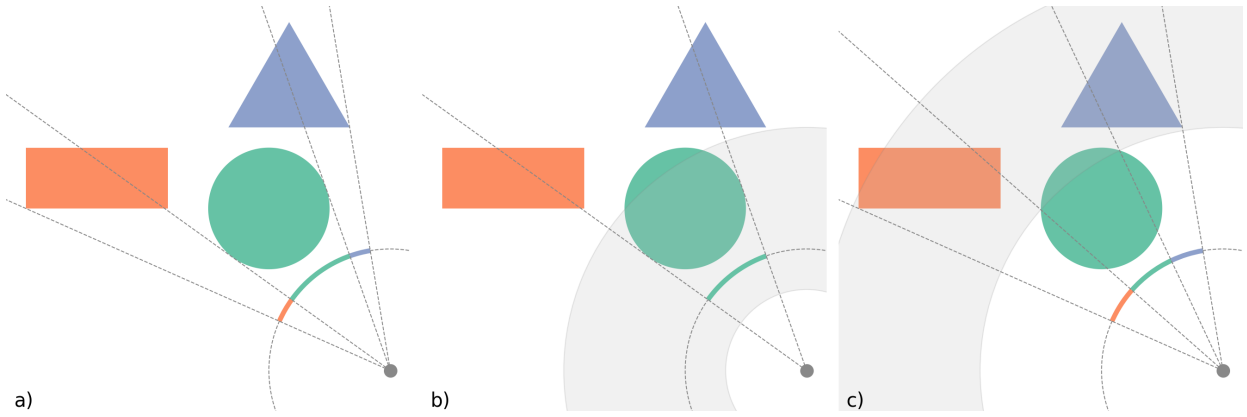


Figure 1: Concept of radiance intervals illustrated against standard long-characteristics. Taken from [14] . Caption from original source: *Comparison of long characteristics style ray-casting from a point against two different radiance intervals over the same field. The three coloured primitives can be considered opaque emissive sources. In (a) we show the conventional ray-casting approach, whilst (b) and (c) show the radiance found inside annuli described by closer and further radiance intervals (shown in grey). Note that in (b) only the blue circle is found by these samples, whilst in (c) obscured components of the orange rectangle and green triangle are sampled.*

3.3.1 Formalization of Radiance Intervals

The main idea of radiance intervals is that each ray can be decomposed into adjacent segments, solved independently, and then merged to determine the angular flux at the start point of the ray. We seek to derive the radiance intervals, and how they relate to the angular flux, for the Linear Boltzmann Equation (LBE). To do so, we begin with the characteristic form of the LBE as

$$\frac{\mathrm{d}}{\mathrm{d}u}\psi^g\left(\mathbf{r}_0+u\mathbf{\Omega},\mathbf{\Omega}\right)+\Sigma_t^g\left(\mathbf{r}_0+u\mathbf{\Omega},\mathbf{\Omega}\right)\psi^g\left(\mathbf{r}_0+u\mathbf{\Omega},\mathbf{\Omega}\right)=Q^g\left(\mathbf{r}_0+u\mathbf{\Omega},\mathbf{\Omega}\right). \tag{2}$$

Before continuing, we first perform a change of variables from u to s, such that $u=-s$ and $\frac{d}{du}=-\frac{d}{ds}$, [2, p. 210]. This is done to solve 'backwards' along each characteristic line; instead of solving for the contribution at s from $\mathbf{r_0}$, we solve for the contribution at $\mathbf{r_0}$ from s. Then, the characteristic form using this transform is

$$-\frac{\mathrm{d}}{\mathrm{d}s}\psi^g\left(\mathbf{r}_0-s\mathbf{\Omega},\mathbf{\Omega}\right)+\Sigma_t^g\left(\mathbf{r}_0-s\mathbf{\Omega},\mathbf{\Omega}\right)\psi^g\left(\mathbf{r}_0-s\mathbf{\Omega},\mathbf{\Omega}\right)=Q^g\left(\mathbf{r}_0-s\mathbf{\Omega},\mathbf{\Omega}\right). \tag{3}$$

This is an linear ordinary differential equation, and so it can be solved for by use of an integrating factor. We define the integrating factor as

$$I = e^{-\int_0^s \mathrm{d}s' \Sigma_t^g (\mathbf{r}_0 - s'\mathbf{\Omega}, \mathbf{\Omega})}. \tag{4}$$

Henceforth, the independent variables are dropped purely for conciseness, and everything is assumed to be a function of $\mathbf{r}_0 - s'\mathbf{\Omega}$ and $\mathbf{\Omega}$ unless otherwise specified. Multiplying Eq. 3 by our integrating factor, we obtain

$$-e^{-\int_0^s \mathrm{d}s' \Sigma_t^g} \frac{\mathrm{d}}{\mathrm{d}s} \psi^g + e^{-\int_0^s \mathrm{d}s' \Sigma_t^g} \Sigma_t^g \psi^g = e^{-\int_0^s \mathrm{d}s' \Sigma_t^g} Q^g. \tag{5}$$

The left hand side of this equation is the output of a product rule, specifically

$$\frac{\mathrm{d}}{\mathrm{d}s} \left[-e^{-\int_0^s \mathrm{d}s' \Sigma_t^g} \psi^g \right] = -e^{-\int_0^s \mathrm{d}s' \Sigma_t^g} \frac{\mathrm{d}}{\mathrm{d}s} \psi^g + e^{-\int_0^s \mathrm{d}s' \Sigma_t^g} \Sigma_t^g \psi^g. \tag{6}$$

Then, we insert this relation into Eq. 5, and integrate both sides with respect to s. We set the bounds for this integration to be from a to b, such that $0 \leq a \leq b \leq \tau$ where τ is the endpoint of the ray. These bounds are chosen instead of the typical 0 to τ, because we are only interested in the incoming radiation along this interval at $s = a$.

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}s} \left[-e^{\int_0^s \mathrm{d}s' \Sigma_t^g} \psi^g \right] &= e^{-\int_0^s \mathrm{d}s' \Sigma_t^g} Q^g \\ \int_a^b \mathrm{d}s \, \frac{\mathrm{d}}{\mathrm{d}s} \left[-e^{-\int_0^s \mathrm{d}s' \Sigma_t^g} \psi^g \right] &= \int_a^b \mathrm{d}s \; e^{-\int_0^s \mathrm{d}s' \Sigma_t^g} Q^g \end{aligned} \tag{7}$$

For brevity, we define $\psi^g(x) = \psi^g(\mathbf{r}_0 - x\mathbf{\Omega}, \mathbf{\Omega})$. Evaluating the left hand side integral, rearranging, and solving for $\psi^g(a)$, we find

$$\begin{aligned} e^{-\int_0^a \mathrm{d}s' \Sigma_t^g} \psi^g(a) &= e^{-\int_0^b \mathrm{d}s' \Sigma_t^g} \psi^g(b) + \int_a^b \mathrm{d}s \; e^{-\int_0^s \mathrm{d}s' \Sigma_t^g} Q^g \\ \psi^g(a) &= e^{\int_0^a \mathrm{d}s' \Sigma_t^g} \left[e^{-\int_0^b \mathrm{d}s' \Sigma_t^g} \psi^g(b) + \int_a^b \mathrm{d}s \; e^{-\int_0^s \mathrm{d}s' \Sigma_t^g} Q^g \right] \\ &= e^{\int_0^a \mathrm{d}s' \Sigma_t^g - \int_0^b \mathrm{d}s' \Sigma_t^g} \psi^g(b) + e^{\int_0^a \mathrm{d}s' \Sigma_t^g} \int_a^b \mathrm{d}s \; e^{-\int_0^s \mathrm{d}s' \Sigma_t^g} Q^g \end{aligned} \tag{8}$$

Now, because the exponential in front of the integral is a function of s' and not s, it can be factored into the integral, and thus

$$\begin{aligned} \psi^g(a) &= e^{-\int_a^b \mathrm{d}s' \Sigma_t^g} \psi^g(b) + \int_a^b \mathrm{d}s \; e^{\int_0^a \mathrm{d}s' \Sigma_t^g - \int_0^s \mathrm{d}s' \Sigma_t^g} Q^g \\ &= e^{-\int_a^b \mathrm{d}s' \Sigma_t^g} \psi^g(b) + \int_a^b \mathrm{d}s \; e^{-\int_a^s \mathrm{d}s' \Sigma_t^g} Q^g. \end{aligned} \tag{9}$$

For completeness and clarity, we now expand out the independent variables in each term.

