Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A Monte Carlo synthetic-acceleration method for solving the thermal radiation diffusion equation[☆,☆☆]

Thomas M. Evans[a,1,*], Scott W. Mosher[a,1], Stuart R. Slattery[b,2], Steven P. Hamilton[a,1]

[a] Oak Ridge National Laboratory, 1 Bethel Valley Rd., Oak Ridge, TN 37831, USA
[b] University of Wisconsin–Madison, 1500 Engineering Dr., Madison, WI 53716, USA

ARTICLE INFO

Article history:
Received 14 September 2012
Received in revised form 18 October 2013
Accepted 21 October 2013
Available online 4 November 2013

Keywords:
Radiation diffusion
Synthetic acceleration
Monte Carlo
Sparse matrix systems

ABSTRACT

We present a novel synthetic-acceleration-based Monte Carlo method for solving the equilibrium thermal radiation diffusion equation in three spatial dimensions. The algorithm performance is compared against traditional solution techniques using a Marshak benchmark problem and a more complex multiple material problem. Our results show that our Monte Carlo method is an effective solver for sparse matrix systems. For solutions converged to the same tolerance, it performs competitively with deterministic methods including preconditioned conjugate gradient and GMRES. We also discuss various aspects of preconditioning the method and its general applicability to broader classes of problems.

1. Introduction

In nearly all computational engineering and physics fields, linear and nonlinear solvers form the core components of modeling and simulation applications. Recent focus on multiphysics coupling adds additional complexity to common linear and nonlinear systems as solution strategies change when physical models are coupled. Furthermore, a desire for predictive simulations to enhance the safety and performance of engineered systems creates a need for extremely high fidelity computations to be performed for these coupled systems as a means to capture effects not modeled by coarser methods. In order to achieve this high fidelity, state-of-the-art computing facilities must be leveraged in a way that is both efficient and considerate of hardware-related issues. As scientific computing moves towards exascale facilities, with machines of $O(1{,}000{,}000)$ cores already coming online, new algorithms to solve these complex problems must be developed to leverage this new hardware. Issues such as resiliency to node failures and scaling to large numbers of heterogeneous computing elements (CPUs and GPUs) will be pertinent to robust algorithms aimed at this new hardware.

☆ Notice: This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

☆☆ This is an open-access article distributed under the terms of the Creative Commons Attribution-NonCommercial-No Derivative Works License, which permits non-commercial use, distribution, and reproduction in any medium, provided the original author and source are credited.

* Corresponding author.
E-mail addresses: evanstm@ornl.gov (T.M. Evans), moshersw@ornl.gov (S.W. Mosher), sslattery@wisc.edu (S.R. Slattery), hamiltonsp@ornl.gov (S.P. Hamilton).

1 Radiation Transport Group, Reactor and Nuclear Systems Division.

2 Department of Engineering Physics, College of Engineering.

0021-9991/$ – see front matter © 2013 The Authors. Published by Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jcp.2013.10.043

In previous work [1], we developed a residual Monte Carlo method that was an application of Halton's sequential Monte Carlo method [2,3] to the one-dimensional (1D) equilibrium thermal radiation diffusion equation. At that time, Halton's method was not widely known in the general computational physics community. While extending our method to three spatial dimensions, we realized that the sequential Monte Carlo method is actually a variant of iterative refinement cast as a residual method. We have developed a new method in which Monte Carlo is used to accelerate a fixed-point (Richardson) iteration. This algorithm surpasses our previous 1D method and allows for extension to more general solution techniques.

The purpose of this study is to demonstrate an efficient Monte Carlo solution method for three-dimensional (3D), time-dependent, discrete systems. In this work, our previous investigations are extended in the following manner:

- we develop a rigorous synthetic-acceleration Monte Carlo method for solving sparse matrix systems;
- we apply this method to the one-temperature (1T) thermal radiation diffusion equation in three dimensions;
- we compare numerical and performance results of the Monte Carlo method against traditional deterministic solution methods.

While parallel scaling and resiliency on Petascale systems and beyond provides strong motivation for pursuing this work, we do not directly address these issues in this study. These topics will be the subject of future investigations.

In Section 2 we review Monte Carlo methods for solving matrix systems, and we discuss our new synthetic-acceleration scheme and connect it with previous work by Halton [3] and others [1]. The 1T radiation diffusion equation and its discrete form will be derived in Section 3. We discuss using Monte Carlo Synthetic Acceleration to solve the discrete radiation diffusion equation in Section 4. Results from two 3D test problems are given in Section 5. We finish with conclusions and ideas for future work in Section 6.

2. Mathematical methods

The idea of using Monte Carlo methods (random walks) to invert linear systems is not new. The earliest referenced work dates to a 1950 paper by Forsythe and Leibler [4]. Their paper credits the idea to unpublished work by J. von Neumann and S.M. Ulam dating back to the 1940's. The basic principles of Monte Carlo matrix inversion were further elucidated in Hammersley and Handscomb's 1964 text [5]. These early methods are distinguished by very slow, statistically noisy convergence properties; thus, they have not made any significant impact in the physics community.

In order to address the convergence issues plaguing Monte Carlo solvers, Halton [2,3] proposed a sequential Monte Carlo method. This algorithm demonstrated dramatically improved convergence over regular Monte Carlo. Nonetheless, Halton's method has not gained widespread use in computational physics applications. In the next sections, we will briefly describe the Monte Carlo methods that can be used to solve linear systems and expand them to present our new synthetic-acceleration method.

2.1. Monte Carlo matrix solution methods

To establish the mathematical framework for Monte Carlo linear solvers, we consider the following matrix equation:

$$\mathbf{Ax} = \mathbf{b}, \tag{1}$$

which can be written as

$$\begin{aligned}\mathbf{x} &= (\mathbf{I} - \mathbf{A})\mathbf{x} + \mathbf{b}\\ &= \mathbf{Hx} + \mathbf{b}.\end{aligned} \tag{2}$$

When the spectral radius (ρ) of the iteration matrix ($\mathbf{H}$) is less than 1, we can expand $\mathbf{A}^{-1}$ using the *Neumann Series*:

$$\mathbf{A}^{-1} = (\mathbf{I} - \mathbf{H})^{-1} = \sum_{k=0}^{\infty} \mathbf{H}^k. \tag{3}$$

Thus, when $\rho(\mathbf{H}) < 1$, we can recast the solution vector $\mathbf{x}$ as a series:

$$\begin{aligned}\mathbf{x} &= (\mathbf{I} - \mathbf{H})^{-1}\mathbf{b}\\ &= \mathbf{b} + \mathbf{Hb} + \mathbf{H}^2\mathbf{b} + \mathbf{H}^3\mathbf{b} + \cdots.\end{aligned} \tag{4}$$

Using this series, we can write an iterative method, known as Richardson iteration, that solves Eq. (1):

$$\mathbf{x}^{k+1} = \mathbf{Hx}^k + \mathbf{b}. \tag{5}$$

Eq. (5) will converge for all $\mathbf{b}, \mathbf{x}^0 \in \mathcal{R}^N$ when $\rho(\mathbf{H}) < 1$ [6]. All of the Monte Carlo methods that are described in Sections 2.1.1 through 2.2 rely on estimating the terms in Eq. (4) through random walks.

2.1.1. Direct method

Now we consider a Monte Carlo method that can be used to estimate the solution to Eq. (1) [5]. First, rewrite Eq. (4) in order to calculate a component of $\mathbf{x}$:

$$\begin{aligned} x_i &= (\mathbf{b})_i + (\mathbf{Hb})_i + \left(\mathbf{H}^2\mathbf{b}\right)_i + \left(\mathbf{H}^3\mathbf{b}\right)_i + \cdots \\ &= \sum_{k=0}^{\infty}\sum_{i_1}^{N}\sum_{i_2}^{N}\ldots\sum_{i_k}^{N} h_{i,i_1}h_{i_1,i_2}\ldots h_{i_{k-1},i_k}b_{i_k}. \end{aligned} \tag{6}$$

Here, k is the random walk step index and the h terms are the matrix elements of $\mathbf{H}$. The terms in the Neumann series can be interpreted as a series of transitions from $i_{k-1} \to i_k$ that can be simulated by a random walk. Let X be a random variable sampled from a random walk with k events that initiates in state i:

$$\begin{aligned} X(i_0 = i) &= \sum_{m=0}^{k} W_m b_{i_m} \\ &= \sum_{m=0}^{k} w_{i,i_1}w_{i_1,i_2}\ldots w_{i_{m-1},i_m}b_{i_m}. \end{aligned} \tag{7}$$

Here, the weight on the mth step is denoted W_m, and each random walk starts with unit weight. At every step, contributions to each component of X are generated only in the starting state of the history, i_0, by the source term in the current state, b_{i_m}. Therefore, for a given random walk permutation, each state must be used as a starting value in order to calculate its component of the expectation value (i.e. a state of size N will have N histories per random walk permutation). When there is a transition between states, for example $i \to j$, the weight is multiplied by the transition factor

$$w_{ij} = \frac{h_{ij}}{p_{ij}}, \tag{8}$$

where p_{ij} denotes the probability of transitioning from state i to j. Then, the expected value of X is

$$\begin{aligned} E\left[X(i_0 = i)\right] &= \sum_{\nu} P_\nu X_\nu \\ &= \sum_{k=0}^{\infty}\sum_{i_1}^{N}\sum_{i_2}^{N}\ldots\sum_{i_k}^{N} p_{i,i_1}p_{i_1,i_2}\ldots p_{i_{k-1},i_k}w_{i,i_1}w_{i_1,i_2}\ldots w_{i_{k-1},i_k}b_{i_k} \\ &= x_i, \end{aligned} \tag{9}$$

where ν denotes a particular random walk permutation. Therefore, the estimator in Eq. (7) is an unbiased estimator of the components of $\mathbf{x}$ provided $\rho(\mathbf{H}) < 1$.

We are left to define the transition probabilities. The most straightforward approach is to set

$$p_{ij} = \frac{|h_{ij}|}{\sum_j |h_{ij}|}. \tag{10}$$

Each row of the transition probability matrix, $\mathbf{P}$, represents a discrete probability density function that can be sampled to select a new state j, given that the current state is i. Random walks can be terminated in two ways: the matrix can be augmented with a terminating event equation that describes the probability that a history ends its walk, or the random walk can be terminated by weight cutoff, W_c. Generally, we choose to terminate random walks using a weight cutoff such that the random walk terminates when $W_m < W_c$.

2.1.2. Adjoint method

An alternative approach to the direct method is to calculate contributions to every component of $\mathbf{x}$ during the random walk. In this method, the weight change from state $i \to j$ is

$$w_{ij} = \frac{h_{ji}}{p_{ij}}. \tag{11}$$

The transition probabilities may be calculated as

$$p_{ij} = \frac{|h_{ji}|}{\sum_j |h_{ji}|}. \tag{12}$$

Note that the indices are reversed so that the probabilities are normalized over a column, as opposed to the forward method in which the probabilities are normalized over a row. This is equivalent to forming the Neumann series in reverse order. Correspondingly, the estimator for this method is

$$\begin{aligned} X &= \sum_{m=0}^{k} W_m \delta_{i_m,i} \\ &= \sum_{m=0}^{k} \hat{b}_{i_0} w_{i_0,i_1} w_{i_1,i_2} \ldots w_{i_{m-1},i_m} \delta_{i_m,i}. \end{aligned} \tag{13}$$

Here, $\hat{b}_{i_0}$ is the sampled source and initial weight in state i_0. The Kronecker delta implies that tallies are only made in the state where the random walk currently resides. Thus, the adjoint method is equivalent to the common approach in standard Monte Carlo transport simulations. As opposed to the direct method, the adjoint method does not require starting a history in each state for each random walk permutation because tallies are made in the state in which the random walk currently resides. Instead, sampling the source is sufficient, and one random walk per permutation is required.

