AN ABSTRACT OF THE THESIS OF

Bradley R. Eccleston for the degree of Master of Science in Nuclear Engineering presented on July 13, 2000.

Title: An Inherently Parallel Monte Carlo Linear Systems Solver Applied to Neutron Diffusion Equations.

Redacted for privacy

Abstract approved:

Todd S. Palmer

Linear solvers are often used to solve neutron diffusion problems. These tools have two significant shortcomings. First, parallel implementations provide only a modest speedup. The operations cannot be divided cleanly between processors. Second, for large matrices they can be very slow. Our primary goal is to find a new method for solving linear systems which reduces the impact of these two problems.

In this study, we consider a different kind of approach. We employ a Monte Carlo algorithm in two dimensions to solve our linear systems probabilistically. We develop our probabilistic model and describe the formulation of our linear system. We also discuss our random sampling technique in some detail. We tally our solutions for both the forward and adjoint problems using path length and last event estimators.

Computational results are compared to analytic and numerical benchmark solutions for three metrics: accuracy, convergence, and efficiency. The results detailed herein indicate that the method we have developed can be competitive with common linear solvers.

We develop an on-the-fly algorithm as well, which is intended to make more efficient use of our computing resources. While this algorithm exhibits longer run-times, it is far less taxing on the system memory.

An Inherently Parallel Monte Carlo Linear Systems Solver Applied to Neutron Diffusion Equations

by

Bradley R. Eccleston

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented July 13, 2000
Commencement June, 2001

Master of Science thesis of Bradley R. Eccleston presented on July 13, 2000.

APPROVED:

Redacted for privacy

Major Professor, representing Nuclear Engineering

Redacted for privacy

Chair of Department of Nuclear Engineering

Redacted for privacy

Dean of Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for privacy

Bradley R. Eccleston, Author

ACKNOWLEDGEMENTS

I want first to thank Dr. Todd S. Palmer for his guidance and wisdom. He offered me excellent opportunities and made my experience at Oregon State University an enriching one. I would also like to thank the other members of my masters committee, Dr. Jose Reyes, Dr. Rubin Landau, and Dr. Solomon Yim. The faculty of the Oregon State University Department of Nuclear Engineering deserve my thanks as well. Their enthusiasm and expertise have greatly enhanced my life, providing me with more than an education, but also an appreciation for the technical and social challenges of this science.

A number of people contributed to this work. Many of these individuals are involved in their own research at Lawrence Livermore National Laboratory. They include: Nick Gentile, Jim Rathkopf, Mike Zika, and Pete Eltgroth. Todd Urbatsch and Forrest Brown of Los Alamos National Laboratory were also very helpful. In spite of their own responsibilities all of these people found time to share their knowledge with me and I humbly thank them.

The members of my research group have had a substantial impact on me. By virtue of timing and luck I found myself in the frequent company of John Gulick, Brenton Ching, and Kang-Seog Kim. I have as much to thank these gentlemen for as I do practically anyone. I would also like to acknowledge Lawrence Livermore National Laboratory, the Department of Energy, the American Nuclear Society, and the Oregon State University Department of Nuclear Engineering for their financial support.

My family is the single most important thing in my life. The value that I place on my wife Vicki and daughter Katilyn cannot be put into words. I know that they have struggled with these years as much as I have and it is that struggle that has helped me to maintain my focus. My father and stepmother Richard and Sharon Eccleston

have been staunch supporters of my efforts. Always quick with an inspiring word or pat on the back, they have helped me to keep things in perspective. I also thank Jeff and Staci, Milissa, Ed and Holly, Ruth, Ron and Sally, and their $11\frac{1}{2}$ collective children for reminding me what is really important.

This work was performed in part under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract #344348.