$$\begin{aligned}\psi^g\left(\mathbf{r}_0 - a\boldsymbol{\Omega}, \boldsymbol{\Omega}\right) = e^{-\int_a^b \mathrm{d}s' \Sigma_t^g(\mathbf{r}_0 - s'\boldsymbol{\Omega}, \boldsymbol{\Omega})} \psi^g\left(\mathbf{r}_0 - b\boldsymbol{\Omega}, \boldsymbol{\Omega}\right) \\ + \int_a^b \mathrm{d}s\, e^{-\int_a^s \mathrm{d}s' \Sigma_t^g(\mathbf{r}_0 - s'\boldsymbol{\Omega}, \boldsymbol{\Omega})} Q^g\left(\mathbf{r}_0 - s\boldsymbol{\Omega}, \boldsymbol{\Omega}\right)\end{aligned} \tag{10}$$

Now, we formally define the transparency ($\beta_{a,b}^g$) and the radiance interval ($\psi_{a,b}^g$) as

$$\beta_{a,b}^g\left(\mathbf{r}_0, \boldsymbol{\Omega}\right) = e^{-\int_a^b \mathrm{d}s' \Sigma_t^g(\mathbf{r}_0 - s'\boldsymbol{\Omega}, \boldsymbol{\Omega})} \tag{11}$$

$$\psi_{a,b}^g\left(\mathbf{r}_0, \boldsymbol{\Omega}\right) = \int_a^b \mathrm{d}s\; e^{-\int_a^s \mathrm{d}s' \Sigma_t^g(\mathbf{r}_0 - s'\boldsymbol{\Omega}, \boldsymbol{\Omega})} Q^g\left(\mathbf{r}_0 - s\boldsymbol{\Omega}, \boldsymbol{\Omega}\right). \tag{12}$$

The naming 'transparency' and 'radiance interval' is used for clarity, as the original derivation of RCs by Sannikov [16] used these terms. Additionally, it should be noted that these definitions differ slightly from Sannikov's [16] in the arguments of the cross-section and source term, but are fundamentally identical. The notation used here is to be consistent with the standard Method of Characteristics (MOC) formulation within the field of computational neutron transport. Finally, we can insert our definitions for the transparency and radiance interval into Eq. 10, finding the simplified definition for the radiation contribution at $s = a$ from the interval $[a, b]$ as

$$\psi^g\left(\mathbf{r}_0 - a\boldsymbol{\Omega}, \boldsymbol{\Omega}\right) = \beta_{a,b}^g\left(\mathbf{r}_0, \boldsymbol{\Omega}\right) \psi^g\left(\mathbf{r}_0 - b\boldsymbol{\Omega}, \boldsymbol{\Omega}\right) + \psi_{a,b}^g\left(\mathbf{r}_0, \boldsymbol{\Omega}\right). \tag{13}$$

3.3.2 Validity of Radiance Interval Formulation

To illustrate that this formulation is a valid method of breaking a ray into segments, we select an arbitrary ray broken into two segments. We define the interval ranges as $[a = 0, b]$ and $[b, c = \tau]$. Using equation 13 we define $\psi^g\left(\mathbf{r}_0, \boldsymbol{\Omega}\right)$ as

$$\psi^g\left(\mathbf{r}_0, \boldsymbol{\Omega}\right) = \beta_{0,b}^g\left(\mathbf{r}_0, \boldsymbol{\Omega}\right) \psi^g\left(\mathbf{r}_0 - b\boldsymbol{\Omega}, \boldsymbol{\Omega}\right) + \psi_{0,b}^g\left(\mathbf{r}_0, \boldsymbol{\Omega}\right). \tag{14}$$

Now, again using equation 13, we define $\psi^g\left(\mathbf{r}_0 - b\boldsymbol{\Omega}, \boldsymbol{\Omega}\right)$ as

$$\psi^g\left(\mathbf{r}_0 - b\boldsymbol{\Omega}, \boldsymbol{\Omega}\right) = \beta_{b,\tau}^g\left(\mathbf{r}_0, \boldsymbol{\Omega}\right) \psi^g\left(\mathbf{r}_0 - \tau\boldsymbol{\Omega}, \boldsymbol{\Omega}\right) + \psi_{b,\tau}^g\left(\mathbf{r}_0, \boldsymbol{\Omega}\right). \tag{15}$$

Inserting this definition into that of $\psi^g(\mathbf{r}_0, \mathbf{\Omega})$, again dropping arguments for brevity, we find

$$\begin{aligned}
\psi^g(\mathbf{r}_0, \mathbf{\Omega}) &= \beta^g_{0,b}\left[\beta^g_{b,\tau}\psi^g(\tau) + \psi^g_{b,\tau}\right] + \psi^g_{0,b} \\
&= e^{-\int_0^b \mathrm{d}s' \Sigma^g_t}\left[e^{-\int_b^\tau \mathrm{d}s' \Sigma^g_t}\psi^g(\tau) + \int_b^\tau \mathrm{d}s\; e^{-\int_b^s \mathrm{d}s' \Sigma^g_t} Q^g\right] + \psi^g_{b,\tau} \\
&= e^{-\int_0^\tau \mathrm{d}s' \Sigma^g_t}\psi^g(\tau) + e^{-\int_0^b \mathrm{d}s' \Sigma^g_t}\int_b^\tau \mathrm{d}s\; e^{-\int_b^s \mathrm{d}s' \Sigma^g_t} Q^g + \psi^g_{b,\tau} \\
&= e^{-\int_0^\tau \mathrm{d}s' \Sigma^g_t}\psi^g(\tau) + \int_b^\tau \mathrm{d}s\; e^{-\int_0^s \mathrm{d}s' \Sigma^g_t} Q^g + \int_0^b \mathrm{d}s\; e^{-\int_0^s \mathrm{d}s' \Sigma^g_t} Q^g \\
&= e^{-\int_0^\tau \mathrm{d}s' \Sigma^g_t}\psi^g(\tau) + \int_0^\tau \mathrm{d}s\; e^{-\int_0^s \mathrm{d}s' \Sigma^g_t} Q^g. \qquad (16)
\end{aligned}$$

This finding is consistent with the long-characteristic solution that is the standard in MOC, and so we conclude that rays can be broken into radiance intervals as we have defined them.

3.3.3 Radiance Interval Merging

The definition for $\psi^g(\mathbf{r}_0 - a\mathbf{\Omega}, \mathbf{\Omega})$ also defines how radiance intervals are combined to find the final solution at $\mathbf{r_0}$. The radiance interval merging formula, written more generally than Eq. 13, is

$$\psi^g(\mathbf{r}_0 - s_i\mathbf{\Omega}, \mathbf{\Omega}) = \beta^g_{i,i+1}(\mathbf{r}_0, \mathbf{\Omega})\,\psi^g(\mathbf{r}_0 - s_{i+1}\mathbf{\Omega}, \mathbf{\Omega}) + \psi^g_{i,i+1}(\mathbf{r}_0, \mathbf{\Omega}), \qquad (17)$$

where the dummy index i denotes the interval bounds. It is important to notice that two intervals can only be merged if they are directly adjacent: they must share a bound. Finally, it should be evident that if a ray is broken into n intervals, the solution to the ray is found through recursively merging the radiance intervals, collapsing towards $\mathbf{r_0}$.

3.4 Radiance Probes and Cascades

This section formally defines radiance probes and radiance cascades, demonstrating how the radiation field about is solved for. To do so, we utilize our previously derived radiance intervals (Equation (11)) and the interval merging relation (Equation (17)) to solve the radiation field naively. Then, we present the main result of RCs, which enables significantly more efficient encoding of the radiation field by taking advantage of the key observations of the penumbra criterion. Finally, we describe the cascade merging algorithm, which efficiently computes for the radiation field.

3.4.1 Radiation Field Representation

From the penumbra criterion, the angular discretization required to accurately resolve near-field contributions is not the same as is required for far-field contributions at a point. Specifically, greater angular discretization is required for far-field resolution than near-field. Using this, we break the radiation field about a point into a series of adjacent radiance intervals, with the increase

in angular discretization between intervals defined as the branching ratio, α. We simply prescribe $\alpha = 2$ for simplicity in presentation, but this is not required in general for RCs.

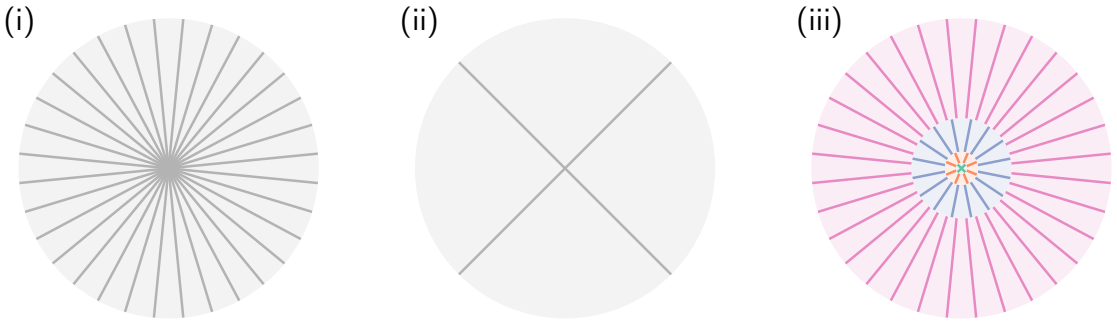


Figure 2: Comparison of different long characteristic angular representations of a point. (i) and (ii) are typical long-characteristics representations of the radiation field about a point, with the 32 and 4 rays cast, respectively. (iii) uses radiance intervals, with a branching ratio of 2. The lowest interval (in green) casts 4 partial rays, and the highest interval (in pink) casts 32 partial rays.