Similar to the direct method, the adjoint method random walk process requires a terminating condition. In all of the work that follows we utilize a relative weight cutoff. The relative weight cutoff is defined as a fraction of the starting weight such that

$$W_f = W_c \hat{b}_{i_0}, \tag{14}$$

where W_f is the terminating weight of the random walk and W_c is the input relative weight cutoff. The random walk terminates on the mth step if $W_m < W_f$.

2.2. Monte Carlo synthetic acceleration

The methods presented in Section 2.1 are characterized by slow convergence rates bound by the Central Limit Theorem. Halton [2,3] proposed a staged residual scheme called sequential Monte Carlo to speed up the convergence of these methods. A variant of this scheme has been successfully applied to the 1D nonlinear thermal radiation diffusion equation in Ref. [1].

Concisely, the sequential Monte Carlo method solves Eq. (1) using the adjoint solution technique described in Section 2.1.2. The following iteration scheme is applied:

$$\mathbf{r}^l = \mathbf{b} - \mathbf{A}\mathbf{x}^l, \tag{15a}$$

$$\hat{\mathbf{A}}\delta\mathbf{x}^{l+1} = \mathbf{r}^l, \tag{15b}$$

$$\mathbf{x}^{l+1} = \mathbf{x}^l + \delta\mathbf{x}^{l+1}. \tag{15c}$$

In this scheme, the Monte Carlo adjoint method is used to estimate the solution to Eq. (15b) as represented by the hat on **A**, and the residual is iterated to convergence. This iteration sequence is closely related to iterative refinement, the exception being that there is no update to x^{l+1} between residual iterations.

We propose a modification of this scheme that uses Monte Carlo as a synthetic acceleration [7] for the fixed-point iteration given by Eq. (5). We begin by subtracting Eq. (5) from Eq. (2):

$$\delta\mathbf{x}^{l+1} = (\mathbf{I} - \mathbf{A})\delta\mathbf{x}^l, \tag{16}$$

where

$$\delta\mathbf{x}^l = \mathbf{x} - \mathbf{x}^l \tag{17}$$

is defined as the error at iteration l. We then subtract $(\mathbf{I} - \mathbf{A})\delta\mathbf{x}^{l+1}$ from Eq. (16), giving a linear system to solve for the error:

$$\begin{aligned} \mathbf{A}\delta\mathbf{x}^{l+1} &= (\mathbf{I} - \mathbf{A})\left(\mathbf{x}^{l+1} - \mathbf{x}^l\right) \\ &= \mathbf{r}^{l+1}. \end{aligned} \tag{18}$$

The following scheme will then converge in one iteration:

$$\mathbf{x}^{l+1} = (\mathbf{I} - \mathbf{A})\mathbf{x}^l + \mathbf{b}, \tag{19a}$$

$$\mathbf{A}\delta\mathbf{x}^{l+1} = \mathbf{r}^{l+1}, \tag{19b}$$

$$\mathbf{x} = \mathbf{x}^{l+1} + \delta\mathbf{x}^{l+1}. \tag{19c}$$

These steps can be written as an accelerated iteration scheme that applies the Monte Carlo method to estimate the error in each iteration. The Monte Carlo method is used to approximately invert $\mathbf{A}$ in Eq. (19b). This updated scheme, the Monte Carlo Synthetic-Acceleration (MCSA) method, can be defined as follows:

$$\mathbf{x}^{l+1/2} = (\mathbf{I} - \mathbf{A})\mathbf{x}^l + \mathbf{b}, \tag{20a}$$

$$\mathbf{r}^{l+1/2} = \mathbf{b} - \mathbf{A}\mathbf{x}^{l+1/2}, \tag{20b}$$

$$\hat{\mathbf{A}}\delta\mathbf{x}^{l+1/2} = \mathbf{r}^{l+1/2}, \tag{20c}$$

$$\mathbf{x}^{l+1} = \mathbf{x}^{l+1/2} + \delta\mathbf{x}^{l+1/2}. \tag{20d}$$

As in the sequential method, the hat on $\mathbf{A}$ in Eq. (20c) indicates that the Monte Carlo solution only approximately inverts this operator. Thus, we have defined a scheme in which the initial estimate of $\mathbf{x}$ in each iteration is updated using a single fixed-point iteration.[3] The residual is calculated and used as a source to estimate the error, $\delta\mathbf{x}^{l+1/2}$, by solving Eq. (20c) via the Monte Carlo adjoint method. The error is then used to calculate an updated iterate of the solution vector, $\mathbf{x}^{l+1}$. The entire sequence is iterated to convergence based on the following stopping criterion [6]:

$$\|\mathbf{r}\|_2 < \epsilon \cdot \|\mathbf{b}\|_2. \tag{21}$$

A closer examination of the sequential method in Eq. (15) shows that this scheme is equivalent to a single preconditioned Richardson iteration,

$$\mathbf{x}^{l+1} = \mathbf{x}^l + \hat{\mathbf{A}}^{-1}\mathbf{r}^l, \tag{22}$$

in which $\hat{\mathbf{A}}^{-1}$ indicates the approximate inversion of $\mathbf{A}$ via the adjoint Monte Carlo method. MCSA, on the other hand, is identical to a Richardson iteration followed by a preconditioned Richardson iteration,

$$\mathbf{x}^{l+1/2} = \mathbf{x}^l + \mathbf{r}^l, \tag{23}$$

$$\mathbf{x}^{l+1} = \mathbf{x}^{l+1/2} + \hat{\mathbf{A}}^{-1}\mathbf{r}^{l+1/2}. \tag{24}$$

As above, the adjoint Monte Carlo process is used to approximately invert $\mathbf{A}$.

2.3. Monte Carlo method selection

The MCSA method defined in Eq. (20) uses the adjoint method to estimate the error in Eq. (20c). To demonstrate the effectiveness of the adjoint method over the direct method within the context of MCSA, we choose the 2D time-dependent Poisson equation as a simple model problem:

$$\frac{\partial \mathbf{u}}{\partial t} = \nabla^2 \mathbf{u}. \tag{25}$$

For all comparisons, a single time step is computed with backwards Euler time integration. The Laplacian is differenced on a square Cartesian grid with a second-order five-point stencil,

$$\nabla_5^2 = \frac{1}{\Delta^2}[u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}], \tag{26}$$

and a fourth-order nine-point stencil,

$$\begin{aligned}\nabla_9^2 = \frac{1}{6\Delta^2}[&4u_{i-1,j} + 4u_{i+1,j} + 4u_{i,j-1} + 4u_{i,j+1} + u_{i-1,j-1} \\ &+ u_{i-1,j+1} + u_{i+1,j-1} + u_{i+1,j+1} - 20u_{i,j}],\end{aligned} \tag{27}$$

both assuming a grid size of Δ in both the i and j directions. For a single time step solution, we then have the following sparse linear system to be solved with the MCSA method:

$$\mathbf{A}\mathbf{u}^{n+1} = \mathbf{u}^n. \tag{28}$$

Both the stencils will be used to vary the size and density of the sparse linear system in Eq. (28).

A timing and convergence study is used to demonstrate the improved effectiveness of the adjoint method as compared to the direct method. To assess both the CPU time and number of iterations required to converge to a solution, a problem of constant Δ was used with varying number of grid points, fixing the spectral radius of the system at a constant value for each variation. Both the five-point and nine-point stencils were used with both the direct and adjoint solvers. For each

[3] A subspace method could be used instead of a stationary method here.

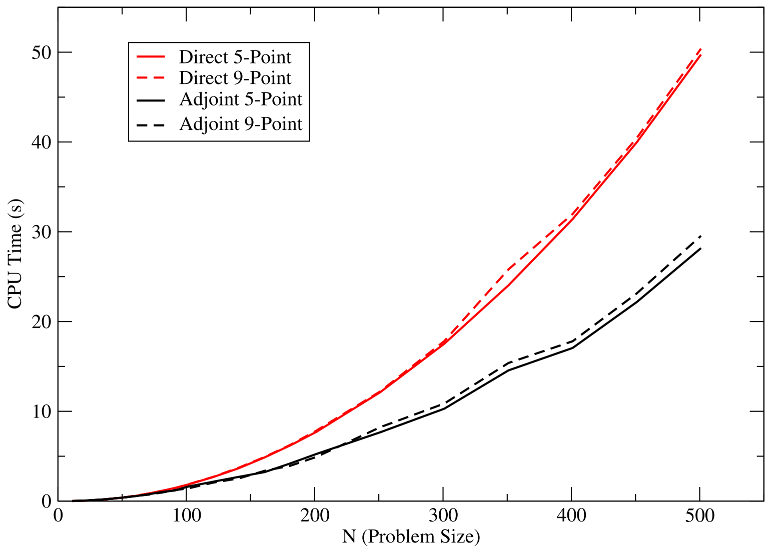

Fig. 1. CPU time (s) to converge vs. problem size (N for an $N \times N$ square mesh). Both the adjoint and direct solvers are used with the five-point and nine-point stencils. A CPU time speedup is achieved with the adjoint method due to the higher density of random walk events in regions with a large residual.

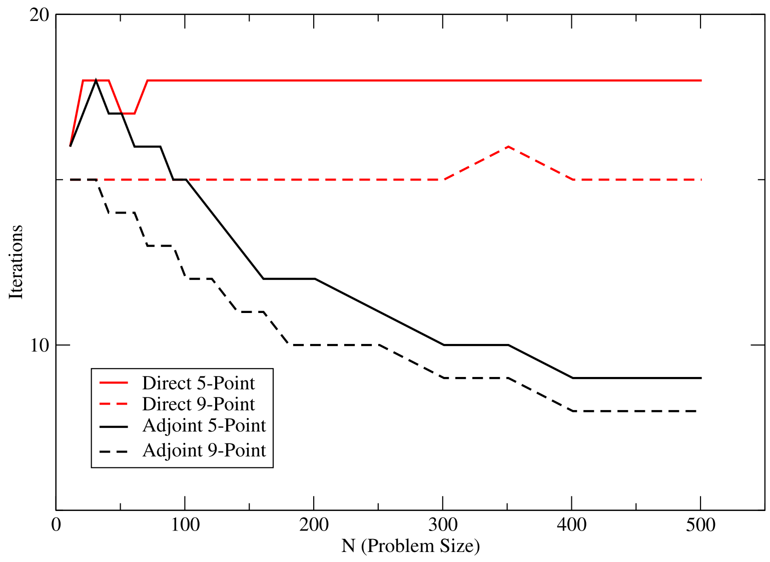

Fig. 2. Iterations to converge vs. problem size (N for an $N \times N$ square mesh). Both the adjoint and direct solvers are used with the five-point and nine-point stencils.

case, $N \times N$ total random walk permutations were computed per MCSA iteration where $N \times N$ is the number of discrete grid points in the system. Solver parameters were set to a weight cutoff of 1×10^{-4} for the stochastic linear solver and a convergence tolerance of 1×10^{-8} for the MCSA iterative solver. Fig. 1 gives the CPU time needed for each case to converge in seconds and Fig. 2 gives the number of iterations needed for each case to converge to the specified tolerance as a function of the problem size. All computations presented in this section and Section 2.4 were completed on a 3.0 GHz Intel Core 2 Quad Q9650 CPU machine with 16 GB 1067 MHz DDR3 memory.

