The Random Ray Method versus Multigroup Monte Carlo: The Method of Characteristics in OpenMC and SCONE

Paul Cosgrove\textsuperscript{a}, John R. Tramm\textsuperscript{b}

\textsuperscript{a}Department of Engineering, University of Cambridge, Trumpington Street, Cambridge, CB2 1PZ, United Kingdom
\textsuperscript{b}Argonne National Laboratory, 9700 S. Cass Avenue, Lemont, IL, USA

Abstract

The Random Ray Method (TRRM) is a recently developed approach to solving neutral particle transport problems based on the method of characteristics. While the method previously has been implemented only in closed-source or limited-functionality codes, this work describes its implementation in two open-source Monte Carlo codes, OpenMC and SCONE. The random ray implementations required small modifications to the existing multigroup Monte Carlo (MGMC) solvers, offering a rare venue for redundant, fine-grained, “apples-to-apples” speed and accuracy comparisons between transport methods. To this end, TRRM and MGMC solvers are evaluated against each other using each code’s native capabilities on reactor eigenvalue problems with different degrees of energy discretization. On the C5G7 benchmark (featuring only seven energy groups), TRRM achieves a maximum pin power error comparable to or lower than that of MGMC for a given runtime. On a problem with 69 energy groups, MGMC is found to scale more efficiently, obtaining a lower pin power error for a given runtime. However, the defining difference between the two transport methods is found to be their vastly different uncertainty distributions. Specifically, TRRM is found to maintain similar levels of accuracy and uncertainty throughout the simulation domain, whereas MGMC can exhibit orders of magnitude greater errors in areas of the problem that feature low neutron flux. For instance, TRRM provided up to a 373\times speed advantage compared with MGMC for computing the flux in low-flux regions in the moderator surrounding the C5G7 core.

Keywords: Monte Carlo, The Random Ray Method, Neutron Transport, Method of Characteristics

1. Introduction

Recently, The Random Ray Method (TRRM) has been developed as a scalable and low-memory particle transport algorithm for exascale computing (Tramm et al., 2017, 2018\textsuperscript{b}; Tramm, 2018). It is based on the method of characteristics (MoC) (Askew, 1972) but using a stochastic quadrature that is resampled each transport sweep—in effect converting the traditionally deterministic MoC into a fully stochastic process. This stochastic quadrature and the ensuing on-the-fly ray tracing allow for massive reductions in memory and ray density requirements as compared with MoC, while still sharing the MoC algorithm’s ease of SIMD vectorization across energy groups (Boyd et al., 2014; Tramm et al., 2017; Giudicelli et al., 2019), better utilizing vector registers on modern processors.

To date, only three implementations of TRRM exist: ARRC (Tramm et al., 2020), MOCKingbird (Gaston et al., 2021), and Minray (Tramm, 2023a). Both ARRC and MOCKingbird are optimized, high-performance implementations of TRRM that are capable of performing full-core simulations of light water reactors on supercomputers. ARRC uses a constructive solid geometry (CSG) representation whereas MOCKingbird operates on an unstructured mesh representation. While the numerical performance of ARRC’s TRRM on light water reactor problems is well detailed in Tramm et al. (2018\textsuperscript{b,a}), MOCKingbird’s TRRM implementation has not yet been fully detailed in the literature. Both of these implementations are also closed source. The third application, Minray, is in miniapp form, which is intended only for exploring scalability in a simple manner and is limited to highly simplified Cartesian mesh geometries.
This paper has two goals. The first and most important is to investigate the relative performances of TRRM and multigroup Monte Carlo (MGMC). Many parallel exists between these two transport methods, but no critical comparison has been performed so far. We do this by implementing TRRM in two open-source Monte Carlo codes, OpenMC (Romano et al., 2015) and SCONE (Kowalski et al., 2021). By using two separate codes, we hope to draw conclusions about the relative performance of the two methods which are independent of any design decisions taken by either OpenMC or SCONE. The implementations in these codes also has the added benefit of allowing a performance evaluation of TRRM in a general universe-based CSG geometry, as opposed to the more limited CSG geometry implemented in ARRC.

The second goal of the work is to demonstrate that one can straightforwardly modify a MGMC code to support TRRM. In fact, this is arguably much simpler to do than it is to modify an MoC code to support TRRM. Implementing TRRM in a Monte Carlo code has several advantages. First, there may be synergies in having a second type of transport solver tightly coupled to a Monte Carlo code with respect to, for example, variance reduction. Second, since the geometry routines in Monte Carlo solvers are much more general than those in ARRC, TRRM can be applied to a wider variety of problems than before. Third, implementing TRRM amortizes the effort of supporting multigroup nuclear data in these applications. In fact, in Askew’s original manuscript on MoC, he says the following:

One of the features of the method proposed [MoC] is that . . . the tracking process needed to perform this operation is common to the proposed method . . . and to Monte Carlo methods. Thus a single tracking routine capable of recognizing a geometric arrangement could be utilized to service all types of solution, choice being made depending which was more appropriate to the problem size and required accuracy. [Askew, 1972]

This paper is laid out as follows: an overview of both TRRM and MGMC is given; the two methods are compared; the changes required by OpenMC and SCONE to support TRRM are detailed and discussed, including new optimizations; numerical tests are described and the results discussed; and a summary and conclusion are given.

For the curious reader, the implementations can be found at Tramm (2023b) and Cosgrove (2023), respectively, with the hope of merging each into the main branch of their code bases in the near future.

2. Methodology

2.1. The Random Ray Method

Although TRRM has been described in detail elsewhere (Tramm et al., 2017), a summary and discussion of TRRM are warranted here. TRRM is a variant of MoC; that is, it numerically approximates the solution to the Boltzmann neutron transport equation (a partial differential equation, or PDE) by decomposing the PDE into a family of characteristic ordinary differential equations (ODEs) that can each be solved for analytically. Each ODE represents a single characteristic line through the geometry, upon which the transport equation simplifies and can thus be written and solved for with the use of a multigroup approximation in energy. In this work we refer specifically to the steady-state characteristic equation:

\[
\frac{d}{ds} \psi_g + \Sigma_t \psi_g = Q_g .
\]  (1)

Here \( \psi_g \) is the neutron angular flux in energy group \( g \) along the direction of the given characteristic of the transport equation, \( \Sigma_t \) is the total or transport macroscopic cross section in group \( g \), \( s \) is the distance along the characteristic, and \( Q_g \) is the neutron source in group \( g \) producing neutrons (both from fission and in-scattering) in the direction of the characteristic. The characteristics of the transport equation are straight lines in particular directions; following some additional approximations, Eq. [1] is solved along many different characteristics to give a numerical solution to the full neutron transport equation.

The additional approximations that are commonly applied assume flat source regions (FSRs) where the neutron source is spatially constant. One also commonly assumes that the neutron source is isotropic, giving \( q_g = \int \Omega Q_g / 4\pi \). These approximations still result in overall high fidelity in the context of reactor simulation problems that utilize transport-corrected multigroup cross sections and fairly small FSRs (typically with dimensions well below that of a neutron’s mean free path) and hence are used in this work. That said, it is possible to relax both the flat source approximation and the assumption of source isotropy (Ferrer and Rhodes, 2017). However, given these assumptions, one can explicitly solve Eq. [1] across an FSR as

\[
\psi_g(s) = \psi_g(0) e^{-\Sigma_t s} + \frac{q_g}{\Sigma_t} \left( 1 - e^{-\Sigma_t s} \right) .
\]  (2)

Here \( s \) is the distance that a characteristic traverses across the FSR, \( \psi_g(0) \) is the angular flux of the characteristic entering the FSR in group \( g \), and both \( q_g \) and
\( \Sigma_{\text{g}} \) are unique to the given FSR. As stated above, MoC solves this equation across many distinct characteristic tracks, aiming to adequately cover the spatial and angular phase space of the problem. In solving this equation, one obtains an angular flux update across the FSR for a given ray:

\[
\Delta \psi_g = \left( \psi_g(0) - \frac{q_g}{\Sigma_{\text{g}}} \right) \left( 1 - e^{-\Sigma_{\text{g}} r} \right). \tag{3}
\]

As well as updating the angular flux, when a ray traverses an FSR, it must also increment the scalar flux tally in the FSR, \( \psi_g \) (which is initialized at zero to begin), as

\[
\psi_g \rightarrow \Delta \psi_g. \tag{4}
\]

In traditional MoC, this update typically includes some weighting to account for deterministic quadrature biases. In random ray, however, no weighting scheme is needed because all rays are generated uniformly in space and angle and all rays traverse the same distance. MoC is particularly amenable to vectorization because once the distance across an FSR is determined, the ensuing operations (evaluating an exponential, evaluating \( \Delta \psi_g \), and updating the angular and scalar flux) can be performed across all energy groups simultaneously.