TABLE OF CONTENTS

Page

1 INTRODUCTION 1

2 METHODS 5

2.1 Monte Carlo Linear Solver 5

2.1.1 Probablistic Interpretation of the Matrix 5
2.1.2 Random Walk 8
2.1.3 Forward vs. Adjoint 10
2.1.4 Estimators 10
2.1.5 Constraints 13

2.2 Application to Discretized Diffusion Equations 14

2.2.1 Derivation of Equations 14
2.2.2 Algorithm 18

2.3 Parallelization 20

2.3.1 Message-Passing Interface 20
2.3.2 Machine Architecture 20

3 RESULTS 21

3.1 Accuracy 21

3.1.1 Test Problem 1: Reactor Fuel Assemblies 25
3.1.2 Test Problem 2: Source/Shield Assembly 26
3.1.3 Test Problem 3: Relative Error 29

3.2 Convergence 29

3.3 Efficiency 34

3.3.1 Test Problem 1 35
3.3.2 Test Problem 2 36
3.3.3 Test Problem 3 37

TABLE OF CONTENTS (Continued)

Page

3.3.4 Test Problem 4 37

3.4 On-the-Fly versus Standard Monte Carlo 39

4 CONCLUSION 42

4.1 Summary of Results 42

4.2 Discussion 43

4.3 Suggestions for Future Work 45

BIBLIOGRAPHY 46

APPENDIX A 47

LIST OF FIGURES

Figure		Page
1	Association of the probability matrix with the physical problem.	7
2	Conversion of PDF to Aliased PDF.	9
3	The adjoint path length estimator.	11
4	The adjoint last event estimator.	12
5	The forward path length estimator.	12
6	The forward last event estimator.	13
7	Representative control volumes.	14
8	Monte Carlo: Infinite, Homogeneous Medium, Adjoint Estimators.	22
9	Monte Carlo: Infinite, Homogeneous Medium, Forward Estimators.	22
10	PETSc: Infinite, Homogeneous Medium.	23
11	Monte Carlo: Source-Free, Pure Absorber, Adjoint Estimators.	24
12	Monte Carlo: Source-Free, Pure Absorber, Forward Estimators.	24
13	PETSc: Source-Free, Pure Absorber.	25
14	Test Problem 1: Monte Carlo, Adjoint Estimator.	26
15	Test Problem 1: Monte Carlo, Forward Estimator.	27
16	Test Problem 1: PETSc solution.	27
17	Test Problem 2: Monte Carlo, Adjoint Estimator.	28
18	Test Problem 2: Monte Carlo, Forward Estimator.	28
19	Test Problem 2: PETSc solution.	28
20	L_2 norm of the Relative Error, Adjoint Estimators.	30
21	L_2 norm of the Relative Error, Forward Estimators.	30

LIST OF FIGURES (Continued)

Figure		Page
22	Standard Deviation, Adjoint Last Event Estimator.	31
23	Standard Deviation, Adjoint Path Length Estimator.	32
24	Standard Deviation, Forward Last Event Estimator.	32
25	Standard Deviation, Forward Path Length Estimator.	33
26	Average Standard Deviation, Both Adjoint Estimators.	33
27	Average Standard Deviation, Both Forward Estimators.	34
28	Efficiency Test Problem 1 Speedup: Scattering ratio $c = 0.9$.	35
29	Efficiency Test Problem 2 Speedup: Scattering ratio $c = 0.5$.	36
30	Efficiency Test Problem 3 Speedup: Scattering ratio $c = 0.1$.	37
31	Efficiency Test Problem 4: Layout of Broad Range Problem.	38
32	Efficiency Test Problem 4 Speedup: Broad Range Problem.	38
33	CPU Time: On-the-Fly Monte Carlo vs. Standard Monte Carlo. . . .	40
34	Memory Load: On-the-Fly Monte Carlo vs. Standard Monte Carlo. .	41