We see in Figs. 1 and 2 that the adjoint solver with MCSA is faster and requires fewer iterations than the direct method. Several reasons explain this result. First, with an equivalent number of histories specified for both solvers per MCSA iteration and a system of size $N \times N$, the direct solver will compute a single random walk for each state in the system per iteration to acquire a solution in that state, regardless of the size of the residual in that state. This is necessary in the direct method to ensure a contribution from each state as the random walk sequence will only contribute to the starting state. For the adjoint method, a total of $N \times N$ random walk events will have their starting state determined by sampling the residual vector. Because the random walk sequence contributes to the state in which it currently resides, sampling the residual vector as the Monte Carlo source gives a higher density of random walk events in regions with a high residual, thus giving a more

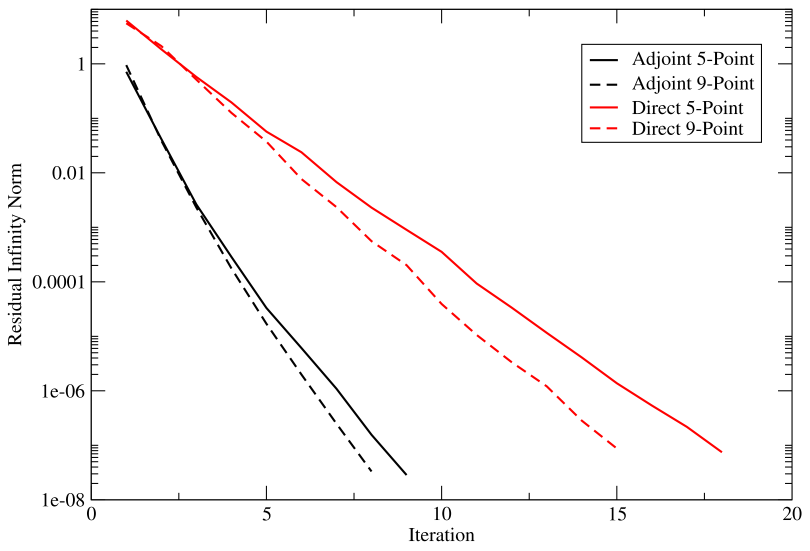

Fig. 3. Infinity norm of the solution residual vs. iteration number for a problem of size $N = 500$. Both the adjoint and direct solvers are used with the five-point and nine-point stencils. A higher rate of convergence is observed for MCSA using the adjoint Monte Carlo solver as compared to the direct method when both solvers compute the same number of random walks per iteration.

effective correction in those regions due to reduced statistical error. From an iteration perspective, Fig. 2 shows that using the direct method yields a roughly unchanging number of iterations required to converge as the problem size increases. Again, if we desire a correction value for all states in the problem, then we must start a random walk in each state in the system which does not reduce the number of iterations needed as the problem size grows. Conversely, as the problem size grows in the adjoint method, the additional stochastic histories that will be computed are concentrated in regions with a large residual, further reducing the stochastic error in the correction in those regions and ultimately reducing the required number of iterations to converge.

As an additional comparison, the convergence behavior of MCSA can be analyzed using both the adjoint and direct solvers to detect any performance benefits. To assess the convergence properties of MCSA using each solver and stencil, the infinity norm of the residual computed in Eq. (20b) was collected at each iteration for a fixed problem size of $N = 500$. Fig. 3 gives the results of these computations. First, it is worthy to note on the semilog plot that we are indeed achieving the expected exponential convergence from MCSA with both Monte Carlo solvers. Second, we note that using the adjoint method with the same number of stochastic histories per MCSA iteration gives a faster rate of converge for the same reasons as above. We also note here that fewer iterations are required for convergence when the 9-point stencil is used to discretize the Laplacian operator (although at no gain in speed as given by the results in Fig. 1). This is due to the fact that the smaller discretization error directly corresponds to a more well defined residual source generated by the Richardson iteration for the Monte Carlo calculation. In addition, the better defined source is transported through a domain described more accurately by the 9-point stencil, thus yielding a more accurate correction vector from the Monte Carlo calculation.

2.4. Comparison to sequential Monte Carlo

To further motivate using Monte Carlo Synthetic Acceleration, we compare its performance to Halton's sequential Monte Carlo method on which our previous work in this area was based. For this comparison, we use the same transient Poisson problem as described in the previous section and choose only the 5-point stencil to discretize the Laplacian operator because the previous results yielded little qualitative difference between the discretizations. Both MCSA and Halton's method are used with the adjoint Monte Carlo solver. In order to complete the same study as in the previous section, the number of histories computed by the Monte Carlo solver at each iteration had to be doubled to $2 \times N \times N$ in order to ensure convergence when using the sequential Monte Carlo Method. Fig. 4 gives the CPU time results for this comparison as a function of problem size, and Fig. 5 gives the number of iterations to converge as a function of problem size. In both cases, using the Monte Carlo solver as a synthetic acceleration rather than in a sequential Monte Carlo scheme resulted in a reduction in both CPU time and iterations required to converge. The additional Richardson iteration between each Monte Carlo solve in the MCSA method gives a better converged residual source to use with the Monte Carlo calculation while the sequential method requires more iterations to achieve the same level of convergence in the residual.

The benefits of using a synthetic-acceleration scheme are also illustrated when comparing the residuals at each iteration. Figs. 6 and 7 show the infinity norm of the residual computed at each iteration for both methods for fixed problems sizes of $N = 100$ and 500, respectively. In both cases, the sequential method is characterized by a slow approach to the asymptotic rate of convergence. Conversely, MCSA reaches its asymptotic convergence rate more quickly and exhibits a faster final rate of

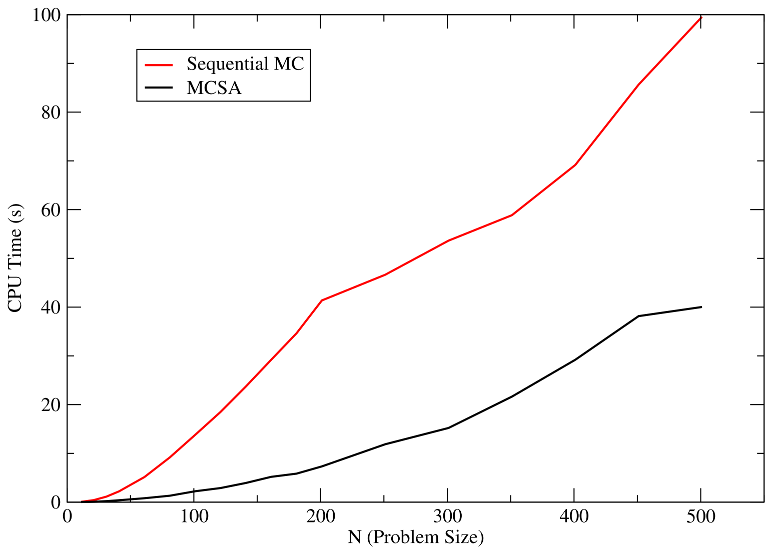

Fig. 4. CPU time (s) to converge vs. problem size (N for an $N \times N$ square mesh). Both the sequential Monte Carlo and MCSA solvers are used with the five-point stencils and the adjoint Monte Carlo solver. The number of random walks was twice the number of discrete states in the system in order to ensure convergence in the sequential Monte Carlo method.

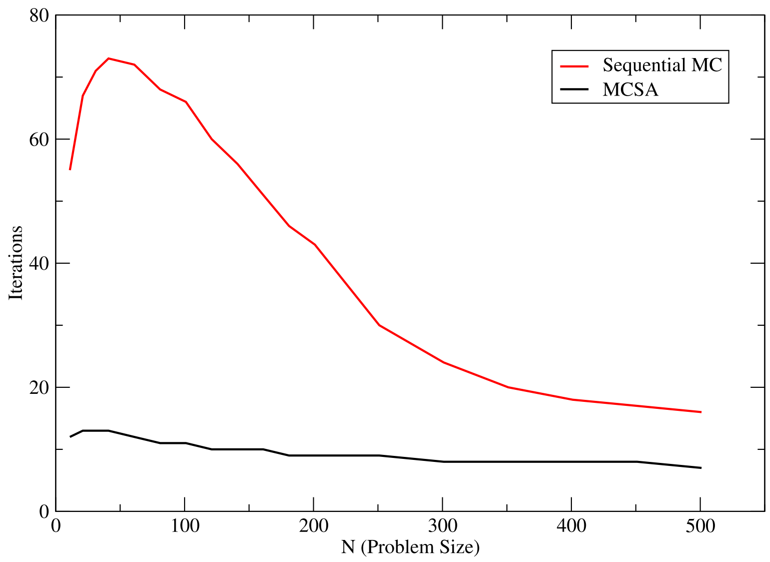

Fig. 5. Iterations to converge vs. problem size (N for an $N \times N$ square mesh). Both the sequential Monte Carlo and MCSA solvers are used with the five-point stencils and the adjoint Monte Carlo solver.

convergence than Halton's method. Even with the doubling of the number of stochastic histories computed per time step in order to ensure convergence for the sequential method, we still see robustness issues with a non-monotonically decreasing residual observed for the $N = 100$ case. In both cases the MCSA solver is observed to be robust with a monotonically decreasing residual.

3. Discrete form of the radiation diffusion equation

Using the mathematical preliminaries developed in the previous section, we can now generate a discrete form of the radiation diffusion equation. We will show that this discrete form is amenable to interpretation using Monte Carlo concepts and can be written in a way that generates a random walk sequence.

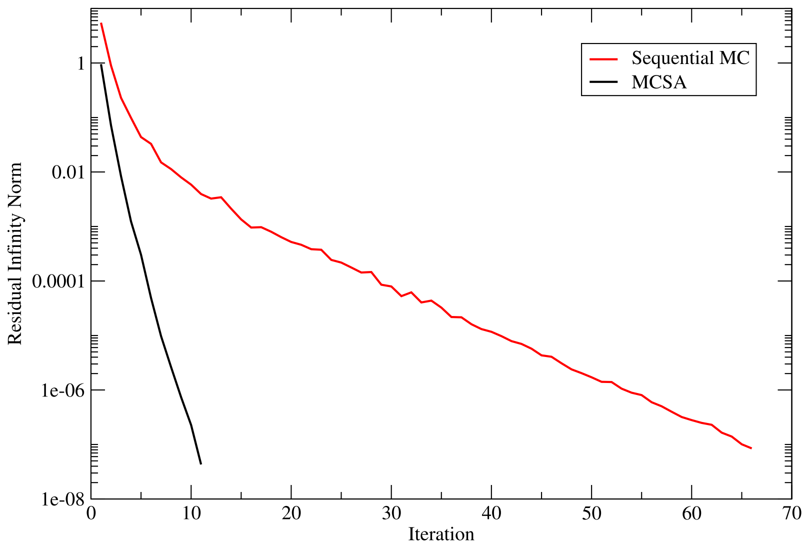

Fig. 6. Infinity norm of the solution residual vs. iteration number for a problem of size $N = 100$. Both the sequential Monte Carlo and MCSA solvers are used with the five-point stencils and the adjoint Monte Carlo solver.