Investigating Figure 2, the radiation field discretization in (i) might accurately resolve the contributions at the edge of the circle, but this discretization is massively overkill for resolving near the point. Inversely, the radiation field discretization in (ii) might adequately resolve near the source, but will poorly resolve at the edge of the circle. The inefficiency and failure of these discretizations is because the angular step (arc length) between two cast rays is not constant, and grows smaller further from the point. This causes either inefficient computation near the point or incorrect solutions far from the point. Conversely, the radiation field discretization in (iii), the RCs discretization, efficiently resolves near field while accurately resolving far-field. By using radiance intervals with increasing angular discretization, the angular step is roughly constant between any given radiance interval, and so the solution is both efficiently computed and accurate.

To solve for the radiation field, we discretize our problem domain into equidistant, discrete points, using the radiation field angular discretization employed in Figure 2 (iii). This discretization scheme applied across the problem domain is shown in Figure 3 (i). Now, solving for the angular flux at each point is just a matter of solving all radiance intervals, and then merging these intervals down to each point.

Although this discretization scheme is more efficient or more accurate than the full ray-casting discretizations, it only utilizes one part of the penumbra criterion. In using the other, serious efficiency improvements are made. The other observation is that the spatial discretization to accurately resolve near-field contributions is greater than the requirement for far-field contributions.

Applying this observation to the set of equidistant points, each with their own distinct series of adjacent radiance intervals, tells us that radiance intervals (with the same range) of nearby points are effectively identical. Thus, they can be compressed into one radiance interval, drastically decreasing the computational cost of the method; with the decrease in cost being directly related to the branching ratio.

Now, we can more intuitively define the terms radiance probes and radiance cascades. A probe

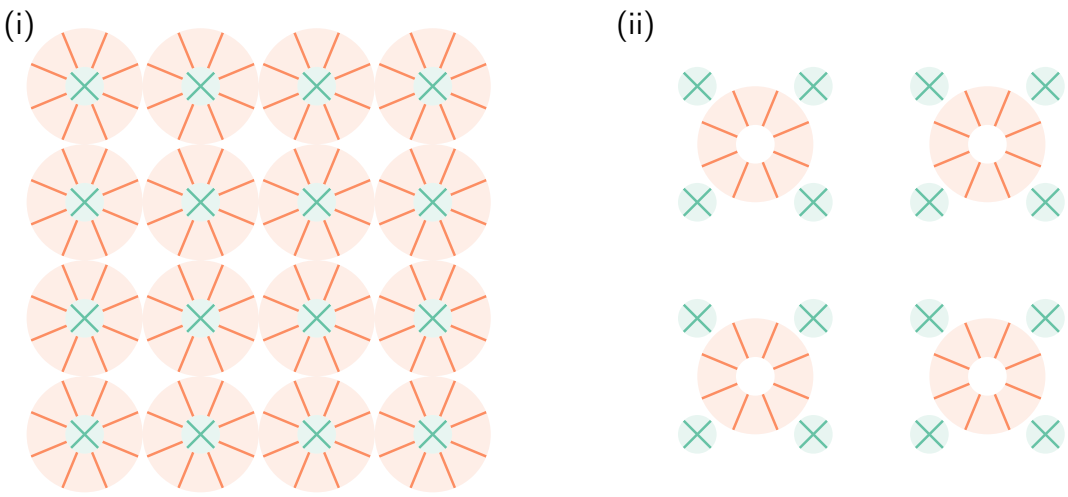


Figure 3: Comparison of spatial dependence of radiance intervals. (i) shows the naive approach to RCs, applying a full set of radiance intervals to each point. (ii) demonstrates the compression of nearby same-range radiance intervals, the method employed in true RCs. In both cases, the radiance interval ranges are shortened and the number of radiance intervals is decreased for clarity.

is a simulation structure, storing a single radiance interval and the point the interval is centered about [16]. In Figure 3 (ii), each green and orange region are distinct radiance intervals encoded by distinct radiance probes. Specifically, there are 16 'green' probes and 4 'orange' probes, for a total of 20 probes. A radiance cascade is a complete set of probes that encode same-range radiance intervals. By convention, the cascade that contains probes with the lowest angular discretization is cascade 0 (C_0), and the cascade that contains probes with highest angular discretization is cascade n (C_n). In Figure 3 (ii), there are two cascades, with C_0 being the complete set of all 16 green probes, and C_1 being the complete set of all 4 orange probes.

3.4.2 Standard Radiation Field Solution

Before beginning, recall the radiance interval merging equation,

$$\psi^g\left(\mathbf{r}_0 - s_i\mathbf{\Omega}, \mathbf{\Omega}\right) = \overbrace{\beta^g_{i,i+1}\left(\mathbf{r}_0, \mathbf{\Omega}\right)\psi^g\left(\mathbf{r}_0 - s_{i+1}\mathbf{\Omega}, \mathbf{\Omega}\right)}^{A} + \overbrace{\psi^g_{i,i+1}\left(\mathbf{r}_0, \mathbf{\Omega}\right)}^{B}. \tag{18}$$

The first term, A, is the radiation contribution at the end point of the ray transported over the ray to the start point. The second term, B, is the radiance interval over the ray, and is the integral of the source term transported over the ray. Importantly, A is a straight forward calculation that requires the radiation contribution at the end point, which is solved for recursively (as discussed in Section 3.3.3). On the other hand, B is completely unique to the specific ray being solved over. Thus, the standard method for solving the radiation field globally splits this calculation into two steps. First, all radiance intervals ($\psi^g_{i,i+1}\left(\mathbf{r}_0, \mathbf{\Omega}\right)$) are computed independently. Second, radiance intervals are recursively merged down from the highest cascade to the lowest.

The general algorithm for merging downward is

1. Identify the neighboring probes in the next lower cascade. In Figure 3(ii), this consists of all shown probes in C_1.
2. For each ray traced from the target probe, determine the corresponding rays traced from the higher cascade probes. The corresponding rays are those within the same angular cone.
3. For each target probe ray, sum each associated set of higher cascade rays and bilinearly interpolate the result to the endpoint of the target probe ray.
4. Transport the bilinearly interpolated value from the endpoint of the ray back to the starting point of the ray.

For the simple case presented in Figure 3(ii), we investigate one of the C_0 probes that lies between the four C_1 probes. The schematic for solving for one of the rays on this probe is shown in Figure 4.

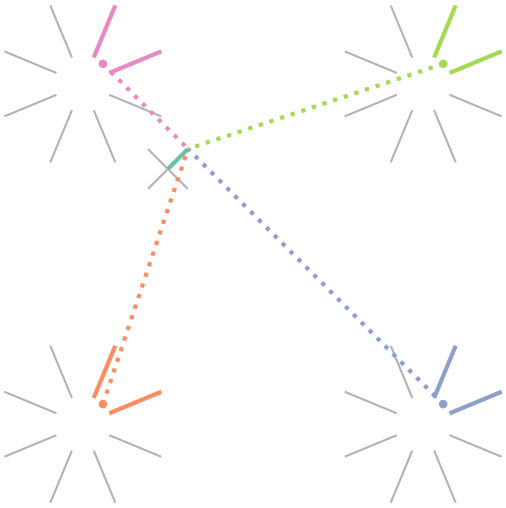


Figure 4: Example of standard merging procedure for single ray on probe of C_0. The highlighted ray on the C_0 probe is the target ray for merging. The highlighted rays on each C_1 probe are summed together to the same colored point. Then, these sums are bi-linearly interpolated from their respective points onto the endpoint of the target ray.

Parallelization The first step, in which all radiance intervals are computed, is trivially parallel. Each radiance interval is completely independent of all others, and so can be computed completely in parallel. The second step complicates this slightly, but is still extremely Parallelizable. Each ray requires read-only access to the associated rays in the nearby probes of the next highest cascade. This results in a graph, specifically a layered directed acyclic graph where each node is a probe and the edge weights are the bilinear interpolation weights. Additionally, each layer is bipartite

with respect to adjacent layers: no ray in C_i requires any information from another ray in C_i, only from C_{i+1}. For the 2-D case with the branching ratio $\alpha = 2$, the majority of probes require communication from 4 probes from the adjacent higher cascade (3-D requires 8). An example graph for the 2-D case with $\alpha = 2$ is shown in Appendix A.

Thus, each merge step, all merges in a single layer (cascade), can be computed completely in parallel. However, the overall merge process is not trivially parallel, as each merge step must be computed in consecutive order.