When all rays have traversed their allotted portion of phase space and accumulated updates to the scalar flux tally, a transport sweep is complete, and the scalar flux in each FSR and group can be calculated as

\[
\phi_g = \frac{4\pi w \psi_g}{\Sigma_{\text{g}} V} + \frac{4\pi q_g}{\Sigma_{\text{g}}} . \tag{5}
\]

Here \( V \) is the unitless (or relative) volume of the FSR, which is not typically known a priori (because of the often odd shapes of FSRs in arbitrary user-defined constructive solid geometry), so it is estimated numerically based on the total distance traveled by all rays that have crossed the region compared with the total distance traveled by all rays throughout the entire simulation geometry. Notably, \( V \) is averaged across the entire simulation (not just for a single iteration), to avoid the equation becoming a ratio estimator, a simple method that is discussed in more depth in Tramm et al. (2020). If desired, \( V \) can also be converted to a physical volume by simply multiplying by the full simulation domain volume. Additionally, \( w \) represents the area of each ray which, since all rays are equally weighted, has a value of 1. Subsequently, the criticality eigenvalue, \( k \), can be updated as

\[
k^{(n+1)} = k^{(n)} \frac{\sum_r V_r \sum_{\text{g}} \psi_{\text{g},r}^{(n+1)} \chi_{\text{g}}}{\sum_r V_r \sum_{\text{g}} \psi_{\text{g},r}^{(n)} \chi_{\text{g}}}. \tag{6}
\]

Here the \( r \) index is used to denote unique FSRs and their associated attributes, \( n \) denotes the transport sweep iteration number, and \( \psi_{\text{g},r} \) is the fission neutron production cross section in energy group \( g \) in region \( r \). The source terms in each region are also updated as

\[
F_g = \chi_g \sum_{g'} \nu \Sigma_{\text{g},r} \phi_{g'}^g, \tag{7}
\]

\[
S_g = \sum_{g'} \Sigma_{\text{g},r} \phi_{g'}^g \tag{8}
\]

\[
q_g = \frac{1}{4\pi} \left( S_g + \frac{1}{k} F_g \right). \tag{9}
\]

Here \( F_g \) is the fission source in group \( g \), \( \chi_g \) is the fission spectrum in group \( g \), \( S_g \) is the scattering source in group \( g \), and \( \Sigma_{\text{g},r} \) is the scattering cross section from group \( g' \) to group \( g \).

Boundary conditions are required to solve the transport equation: these are most commonly reflective or vacuum in MoC. In standard deterministic MoC implementations that leverage a cyclical quadrature, the characteristic tracks are each swept once per iteration. Each track begins starting on a boundary where the angular flux values are known (e.g., as input from a ray with a complementary angle terminating at that location, the previous iteration \( n - 1 \), or zero if on a vacuum boundary condition) and is swept through the domain, applying Equation 3 for each FSR it traverses, until it reaches another boundary where its angular flux is banked as a starting condition for a track in the subsequent iteration.

TRRM differs from standard MoC implementations in several ways. First, there is no predetermined characteristic quadrature: whereas standard deterministic MoC normally uses a cyclic quadrature, in TRRM all characteristics begin with a randomly sampled initial position and direction that vary from one transport sweep to the next. Two key implications of this difference are that ray tracing information cannot be computed once and reused over all iterations (i.e., on-the-fly ray tracing must be performed), and the initial angular flux spectrum of each sampled ray is unknown. On-the-fly ray tracing adds additional computational cost but also reduces memory overhead. Not storing reflective boundary angular fluxes can greatly reduce memory requirements but requires some method for approximating the starting angular fluxes. Starting flux conditions can be crudely approximated by taking the initial angular flux in every energy group as the local isotropic neutron source \( q_g/\Sigma_{\text{g}} \). This approximation would naively induce a large bias, but this is substantially mitigated through the use of a
“dead zone length”: the characteristic is tracked for a user-specified distance without accumulating updates to the scalar flux; that is, Eq. 4 is not applied.

Characteristics also have no natural termination point: in standard MoC a sweep over a given characteristic is typically terminated when it reaches a boundary. As opposed to traditional deterministic MoC where one must specify azimuthal and vertical track separations and numbers of azimuthal and polar angles, TRRM’s track density is set by the the number of rays to track per iteration, their active length, and their “dead length.” These parameters should be chosen such that nearly all FSRs in the geometry are traversed by at least one ray each transport sweep, typically with ray length chosen to be several times longer than the dead zone length so as to amortize its cost, yet not so long that too few angular samples are taken for a given minimal ray density. Volumes of FSRs in standard MoC are estimated during the initial ray trace based on the volume implied by the track separation and tracking distance traversed through each cell. In TRRM, because of on-the-fly ray tracing changing the perceived volume of a given cell each iteration, and combined with the need to avoid usage of a per-iteration ratio estimator (as detailed in Tramm et al. (2020)), the volume estimate is typically accumulated over all transport sweeps in the simulation since, unlike the scalar flux distribution, the true underlying volume distribution does not progress or vary between iterations (Tramm et al. 2020).

Significant complications arise from the stochastic nature of TRRM, although these are often offset by other computational gains resulting from the method. First, the determination of when the source has reached a stationary distribution (such that active iterations begin and statistics can be accumulated on unknowns) convergence can no longer be measured through a deterministic criterion since flux and eigenvalue estimates on a given iteration are subject to stochastic noise. However, methods used in Monte Carlo simulation to determine when the source has reached a stationary distribution, for example, Shannon entropy, are easily deployed in the same manner as in random ray analysis, as has been shown in ARRC (Tramm et al. 2018b). Second, just as in Monte Carlo eigenvalue calculations, both active and inactive iterations are required: inactive iterations are required to evolve the scalar flux distribution from its initial guess value to its converged (stationary) distribution, while active iterations are required to accumulate statistics on the scalar flux and to obtain a sufficiently low variance. These iterations can also be terminated through some stochastic criterion, for example, when some chosen standard deviation is sufficiently low (Tramm et al. 2018b). Although the randomly varying quadrature induces uncertainty in the estimates of the flux, it has been shown numerically that TRRM obtains more accurate results for a fixed number of integrations (ray traces and flux updates in each energy group) as compared with standard MoC (Tramm et al. 2017) in the context of eigenvalue simulations of light water reactors (and absent any numerical acceleration). This unintuitive effect is due in large part to the high number of power iterations required by the MoC process to converge the scattering and fission source distributions for large light water reactor problems featuring very high dominance ratios. In these problems, random ray offers significant numerical performance advantages by allowing for very low ray density iterations to be used through the inactive region of the simulation without inducing any bias. Conversely, in the absence of a secondary nonlinear numerical acceleration technique such as coarse mesh finite difference (CMFD) (Smith 1983 Smith and Rhodes 2002), traditional deterministic MoC must use the same high-fidelity ray density quadrature for all iterations. The effect is demonstrated for a full-core light water reactor simulation in Tramm et al. (2018b), where over 10,000 power iterations were found to be required to converge the source to a stationary distribution before entering the active phase of the simulation. In this simulation, an overall ray density for the random ray solver during the inactive phase was three or more orders of magnitude lower than what would be required for a similar fidelity solve using traditional deterministic MoC (as described in Gunow 2018). Gaston et al. (2021).

Although not yet reported in the literature, many other variations on TRRM might also be applied in analogy with standard MoC, for example, the use of convergence acceleration such as CMFD, or linear and anisotropic sources (Ferrer and Rhodes 2017).