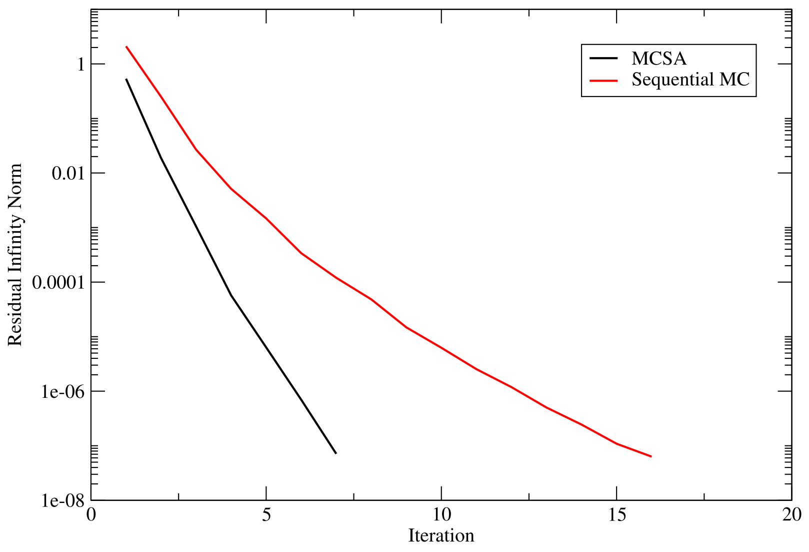

Fig. 7. Infinity norm of the solution residual vs. iteration number for a problem of size $N = 500$. Both the sequential Monte Carlo and MCSA solvers are used with the five-point stencils and the adjoint Monte Carlo solver.

3.1. *Equilibrium diffusion model*

The general form of the equilibrium radiation diffusion equation is [8]

$$\left(\rho C_v + 4aT^3\right)\frac{\partial T}{\partial t} - \nabla \cdot \left(\frac{4acT^3}{3\sigma_R}\right)\nabla T = Q, \tag{29}$$

where $C_v \equiv C_v(\mathbf{r}, T(\mathbf{r}, t))$ [GJ g^{-1} keV^{-1}] is the specific heat capacity of the material, $\rho \equiv \rho(\mathbf{r})$ [g cm^{-3}] is the density of the material, $a = 0.01372$ [GJ cm^{-3} keV^{-4}] is the radiation constant, $c = 29.979$ [cm ns^{-1}] is the vacuum light speed, and $T \equiv T(\mathbf{r}, t)$ [keV] is the temperature that characterizes both the radiation and the material. The opacity, $\sigma_R \equiv \sigma_R(\mathbf{r}, T(\mathbf{r}, t))$ [cm^{-1}], is the Rosseland Mean opacity. The source, $Q \equiv Q(\mathbf{r}, t)$ [GJ cm^{-3} ns^{-1}], is assumed to be independent of T.

We choose to use the angle- and energy-integrated radiation intensity, $\phi \equiv \phi(\mathbf{r}, t)$ [GJ cm^{-2} ns^{-1}], as the primary state variable. The intensity is defined as

$$\begin{aligned}\phi(\mathbf{r},t) &= \int_{4\pi} d\boldsymbol{\Omega} \int_{\nu} d\nu\, \psi(\mathbf{r},\nu,\hat{\boldsymbol{\Omega}},t) \\ &= \int_{\nu} d\nu\, 4\pi B(\nu,T) \\ &= acT^4,\end{aligned} \tag{30}$$

where ψ is the angular radiation intensity and ν is the radiation frequency. The Planck function (or Planckian) is defined as

$$B(\nu,T) = \frac{2h\nu^3}{c^2}\left(e^{\frac{h\nu}{kT}} - 1\right)^{-1}, \tag{31}$$

where h is Planck's constant and k is the Boltzmann constant. Utilizing this definition of ϕ we can write

$$\frac{\partial T}{\partial t} = \frac{1}{4acT^3}\frac{\partial \phi}{\partial t}. \tag{32}$$

Inserting (32) into (29) yields

$$\left(\frac{\rho C_v}{4acT^3} + \frac{1}{c}\right)\frac{\partial \phi}{\partial t} - \nabla\cdot D\nabla\phi = Q, \tag{33}$$

where D is the diffusion coefficient, $D \equiv D(\mathbf{r}, T(\mathbf{r},t))$ [cm], and is defined as

$$D = \frac{1}{3\sigma_{\mathrm{R}}}. \tag{34}$$

Eq. (33) is the model equation we will solve using the Monte Carlo techniques presented in Sections 2 and 2.2.

3.2. Discrete diffusion equation

Applying backward-Euler time differencing to Eq. (33) and lagging the temperature-dependent coefficients at t^n gives

$$-\nabla\cdot D^n\nabla\phi + \tilde{\sigma}^n\phi = q^n, \tag{35}$$

where we have suppressed the $n+1$ indices on ϕ and

$$\tilde{\sigma}^n = \frac{\rho C_v^n}{4ac(T^n)^3\Delta t} + \frac{1}{c\Delta t}, \tag{36}$$

$$q^n = \tilde{\sigma}^n\phi^n + Q^n, \tag{37}$$

$$\phi^n = ac\left(T^n\right)^4. \tag{38}$$

Applying Fick's Law,

$$\mathbf{F} = -D\nabla\phi, \tag{39}$$

where $\mathbf{F}$ is the radiation flux, we define a flux-balance equation,

$$\nabla\cdot\mathbf{F} + \tilde{\sigma}^n\phi = q^n. \tag{40}$$

Integrating Eq. (40) over the cell illustrated in Fig. 8 gives

$$\iiint \nabla\cdot\mathbf{F}\,d\mathbf{r} + \tilde{\sigma}^n_{ijk}\phi_{ijk}V_{ijk} = q^n_{ijk}V_{ijk}, \tag{41}$$

and $V_{ijk} = \Delta_i\Delta_j\Delta_k$. We now apply the divergence theorem,

$$\begin{aligned}\iiint \nabla\cdot\mathbf{F}\,d\mathbf{r} &= \oint \hat{\mathbf{n}}\cdot\mathbf{F}\,dA \\ &= \sum_{f=1}^{6}\hat{\mathbf{n}}_f\cdot\mathbf{F}_f A_f,\end{aligned} \tag{42}$$

to Eq. (41) yielding

$$(F_{i+1/2} - F_{i-1/2})\Delta_j\Delta_k + (F_{j+1/2} - F_{j-1/2})\Delta_i\Delta_k + (F_{k+1/2} - F_{k-1/2})\Delta_i\Delta_j + \tilde{\sigma}^n_{ijk}\phi_{ijk}V_{ijk} = q^n_{ijk}V_{ijk}. \tag{43}$$

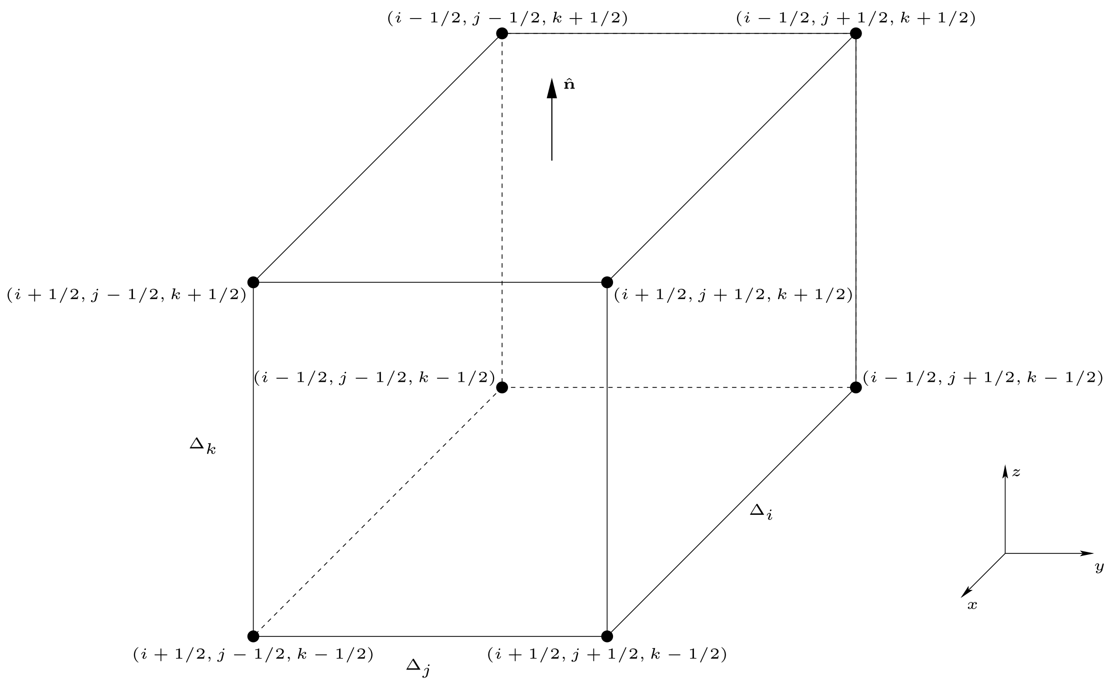

Fig. 8. Orthogonal 3D mesh cell.

Using Fick's Law, Eq. (39), we can write $O(\Delta^2)$ expressions for the face-centered fluxes. Defining $l \in \{i, j, k\}$, the face-centered fluxes are

$$F_{l+1/2} = -D_{l+1/2}\frac{\phi_{l+1} - \phi_l}{\Delta_{l+1/2}}, \tag{44}$$

$$F_{l-1/2} = -D_{l-1/2}\frac{\phi_l - \phi_{l-1}}{\Delta_{l-1/2}}, \tag{45}$$

where

$$\Delta_{l+1/2} = \frac{\Delta_{l+1} + \Delta_l}{2}, \tag{46}$$

$$\Delta_{l-1/2} = \frac{\Delta_l + \Delta_{l-1}}{2}. \tag{47}$$

In order to be conservative, the flux must be continuous across a face. Thus, the face-centered flux evaluated from either side of the face must be equivalent:

$$\begin{aligned}
&F^+_{l+1/2} = F^-_{l+1/2},\\
&2D_{l+1}\frac{\phi_{l+1} - \phi_{l+1/2}}{\Delta_{l+1}} = 2D_l\frac{\phi_{l+1/2} - \phi_l}{\Delta_l},\\
&\frac{1}{3\sigma_{l+1}}\frac{\phi_{l+1} - \phi_{l+1/2}}{\Delta_{l+1}} = \frac{1}{3\sigma_l}\frac{\phi_{l+1/2} - \phi_l}{\Delta_l},
\end{aligned} \tag{48}$$

where we have dropped the subscript R from σ for clarity and the $\pm$ superscripts indicate positive and negative directions across the face. Solving this system for $\phi_{l+1/2}$ and applying to either $F^+_{l+1/2}$ or $F^-_{l+1/2}$ and setting the resulting expression equal to Eq. (44), one finds the face-centered diffusion coefficient,

$$D_{l+1/2} = \frac{2\Delta_{l+1/2}}{3\sigma_l\Delta_l + 3\sigma_{l+1}\Delta_{l+1}}. \tag{49}$$

Following the same procedure at the $l - 1/2$ face gives the diffusion coefficient on the low-side face:

$$D_{l-1/2} = \frac{2\Delta_{l-1/2}}{3\sigma_l\Delta_l + 3\sigma_{l-1}\Delta_{l-1}}. \tag{50}$$