The first step, computing all radiance intervals, is trivially parallel with both shared and distributed memory parallelism: no rank communication is required whatsoever and each radiance interval is independent. The second step, merging, is more challenging to parallelize with distributed memory parallelism.

3.5 Bilinear Fix

The most notable issue with RCs is ringing artefacts, as shown in the upper image of Figure 5. As discussed by Osborne [14], the origin of this issue lies in the oversampling of a source across cascades. This issue is remedied through the use of the so-called "bilinear fix."

Consider a spherical source in space. Because each probe both within and across cascades is located a different point in space, each will "see" a different contribution of the source to the angular flux at the point. When interpolation is carried out between a higher and lower cascade, we unfairly increase the contributions of the source on the probe; the source is not properly blocked ("occluded") when stepping to a lower cascade. This leads to energy not being conserved at locations where cascades overlap a source.

For illustration, let's say we have two cascades, 0 and 1, where Cascade 0 has 4 equidistant angular directions, and Cascade 1 has 16. Additionally let's assume that the penumbra criterion holds, so there are twice as many probes in Cascade 0 as there are in Cascade 1 to reflect the higher spatial density. As discussed previously, the method of Radiance Cascades evaluates contributions to each probe within distances defined by the radiance interval. This interval is an annulus is 2D and a shell in 3D. The fix is thus:

1. Impose the radiance interval and angular discretization of Cascade i onto the probes in Cascade $i + 1$.

2. Trace rays along the imposed radiance intervals and angular discretizations on Cascade $i + 1$.

3. Use bilinear interpolation to merge these contributions to Cascade i.

This process minimizes parallax effects and ensures occlusion (blocking) from Cascade i is correctly applied. A source of confusion that still exists are the descriptions of additional ray tracing from the higher cascades to the lower. It is not immediately clear from the descriptions and figures provided by Osborne [14] where these steps emerge in the process and what their purpose is.

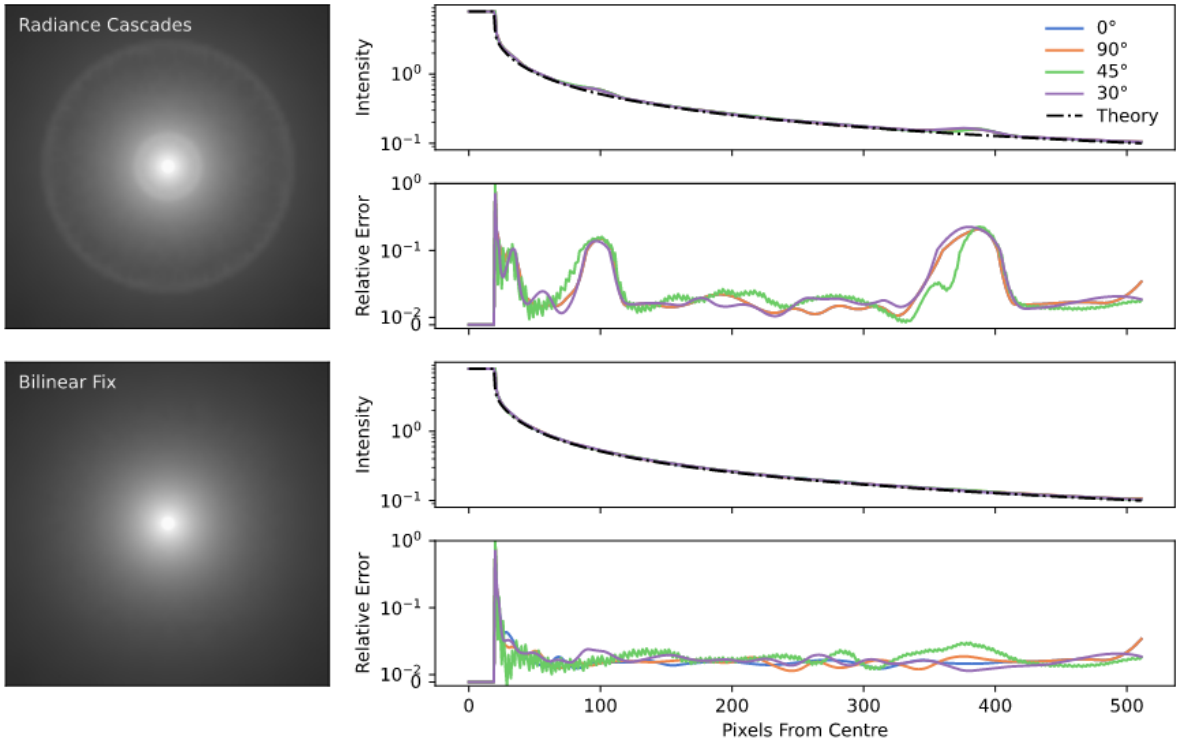


Figure 5: Ringing effect. Taken from [14], figure 7. Caption from original source: *Comparison of falloff around an opaque circular source using basic radiance cascades, and those with the bilinear fix. On the left-hand side, a tonemapped representation of the radiation field is shown. The upper panel of each pair on the right-hand side is a comparison of the intensity falloff against a theoretical model shown for cuts at different angles starting from the centre of the source, whilst the lower panel of each pair is the relative error between the solution along these different cuts and the expected falloff. We note that this is the worst-case scenario for ringing from the basic radiance cascades method as it shows an extremely opaque source embedded in a completely transparent medium.*

4 Proposed Work

This section discusses the implementation of Radiance Cascades, and our proposed work.

4.1 Formalization of the Penumbra Criteria

As discussed in Section 3.2, the penumbra criteria was originally formulated for global illumination problems in video game development. As such, the motivations for its formulation differ from those in the scientific community. That is, greater emphasis is placed on the *speed* of the algorithm, rather than the numerical accuracy. This is clearly seen in the penumbra criteria, which lacks rigor in its definition. Indeed, though initial papers by Sannikov and Osborne acknowledge the proportionality of the required step size needed in the angular and spatial discretizations, neither attempt to describe additional constraints to the requirements; both papers simply assume a coefficient of 1 and claim that this holds generally. As part of this work, we propose to formalize

the penumbra criteria so that the RCs technique and its limitations are clearly understood.

4.2 Prerequisite Code Features

Prior to implementing into a codebase, there are certain features that must exist in order to implement RCs. Additionally, there are highly-desired features that we strongly believe make a code more or less ideal to implement in, but are not explicitly required for implementation.

4.2.1 Required Features

Appropriate Geometric Representation In the current form of RCs, the spacing between probes of the same cascade must be identical. Because of this, the geometric representation in the code must be either structured rectilinear meshes or constructive solid geometry. If the geometric representation is a structured rectilinear mesh, then the vertices would be the centroids of probes. This is the representation used by Osborne for non-LTE radiative transfer [14]. If the geometric representation is constructive solid geometry, then we would simply prescribe the probe centroids and would not be restricted by the mesh. Additionally, using constructive solid geometry would enable simple plug in to Monte Carlo codes down the line.

Multigroup Cross Sections Radiance Cascades is a deterministic method, and so the energy dependence must be discretized using either the multigroup approximation or some other discretization scheme (e.g., charged particle transport). As energy discretization is required, there must be some sort of cross section module in which users specify energy-discretized cross sections for geometric regions.

4.2.2 Desired Features

Shared Memory Parallelism RCs are optimized and well-suited for shared memory parallel execution (trivially parallel). Specifically, radiance intervals can be calculated independently, and then after all intervals have been calculated, interval merging can be calculated independently with read-only access on 'parent' probes. Unfortunately, distributed memory parallelism is a bit more challenging and should be left for future work. Ideally, a code base would already have some notion of shared memory parallelism, especially GPU based parallelism. This would be in the form of OpenMP or CUDA, or ideally a general parallelism interface like OneAPI, Kokkos, RAJA, etc. The benefits of using a general interface are that they are externally maintained and updated, interface with all types of GPUs (NVIDIA, AMD, Intel...) and CPUs (Intel, AMD...) with no additional user effort or code written, and would be updated to reflect the state of the industry (e.g.,new GPU architecture or manufacturer). Among all of the general parallelism interfaces, Kokkos seems to be the front runner because of its simple interface, large user community, highly active development team, large device support, and backing from the Linux foundation. The

next best would be RAJA, which has a smaller community overall and less backing, but has been shown to outperform Kokkos for simple instruction programs [19].

Simple Code Base Ideally, the code base would be relatively simple to develop in. This means the code for the executable would be written in one language, and not, for example, half-python and half-c++ (SWIG, Cython). In our opinion (Nathan), SWIG and Cython drastically worsen the complexity of the build system, without equivalently improving user experience. A system like OpenMC's, where there is a python interface for generating input files but actual code execution occurs entirely in C++, is desired.