2.2. Multigroup Monte Carlo

Monte Carlo neutron transport simulates individual neutrons from their birth in fission to their death in absorption or leakage. No angular or spatial discretization is typically required to facilitate transport—space needs only to be discretized to separate materials of different properties, although when incorporating temperature feedback and depletion physics into a model, the spatial discretizations for Monte Carlo can often closely resemble the finer meshes typically used by deterministic methods and random ray. When using continuous-energy cross section data, none of phase space is discretized, and all the neutron physics can be treated with
a minimal number of approximations, resulting in an accuracy limited only by the quality of the underlying nuclear data and physics models. However, it is nuclear data that makes Monte Carlo infamously slow: Monte Carlo’s runtime is dominated by memory look-ups, due to the size of cross-section data and the necessity to search through and interpolate cross section tables repeatedly during transport (Tramm and Siegel [2015]).

Alternatively, one can represent nuclear data in a more compact form to alleviate the expense of memory access. The most common approach is the multigroup representation, identical to that used in deterministic transport methods, which can substantially accelerate Monte Carlo in certain cases. Several costs are incurred in using a multigroup representation, however. The accuracy of MGMC is dependent on the accuracy of the cross sections, which must be produced separately (as is also the case for deterministic transport methods). Accuracy is also lost due to the representation of scattering anisotropy and kinematics—most multigroup codes represent these using the $P_n$ approximation, and this is frequently truncated to first order at most. In fact, MGMC struggles to use higher-order scattering (or transport corrected $P_0$ in some cases) without significant modification due to the occurrence of negative cross sections (Martin et al., 2011). There is also some minor limitation to the geometric flexibility in that phenomena such as spatial self-shielding must be represented by discretizing the geometry to some extent and using different cross-section data in different regions, even if the material is nominally the same between these regions.

2.3. Comparing MGMC and TRRM

Both methods discussed have strong similarities; indeed it has even been said that TRRM “is MGMC by another name.” As will be seen, this is not the case. Most obviously, both methods are stochastic in nature, and their “transport sweep” consists of following many individual neutrons/characteristics. Thus, they are both straightforwardly capable of task-based parallelism and parallelism through decomposition. This is as opposed to $S_n$ solvers, for example, where a sweep involves several wavefronts moving across the geometry in different directions; because of the cell-based sweep, these methods struggle to expose as much concurrency as particle- or track-based methods (Adams et al., 2020). Additionally, as opposed to other sweep-based transport methods, curvilinear geometries—limited only by the generality of the ray tracing subroutines—can be handled without any approximation, ambiguity, or special treatment in the sweep (Plimpton et al., 2005). Both methods are also capable of modeling void or highly absorbing regions of geometry, which some deterministic methods struggle with.

Both methods also treat both energy and angle identically. While the agreement in energy representation is shared across all multigroup methods, the angular representation is more remarkable. In common deterministic methods, angular phase space is handled either by using a discrete set of angles (as in the discrete ordinates approximation) or by using a finite angular basis (as in spherical harmonics methods), both of which will introduce some truncation error. MGMC and TRRM treat the angular portion of phase space continuously: there is no bias in angle resulting from the methods, although there is a risk of undersampling the phase space by simulating too few particles/rays.

Many differences between the methods remain, however. Some of these are as follows.

1. Ray positions in random ray are sampled uniformly across the geometry and change direction only on collision with a boundary. Neutrons in a light water reactor MGMC simulation are always sampled in the fuel and, due to their physics, tend to stay in the vicinity of the fuel while changing direction many times. Thus, the physical volume of a given geometry is more uniformly sampled by TRRM; variance reduction methods such as uniform fission sampling (Kelly et al., 2012) must be used by MC to attempt to flatten statistical uncertainties.

2. While both methods can be affected by statistical correlation, there are fewer avenues for correlation to develop in random ray than in MGMC. In other words, both methods will ideally reach a $1/\sqrt{N}$ statistical convergence rate (where $N$ is the number of particles/rays sampled), but this is harder to attain for MGMC. The reason is that the sampled locations of rays do not depend at all on the previous iteration. Conversely, in MGMC, particles born in one iteration depend on the fission locations in the previous iteration. Both methods still have a dependence on the source estimate in the previous iteration, but the correlation effect tends to be more pronounced in MGMC (Herman, 2014; Dumonteil et al., 2014; Nowak et al., 2016; Miao, 2018) than in TRRM (Tramm et al., 2017).

3. The flat source approximation necessitates a finer discretization of TRRM than MGMC. In MGMC, scattering and fission sources vary continuously in space. In order to obtain accurate results with flat source MoC, fuel pins essentially must be divided
radially and azimuthally, and the moderator must also be subdivided, whereas this is not necessary in MGMC. This will impose an additional computational burden on TRRM over MGMC, although it can be mitigated by using higher-order sources that allow coarser spatial discretization (Ferrer and Rhodes, 2017). That being said, when performing depletion calculations or when incorporating temperature feedback, regardless of the method, it is often desirable to have such radial and azimuthal divisions in fuel pins in order to resolve plutonium build-up or gadolinium depletion and to account for different temperatures within the fuel.

4. In MGMC one iterates on the fission source, whereas in TRRM (like all deterministic methods) one iterates on the flux or, equivalently, the total neutron source (fission and scattering). In MGMC, a neutron can undergo any number of scattering events during a given batch (implying that the scattering source is “converged” with the fission source). In TRRM, each batch corresponds to neutrons undergoing only a single collision (Adams and Larsen, 2002), implying expending additional computational effort on converging the scattering source, as well as the fission source. This has the effect of slowing the iteration-wise convergence rate of TRRM relative to MGMC since the neutron spectrum is not immediately converged upon. In the context of light water reactor simulation, this difference results in the random ray method typically requiring far more inactive batches than MGMC.

5. In TRRM, rays hold angular fluxes in all energy groups simultaneously, while in MGMC particles can have only one energy group. This allows for straightforward vectorization in the TRRM algorithm, which is much more difficult to achieve with Monte Carlo. Furthermore, by scoring fluxes in all energy groups simultaneously, the variance in values derived from these fluxes is reduced; MGMC, on the other hand, may have energy groups that are rarely visited, depending on the cross sections, inducing higher variance.

6. Neutron lifetimes may be extremely disparate in MGMC, as opposed to TRRM where all characteristics have an identical “lifetime” in terms of the distance they traverse. That said, the amount of work that each characteristic must perform can vary significantly depending on how finely discretized the geometry is in its vicinity. For example, it is conceivable that, in a coarsely discretized region, one ray would need to perform only a single ray trace to traverse its entire termination length, whereas another ray in a finely discretized region may need to perform many traces.

7. MGMC does not necessarily require explicit ray tracing: delta tracking can be used that may be substantially more efficient in certain geometries. Delta tracking only requires identifying which region of a geometry a particle is in and does not use distance-to-surface calculations. On the other hand, delta tracking may be inefficient in cases where there are localized strong absorbers, but this situation can be mitigated by using a combination of delta and surface tracking (Leppanen, 2010). Another drawback of delta tracking is that it tends to require precomputing the majorant cross section to be efficient, although this is simple in MGMC, especially with few unique materials present in the geometry.

8. While both methods are straightforwardly capable of MPI parallelism, the optimal approaches differ between them. In Monte Carlo, MPI parallelism is often performed by replicating the domain of interest across MPI nodes, only sharing eigenvalue and fission bank information to minimise communication—this is even the case for whole-core problems (Romano and Forget, 2012). In problems featuring many unique compositions or finely discretized tally regions (such as whole-core burn-up), data and domain decomposition become the preferred approaches because of the large memory footprint no longer fitting on a single node (Dun et al., 2015; Garcia et al., 2021). In TRRM, on the other hand, domain decomposition is essentially required for medium to large simulations because of the memory footprint resulting from the necessary spatial discretization and resulting large scalar flux vectors (Tramm, 2018; Tramm et al., 2018b). Domain replication is not scalable in random ray because of the need to perform an MPI “all reduce” of the full scalar flux vector between all MPI ranks after each power iteration. Given that the size of the scalar flux vector can be very large for full-core simulation problems, this reduction operation tends to become prohibitively expensive for all but the smallest problems.