Using Eqs. (44), (45), (49), and (50) in the discrete flux-balance equation, (43), we obtain

$$\begin{aligned}&\sigma_{T\,ijk}\phi_{ijk} - \sigma^-_{i+1}\frac{\Delta_{i+1}}{\Delta_i}\phi_{i+1\,jk} - \sigma^+_{i-1}\frac{\Delta_{i-1}}{\Delta_i}\phi_{i-1\,jk} - \sigma^-_{j+1}\frac{\Delta_{j+1}}{\Delta_j}\phi_{i\,j+1\,k} - \sigma^+_{j-1}\frac{\Delta_{j-1}}{\Delta_j}\phi_{i\,j-1\,k}\\ &\quad - \sigma^-_{k+1}\frac{\Delta_{k+1}}{\Delta_k}\phi_{ij\,k+1} - \sigma^+_{k-1}\frac{\Delta_{k-1}}{\Delta_k}\phi_{ij\,k-1}\\ &\quad = q^n_{ijk}. \end{aligned} \tag{51}$$

Here,

$$\sigma_{T\,ijk} = \sigma^+_i + \sigma^-_i + \sigma^+_j + \sigma^-_j + \sigma^+_k + \sigma^-_k + \tilde{\sigma}^n_{ijk}, \tag{52}$$

and

$$\sigma^+_i = \frac{1}{\Delta_i}\frac{2}{3\sigma_{ijk}\Delta_i + 3\sigma_{i+1\,jk}\Delta_{i+1}}, \tag{53a}$$

$$\sigma^-_i = \frac{1}{\Delta_i}\frac{2}{3\sigma_{ijk}\Delta_i + 3\sigma_{i-1\,jk}\Delta_{i-1}}, \tag{53b}$$

$$\sigma^+_j = \frac{1}{\Delta_j}\frac{2}{3\sigma_{ijk}\Delta_j + 3\sigma_{i\,j+1\,k}\Delta_{j+1}}, \tag{53c}$$

$$\sigma^-_j = \frac{1}{\Delta_j}\frac{2}{3\sigma_{ijk}\Delta_j + 3\sigma_{i\,j-1\,k}\Delta_{j-1}}, \tag{53d}$$

$$\sigma^+_k = \frac{1}{\Delta_k}\frac{2}{3\sigma_{ijk}\Delta_k + 3\sigma_{ij\,k+1}\Delta_{k+1}}, \tag{53e}$$

$$\sigma^-_k = \frac{1}{\Delta_k}\frac{2}{3\sigma_{ijk}\Delta_k + 3\sigma_{ij\,k-1}\Delta_{k-1}}. \tag{53f}$$

Also, note that

$$\begin{aligned}\sigma^+_{i-1} &= \frac{1}{\Delta_{i-1}}\frac{2}{3\sigma_{ijk}\Delta_i + 3\sigma_{i-1\,jk}\Delta_{i-1}},\\ \sigma^-_{i+1} &= \frac{1}{\Delta_{i+1}}\frac{2}{3\sigma_{ijk}\Delta_i + 3\sigma_{i+1\,jk}\Delta_{i+1}},\\ &\ldots.\end{aligned}$$

Eqs. (51) through (53) represent the discretized, 1T diffusion equation on an orthogonal 3D grid.

Eqs. (51) through (53) have been written in a manner that is amenable to Monte Carlo interpretation. One can envision the terms in Eqs. (53) representing leakage out of each face of a cell. The off-diagonal terms then represent sources into the cell, or, equivalently, these terms are leakage out of the neighboring cells. A limitation of this interpretation is that it requires a symmetric system to implement the Monte Carlo scheme. More specifically, the operator **D** must be an M-matrix [6] such that the off-diagonal elements are negative and symmetric. In other words, the leakage out of a cell must be equivalent to its corresponding source into the adjacent cell.

The methods described in Sections 2 and 2.2 do not suffer this limitation. Thus, we will interpret the solution via MCSA as a transport-like process, but we do not require any constraints on the system other than the spectral radius must be less than 1.

Before completing the derivation of discrete equations required to solve Eq. (51), we must specify boundary conditions. For this work, the Marshak boundary condition suffices:

$$f^\pm_b = \frac{1}{4}\phi - \frac{1}{2}\mathbf{F}\cdot\hat{\mathbf{n}}, \quad \mathbf{r}\in\partial\Gamma, \tag{54}$$

where f_b is the incoming partial current on the problem boundary, $\partial\Gamma$. On any low boundary we can write the flux using a first-order discrete form of Fick's Law:

$$F_{1/2} = -\frac{2}{3\sigma_1}\frac{\phi_1 - \phi_{1/2}}{\Delta_1}. \tag{55}$$

As before, we enforce continuity of flux on the boundary using the Marshak boundary condition:

$$-\frac{2}{3\sigma_1}\frac{\phi_1 - \phi_{1/2}}{\Delta_1} = 2f^+_b - \frac{1}{2}\phi_{1/2}. \tag{56}$$

Solving for $\phi_{1/2}$ and applying to Eq. (55) yields an expression for the low-side boundary flux:

$$F_{1/2} = \frac{2}{3\sigma_1\Delta_1 + 4}\left(4f_b^+ - \phi_1\right). \tag{57}$$

Following an identical procedure on high-side problem boundaries gives

$$F_{N_l+1/2} = \frac{2}{3\sigma_{N_l}\Delta_{N_l} + 4}\left(\phi_{N_l} - 4f_b^-\right), \tag{58}$$

where $l \in \{i, j, k\}$ is defined over the range $[1, N_l]$.

These boundary fluxes take the place of the interior face-centered fluxes (defined in Eqs. (44) and (45)) in Eq. (43) yielding the following modifications to Eqs. (53) on a boundary:

$$\text{low boundary:} \quad \sigma_1^- = \frac{1}{\Delta_1}\frac{2}{3\sigma_1\Delta_1 + 4}, \tag{59}$$

$$\text{high boundary:} \quad \sigma_{N_l}^+ = \frac{1}{\Delta_{N_l}}\frac{2}{3\sigma_{N_l}\Delta_{N_l} + 4}. \tag{60}$$

Furthermore, the following contributions are added to the source, q_{ijk}^n on a boundary cell:

$$\text{low boundary:} \quad \frac{8}{\Delta_1}\frac{1}{3\sigma_1\Delta_1 + 4}f_b^+, \tag{61}$$

$$\text{high boundary:} \quad \frac{8}{\Delta_{N_l}}\frac{1}{3\sigma_{N_l}\Delta_{N_l} + 4}f_b^-. \tag{62}$$

Note that for vacuum boundary conditions, $f_b^\pm = 0$.

For reflective boundary conditions we have the simple modification that

$$F_{1/2} = 0, \tag{63}$$

$$F_{N_l+1/2} = 0. \tag{64}$$

Thus, there are no leakage or adjacent cell source terms resulting from this boundary face.

3.3. Preconditioning

The discrete diffusion equation in (51) can be written in matrix–operator form as follows:

$$\mathbf{D}\boldsymbol{\phi} = \mathbf{q}, \tag{65}$$

where $\mathbf{D}$ is a symmetric, positive-definite (SPD) operator. Unfortunately, for many problems $\rho(\mathbf{D}) > 1$. Therefore, we must precondition Eq. (65) in order to use the MCSA method. We apply left-preconditioning and write

$$\mathbf{M}^{-1}\mathbf{D}\boldsymbol{\phi} = \mathbf{M}^{-1}\mathbf{q}. \tag{66}$$

A good choice for the preconditioner, $\mathbf{M}$, is to use Jacobi preconditioning:

$$\mathbf{M} = \operatorname{diag}(\mathbf{D}). \tag{67}$$

Setting $\mathbf{M}$ equal to the diagonal elements of $\mathbf{D}$ accomplishes several objectives:

- For diagonally dominant systems like Eq. (65), it reduces the spectral radius such that $\rho(\mathbf{M}^{-1}\mathbf{D}) < \rho(\mathbf{D})$ and $\rho(\mathbf{M}^{-1}\mathbf{D}) < 1$ [9]. Thus, convergence is guaranteed and fewer iterations are required, resulting in accelerated convergence.
- The diagonal elements of the preconditioned iteration matrix are zero: $\operatorname{diag}(\mathbf{I} - \mathbf{M}^{-1}\mathbf{D}) = 0$. Thus, there will be no computations wasted for in-state transitions $i \to i$ during the random walk process.

The second item is quite important for Monte Carlo solutions of linear systems, and this type of scaling should be performed regardless of any additional preconditioner choices.

An advantage of MCSA is that the resulting system does not have to be SPD; thus, any suitable choice of $\mathbf{M}$ will suffice. The preconditioned system we will solve is defined in Eq. (66). Applying $\mathbf{M}^{-1}$ to Eq. (51) yields

$$\begin{aligned}
&\phi_{ijk} - \frac{\sigma^-_{i+1}}{\sigma_{T\,ijk}}\frac{\Delta_{i+1}}{\Delta_i}\phi_{i+1\,jk} - \frac{\sigma^+_{i-1}}{\sigma_{T\,ijk}}\frac{\Delta_{i-1}}{\Delta_i}\phi_{i-1\,jk} - \frac{\sigma^-_{j+1}}{\sigma_{T\,ijk}}\frac{\Delta_{j+1}}{\Delta_j}\phi_{i\,j+1\,k} - \frac{\sigma^+_{j-1}}{\sigma_{T\,ijk}}\frac{\Delta_{j-1}}{\Delta_j}\phi_{i\,j-1\,k} \\
&\quad - \frac{\sigma^-_{k+1}}{\sigma_{T\,ijk}}\frac{\Delta_{k+1}}{\Delta_k}\phi_{ij\,k+1} - \frac{\sigma^+_{k-1}}{\sigma_{T\,ijk}}\frac{\Delta_{k-1}}{\Delta_k}\phi_{ij\,k-1} \\
&= \frac{q^n_{ijk}}{\sigma_{T\,ijk}}. \qquad (68)
\end{aligned}$$

4. Solution of the radiation diffusion equation

Here we derive the equations required to solve Eq. (68) using the MCSA method. As discussed in Section 2.2, the MCSA method requires two solutions:

1. A Richardson iteration to estimate $\boldsymbol{\phi}^{l+1/2}$. This is a matrix–vector multiply operation.
2. An adjoint Monte Carlo solve to estimate $\delta\boldsymbol{\phi}^{l+1/2}$.

For clarity, we rewrite the iteration scheme in Eq. (20) using the diffusion operator notation from Eq. (66):

$$\begin{aligned}
&\boldsymbol{\phi}^{l+1/2} = (\mathbf{I} - \mathbf{T})\boldsymbol{\phi}^l + \mathbf{M}^{-1}\mathbf{q}, \quad \text{(Richardson iteration)} \\
&\mathbf{r}^{l+1/2} = \mathbf{M}^{-1}\mathbf{q} - \mathbf{T}\boldsymbol{\phi}^{l+1/2}, \quad \text{(compute residual)} \\
&\hat{\mathbf{T}}\delta\boldsymbol{\phi}^{l+1/2} = \mathbf{r}^{l+1/2}, \quad \text{(estimate } \delta\phi \text{ using adjoint Monte Carlo method)} \\
&\boldsymbol{\phi}^{l+1} = \boldsymbol{\phi}^{l+1/2} + \delta\boldsymbol{\phi}^{l+1/2}, \quad \text{(update } \boldsymbol{\phi}\text{)}
\end{aligned}$$

where $\mathbf{T} = \mathbf{M}^{-1}\mathbf{D}$. As described in Section 2.2, the adjoint Monte Carlo method only approximately inverts $\mathbf{T}$ as indicated by the $\hat{}$ notation. We now describe each of these steps in turn.