4.3 Radiance Cascades Implementation

4.3.1 Toy Problem

Rather than immediately implementing RCs in the general case, we propose to begin with a simple implementation applied to no more than two "toy" problems. These will consist of basic two-dimensional transport problems solved using either Python or C++. The purpose is purely educational and serves two goals:

1. Provide an opportunity to practice implementing RCs on a small, well-defined problem before moving to a generalized code base.

2. Demonstrate early results for the method as applied to neutronics.

To accomplish this, we will start from example problems already developed by the community and tailor the corresponding code to match our problem of interest[4]. The first problem will consider a purely absorbing material with either a simple boundary source or a uniform volumetric source on a basic two-dimensional Cartesian geometry subject to vacuum boundary conditions.

4.3.2 Implementation Target

We investigated four codes to potentially implement RCs in: OpenMC, OpenSN, DexRT, and OpenMOC.

OpenMC OpenMC [20] has a mature code base featuring CSG and MGXS modules, and so it satisfies both required prerequisites. Additionally, OpenMC satisfies both desired features, as it supports shared memory parallelism via OpenMP and has a very developer friendly code base. Unfortunately, we believe OpenMC is not the correct code to implement RCs in, as it is primarily a Monte Carlo code. While OpenMC does have a deterministic solver, TRRM solver, this is only supported due to the inherently stochastic nature of TRRM. Radiance Cascades do not have a stochastic element, and thus do not belong in OpenMC.

[4]This approach was personally recommended by C. M. J. Osborne.

OpenSN OpenSN [10] is a discrete ordinates research code developed by Dr. Ragusa's group at Texas A&M University. OpenSN has a MGXS module and supports shared memory parallelism through OpenMP. OpenSN explicitly supports unstructured meshes, and should be capable of supporting structured meshes through input files, but this feature is unclear. OpenSN is actively maintained, and recently underwent a massive update to the code base, in which a python interface was added. Although OpenSN is now run through it's python interface, the backend is entirely C++, and the interfacing is not done through Cython or SWIG. While OpenSN is an actively maintained deterministic research code, we believe this code is not the correct code to implement in. The goal of OpenSN is to investigate different sweeping algorithms for the discrete ordinates method, and so implementing RCs does not align with the goal of the code.

DexRT DexRT [14] is a non-LTE radiative transfer code, and is the first published RCs implementation for physics applications. DexRT supports structured rectilinear meshes, and uses Kokkos for portable shared-memory parallelism. Unfortunately, due to the nature of non-LTE radiative transfer, the representation for 'opacity' (the non-LTE radiative transfer equivalent to cross-sections) differs from a standard MGXS representation. The representation for opacity is significantly more complicated, as the opacity is strongly coupled to the radiation field, but still stores mean values for each discretized wavelength (the equivalent to discrete energy groups). Thus, DexRT is capable of representing MGXS, and the implementation would only require adding input file parsing and a removal of cross-section dependence on the radiation field. Additionally, as DexRT is a implementation of RCs for a physics application, it is adaptable to solving the LBE. Despite mostly satisfying the prerequisites, there are a few notable considerations for implementing in DexRT:

1. DexRT is in rapid but sporadic development, and is maintained by a single person.
2. DexRT solves for non-LTE radiative transfer, and this problem is heavily baked into the code base. An abstraction to also solve the LBE could be tricky to implement, and may not be supported by Osborne, as the problems are very different.

Given these, we believe if DexRT is chosen as the implementation target, the best course of action would be to create a fork and alter the code base independent of DexRT.

OpenMOC OpenMOC [11] is an open source MOC code developed by Dr. Forget's group at the Massachusetts Institute of Technology. OpenMOC uses CSG for geometric representation, has a MGXS module, and uses OpenMP or CUDA for shared memory parallelism. In our opinion, the code base is relatively complex, due mostly to design choices motivated by the computational environment at the time of writing. For example, the code uses SWIG for python bindings of C++ code, which adds a complicated step to the build system. OpenMOC also has distinct code written for CPUs and GPUs, and, due to the state of OpenMP and CUDA at the time of writing, has many explicit memory calls. These features add maintenance burden and are no longer required

given the recent introduction of portable parallelism interfaces like Kokkos and RAJA. Additionally, OpenMOC is outdated and will require code maintenance and updates prior to implementing RCs. This will be discussed further in Section 4.3.3. That said, we believe that of the four codes investigated, OpenMOC is clearly the code to implement RCs into for a few reasons. OpenMOC satisfies most of the prerequisite features we defined previously, only missing the mark on code base simplicity; something that can be remedied through maintenance and updates. Additionally, the intent of OpenMOC was to investigate and demonstrate long characteristic methods for solving the LBE. OpenMOC is also an open-source package, that is nearly in the code graveyard of github, and could be revived if we implement RCs into OpenMOC.

4.3.3 Implementation Outline

This section assumes we will be implementing into OpenMOC, and will layout a rough estimate on the work for the initial RCs implementation.

Required Maintenance OpenMOC is sporadically maintained, and as such has a few outdated dependencies. Specifically, the build system uses some deprecated packages, like `distutils`, and the minimum CMake version (2.8) has been deprecated since 2020.

More significantly, OpenMOC was intended to be able to interface with OpenMC, specifically for geometry and cross section generation. However, OpenMC has developed new surfaces that are not implemented in OpenMOC, such as `x/y` cylinders, quadrics, spheres, cones, and torii. Additionally, OpenMOC only supports reading MGXS from OpenMC for cell and material domain types, while OpenMC supports generating cross sections for distributed cells, meshes, and universes. Thus, OpenMOC is not currently fully compatible with OpenMC geometry or cross section generation. Before implementing RCs, we believe OpenMOC should be able to do all the things it's documentation says it can do, and so these must be updated.

Desired Updates There are *several* desired updates to OpenMOC that we believe will be extremely helpful for implementing RCs:

1. Code organization: The C++ source code is currently all stored in the same directory. We propose organizing the code into logical sub-directories, similar to the organization of INL's MOOSE framework.

2. Code complexity: The C++ source integration into Python using SWIG complicates the code and build system. We propose separating the C++ source and Python interface, similair to how OpenMC works.

3. Code cleanliness: Some of the source files are extremely long, and could benefit from templating and unnecessary function/variable removal.

4. Linear Algebra: OpenMOC has a CMFD solver implemented, where the data storage (sparse matrix) and the solver itself are written natively. For the sake of conciseness and performance, we propose bringing in PETSc for the sparse linear algebra that CMFD requires. This would drastically decrease the volume of code written for the CMFD solver, and entirely remove the need of the `vector` class. Additionally, using PETSc guarantees adequate performance, scalability, and portability (PETSc supports Kokkos).

5. Portable Parallelism: The parallel solvers currently use CUDA or OpenMP for shared memory parallelism. Because of this, there are two families of solvers in OpenMOC, those written with CUDA and those written with OpenMP, that have nearly identical code. Additionally, because the GPU solver is written in CUDA, it is not portable between GPU systems. We propose bringing in a package like Kokkos or RAJA for shared (and distributed) memory parallelism. This change would drastically reduce the size of the codebase, enable portable parallelism in the future, and simplify the codebase (increased readability); all without losing functionality or performance.

Implementation Goals For the initial implementation, we propose the following required features:

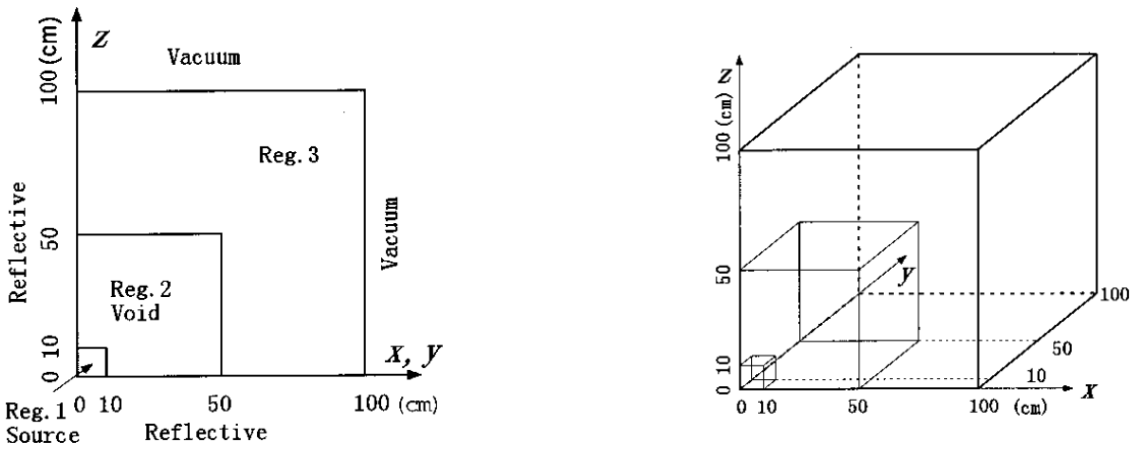
1. Support for arbitrary geometries through CSG representation
2. MGXS input parsing from a tool like OpenMC, NJOY, or some other code
3. Steady-state 2D RCs solver
4. Capability to handle both multiplying and fixed-source problems
5. The standard RCs merging algorithm and Bilinear fix merging algorithm

The first two features already exist in OpenMOC, but are included for completeness and clarity.