LIST OF TABLES

Table		Page
1	Side node multipliers	18
2	Corner node multipliers	19

In memory of my mother, Marian Eccleston

AN INHERENTLY PARALLEL MONTE CARLO LINEAR SYSTEMS SOLVER APPLIED TO NEUTRON DIFFUSION EQUATIONS

1 INTRODUCTION

The past several decades have seen substantial gains in computer performance. Improvements have been made in CPU speed, memory size, and architecture. Numerical simulation has been greatly enhanced by these advances. Many problems considered too complicated or time consuming in the past are now being solved routinely. A common type of tool used to solve many problems is a linear solver. For large matrices, typical linear solvers can be inefficient. This investigation considers a technique developed by Hammersley and Handscomb [HaHa 64] to solve linear systems. The essence of the scheme is to interpret the coefficient matrix probablistically and find the solution to the system using the Monte Carlo technique. There are a number of potential advantages to a Monte Carlo approach:

- Analog Monte Carlo techniques are inherently parallel; this is not necessarily true of today's more advanced linear equation solvers (multigrid, conjugate gradient, etc.).

- Some forms of this technique are adaptive, in that they allow the specification of locations in the problem where resolution is of particular importance and the concentration of work at those locations.

- These techniques permit the solution of very large systems of equations, in that matrix elements need not be stored. Calculational speed can be traded for storage if elements of the matrix are calculated "on-the-fly".

There is a substantial disadvantage to Monte Carlo. The premise behind Monte Carlo is to simulate the effect of the matrix by randomly sampling its effect on the

solution. This means that the method is subject to sampling statistics. To obtain a more accurate solution, more sampling is required. Thus, in Monte Carlo algorithms, accuracy is typically obtained at the cost of increased CPU time. However, with increased computer performance, specifically massive parallelism, there may be regimes in which Monte Carlo is a more desirable approach than more common linear solver techniques.

A great deal of work has been performed using Monte Carlo to solve particle physics problems. The preponderance of the effort in recent years has been in neutron, electron, and photon transport. Monte Carlo has been shown to be an effective tool [Jen 88] for this type of application. This is made apparent by its wide use in production tools such as MCNP, SCALE, and EGS4. Moreover, implementation of Monte Carlo particle transport algorithms on parallel computers has become a topic of considerable research [Jen 88, Mar 91, Mar 92]. Such investigations have resulted in significant speedup on both vector and scalar parallel machines.

Much has been published on the topic of parallel computing, and a review of Martin's papers [Mar 91, Mar 92] can shed some light on the benefits of running Monte Carlo problems on various machine configurations and hardware architectures. Select studies [Del 85, McK 88, Fin 88] have been performed specifically for the solution of neutron diffusion problems on parallel computing systems. The Delves paper [Del 85] discusses implementation on the ICL DAP. He shows speedup ratio values ranging from 2.5:1 over serial implementations up to 18.5:1 depending on the parallelization strategy. An extension of that work is the McKerrell and Delves paper [McK 88] where a speedup ratio of 6.3:1 was achieved. Both of these papers considered multiplying mediums to simulate reactor problems. This type of source term is not considered in our study. A third paper, by Finnemann et al. [Fin 88], considers multigrid techniques rather than Monte Carlo and solves non-linear equations. The relevance of this paper is the high degree of speedup achieved by their algorithm. They reported

speedup ratios for several different numbers of processors up to 24 and published speedup ratios that tracked with the number of processors at 80% of linear and up. For instance, 12 processors resulted in a speedup ratio of 10.7:1 or (10.7/12)*100% = 89% of linear. This degree of speedup or better, is a goal of our study.

We have developed algorithms that use the Monte Carlo technique to solve the one-group, discretized, fixed-source diffusion equation for one-dimensional and orthogonal, two-dimensional grids. In this paper, we compare the parallel performance of our Monte Carlo algorithm to that of a more traditional parallelized linear solver. The metrics of comparison are accuracy, convergence, and efficiency.

The remainder of this paper is organized in three sections. In Section 2 we introduce the three major concepts that are used in this study: Monte Carlo solution of linear systems, discretized diffusion equtions, and algorithm parallelization. We examine Monte Carlo as it applies to the probablistic interpretation of the matrix form of our diffusion operator. We survey sampling in some detail and discuss our chosen sampling technique. Random walks and solution estimators are discussed to establish a working knowledge of how the solution is assembled. We then look at how the forward and adjoint problems compare in our formulation. Last, we consider the constraints of using our technique. Our discussion of diffusion begins with the equation discretization and matrix formulation. That is followed by a brief description of the algorithm flow. This section concludes with details of our parallelization strategy and includes information about the software required and the hardware architectures used.