First, we use Eq. (68) to evaluate $\boldsymbol{\phi}^{l+1/2}$ via a single fixed-point iteration as follows:

$$\begin{aligned}
\phi^{l+1/2}_{ijk} = {} & \frac{\sigma^-_{i+1}}{\sigma_{T\,ijk}}\frac{\Delta_{i+1}}{\Delta_i}\phi^l_{i+1\,jk} + \frac{\sigma^+_{i-1}}{\sigma_{T\,ijk}}\frac{\Delta_{i-1}}{\Delta_i}\phi^l_{i-1\,jk} + \frac{\sigma^-_{j+1}}{\sigma_{T\,ijk}}\frac{\Delta_{j+1}}{\Delta_j}\phi^l_{i\,j+1\,k} + \frac{\sigma^+_{j-1}}{\sigma_{T\,ijk}}\frac{\Delta_{j-1}}{\Delta_j}\phi^l_{i\,j-1\,k} \\
& + \frac{\sigma^-_{k+1}}{\sigma_{T\,ijk}}\frac{\Delta_{k+1}}{\Delta_k}\phi^l_{ij\,k+1} + \frac{\sigma^+_{k-1}}{\sigma_{T\,ijk}}\frac{\Delta_{k-1}}{\Delta_k}\phi^l_{ij\,k-1} + \frac{q^n_{ijk}}{\sigma_{T\,ijk}}. \qquad (69)
\end{aligned}$$

This result is used to compute the residual of the system that, in turn, becomes the source for the Monte Carlo simulation.

The adjoint Monte Carlo method described in Section 2.1.2 is used to estimate

$$\hat{\mathbf{T}}\delta\boldsymbol{\phi}^{l+1/2} = \mathbf{r}^{l+1/2}. \qquad (70)$$

As noted above, the residual acts as the source for this simulation, and the operator $\mathbf{T}$ is implicitly defined in Eq. (68). There are two Monte Carlo interpretations that can be applied to this system. The first follows the mathematical presentation given in Section 2. Namely, we form probabilities from the iteration matrix and perform random walks to generate unbiased estimates of the solution vector, $\delta\boldsymbol{\phi}^{l+1/2}$, using the estimator in Eq. (13).

However, a more natural interpretation for Monte Carlo transport practitioners is to use the machinery described in Section 2 to define Probability Distribution Functions (PDF) that describe the transport of particles between cells. We then do a Monte Carlo transport calculation using Eq. (11) to calculate the weight change at each collision. Eq. (12) gives the probability of scattering to an adjacent cell. We tally the contribution of ϕ in each cell using Eq. (13). The transport is terminated using a relative weight cutoff that is defined in Eq. (14). We now examine each of these in more detail.

The connectivity of an arbitrary cell is illustrated in the 3D mesh-segment in Fig. 9. For a particle residing in the blue cell, any collision results in a scatter to one of the six adjacent cells since $p_{ii} = 0$ in the preconditioned system. We can easily calculate the probabilities for each scattering event from Eq. (68) without explicitly forming the iteration matrix. In this mesh cell, the probabilities for leakage out of each face are derived from the equations from each adjacent cell. Using Eq. (68) we can explicitly write the equations from each adjacent cell, here only noting the contribution from cell 7:

$$\begin{aligned}
\phi_1 &= \frac{\sigma^-_{7\to 1}}{\sigma_{T\,1}}\frac{\Delta_7}{\Delta_1}\phi_7 + \cdots, \\
\phi_2 &= \frac{\sigma^+_{7\to 2}}{\sigma_{T\,2}}\frac{\Delta_7}{\Delta_2}\phi_7 + \cdots, \\
\phi_3 &= \frac{\sigma^-_{7\to 3}}{\sigma_{T\,3}}\frac{\Delta_7}{\Delta_3}\phi_7 + \cdots,
\end{aligned}$$

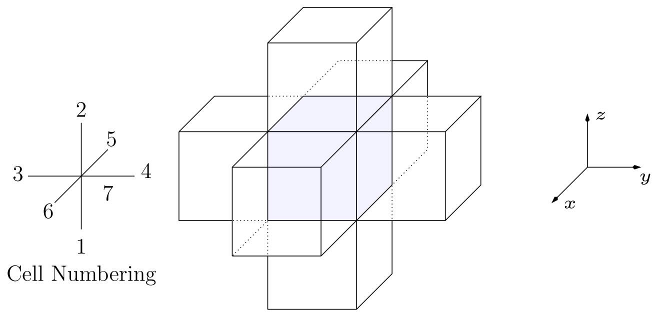

Fig. 9. 3D mesh-segment. The particle's current cell (shaded in blue) is connected to 6 adjacent cells. The cell numbers are plotted alongside the mesh such that the blue cell has index 7, the high z cell has index 2, etc. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$\phi_4 = \frac{\sigma^+_{7\to 4}}{\sigma_{T\,4}} \frac{\Delta_7}{\Delta_4} \phi_7 + \cdots,$$
$$\phi_5 = \frac{\sigma^-_{7\to 5}}{\sigma_{T\,5}} \frac{\Delta_7}{\Delta_5} \phi_7 + \cdots,$$
$$\phi_6 = \frac{\sigma^+_{7\to 6}}{\sigma_{T\,6}} \frac{\Delta_7}{\Delta_6} \phi_7 + \cdots.$$

A quick view of these equations reveals that the only non-zero entries in column 7 of the iteration matrix are

$$(h_{1,7} \quad h_{2,7} \quad h_{3,7} \quad h_{4,7} \quad h_{5,7} \quad h_{6,7})^T = \begin{pmatrix} \frac{\sigma^-_{7\to 1}}{\sigma_{T\,1}} \frac{\Delta_7}{\Delta_1} \\ \frac{\sigma^+_{7\to 2}}{\sigma_{T\,2}} \frac{\Delta_7}{\Delta_2} \\ \frac{\sigma^-_{7\to 3}}{\sigma_{T\,3}} \frac{\Delta_7}{\Delta_3} \\ \frac{\sigma^+_{7\to 4}}{\sigma_{T\,4}} \frac{\Delta_7}{\Delta_4} \\ \frac{\sigma^-_{7\to 5}}{\sigma_{T\,5}} \frac{\Delta_7}{\Delta_5} \\ \frac{\sigma^+_{7\to 6}}{\sigma_{T\,6}} \frac{\Delta_7}{\Delta_6} \end{pmatrix},$$

where, following Eq. (2), $\mathbf{H} = (\mathbf{I} - \mathbf{T})$ is the iteration matrix. Now, using Eq. (12), we define the scattering probabilities for cell 7 as

$$P_{7\to n} = p_{7,n} = \frac{|h_{n,7}|}{|h_{1,7} + h_{2,7} + h_{3,7} + h_{4,7} + h_{5,7} + h_{6,7}|}, \quad n \in \{1, \ldots, 6\}.$$

The weight change for each scattering event is calculated using Eq. (11):

$$W_m = \frac{h_{n,7}}{p_{7,n}} W_{m-1}, \quad n \in \{1, \ldots, 6\},$$

where m is the step index in the particle history. We also note that, for the 3D Cartesian mesh considered in this work, the maximum number of non-zero entries in columns of the iteration matrix is 6, generating a sparse system. Cells along the problem boundary can have fewer than six entries.

5. Results

Here we examine the effectiveness of the MCSA method and compare it to traditional deterministic solution methods. We implemented Krylov subspace methods to solve the radiation diffusion equation using the Trilinos solver library [10]. Conjugate Gradient (CG) is the standard method for symmetric positive-definite (SPD) systems like that defined in Eq. (51). However, for completeness we include some comparisons using restarted GMRES(m) so that asymmetric preconditioners can be tested. The stopping criterion in Eq. (21) is used for all calculations. All methods are tested on two 3D problems: a uniform-medium Marshak wave problem and a multi-material duct problem.

5.1. Marshak wave

The Marshak wave problem under consideration in this study has an incoming 1 keV blackbody boundary flux at $x = 0$ and propagates in the x-dimension in time as outlined in Ref. [11]. In other words, this is a 3D representation of a 1D problem. The full set of problem conditions is as follows:

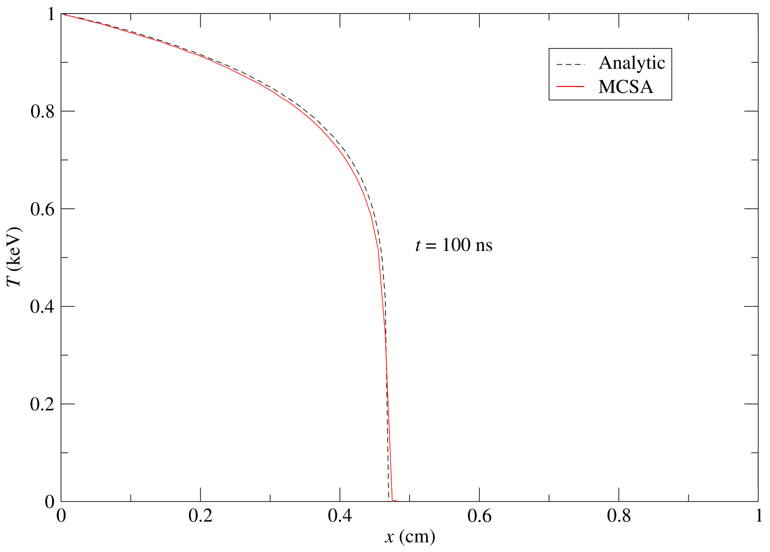

Fig. 10. Solution of the Marshak wave problem at $t = 100$ ns. The analytic solution is compared to the MCSA solution. All methods are converged to the same tolerance and therefore give numerically equivalent answers.

B.C.	$f_b^+(x=0, T=1.0) = \frac{ac}{4}T^4$
	$\mathbf{F} = 0 \quad \partial\Gamma \in \{x^+, y^-, y^+, z^-, z^+\}$
material properties	$\rho = 3.0\ \mathrm{g\,cm^{-3}}$
	$C_v = 0.1\ \mathrm{GJ\,g^{-1}\,keV^{-1}}$
	$T(t=0) = 1 \times 10^{-4}$ keV
opacity	$\sigma = 100T^{-3}\ \mathrm{cm^2\,g^{-1}}$
mesh	$\Delta_i = \Delta_j = \Delta_k = 0.01$ cm
	$N_i = 100 \quad N_j = N_k = 4$

The problem is run to 100 nanoseconds. The variable timestep control started with 1.25×10^{-9} ns and increased to 0.01 ns using a maximum change in Δt of 10% per cycle. This maximum timestep was selected based on numerical experiments that showed good agreement with analytic solutions of the Marshak wave problem for the time-discretization employed in Eq. (35).

Fig. 10 shows the analytic Marshak wave solution compared to the MCSA solution. As opposed to most Monte Carlo techniques, the MCSA method generates solutions that are numerically equivalent to the deterministic results. Because the MCSA method converges the residual of the system, the numerical accuracy is determined by the stopping criterion shown in Eq. (21). Also, the MCSA method is not bound by the Central Limit Theorem so convergence is not restricted to $1/\sqrt{N}$ [3,1].

As stated in Section 2, the MCSA method will converge when $\rho(\mathbf{H}) < 1$, where $\mathbf{H}$ is the iteration matrix and is defined $\mathbf{H} = (\mathbf{I} - \mathbf{T})$. The spectral radius for the Marshak problem is plotted in Fig. 11. As described in Section 3.3, the preconditioning effectively reduces the spectral radius of $\mathbf{H}$ to less than 1.