9. TRRM should be able to handle most difficulties associated with anisotropic scattering, whereas, as mentioned previously, MGMC cannot straightforwardly handle the resulting negative cross sections without using a different angular representation (Martin et al., 2011). When applying a transport correction that results in nega-
tive diagonal elements of the $P_0$ scattering matrix, stabilization techniques can be used with MoC and TRRM (Gunow et al., 2019). Additionally, methods for including arbitrary-order scattering anisotropy in MoC have previously been demonstrated (Ferrer and Rhodes, 2017).

2.4. Monte Carlo modifications to enable TRRM

Both OpenMC and SCONE are open-source, performant MGMC codes. They have well-optimized constructive solid geometry and ray tracing subroutines. From this point, several small changes and additions are necessary to create a working TRRM solver.

First, all material-containing cells in the geometry must have a unique ID, corresponding to the elements of the scalar flux and neutron source vectors. This is not necessarily a given in Monte Carlo geometry implementations—there is no algorithmic necessity, for example, to uniquely identify one pin cell in a lattice from another with the same composition. In SCONE, this feature was already present because of how geometry is constructed as a directed acyclic graph and there is an option to uniquely ID every end point of the resulting graph. In OpenMC, routines were already in place to provide unique IDs for each cell in service of OpenMC’s “distributed cell” and “cell instance” tally features.

While Monte Carlo particle objects could essentially be left unchanged, how they move through a geometry required modification in the case of vacuum boundary conditions. Normally, this would terminate a particle, but for a characteristic, a reflection must be performed (for reasons discussed in Tramm et al. (2017)) in addition to applying the vacuum condition (angular flux spectrum of zero) to the ray. In SCONE, this was handled by creating a new particle movement subroutine. This subroutine moves particles as normal but enforces a reflection at all boundaries and also has an additional argument that returns whether a vacuum was struck; this is used to zero the angular fluxes during a transport sweep. In OpenMC, a small amount of additional logic was added to the reflection routine that applies appropriate reflection and flux zeroing operations to the ray when running in random ray mode. Particle objects could also be modified to hold angular flux information. This was not done in SCONE to minimize changes to the particle object for the moment. Instead all angular flux information remains inside the transport sweep subroutine. Conversely, OpenMC does provide particles with angular flux information. This is the natural approach if the “immortal rays” variant of TRRM were to be pursued in the future, and angular fluxes would need to be stored between iterations (Tramm and Siegel, 2021).

To maximize performance, most MoC solvers do not use the intrinsic exponential function. Although exponential lookup tables are popular (Yamamoto et al., 2004), recently a rational approximation to evaluate the $1 - e^{-x}$ term in Eq. [3] was developed by Josey and detailed in Giudicelli et al. (2019) as a simple, fast, and numerically robust method to vectorize the function across energy groups during the transport sweep. This was implemented inline in OpenMC and as an elemental function in SCONE to facilitate vectorization during compilation. Additionally, other than ray tracing routines and integral variables like k-effective, all other data storage and arithmetic operations use single precision.

The basic TRRM algorithms are straightforward to implement, such as updating the eigenvalue, calculating sources, and performing the transport sweep. Extra care was taken in writing both the transport sweep and source calculation routines to ensure that they would be vectorized by the compiler.

In MoC calculations on problems featuring pin cells, it is necessary to both radially and azimuthally discretize their geometry to obtain accurate results (at least when using the flat source approximation). OpenMC does not currently feature native pin universes, but enhancements to OpenMC’s Python interface were made to facilitate automatic creation of discretized pin cells using cylindrical and plane surfaces. SCONE already supported radially divided pin cells, but not azimuthal: this new universe type was added to the list of available universes since it may be useful in Monte Carlo burn-up calculations also.

Neither code has yet implemented all of the robustness/ease-of-use features that are included with ARRC. These include options to detect inactive iteration convergence through Shannon entropy, options to detect active iteration convergence through monitoring the variance of the global scalar flux vector, or automated selection of minimal track density to ensure sufficient coverage of the geometry (Tramm et al., 2018). Additionally, neither solver supports domain decomposition through MPI as of yet, although an older experimental branch of OpenMC with domain decomposition may be leveraged in the future if needed to support random ray transport. As mentioned in Section 2.3, this will be vital if the codes are to be used to simulate large problems, such as high-fidelity simulations of full nuclear reactors.

The final number of additional lines of code required to implement TRRM in OpenMC and SCONE is 800 and 1400, respectively. Substantially more lines of
code are required by SCONE because of the approach SCONE developers are encouraged to take in writing a new type of calculation scheme, featuring a reasonably large amount of boilerplate code. Additional code overhead is due to the creation of a new universe type as described previously, adding an extra 460 lines; this code is also likely to be useful for MC calculations. In OpenMC, an approach was taken to reuse and repurpose functions as needed, which results in a closer to minimal code delta, but with the downside of increasing the complexity of some routines (e.g., the aforementioned reflection function).

Notable differences exist between the codes that will affect the performance of the TRRM implementations. The first prominent difference is that OpenMC supports cell neighbor lists while SCONE does not at present. This can greatly accelerate the cell search necessary when a particle/ray exits one cell and must locate which cell it has entered.

Second, SCONE supports several structured universe classes other than lattices (such as a pin universe, from which the azimuthally divided pin universe was created), while OpenMC does not at present. This, too, can be used to accelerate cell search operations given that in an arbitrarily radially divided pin universe a particle’s/ray’s cell can be located rapidly by obtaining its radial position in the universe and comparing it with the radial boundaries of the cells.

In general unstructured geometries, OpenMC’s ray tracing subroutines are likely to outperform SCONE given the generality of cell neighbor lists. On the lattice-oriented geometries representative of most nuclear reactors, however, these two features essentially play identical roles. Thus, while some difference in the efficiency of the ray tracing subroutines present in OpenMC and SCONE is to be expected, it should not be substantial. Any difference should also diminish as larger numbers of energy groups are used and ray tracing costs are amortized.

2.5. Ray tracing with distance caching

During the course of implementing TRRM in SCONE, the use of a previously implemented optimization for Monte Carlo was explored, namely, the method referred to as distance caching (Kowalski and Cosgrove 2021). Distance caching is a means of accelerating ray tracing in constructive solid geometries consisting of multiple nested universes, a common means of describing reactor geometries in Monte Carlo calculations.

In Monte Carlo, when calculating the distance that a particle can travel before crossing a surface into another cell, this distance must be evaluated at all universe levels of the geometry. This evaluation is done in order to guard against cases such as when only a portion of a fuel pin is inside its lattice position in an assembly or where a cell is truncated by a surface at a higher level in the universe structure. For problems such as C5G7 (Lewis et al., 2001), the natural and efficient way to write this input in SCONE would have five nested universe levels. Therefore, every time a particle moves, one must perform five distance-to-surface calculations.

Distance caching performs this distance calculation at all universe levels the first time a particle moves. If the particle then crosses a surface, the distance to the boundary at that universe level and all lower levels must subsequently be recalculated. However, the distances to the boundary at higher universe levels can simply be decremented by the distance the particle just traveled. This remains true until the particle either crosses a surface at a higher universe level or until it changes direction.

The paper that introduced distance caching, Kowalski and Cosgrove (2021), applied it to continuous energy Monte Carlo. In that work, distance caching provided an acceleration of up to 20% in finely divided reactor geometries that might be applicable, say, when one is concerned with the radial burn-up of pin cells or in fast reactors where neutrons cross many surfaces before colliding. In more standard geometries, however, the acceleration was not as substantial, since particles scatter and change direction often, meaning that all distances must be recomputed frequently. However, in any MoC implementation using on-the-fly ray tracing written in a nested universe geometry paradigm, substantial acceleration might be expected. The reason is that geometries will typically be more finely divided in MoC than in Monte Carlo and the rays only change direction on colliding with a boundary, meaning the distances at higher universes levels are valid for much longer.

Given all of the (relatively simple) mechanics for performing ray tracing with distance caching that already existed in SCONE, it was natural to apply this feature to TRRM also. Thus, another movement subroutine was written to permit this. The speed-up that this provides will be discussed in the results section. On-the-fly ray tracing is an increasingly common feature of 3D MoC solvers, and so distance caching may be of use elsewhere (Gunow 2018; Gaston et al., 2021).