4.4 Benchmarks

Benchmark cases for RCs must be chosen to specifically show the advantages of the method and directly test for known problems in other methods, such as ray-effects in SN. To be considered successful, we will need to implement RCs in such a way that not only converges to the correct solution, but can also do so faster than proven methods such as Discrete Ordinates (SN). Or, alternatively, must be proven to scale well using GPU parallelism. We propose the following benchmark cases be used for the development and testing of RCs.

The application of RCs to the Boltzmann Transport Equation (BTE) has been largely proven by Osborne and Sannikov [14], though with a particular application to photon transport in astrophysics. There are two main differences between this application and ours:



(a) 2D **(b)** 3D

Figure 6: First Kobayashi benchmark (taken from [21]).

1. Neutron transport simulations are typically focused on length scales in the centimeters to meters, not megameters and light-years.

2. The source term is dependent on the flux itself in both multiplying and nonmultiplying systems.

Item 1 will be straightforward to show, as there are many benchmarks that allow for testing neutron streaming. One such set, known as the Kobayashi benchmarks [21], is an industry standard and has been used consistently in modern nuclear software, such as the discrete ordinates code Denovo developed by Oak Ridge National Laboratory [8]. The benchmarks look to test streaming through voids from a fixed source into a non-multiplying material and can be extended to also test for scattering source convergence with minor changes to material properties.

There are three associated geometries:

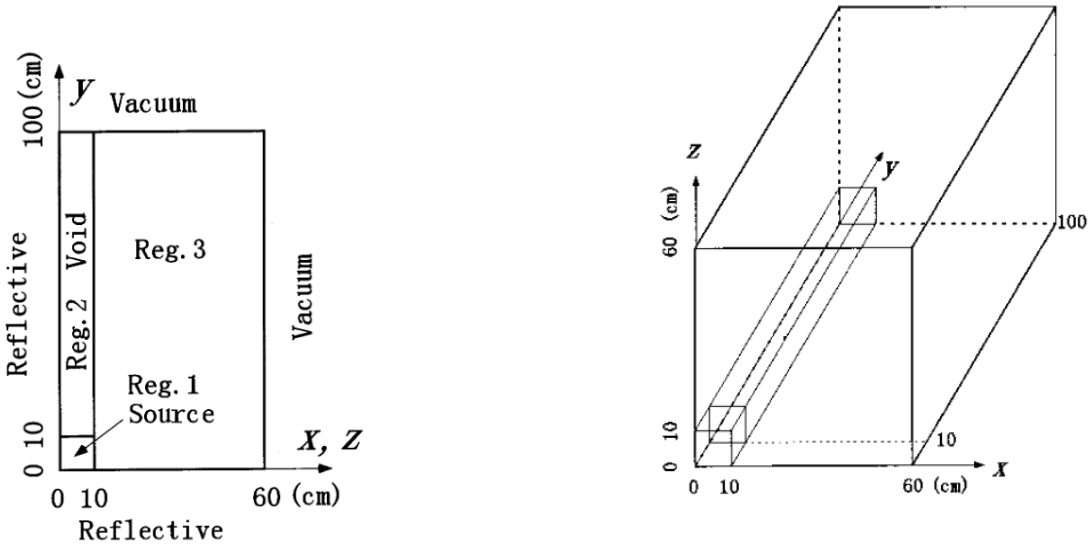
1. A source surrounded by a cubic void (Figure 6).

2. A source with a vertical (chimney-like) void (Figure 7).

3. A source with a "dog leg"[5] void duct (Figure 8).

The surrounding non-void region is either strictly absorbing or 50% absorbing with 50% scattering (of the total cross section). Values for these cases can be found in Table 1. The ultimate goal will be to report on RCs's capabilities on the 3D problems as they were originally published. However, we propose to start with 2D "flattened" versions of these problems for a proof-of-concept internally, then move on to 3D implementation. As mentioned above, the strengths in this suite of benchmarks lie in the ability to provide both challenging streaming geometries while also testing for accurate source terms from phase space in-scattering.

[5]Yes, that is the actual name in the paper.

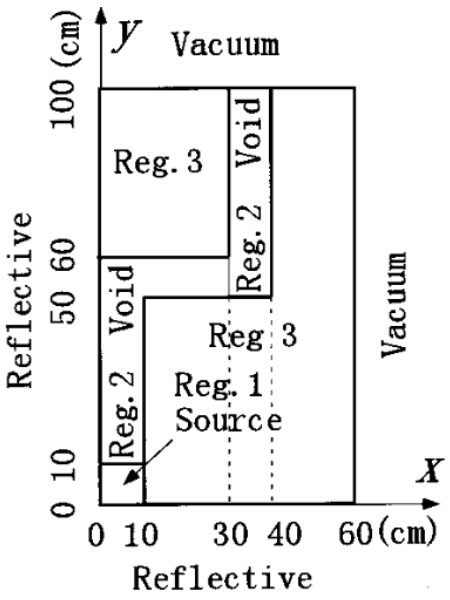


(a) 2D **(b)** 3D

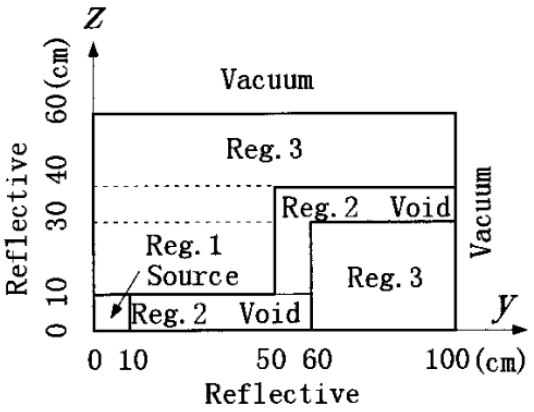
Figure 7: Second Kobayashi benchmark (taken from [21]).

			Problem 1	**Problem 2**
Region	**Source** [n/cm^3-s]	Σ_t [1/cm]	Σ_s [1/cm]	
1	1	0.1	0	0.05
2	0	1×10^{-4}	0	0.5×10^{-4}
3	0	0.1	0	0.05

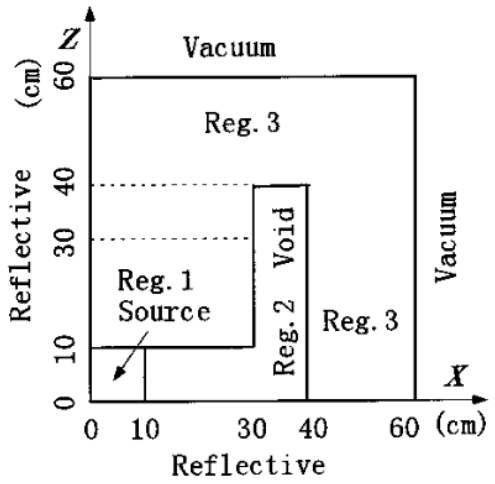
Table 1: Source strength and material properties for Kobayashi benchmark problems.



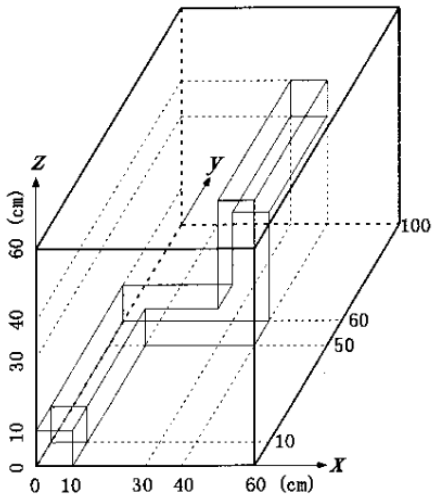
(a) 2D $x - y$ plane.



(b) 2D $y - z$ plane.



(c) 2D $x - z$ plane.



(d) 3D

Figure 8: Third Kobayashi benchmark (taken from [21]).

4.5 Points of Discussion

This section highlights specific ideas we believe may fit into both future work and a first implementation.

3D Implementation A 3D implementation of RCs is absolutely feasible, however the scaling law that makes RCs so interesting in 2D does not apply in 3D (at least not as trivially as 2D).