There are two principal parameters when utilizing the MCSA method: (1) the number of particles per iteration and (2) the weight cutoff. Fig. 12 shows the CPU time as a function of the requested number of particles per iteration for a relative weight cutoff of 1×10^{-4}. For fewer than five particles per iteration, the method does not converge. Fig. 13 shows the variation in the maximum number of iterations per cycle with number of particles. Analysis of these two figures shows that increasing the number of particles per iteration reduces the number of iterations required to converge the solution. However, the cost of the Monte Carlo transport is high; thus, better performance is attained by running more iterations with fewer particles per iteration [1].

Fig. 14 shows the CPU time as a function of relative weight cutoff. These results indicate that the performance of the MCSA method is relatively insensitive to the weight cutoff. This is especially true when running small numbers of particles per iteration. The weight cutoff has a more pronounced effect on the efficiency of the method when larger numbers of particles are used.

Having analyzed the variation of the MCSA algorithm efficiency with number of particles and weight cutoff, we now compare to traditional deterministic algorithms. Table 1 shows timing results for the Marshak problem using CG with Jacobi and algebraic multigrid preconditioning and GMRES with incomplete LU (ILU) and ILU with threshold cutoff (ILUT). The Krylov methods and preconditioners are implemented through the Trilinos AztecOO package. The multigrid preconditioner is the smoothed aggregation algebraic multigrid method implemented in the Trilinos ML package [12]. In all the problems that follow we use a 4/3 damping factor in the aggregation and a symmetric Gauss–Seidel smoother with 2 sweeps per level, which are the defaults for the ML smoothed aggregation option.

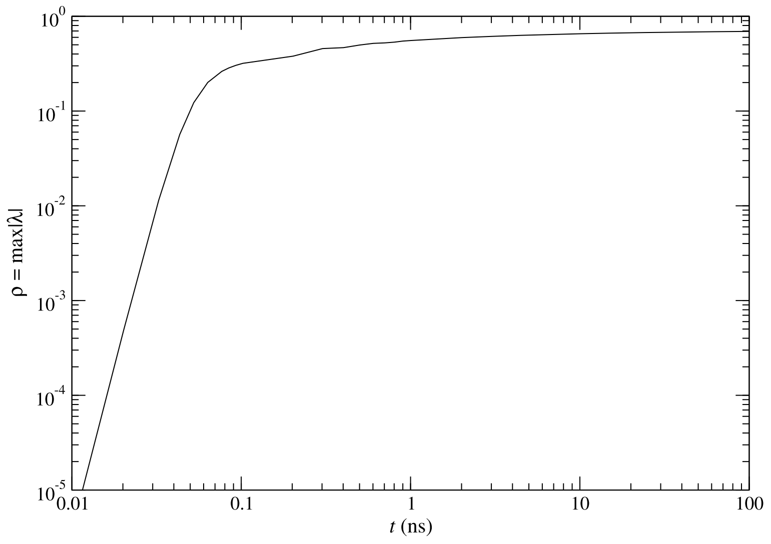

Fig. 11. Spectral radius of the iteration matrix, **H**, for the Marshak wave problem as a function of elapsed problem time.

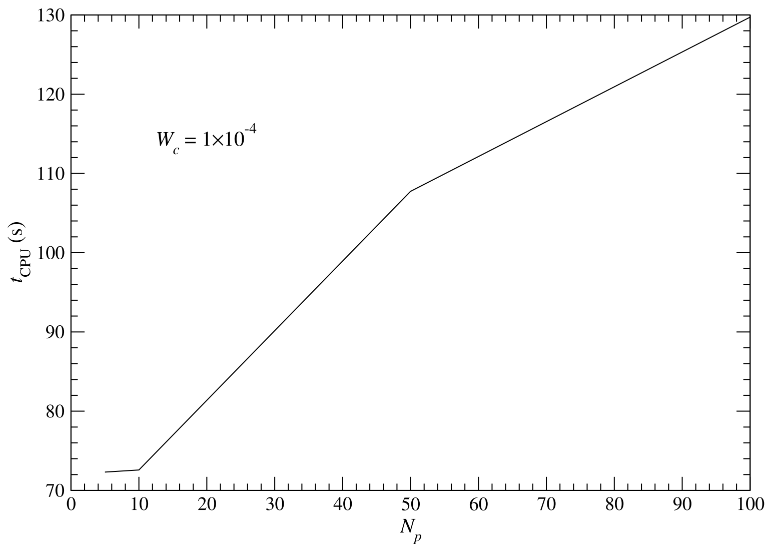

Fig. 12. CPU time versus number of particles per iteration for the Marshak problem. The number of particles is the requested number. In each iteration the exact number can vary depending upon the number of cells and source strength per cell.

Analyzing Table 1, we see that the overall iterations are more than a factor of 2 less for the Krylov solvers compared to MCSA; however, the cost per iteration is significantly more expensive. For any given problem, the cost per iteration is roughly linear with particle count as illustrated in Fig. 12. Because this problem converges running only a maximum of 10 particles per iteration, 156 464 total particles over 10 157 cycles resulting in an average of ∼1.6 particles per iteration, the cost per iteration is exceptionally small. Running more particles per iteration would reduce the number of iterations, but at a greater cost per iteration.

5.2. Multi-material problem

We now test the performance of the MCSA method on a large multi-material problem. The multi-material problem has a dog-legged duct through a thick wall where the radiation flows into a thin region bound by a foil on one side. The mesh and geometry are illustrated in Fig. 15. A blackbody boundary flux is defined on the low-x face at $T = 0.5$ keV. The full set of problem conditions is as follows:

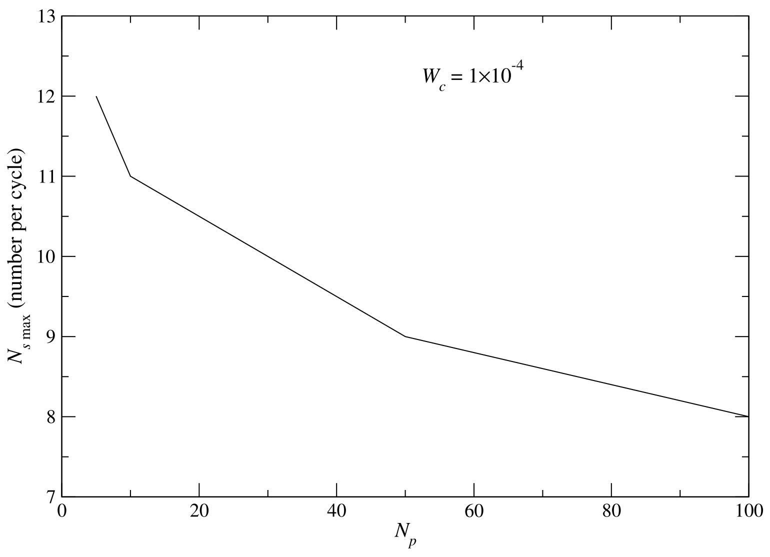

Fig. 13. Maximum number of iterations per cycle versus number of particles per iteration for the Marshak problem. The number of particles is the requested number. In each iteration the exact number can vary depending upon the number of cells and source strength per cell.

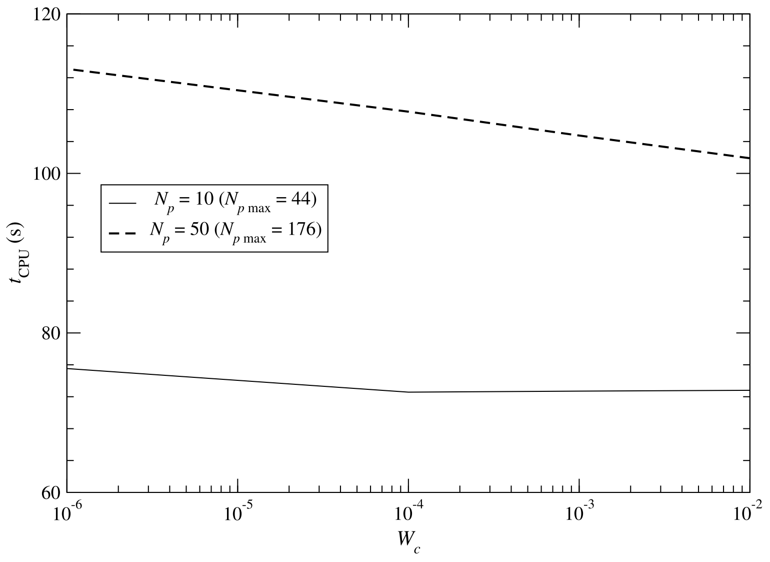

Fig. 14. CPU time versus relative weight cutoff for the Marshak problem. Results are shown for 10 and 50 requested particles per iteration. The numbers in parentheses show the maximum number of particles run in an iteration during the course of the problem.

B.C.	$f_b^+(x=0, T=0.5) = \frac{ac}{4}T^4$
	$\mathbf{F}=0 \quad \partial\Gamma \in \{x^+, y^-, y^+, z^-, z^+\}$
material properties	$\rho = 1.5\ \mathrm{g\,cm^{-3}}$ (duct)
	$\rho = 8.0\ \mathrm{g\,cm^{-3}}$ (wall)
	$\rho = 4.0\ \mathrm{g\,cm^{-3}}$ (foil)
	$C_v = 0.1\ \mathrm{GJ\,g^{-1}\,keV^{-1}}$
	$T(t=0) = 1\times10^{-4}$ keV
opacity	$\sigma = T^{-3}\ \mathrm{cm^2\,g^{-1}}$ (duct)
	$\sigma = 1000T^{-1}\ \mathrm{cm^2\,g^{-1}}$ (wall)
	$\sigma = 5T^{-2}\ \mathrm{cm^2\,g^{-1}}$ (foil)
mesh	$\Delta_i = \Delta_j = \Delta_k = 0.025$ cm
	$N_i = N_j = N_k = 60$

Table 1
Comparison of solution methods for the Marshak problem. All methods were run with a stopping criterion of 1×10^{-8}. The MCSA method used $N_p = 10$ and $W_c = 1 \times 10^{-4}$. Runtimes are relative to the MCSA performance that required 57.72 seconds on a 2.2 MHz AMD Opteron.

Method	Preconditioner	Total iterations	Average iterations	Relative time per iteration	Relative runtime
CG	Jacobi	46 746	4.60	25.91	3.29
CG	ML	12 150	1.20	74.72	2.69
GMRES	ILU	40 294	3.97	103.02	9.19
GMRES	ILUT[†]	40 447	3.98	101.31	9.09
MCSA	Jacobi	98 863	9.73	1.0	1.0

[†] The threshold cutoff was set to 0.01, and the fill was 1.