One point to note is that Kowalski and Cosgrove (2021) previously hypothesized loss of precision due to floating-point tolerance but did not observe it. In the present work, we did rarely observe ray positions being lost due to distance caching, although this problem
was remedied using (as suggested in the original paper) a periodic reset of the cache: after 100 ray intersections, the cache is reset, and all distances to boundaries are recalculated. A more robust approach also suggested by Kowalski and Cosgrove [2021] but not yet implemented was the use of Kahan summation.

3. Numerical Investigations

With TRRM now implemented in both OpenMC and SCONE, the remaining goals of the paper are to to critically compare their runtime performance and accuracy against the native MGMC implementations. This was done for two types of problem: one with few energy groups and one with many. All OpenMC and SCONE simulations were performed on a dual-socket Intel Xeon Gold 6336Y node (48 cores, 96 threads). All parallel results produced were averaged using 100 independent runs. The runtime performance of both codes in serial was also investigated, although not to the same extent because of resource constraints: the performance results for serial simulations were averaged over 5 independent runs.

This section is divided into three parts covering the three investigations performed. These investigations concerned the performance of the methods when different numbers of energy groups are used and the differences in uncertainty distributions between the methods.

3.1. C5G7

The standard problem on which to evaluate reactor-oriented multi-group transport methods is the C5G7 benchmark (Lewis et al., 2001). This is a mini-reactor geometry, featuring a quarter core of four assemblies and with given cross sections in seven energy groups. The geometry is shown in Fig. 1a. The problem features a large reflector which tends to require relatively fine discretization with deterministic methods in order to accurately resolve the steep thermal flux gradient. The 2D version of the benchmark is used for simplicity and to allow for large numbers of independent simulations to be run to understand the true accuracy and variance of all reported results. Despite the simulation problem being a 2D geometry, both the MGMC and TRRM solvers in OpenMC and SCONE are natively 3D, so do not take any special advantage of the axial dimension (i.e., no discretized polar quadratures are used, and full 3D ray tracing is still performed). It is also important to note that the C5G7 benchmark is not necessarily the most performant problem type for random ray, at least as compared to traditional deterministic MoC. This is because with only 7 energy groups the cost of on-the-fly ray tracing becomes more dominant and is not well amortised due to fewer energy integrations to perform per ray trace.

As mentioned in Section 2.3, there is a significant difference in the necessary geometry discretization between the two methods. The unique cell discretization for both MGMC and TRRM are shown in Figs. 1b and 1c. To be precise, TRRM requires subdivision of the fuel pins and moderator to obtain accurate results (due to the flat source approximation) while MGMC does not due to a spatially-continuous representation of neutron sources. To highlight this fine discretization of fuel pins and the adjacent moderator, a zoomed in picture in the vicinity of the corner fuel pin is shown in Fig. 1d. The discretization used for TRRM here is relatively coarse nevertheless, with about 142,000 cells used in both OpenMC and SCONE. The MGMC simulations featured 2,317 cells. It should be noted that, were these solvers to be used for burn-up or coupled thermal-hydraulic calculations, MGMC may require additional discretization which would affect its performance, whereas the discretization used by TRRM would be adequate for most pin cells. Conversely, if linear sources were implemented for TRRM then this discretization could be substantially coarsened and improvements in speed and accuracy might result.

In order to compare TRRM and MGMC, we investigate the computational time required to achieve various levels of accuracy on the C5G7 problem. While eigenvalues are reported and compared, accuracy is better measured with respect to the reference spatial power distribution, namely using the maximum absolute relative pin power error and average absolute relative pin power error metrics. Based on previous investigations on the C5G7 benchmark (Tramm et al., 2017; Tramm, 2018), TRRM used 1,750 rays per iteration, a dead length of 20 cm, a termination length of 220 cm (giving an active length of 200 cm), and 1,000 inactive iterations. The number of active iterations was varied from 325 to 2,600. For MGMC, 10,000 particles per batch were used with 50 inactive batches – the number of active batches was varied from 1,300 to 10,400. The choice of settings for MGMC is based on C5G7 being a low dominance ratio problem; this implies that source convergence will be rapid and inter-generational correlation effects will be minor (Herman, 2014; Dumonteil et al., 2014; Nowak et al., 2016; Miao, 2018). As such, to minimise the computational cost of inactive cycles (which do not contribute towards tallying results), relatively few inactive generations and particles per generation can be used without significant risk of bias, corre-
(a) 2D C5G7 geometry. Gold is water, pale yellow pins are UO$_2$, pale blue pins are guide tubes, purple pins are fission chambers, and other pins are types of MOX.

(b) Unique cells in the multi-group Monte Carlo C5G7 geometry.

(c) Unique cells in the random ray C5G7 geometry.

(d) Unique cells in the random ray C5G7 geometry in the vicinity of the corner fuel pin.

Figure 1: Geometry of the 2D C5G7 problem in terms of materials and unique cells for both multi-group Monte Carlo and random ray.

It is notable that MGMC required only 50 inactive batches, while TRRM required 1,000. This is due to a key numerical difference between the methods, namely, that random ray requires iterations to converge scattering sources unlike in Monte Carlo. While a particle in one source iteration of Monte Carlo can undergo any arbitrarily high number of scattering events (provided
the physics allows for it), the iteration scheme used for MoC in this work only allows the scattering source to advance by a single hop each iteration. This effect may be best demonstrated when considering a fixed source problem without fission, where the first iteration \( n = 1 \) of a random ray or MoC simulation will essentially only be capable of computing first collision sources, with each iteration \( n \) thereafter using a scattering source that has been built considering the effects of up to \( n - 1 \) neutron scatters. MoC can alternatively be formulated to use a similar number of fission source iterations as Monte Carlo, though at the cost of adding an additional loop inside of each fission source iteration to fully converge the scattering source before moving onto the next fission source iteration. However, due to the relatively low cost of re-computing the fission source, it is typically advantageous to combine the scattering and fission source iterations into a single source iteration scheme as we do in this work.

Given the choice of settings, the MGMC runtime is dominated by the useful work during the active batches. It can also be seen that source convergence acceleration would have little effect on the overall runtime, given there are far fewer inactive batches than active. Conversely, due to the slower iteration-wise convergence of deterministic methods (as discussed in Section 2.3), the computational cost of TRRM is initially dominated by the inactive convergence phase, rather than the active phase of results accumulation. Hence, there is substantial incentive to apply convergence acceleration methods to TRRM in future work. That said, for the highest accuracy settings used here, the inactive portion of the TRRM simulation constitutes just under a third of the runtime.

The results for this problem comparing the absolute maximum pin power error and parallel runtimes for both codes and calculations types are shown in Fig. 2. The error bars represent one standard deviation of the error of the mean from the ensemble. A greater range of results and performance characteristics are shown in Table 1

Note that, as the reference solution for C5G7 is generated by MGMC, there should be no discretization error affecting MGMC results – only stochastic error will be present. In fact, the results approximately follow the expected \( 1/\sqrt{N} \) curve, where \( N \) is the number of active particles simulated, which is roughly proportional to the runtime. On the other hand, TRRM does suffer from discretization error due to the flat source approximation; while initially the error may be dominated by the stochastic component, it eventually starts to plateau at the level of discretization error dictated by the particular FSR mesh being used.

As well as serial and parallel runtimes, common performance metrics for both MGMC and TRRM are shown in Table 2. For MGMC, the serial number of particles per second are given – these are differentiated as active and inactive due to the additional burden of tallying during active batches. These values are calculated by timing the history loop of the MGMC simulations and dividing by the total number of particles simulated.

For TRRM, the time per integration (TPI) is shown. This is defined as the time taken to calculate a distance via ray tracing, evaluate the exponential, and update the angular and scalar fluxes for a single energy group. This can be calculated by recording the time taken during a transport sweep and the number of ray-surface intersections that occur during the sweep. Then, the time per integration may be evaluated after a transport sweep as:

\[
\text{TPI} = \frac{\text{sweep time}}{\text{intersections} \times \text{groups}}.
\]

Note that the TPI metric excludes any time spent outside of the transport sweep (e.g., updating the neutron source in between iterations, normalizing the flux after each iteration, etc). However, all overall runtime data for TRRM in this paper includes these tasks. For comparison, ARRC reports a serial TPI of about 30 ns when using 7 energy groups for a variety of simulation problems run on a Xeon 8176 CPU node (Tramm (2018), Figure 3-3), comparable to both solvers implemented here. The serial performance of each method is also illustrated in Fig. 3. Comparing this against Fig. 2 demonstrates that all methods and implementations parallelised with similar efficiencies.