Standalone CSG Package A standalone constructive solid geometry package, shared by major open source Monte Carlo codes such as OpenMC and MCDC, would greatly support the development of deterministic solvers intended to plug into Monte Carlo codes for generating variance-reduction parameters. The idea is that deterministic-method developers could simply import this package and focus solely on implementing their solver, while the shared CSG library would handle geometry as well as both continuous-energy (CE) and multigroup (MG) cross-section representations. Such an approach would significantly accelerate the development of RCs codes, whether integrated into OpenMOC or another code, by eliminating the need to independently update/implement and maintain CSG and MGXS infrastructure within the Radiance Cascades (RCs) codebase. Essentially, this CSG package is intended to serve a similar purpose to LibMESH in FEM problem codes.

Source Representation A major limitation to MOC (and The Random Ray Method (TRRM)) is the required use of source regions. This limitation is entirely because MOC solvers do not know the angular/scalar flux at discrete points throughout the problem domain. RCs do not suffer this limitation, and as such do not necessarily rely on source regions. Instead, the source along each radiance interval can be computed as a continuous function by interpolating from the scalar flux solution of nearby C_0 probes. Although this formulation is more complicated than flat- or linear-source regions, it is still possible to analytically solve the integral of the source term. As such, we think using this representation of the source term would drastically improve the overall accuracy of the method, compared to using source regions, at minimal added computational cost.

Distributed Memory Parallelism A massive limitation of the current versions of RCs is the lack of distributed memory parallelism. This is due predominantly to lack of interest from the computer graphics community, as they interact with single devices and thus rely enitrely upon shared memory parallelism.

This is problematic for a neutronics implementation of RCs, as many nuclear problems are large and complicated, and may need to run on HPCs. For RCs to run on HPC systems, there must be some form of distributed memory parallelism. This can be achieved in a few different ways.

The first method we though of is a form of domain decomposition, and would be generally applicable. Both the standard and bi-linear fix merging problem can be represented by a graph,

Figure 11, that has a structure we can take advantage of. Specifically, the graph is a layered graph, where each layer is bipartite with respect to neighbor layers. Additionally, each node in the graph (probe) follows a common stencil, where it has γ higher probes it interpolates from, and γ^2 lower probes it interpolates on, where γ is 4 and 8 for 2-D and 3-D problems, respectively. Thus, we can perform domain decomposition by partitioning only the set of C_0 probes across MPI ranks. For each owned C_0 probe, we then trace its 'lineage' through the cascade hierarchy by identifying all upstream probes in coarser cascades that contribute to it according to the merge graph. Nearby C_0 probes share most (if not all) higher probe dependencies, causing the number of required C_1 probes to grow sublinearly with the number of owned C_0 probes. This applies for all neighboring cascades, and so the total number of probes owned per MPI rank will be dominated by C_0 probes. As a result, rather than decomposing coarser cascades globally, each MPI rank locally instantiates copies of only the coarser probes appearing in the lineage of its assigned C_0 probes. There will be some overlap in probes from higher cascades across MPI ranks, but this can be treated by communication or by each rank independently computing all higher cascade probes (resulting in no communication, but recomputation of certain probes). If each rank independently computes all higher cascade probes, the only communication required would be at the end of each inner source-iteration step when the source is updated. The benefit to partitioning the C_0 probes decreases as the number of C_0 probes approaches the number of higher cascade probes (mainly C_1). This breakdown occurs when the number of MPI ranks is roughly $1/\gamma$ of the number of C_0 probes in the problem, or $1/4$ and $1/8$ for 2-D and 3-D, respectively.

The second method is a parallel in energy scheme, in which each energy group is solved independently of each other on distinct MPI ranks. This would be distributed in the inner loop of something like source iteration, as after each solution, all groups would need to be brought back to update next iteration sources. Notably, the number of MPI ranks is limited to the number of energy groups, and so this may not be particularly helpful for standard neutronic simulations.

5 Future Work

5.1 Multiplying Media: C5G7 Benchmark

In addition to in-scattering sources, we will also be interested in testing sourcing contributions from a multiplying region (fission). One of the most well-known benchmarks is the C5G7 benchmark, which models a simplified, reflected, quarter-core reactor with mixed-oxide and uranium-oxide fuel. The core is moderated with water. As described in [22], this problem looks to challenge codes' abilities to handle spatial heterogeneity. A general schematic of this benchmark has been given in Figure 9, and we refer the reader to the Nuclear Energy Agency report [22] for specifics. One advantage to choosing this benchmark is the ability to expand to time-dependent testing, as development is currently underway for this extension [23].

The implementation of C5G7 requires a mostly-mature code to implement, with the major challenges being:

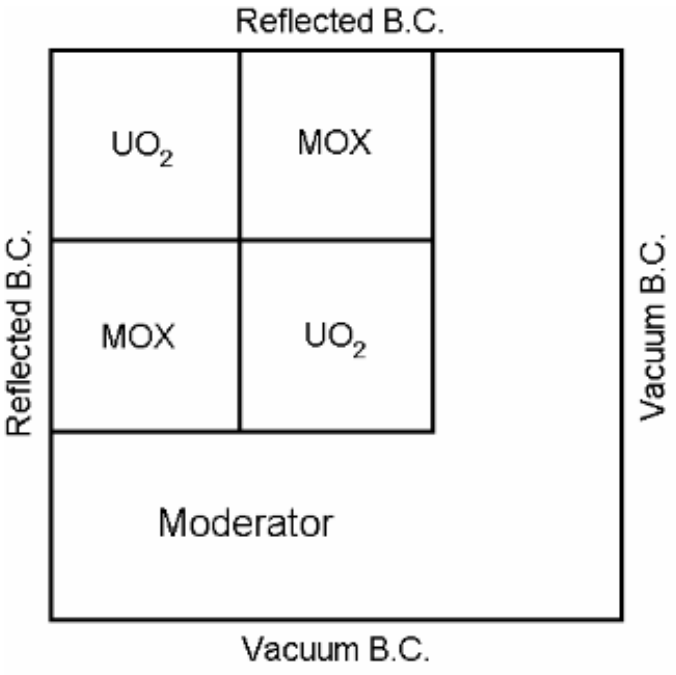


Figure 9: General schematic of the C5G7 benchmark.

1. 3D implementation.
2. Complicated geometry.
3. Multiple materials.
4. Varying length scales (ex. between pincells vs. from assembly to vacuum boundary).

Therefore, we propose to first create test problems using OpenMC, Serpent, or MCNP6 with multiplying media and simple geometry. This will allow for the continued development and refinement of source iteration methods. Then, we will build up to more complicated geometries, finally arriving at the C5G7 benchmark, which will be reported on and published.

5.2 Unstructured Probe Placement and Improved Interpolation Strategies

A natural extension of this work is to explore unstructured probe placement for radiance cascades, especially in constructive solid geometry (CSG) models, and to investigate higher-accuracy interpolation schemes between probes.

Modern mesh generators (e.g., `gmsh`) already provide algorithms capable of placing nodes throughout arbitrary domains. Because Radiance Cascades primarily impose constraints on probe spacing relative to angular resolution, these tools could be adapted directly. In the current formulation, each cascade increases its spatial spacing by a factor of two. Thus, cascade 0 could be

produced using a mesh with a prescribed minimum spacing, and higher cascades could be derived simply by doubling a global spacing parameter. This value must be enforced as a *maximum* spacing to preserve the penumbra criterion.

Unstructured probe locations also raise the question of accurate interpolation. The process resembles finite element interpolation, though without the overhead of element-level basis functions and their gradients. Since RCs only require interpolation between probe values, schemes such as linear, quadratic, or even spline-based interpolation could be evaluated. While spline-type methods may introduce additional computational cost, the mathematical framework is straightforward and could significantly improve accuracy.

Finally, each probe would need to store its angular flux information and a record of its interpolation neighbors. These neighbors could be identified using standard nearest-neighbor search routines or by reusing adjacency information provided by FEM meshers.

Overall, these directions point toward a general framework for extending cascades to unstructured CSG geometries while improving the fidelity of interpolated fields. There is currently interest within the Radiance Cascades community, though it is unclear what the application or progress for the methods are.

5.3 Transient Simulations

Time-dependent problems are the main use case of deterministic solvers, and so a transient solver should be implemented.

5.4 Problem Sparsity

Osborne discusses how in non-LTE radiation transport, many problems are very sparse, and so substantial computation is wasted on void regions [14, 18]. This is similair to space applications, shut down dose rate calculations, and typical charged particle transport problems. Osborne had some interesting ideas, namely using sparse graphs like VDB (which is a large package we could bring in and not worry about maintenance) and prefiltering void probes out of the problem. The prefiltering allows us to skip computing probes that we know do not contribute, drastically saving on compute cost, and the sparse rooted graphs efficiently store linked probes minimizing look up times when merging. This needs to be investigated a lot more before formalizing though, unsure as to how to filter out probes.

5.5 Adjoint Solver

One of the end goals of RCs could be to generate weight windows for variance reduction in a Monte Carlo code. To do so, an adjoint solve mode must be implemented.