We begin the Results section by presenting data concerning the accuracy of the Monte Carlo technique compared to a commonly used parallel linear solver for a number of benchmark problems. Convergence of the method is then quantified to show that the statistical error decreases as the square root of the number of histories. Efficiency is analyzed for both scalar and parallel tests and for variations in the physics

of the test problems. Finally, the timing and load of the Monte Carlo algorithm is compared to a modified version which calculates matrix elements on-the-fly instead of precalculating and storing the matrix.

In the last section, we summarize and discuss our conclusions, and possible areas of future work.

2 METHODS

In this section we will discuss the three major concepts that are used in this study: Monte Carlo solution of linear systems, discretized diffusion equtions, and algorithm parallelization. Our examination of Monte Carlo in section 2.1 includes an explanation of how we interpret the diffusion operator matrix **A** probabilistically. Random walks and solution estimators are discussed to establish a working knowledge of how the solution is assembled and our sampling technique is examined. Next, we consider the similarities and differences in the forward and adjoint problems. Last, we identify the constraints of using our technique.

Our discussion of diffusion in section 2.2 begins with the equation discretization and matrix formulation for the interior and the boundaries including our boundary conditions. That is followed by a brief description of the algorithm flow.

We conclude our discussion of the methods used in section 2.3 with details of our parallelization strategy and includes information about the software required and the hardware architectures used.

2.1 Monte Carlo Linear Solver

2.1.1 Probablistic Interpretation of the Matrix

Our Monte Carlo solution technique requires a modified form of the matrix. A typical linear system may be written in the matrix form as $\mathbf{A}x = b$ where x is the solution vector, b is the source vector, and **A** is an N×N matrix.

If we decompose **A** into its diagonal and off-diagonal elements, such that

$$\mathbf{A} = \mathbf{D} - \mathbf{OD} \tag{1}$$

and define

$$\mathbf{H} = \mathbf{D}^{-1}\mathbf{O}\mathbf{D} \quad \text{and} \quad s = \mathbf{D}^{-1}b \tag{2}$$

we may rewrite (1) as

$$x = s + \mathbf{H}x. \tag{3}$$

It is with this diagonally scaled form that we apply the technique introduced by Hammersley and Handscomb [HaHa 64]. The matrix **H** contains the elements of the matrix **A** normalized in a row-wise manner by the diagonal term. The off-diagonal elements can then be interpreted as directional transition or "jump" probabilities as illustrated in Figure 1. The figure shows a properly dimensioned **H** matrix for the physical problem and the physical problem itself. The diagonal of the matrix is included for point of reference only as the **H** matrix contains zero-valued diagonal terms. All terms not accounted for by the geometric symbols are zero. The values of these non-zero terms are the jump probabilities corresponding to the symbols in the physical problem. We need only define a termination probability (4)

$$p_t = 1 - \sum_i H_{i,j} \tag{4}$$

to completely describe the probable actions of a pseudo-particle through a random process.

Hammersley & Handscomb [HaHa 64] observed that for the adjoint last event estimator, the solution for an element i of the solution vector may be written (5) as an infinite sum.

$$x_i = s_i + \sum_{j=1}^{n} h_{i,j}s_j + \sum_{j=1}^{n}\left(\sum_{k=1}^{n} h_{i,k}h_{k,j}\right)s_j + \sum_{j=1}^{n}\left(\sum_{l=1}^{n} h_{i,l}, \sum_{k=1}^{n} h_{i,k}h_{k,j}\right)s_j + \ldots \tag{5}$$

Equation (5) is equivalent to $(I - H)^{-1}s$, and can be rewritten as a Neumann power series as shown in (6).

$$x_i = s_i + (Hs)_i + \left(H^2 s\right)_i + \left(H^3 s\right)_i + \ldots \tag{6}$$

A small example problem is included in Appendix 1. If the Neumann series $I + H + H^2 + H^3 + \ldots$ converges then the solution does as well. The series will converge as long as diagonal dominace is observed. Similar observations can be made for the other estimators.