There are some differences between the MGMC implementations. For example, SCONE uses delta tracking while OpenMC uses surface tracking. Although SCONE can use surface tracking, it did not perform as well on this particular problem as delta tracking and so delta tracking was used to ensure the best MGMC performance. This precluded the use of a track length estimator (which worsened SCONE’s surface tracking figure of merit notably) and so both codes used a collision estimator. The use of a collision or track length estimator in OpenMC did not significantly affect its performance in terms of error versus runtime. The different tracking algorithms as well as slightly different geometry subroutines is likely to explain most of the performance difference between the two MGMC implementations.

For this particular problem, distance caching shows its usefulness in SCONE’s TRRM implementation. Without using distance caching, the serial TPI is 24.87
Table 1: Performance results for OpenMC and SCONE running MGMC and TRRM on C5G7. Numerical results and parallel runtime results are averaged over 100 ensemble simulations. Serial runtime results are averaged over 5 simulations. APE is Average Pin Error and MPE is Maximum Pin Error (both quantities are absolute values). The uncertainty of k-eff is given in pcm.

<table>
<thead>
<tr>
<th></th>
<th>Active batches</th>
<th>Serial runtime (s)</th>
<th>Parallel runtime (s)</th>
<th>k-eff</th>
<th>APE (%)</th>
<th>MPE (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OpenMC - MGMC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1300</td>
<td>260.09</td>
<td>9.70</td>
<td>1.186453±27.9</td>
<td>0.842±0.031</td>
<td>4.679±0.745</td>
<td></td>
</tr>
<tr>
<td>2600</td>
<td>513.71</td>
<td>20.31</td>
<td>1.186464±18.8</td>
<td>0.605±0.022</td>
<td>3.454±0.573</td>
<td></td>
</tr>
<tr>
<td>5200</td>
<td>1011.60</td>
<td>39.76</td>
<td>1.186481±14.1</td>
<td>0.435±0.016</td>
<td>2.430±0.363</td>
<td></td>
</tr>
<tr>
<td>10400</td>
<td>2021.50</td>
<td>82.16</td>
<td>1.186459±10.6</td>
<td>0.328±0.056</td>
<td>1.816±0.380</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Active batches</th>
<th>Serial runtime (s)</th>
<th>Parallel runtime (s)</th>
<th>k-eff</th>
<th>APE (%)</th>
<th>MPE (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SCONE - MGMC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1300</td>
<td>156.97</td>
<td>4.68</td>
<td>1.186440±27.5</td>
<td>0.837±0.030</td>
<td>4.774±0.671</td>
<td></td>
</tr>
<tr>
<td>2600</td>
<td>312.06</td>
<td>9.33</td>
<td>1.186488±18.1</td>
<td>0.592±0.020</td>
<td>3.337±0.454</td>
<td></td>
</tr>
<tr>
<td>5200</td>
<td>617.70</td>
<td>17.73</td>
<td>1.186472±15.4</td>
<td>0.424±0.016</td>
<td>2.398±0.309</td>
<td></td>
</tr>
<tr>
<td>10400</td>
<td>1230.68</td>
<td>35.99</td>
<td>1.186489±8.8</td>
<td>0.299±0.011</td>
<td>1.685±0.237</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Active batches</th>
<th>Serial runtime (s)</th>
<th>Parallel runtime (s)</th>
<th>k-eff</th>
<th>APE (%)</th>
<th>MPE (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OpenMC - TRRM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>325</td>
<td>1100.50</td>
<td>38.57</td>
<td>1.186736±42.3</td>
<td>0.531±0.095</td>
<td>2.415±0.470</td>
<td></td>
</tr>
<tr>
<td>650</td>
<td>1396.40</td>
<td>51.53</td>
<td>1.186745±32.7</td>
<td>0.444±0.073</td>
<td>2.046±0.345</td>
<td></td>
</tr>
<tr>
<td>1300</td>
<td>1918.50</td>
<td>71.23</td>
<td>1.186762±22.5</td>
<td>0.367±0.046</td>
<td>1.739±0.289</td>
<td></td>
</tr>
<tr>
<td>2600</td>
<td>3002.00</td>
<td>116.53</td>
<td>1.186734±16.6</td>
<td>0.318±0.039</td>
<td>1.561±0.247</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Active batches</th>
<th>Serial runtime (s)</th>
<th>Parallel runtime (s)</th>
<th>k-eff</th>
<th>APE (%)</th>
<th>MPE (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SCONE - TRRM (cache)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>325</td>
<td>415.85</td>
<td>16.06</td>
<td>1.186799±46.6</td>
<td>0.540±0.095</td>
<td>2.474±0.445</td>
<td></td>
</tr>
<tr>
<td>650</td>
<td>514.51</td>
<td>19.54</td>
<td>1.186750±31.8</td>
<td>0.449±0.075</td>
<td>2.018±0.347</td>
<td></td>
</tr>
<tr>
<td>1300</td>
<td>728.54</td>
<td>27.52</td>
<td>1.186787±21.3</td>
<td>0.376±0.050</td>
<td>1.726±0.248</td>
<td></td>
</tr>
<tr>
<td>2600</td>
<td>1171.62</td>
<td>42.65</td>
<td>1.186769±16.3</td>
<td>0.321±0.041</td>
<td>1.446±0.171</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Active batches</th>
<th>Serial runtime (s)</th>
<th>Parallel runtime (s)</th>
<th>k-eff</th>
<th>APE (%)</th>
<th>MPE (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SCONE - TRRM (no cache)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>325</td>
<td>552.21</td>
<td>18.65</td>
<td>1.186670±46.4</td>
<td>0.539±0.095</td>
<td>2.468±0.444</td>
<td></td>
</tr>
<tr>
<td>650</td>
<td>682.07</td>
<td>22.97</td>
<td>1.186628±31.6</td>
<td>0.448±0.075</td>
<td>2.022±0.350</td>
<td></td>
</tr>
<tr>
<td>1300</td>
<td>951.30</td>
<td>30.84</td>
<td>1.186667±21.5</td>
<td>0.373±0.049</td>
<td>1.717±0.247</td>
<td></td>
</tr>
<tr>
<td>2600</td>
<td>1471.00</td>
<td>49.54</td>
<td>1.186650±16.3</td>
<td>0.317±0.041</td>
<td>1.434±0.172</td>
<td></td>
</tr>
</tbody>
</table>
Table 2: Serial performance metrics for OpenMC and SCONE on C5G7.

<table>
<thead>
<tr>
<th>Serial performance metric</th>
<th>OpenMC</th>
<th>SCONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MGMC active particles/s</td>
<td>51,671</td>
<td>84,820</td>
</tr>
<tr>
<td>MGMC inactive particles/s</td>
<td>58,025</td>
<td>116,078</td>
</tr>
<tr>
<td>TPRM TPI (ns)</td>
<td>50.71</td>
<td>19.54</td>
</tr>
</tbody>
</table>

ns; using distance caching this decreases the serial TPI by about 20% to 19.54 ns, which translates into a 10-15% speed-up in parallel (most likely due to parallel inefficiencies outside of the transport sweep). The maximum pin power error against runtime for SCONE with and without distance caching is shown in Fig. 4. This is because there are relatively few energy groups in C5G7, and therefore the cost of ray tracing is relatively high compared to costs associated with exponential evaluation and flux updating. The benefits of distance caching will diminish with more energy groups and improve with fewer.

OpenMC and SCONE differ significantly in the performance of their TPRM solvers. The most likely explanation for this is differences between the geometry/cell-searching subroutines, as the implementations of the other components of the method are similar. As discussed in Section 2.4, SCONE makes substantial use of structured cylindrical universes to minimise the number of operations required when identifying in which cell a point resides. This difference is likely to be more strongly felt during TPRM than during MGMC due to the differences in discretization. In MGMC, a given pin cell universe only has two cells to search, each with one neighbour. For the TPRM discretization, there are forty cells to search, most with four neighbours, increasing the expense of the search.

Across methods, however, TPRM is comparable to or faster than MGMC for this problem.