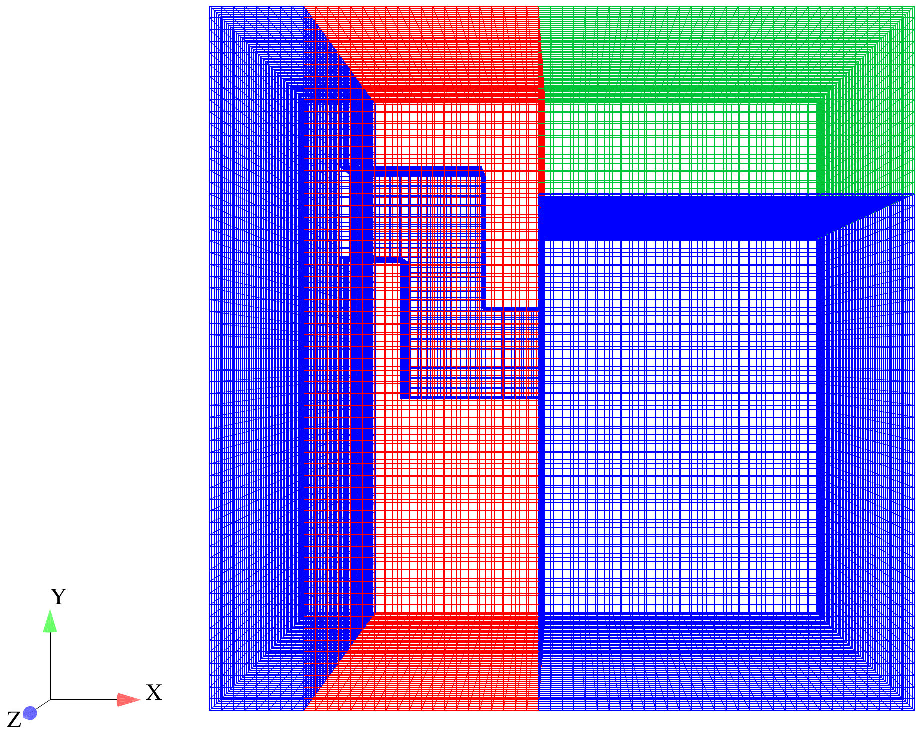

Fig. 15. Multi-material problem mesh. The duct region is blue. The wall is red. The foil is green. The mesh is $60 \times 60 \times 60$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The timestep controls from the Marshak problem were used here resulting in a maximum timestep of 0.01 ns.

A 2D plot of the temperature at 1000 ns elapsed time is illustrated in Fig. 16. The time evolution of the temperature is shown at four edit points in Fig. 17. All three methods give numerically identical answers when using a stopping criterion of 1×10^{-8}.

Table 2 shows the timing results for the multi-material problem using CG with Jacobi and ML preconditioning versus MCSA run out to a time of 100 ns. The MCSA method was set to use 1000 particles based on numerical experiments similar to those applied for the Marshak problem. A value of 1000 particles per iteration yields 74 237 525 total particles over 10.157 cycles or 823.68 particles per iteration. Because more particles are required to get a reasonable estimate of the Neumann series, the relative efficiency per iteration of MCSA versus CG is significantly reduced compared to the smaller Marshak wave problem. Nonetheless, MCSA is still marginally more efficient than CG with Jacobi preconditioning and significantly more efficient than CG with algebraic multigrid. More work is required to investigate automatic techniques for optimizing particle count versus number of iterations with regards to both efficiency and convergence. This will be the topic of a future paper.

6. Conclusions

We have presented a new Monte Carlo solution method for solving the discrete, time-dependent radiation diffusion equation in three dimensions. The MCSA method has been shown to match results using standard solution techniques to arbitrary precision. Also, the new method has been to shown to be faster than preconditioned CG and GMRES for the model problems presented here.

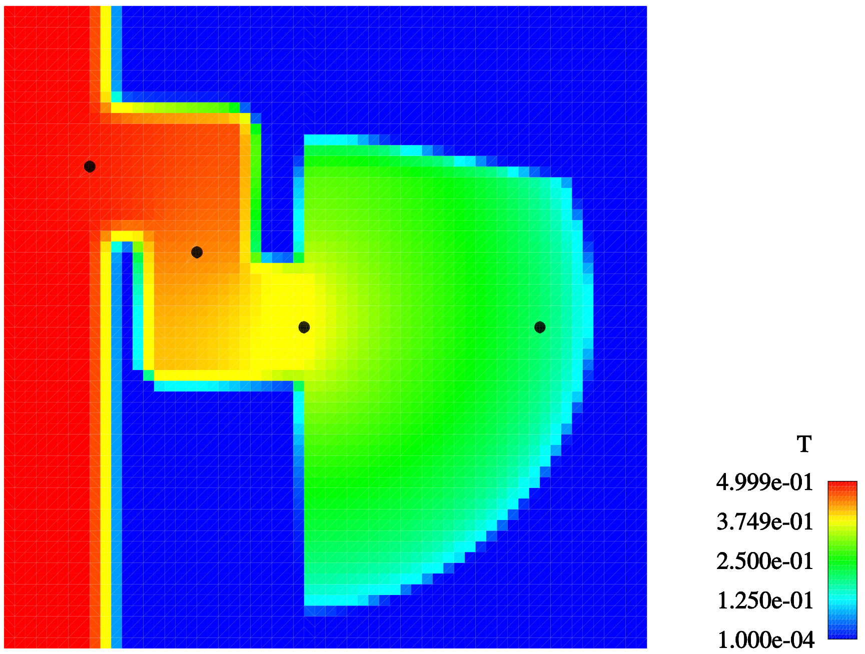

Fig. 16. Two-dimensional plot of the temperature in keV at 1000 ns on a cut-plane positioned at the mid-point of the z-axis. The positions of the edit points where the time evolution of the temperature is plotted in Fig. 17 are indicated by the black disks.

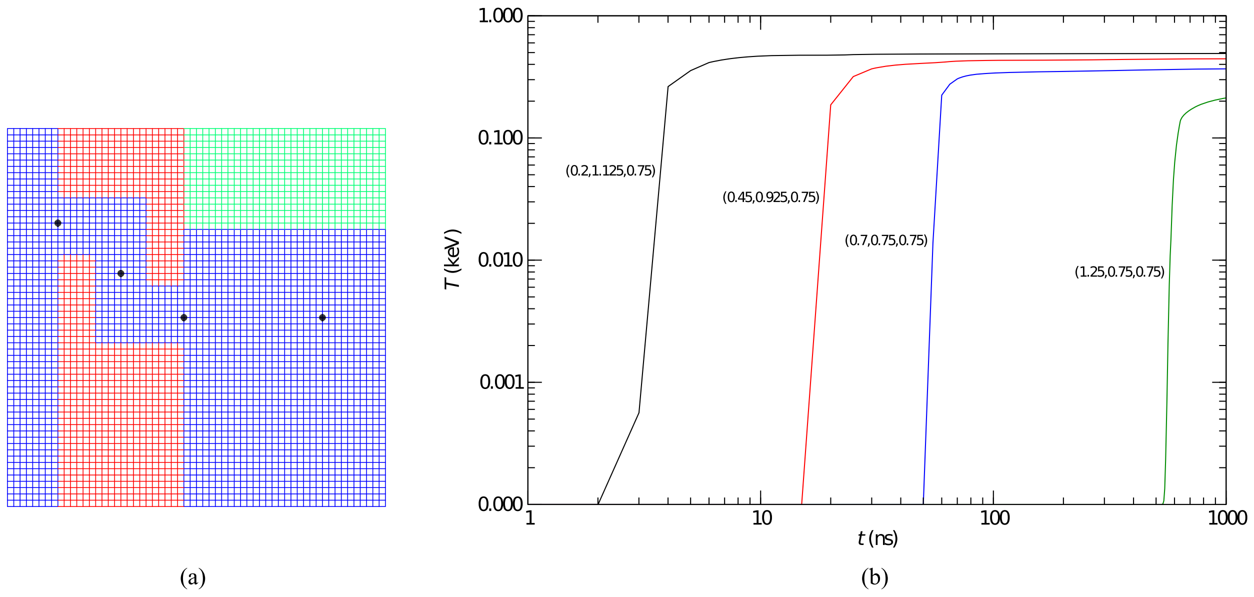

Fig. 17. (a) Edit points and (b) time evolution of the temperature at each point. All points are centered in the z-axis.

Table 2
Comparison of solution methods for the multi-material problem. All methods were run with a stopping criterion of 1×10^{-8}. The MCSA method used $N_p = 1000$ and $W_c = 1 \times 10^{-3}$. The problem was run to an elapsed time of 100 ns. Runtimes are relative to the MCSA performance that required 9862.4 seconds on a 2.2 MHz AMD Opteron.

Method	Preconditioner	Total iterations	Average iterations	Relative time per iteration	Relative runtime
CG	Jacobi	42 881	4.22	2.93	1.02
CG	ML	14 097	1.39	26.60	1.94
MCSA	Jacobi	90 129	8.87	1.0	1.0

While we have demonstrated marginal improvements over standard solution schemes in this study, significant improvements could be realized in fully nonlinear-consistent solutions. In these cases, the MCSA method competes with GMRES, which is more costly in memory and time than CG. Another area where the MCSA scheme may have advantages over traditional solution methods is on dendritic, or adaptive, meshes. These meshes yield matrices with poor condition numbers because of the changing cell volumes at different refinement levels. A smart transport algorithm could be developed that more efficiently solves the residual on these types of meshes. These topics will be the focus of future study.

Finally, exascale hardware will require resiliency to both hard and soft errors. Stochastic solver approaches present a natural framework for both soft and hard errors. In the former case, soft errors can be presented as high variance events. For hard errors, histories from failed nodes can be safely discarded in the final calculation. The MCSA method provides a starting framework towards analyzing both of these failure conditions.

Acknowledgements

Work for this paper was supported by Oak Ridge National Laboratory, which is managed and operated by UT-Battelle, LLC, for the U.S. Department of Energy under Contract No. DEAC05-00OR22725. The early part of this work was performed at Los Alamos National Laboratory under U.S. Government contract W-7405-ENG-36.

References

[1] T. Evans, T. Urbatsch, H. Lichtenstein, J. Morel, A residual Monte Carlo method for discrete thermal radiative diffusion, J. Comput. Phys. 189 (2003) 539–556.
[2] J. Halton, Sequential Monte Carlo, Proc. Camb. Philos. Soc. 58 (1962) 57–78.
[3] J. Halton, Sequential Monte Carlo techniques for the solution of linear systems, J. Sci. Comput. 9 (2) (1994) 213–257.
[4] G. Forsythe, R. Leibler, Matrix inversion by a Monte Carlo method, Math. Tables Other Aids Comput. 4 (1950) 127–129.
[5] J. Hammersley, D. Handscomb, Monte Carlo Methods, Spottiswoode, Ballantyne, and Co., 1964.
[6] C. Kelley, Iterative Methods for Linear and Nonlinear Equations, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1995.
[7] M. Adams, W. Martin, Diffusion-synthetic acceleration of discontinuous finite-element transport iterations, Nucl. Sci. Eng. 111 (2013), in press.
[8] J. Morel, T.A. Wareing, K. Smith, A linear-discontinuous spatial differencing scheme for S_n radiative transfer calculations, J. Comput. Phys. 128 (1996) 445–462.
[9] G. Golub, C. van Loan, Matrix Computations, 3rd edition, The Johns Hopkins University Press, 1996.
[10] M.A. Heroux, R.A. Bartlett, V.E. Howle, R.J. Hoekstra, J.J. Hu, T.G. Kolda, R.B. Lehoucq, K.R. Long, R.P. Pawlowski, E.T. Phipps, A.G. Salinger, H.K. Thornquist, R.S. Tuminaro, J.M. Willenbring, A. Williams, K.S. Stanley, An overview of the Trilinos project, ACM Trans. Math. Softw. 31 (3) (2005) 397–423, http://dx.doi.org/10.1145/1089014.1089021.
[11] E.W. Larsen, G. Pomraning, Asymptotic analysis of nonlinear Marshak waves, SIAM J. Appl. Math. 39 (2) (1980) 201–212.
[12] M.W. Gee, C.M. Siefert, J.J. Hu, R.S. Tuminaro, M.G. Sala, ML 5.0 smoothed aggregation user's guide, Tech. Rep. SAND2006-2649, 2007.