5.6 Acceleration Methods

There are various methods employed in deterministic neutron transport methods for convergence acceleration that could be used for RCs. Specific methods we think would be interesting to investigate are CMFD, DSA, and TSA. CMFD has already been used for MOC codes (and is implemented in OpenMOC and OpenMC) and has proven to be a great accelerator. DSA is a super effective method for accelerating source iteration, but is only effective for highly diffusive problems/regions. TSA is a more robust method for accelerating source iteration, is well suited for highly anisotropic scattering problems/regions and sparse/void/near-void problems and regions; at added computational cost compared to DSA.

5.7 Charged Particle Transport

Nathan's work will be in charged particle transport, so a RCs formulation for charged particle transport will be required. This is in future work because of the added complexity of the Boltzmann-Fokker-Planck equation compared to the LBE.

References

[1] George Bell and Samuel Glasstone. *Nuclear Reactor Theory*. Litton Educational Publishing, inc, 1970.

[2] E. E. Lewis and W. F. Miller. *Computational Methods of Neutron Transport*. 1984.

[3] Thomas Camminady, Martin Frank, Kerstin Küpper, and Jonas Kusch. Ray effect mitigation for the discrete ordinates method through quadrature rotation. *Journal of Computational Physics*, 382:105–123, April 2019.

[4] Stefan T. Thynell. Discrete-ordinates method in radiative heat transfer. *International Journal of Engineering Science*, 36(12):1651–1675, September 1998.

[5] S. Chandrasekhar. *Radiative Transfer*. Dover Publications, New York, 1960–1950.

[6] Bengt G. Carlson. A Method of Characteristics and Other Improvements in Solution Methods for the Transport Equation. *Nuclear Science and Engineering*, 61(3):408–425, November 1976.

[7] B G Carlson and G I Bell. SOLUTION of the TRANSPORT EQUATION by the Sn METHOD. Technical report, Los Alamos Scientific Lab., N. Mex., USA, October 1958.

[8] Thomas M. Evans, Alissa S. Stafford, Rachel N. Slaybaugh, and Kevin T. Clarno. Denovo: A New Three-Dimensional Parallel Discrete Ordinates Code in SCALE. *Nuclear Technology*, 171(2):171–200, August 2010.

[9] RANDAL S BAKER. Time-dependent, parallel neutral particle transport code system. Technical report, Oak Ridge National Laboratory (ORNL), Oak Ridge, TN (United States), 2009.

[10] Open-Sn/opensn. Center for Large Scale Scientific Simulations, December 2025.

[11] William Boyd, Samuel Shaner, Lulu Li, Benoit Forget, and Kord Smith. The OpenMOC method of characteristics neutral particle transport code. *Annals of Nuclear Energy*, 68:43–52, June 2014.

[12] K. D. Lathrop. Ray Effects in Discrete Ordinates Equations. *Nuclear Science and Engineering*, 32(3):357–369, June 1968.

[13] K. D. Lathrop. Remedies for Ray Effects. *Nuclear Science and Engineering*, 45(3):255–268, September 1971.

[14] C M J Osborne and A Sannikov. Radiance cascades: A novel high-resolution formal solution for multidimensional non-LTE radiative transfer. *RAS Techniques and Instruments*, 4:rzae062, January 2025.

[15] Amit Ofer. A summary of real time ray tracingn techniques in video games and simulations. Master's thesis, The Open University of Israel, March 2023.

[16] Alexander Sannikov. Radiance Cascades: A Novel Approach to Calculating Global Illumination[WIP].

[17] Rouli Freeman, Alexander Sannikov, and Adrian Margel. Holographic Radiance Cascades for 2D Global Illumination, May 2025.

[18] Christopher M. J. Osborne. A Simple Ray Acceleration Structure for Non-LTE Radiative Transfer, November 2025.

[19] J H Davis, P Sivaraman, I Minn, K Parasyris, H Menon, G Georgakoudis, and A Bhatele. An Evaluative Comparison of Performance Portability across GPU Programming Models.

[20] Paul K. Romano, Nicholas E. Horelik, Bryan R. Herman, Adam G. Nelson, Benoit Forget, and Kord Smith. OpenMC: A state-of-the-art Monte Carlo code for research and development. *Annals of Nuclear Energy*, 82:90–97, August 2015.

[21] Keisuke Kobayashi, Sugimura Naoki, and Yasunobu Nagaya. 3-D RADIATION T RANSPORT BENCHMARK PROBLEMS AND RESULTS FOR SIMPLE GEOMETRIES WITH VOID REGIONS, November 2000.

[22] *Benchmark on Deterministic Transport Calculations without Spatial Homogenisation: MOX Fuel Assembly 3-D Extendion Case*. OECD Nuclear Energy Agency, Paris, 2005.

[23] Deterministic Time-Dependent Neutron Transport Benchmark without Spatial Homogenisation (C5G7-TD). https://www.oecd-nea.org/jcms/pl_32145/deterministic-time-dependent-neutron-transport-benchmark-without-spatial-homogenisation-c5g7-td.

A Graph Representation of Merge Step

For the example problem presented in Figure 10, the graphical represenation of the merge step is presented in Figure 11.

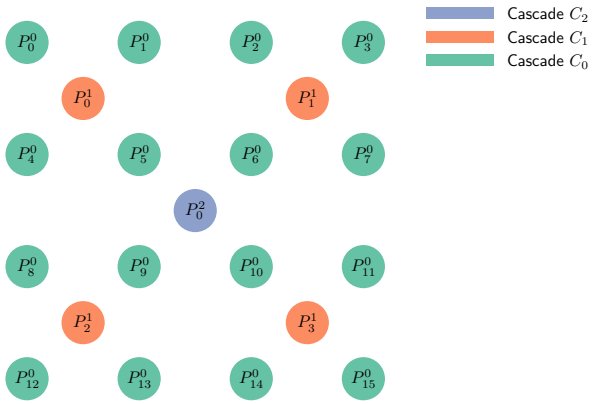


Figure 10: Example problem of 3 cascades, with 16 probes in C_0 for simplicity. The upper index of each probe corresponds to the cascade, and the lower index is the probes index in the cascade.

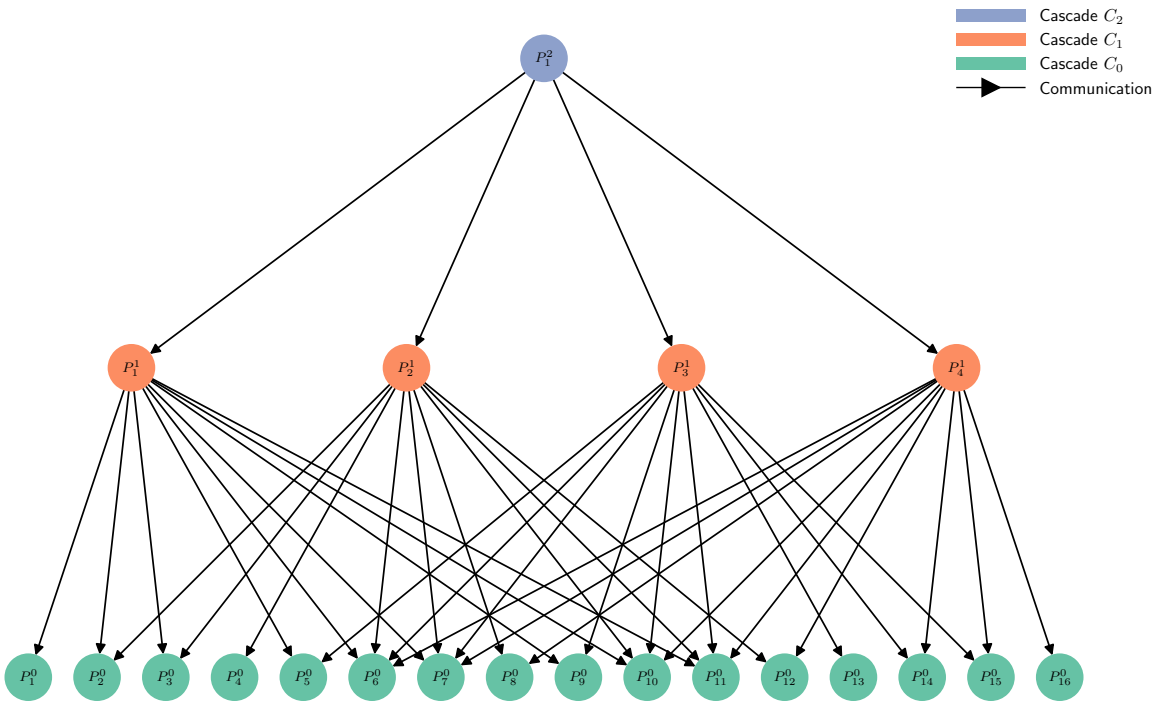


Figure 11: Example merge graph for 3 cascades, with 16 probes in C_0 for simplicity. The upper index of each probe corresponds to the cascade, and the lower index is the probes index in the cascade.