3.2. C5G69

A ‘higher fidelity’ version of C5G7 was also considered, instead with 69 energy groups; this will be referred to as C5G69. Here multi-group cross sections in the WIMS69 group structure were generated using continuous energy SCONE on a constructed continuous energy version of C5G7 featuring realistic nuclide densities. To avoid complications associated with negative cross sections, no transport correction was applied and only the $P_0$ scattering matrix was used. These cross sections are provided in the Supplementary Data. The geometry was otherwise identical to C5G7. MGMC SCONE was used to generate a reference solution using 20,000,000 particles per batch, with 50 inactive batches and 400 active batches. The reference fission rate distribution is also provided in the Supplementary Data.

This problem is considered interesting because with more energy groups the cost of on-the-fly ray tracing in TPRM will be substantially amortised. Additionally, due to the vectorisation of evaluating the exponential and updating the scalar flux, the expense of the TPRM solution should only grow sub-linearly with the number of energy groups.

Once again comparing the maximum absolute pin power error for each code and method as a function of parallel runtime, the results are plotted in Fig. 5. Additional results are shown in Tables 3 and 4.

For this problem, as one would expect, there was essentially no difference whether or not distance caching was used. Hence only one set of SCONE TPRM results (using caching) are shown.

Fig. 5 has two striking features. First, as expected from TPRM, the computational cost does not grow in proportion to the number of groups – while there are nearly 10× more equations to solve than in C5G7, efficient vectorisation of the operations occurring during the transport sweep results in only ~ 3× the parallel runtime for SCONE and ~ 1.5× that for OpenMC. However, while the TPRM solvers experience only a modest increase in computational expense, the MGMC solvers suffer no additional expense. This can be seen by comparing Fig. 5 with Fig. 2 where the MGMC curves are essentially unchanged. This is because the length of a neutron history between each problem is similar – whether in 7 groups or 69, neutrons take (on average)
Table 3: Performance results for OpenMC and SCONE running MGMC and TRRM on CSG69. Numerical results and parallel runtime results are averaged over 100 ensemble simulations. Serial runtime results are averaged over 5 simulations. APE is Average Pin Error and MPE is Maximum Pin Error (both quantities are absolute values). The uncertainty of k-eff is given in pcm.

<table>
<thead>
<tr>
<th></th>
<th>Active batches</th>
<th>Serial runtime (s)</th>
<th>Parallel runtime (s)</th>
<th>k-eff</th>
<th>APE (%)</th>
<th>MPE (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenMC - MGMC</td>
<td>1300</td>
<td>282.84</td>
<td>10.90</td>
<td>1.221555±23.3</td>
<td>0.896±0.038</td>
<td>5.074±0.674</td>
</tr>
<tr>
<td></td>
<td>2600</td>
<td>559.10</td>
<td>21.91</td>
<td>1.221571±19.2</td>
<td>0.634±0.031</td>
<td>3.656±0.609</td>
</tr>
<tr>
<td></td>
<td>5200</td>
<td>1107.50</td>
<td>40.53</td>
<td>1.221586±13.7</td>
<td>0.476±0.098</td>
<td>2.680±0.572</td>
</tr>
<tr>
<td></td>
<td>10400</td>
<td>2255.60</td>
<td>82.71</td>
<td>1.221574±9.8</td>
<td>0.318±0.014</td>
<td>1.791±0.260</td>
</tr>
<tr>
<td>SCONE - MGMC</td>
<td>1300</td>
<td>203.07</td>
<td>5.30</td>
<td>1.221574±26.1</td>
<td>0.893±0.038</td>
<td>5.019±0.729</td>
</tr>
<tr>
<td></td>
<td>2600</td>
<td>398.34</td>
<td>9.98</td>
<td>1.221583±17.9</td>
<td>0.634±0.028</td>
<td>3.632±0.546</td>
</tr>
<tr>
<td></td>
<td>5200</td>
<td>784.43</td>
<td>21.03</td>
<td>1.221555±13.3</td>
<td>0.452±0.022</td>
<td>2.554±0.362</td>
</tr>
<tr>
<td></td>
<td>10400</td>
<td>1587.51</td>
<td>41.77</td>
<td>1.221562±9.4</td>
<td>0.322±0.017</td>
<td>1.832±0.333</td>
</tr>
<tr>
<td>OpenMC - TRRM</td>
<td>325</td>
<td>1872.90</td>
<td>58.29</td>
<td>1.221534±54.9</td>
<td>0.633±0.115</td>
<td>2.882±0.499</td>
</tr>
<tr>
<td></td>
<td>650</td>
<td>2389.90</td>
<td>69.38</td>
<td>1.221582±40.6</td>
<td>0.535±0.104</td>
<td>2.377±0.462</td>
</tr>
<tr>
<td></td>
<td>1300</td>
<td>3329.60</td>
<td>106.07</td>
<td>1.221597±24.3</td>
<td>0.434±0.056</td>
<td>1.957±0.298</td>
</tr>
<tr>
<td></td>
<td>2600</td>
<td>5282.80</td>
<td>160.96</td>
<td>1.221606±16.3</td>
<td>0.389±0.061</td>
<td>1.752±0.268</td>
</tr>
<tr>
<td>SCONE - TRRM</td>
<td>325</td>
<td>1301.33</td>
<td>47.30</td>
<td>1.221638±49.4</td>
<td>0.650±0.127</td>
<td>2.996±0.617</td>
</tr>
<tr>
<td></td>
<td>650</td>
<td>1627.09</td>
<td>59.04</td>
<td>1.221606±33.6</td>
<td>0.537±0.105</td>
<td>2.448±0.464</td>
</tr>
<tr>
<td></td>
<td>1300</td>
<td>2257.52</td>
<td>82.62</td>
<td>1.221656±22.6</td>
<td>0.450±0.075</td>
<td>2.061±0.353</td>
</tr>
<tr>
<td></td>
<td>2600</td>
<td>3551.32</td>
<td>128.99</td>
<td>1.221643±17.0</td>
<td>0.384±0.063</td>
<td>1.755±0.250</td>
</tr>
</tbody>
</table>
nearly the same number of collisions to thermalise and undergo absorption or leakage.

While OpenMC’s serial particles per second is consistent between C5G7 and C5G69, SCONE’s performance degrades slightly. This is due to the additional expenses incurred by having more energy groups, e.g., due to sampling the outgoing group during scattering. Admittedly, this might be avoided with some small optimisations, but these differences are drowned out in parallel due to inefficiencies from atomic operations required for tallying.

Unsurprisingly, the two TRRM solvers begin to converge in runtime and TPI as more groups are added. This is for the reason mentioned previously: differences in geometry routines become less important as more time is spent evaluating the MoC equations. The TPIs reported here are comparable with that of ARRC which achieves a serial TPI of just under 4 ns on a 2D pin cell problem with 69 energy groups on a Xeon 8176 node (Tramm (2018), Figure 3-3).

For this problem, both TRRM solvers are either comparable to or slower than the MGMC solvers. There is relatively little that can be done to further optimise the TRRM subroutines, but a significant speed-up might be obtained by implementing convergence acceleration in the TRRM solvers. For example, the TRRM runs using the coarsest statistics spend approximately three quarters of their runtime on convergence. In larger problems, where MGMC must spend longer converging and is more affected by correlation, it is possible that TRRM may be more appealing. This is because MGMC must use more inactive batches, and either estimates will have a higher true variance or more particles must be simulated each batch to reduce this variance, making inactive batches more expensive. While TRRM would also have to use more inactive batches, it can still use a coarse quadrature to converge relatively quickly, without bias and with little correlation in the results. Additionally, as already discussed, the addition of linear sources in TRRM might provide significant acceleration while greater pin discretization in MGMC (perhaps in service of burn-up or thermal-hydraulics calculations) may hamper its performance.

### 3.3. Stochastic uncertainty distributions

An advantage that TRRM enjoys over MGMC (in common with deterministic methods) is its ability to uniformly integrate the simulation domain in a manner that is not strongly dependent on the physical flux of the model. Although, like MGMC, this solution will be subject to stochastic noise, the uniform sampling of phase space results in a relatively uniform uncertainty distribution. Roughly speaking, uncertainty in Monte Carlo is inversely proportional to flux, i.e., it is lowest in regions that many neutrons visit and highest in regions that few neutrons visit. To demonstrate this key
To more precisely quantify the differences in uncertainty, we also examined the total flux and uncertainties along a single diagonal line through the CSG7 benchmark reactor that cuts from the center of the core out to the peripheral edge of the reflector region. While the flux distribution in the outer reflector regions of the moderator is not an official part of the CSG7 benchmark, one can imagine a variation of the problem where a neutron detector is placed deep into the reflector and requires a highly accurate solution. In particular, a location of interest is the area with the lowest neutron flux in the simulation (i.e., the area in the far corner of the reflector that borders two vacuum boundaries).

To determine the error in the flux solution along this diagonal line, we first needed a reference solution to compare against, as the CSG7 benchmark specification only supplies a reference pin power distribution. This was done by running OpenMC’s MGMC solver using 10 million particles per batch to simulate a total of 216 billion active particles, resulting in a standard deviation of 0.36% in the corner flux tally. The resulting flux map was then used as the reference solution for evaluating the numerical performance of MGMC and TRRM.

With a reference solution generated, we then ran the MGMC and TRRM solvers 10 times to generate ensemble flux error and runtime data along the diagonal. Simulation configurations generally used parameters from the most finely resolved tests from section 3.1 though a few small changes were made. First, to ensure that an apples-to-apples runtime comparison is made, the number of particles per batch with MGMC is increased to 20,350 (with 10,400 active batches), while random ray is configured with 1,750 rays per batch and 1300 active batches. The dead length in random ray was increased to 200 cm to ensure that uncollided flux contributions from the core were captured; given the extremely low flux in the periphery, these contributions can be significant. Additionally, the random ray mesh used for the pin power analysis (as shown in Figure 1c) is refined in the outer periphery of the moderator region, such that all pincell sized regions in the reflector are subdivided with a 10 × 10 mesh rather than only using this finer subdivision in areas closer in to the core. This increases the random ray mesh to a total of 199,988 FSRs. With these configurations, OpenMC was found to run both methods in approximately the same serial runtime (with random ray requiring 4756 seconds and MGMC requiring 4763 seconds).

The average results of our ensemble analysis are shown in Figure 7 which shows that both methods have well converged (and highly accurate) estimates of the flux within the fueled region of the reactor (with errors less than 1%). However, in the reflector region of the reactor, as the total flux drops off by 4 orders of magnitude, the error in the MGMC flux estimate increases rapidly to about 10.1% ± 6.1% at the outer corner of the reactor, compared to the TRRM estimate which only has 0.52% ± 0.56% error. The higher levels of error in the corner reflector region in MGMC are due to the extremely low numbers of neutrons reaching this corner. The uncertainty (standard deviation) of the MGMC corner flux estimate is about 11.8% whereas the TRRM
uncertainty is about 0.35%. To mitigate this effect, variance reduction techniques (e.g., weight windows) would need to be defined and tuned by the user. The slightly elevated error in the corner reflector region in TRRM is due to this cell having a very steep flux gradient (owing to two of this cell’s edges having vacuum boundary conditions), which combined with the flat source approximation used in OpenMC, results in a small amount of bias. Bias due to the flat source approximation causes the observed uncertainty in the flux at this location to be smaller than the average error compared to the MGMC reference solution. Error in TRRM can be further reduced relatively inexpensively in this cell by simply refining the moderator mesh resolution.

Overall, for equal runtime, the TRRM solver in OpenMC was found to result in about a 19.3× reduction in error as compared to MGMC. Given that error falls off with the square root of the number of particles in MGMC, this means that for equal fidelity in the peripheral cell, MGMC is projected to be about 373× slower than TRRM.

4. Summary and Conclusions

This paper has covered the implementation of TRRM in OpenMC and SCONE and discussed its performance with respect to MGMC. First, a brief description of TRRM was given, as well as of MGMC. Then, similarities and differences between the two methods were thoroughly discussed. The details of the TRRM implementations in each code were given, with additional detail added on the use of distance caching in SCONE.

Several numerical investigations were performed considering the runtime required for each code and method to achieve a certain level of error. These investigations were performed on the standard C5G7 bench-
mark, followed by a newly devised C5G69 model (featuring 69 energy groups as opposed to C5G7’s 7 energy groups) in attempting to exploit TTRM’s vectorisation over energy and resulting increased efficiency.

When considering pin power errors, for C5G7, TTRM was comparable to or slightly faster than MGMC. Relative differences between methods were consistent across codes. However, each solver in SCONET was a little faster than its counterpart in OpenMC. Differences in the TTRM performance between codes are most likely due to differences in the geometry implementation. Differences in the performance of the MGMC solvers are most likely due to the use of delta tracking and structured universes in SCONET as opposed to surface tracking in OpenMC. As well, the use of distance caching was shown to give a reasonable acceleration to TTRM by reducing the expense of ray tracing, on the order of 10-15% for this seven-group problem.

In the C5G69 case, TTRM’s runtime showed small increases due to the increase in energy groups from 7 to 69. Conversely, MGMC’s runtime was observed to be relatively insensitive to the increase in energy group count. For this problem, random ray’s performance in both codes was comparable with or worse than MGMC: this is because while TRRM must explicitly consider every single energy group for every ray that is transported, neutrons in MC simulations will often ‘skip over’ many of these energy groups and so will not necessarily have much additional work to do with an increased number of groups. In this problem, distance caching did not noticeably accelerate TTRM as the vast majority of the computation is spent in exponential evaluation and updating fluxes rather than in ray tracing. Furthermore, as opposed to C5G7 where ray tracing costs were relatively large, for C5G69 the vast majority of the computational effort is spent on evaluating the transport equation/exponentials. Therefore, both TTRM solvers exhibited similar performance as differences in their geometry subroutines were amortised. TTRM’s performance might be improved on this problem if a convergence acceleration scheme were to be implemented or if higher order sources were used.

When considering regions of phase space with very low physical neutron flux, such as the flux deep into a reflector region outside of the fuel, TTRM was substantially more efficient than MGMC. Although variance reduction methods might allow MGMC to more efficiently estimate flux in these remote regions, this typically entails more complication to the algorithm or some degree of iteration, e.g., to calculate weight windows. As such, in common with deterministic methods, TTRM is more efficient at providing a global solution than MGMC. For instance, we found that when considering a low-flux region in the reflector of the C5G7 problem, TTRM was approximately 373× faster than MGMC given the same accuracy target.

The efficiency of the TTRM implementations given
here is quite good with respect to dedicated random ray solvers like ARRC. As such, this paper demonstrates that it is relatively straightforward to implement TRRM in MGMC codes in a performant manner while enabling future work with TRRM using more general constructive solid geometries.

In future, these implementations may be expanded to support MPI parallelism through domain decomposition, linear and anisotropic sources, and many of the other features that modern MoC codes enjoy. These implementations are also open-source, allowing other researchers to more easily engage in developing TRRM. Future research will also explore expanding TRRM to other radiation transport problems, such as fixed source problems, using the implementations that have been developed here.

A natural extension of this work would be to perform similar comparisons but for GPU implementations of both methods. TRRM has shown excellent performance on GPUs (Tramm and Siegel, 2021), as has continuous energy Monte Carlo (Hamilton and Evans, 2019; Choi et al., 2019; Tramm et al., 2022) and MGMC (Hamilton et al., 2018). Given recent GPU developments with numeric precision.

Acknowledgments

P. Cosgrove would like to gratefully acknowledge helpful discussions with M.A. Kowalski on false sharing, its causes, and solutions. Thanks are also due to V. Raffuzzi for generating multi-group cross sections and creating a continuous energy equivalent of CSG7. P. Beenie assisted through maintaining the Cambridge Nuclear Energy Group workstations and for helpful suggestions on numeric precision.

This work was performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service (www.csd3.cam.ac.uk), provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Council (capital grant EP/T022159/1), and DiRAC funding from the Science and Technology Facilities Council (www.dirac.ac.uk).

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. The material was based upon work supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357.

We gratefully acknowledge the computing resources provided and operated by the Joint Laboratory for System Evaluation (JLSE) at Argonne National Laboratory.

References


Herman, B. R., 2014. Monte Carlo and Thermal Hydraulic Coupling