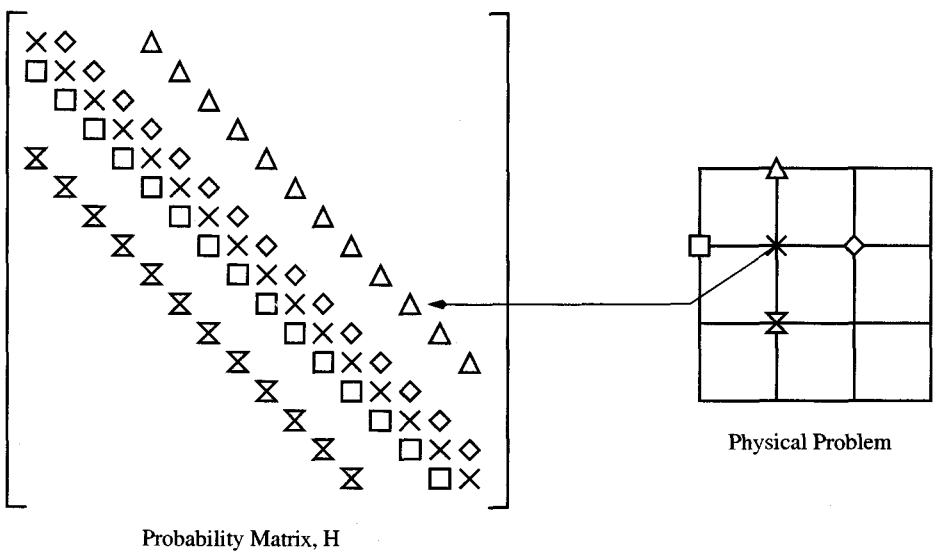

Fig. 1: Association of the probability matrix with the physical problem.

This method of generating the probability matrix is called diagonal scaling and is not unique, however, it is simple and is well suited for our linear systems. The values in the probability matrix for our development must be positive. Negative off-diagonal elements would be indicative of negative jump probabilities. This would result in problems due to the unphysical nature of the event.

2.1.2 *Random Walk*

Pseudo-particle movement is described mathematically by row transitions in the probability matrix. These row transitions occur randomly based on the transition probabilities. This is commonly called a *random walk.*

Under normal conditions, the probability distribution function (PDF) is non-uniform requiring a table lookup to determine the appropriate event given a generated random number. A method called *alias sampling* [Edw 91, Mar 91] was employed to eliminate table lookups and improve efficiency. This method basically takes as input a non-uniform PDF histogram and divides each of its bins into two sections. The components of each section are then calculated as follows. The smallest probability is selected and enough of the largest probability is added to the bin to equal the mean of the original PDF. The next smallest probability is then chosen and so on until the original PDF is converted to a uniform set of binary pairs. This conversion process is shown in Figure 2. Sampling from the aliased distribution is performed by uniformly sampling the bin number, then sampling the event from the bin.

Efficiency comparisons have been performed to compare alias sampling to a standard table lookup. The reported results [Edw 91] show alias sampling to be 30% faster for scalar processors like those used in this study.

Random numbers are selected using the linear congruent method and 48-bit integer arithmetic in the *drand48* function inherent to most UNIX systems. The 48-bit arithmetic results in approximately $2^{48} \simeq 3 \times 10^{14}$ random numbers prior to repetition [Lan 97]. To put this in perspective, a problem containing one million unknowns, running one thousand histories per unknown, with each history undergoing an average of 25 jumps prior to termination would use about 5×10^{10} random numbers. This method of generation offers us the best available random number sequence to avoid potential non-random behavior.

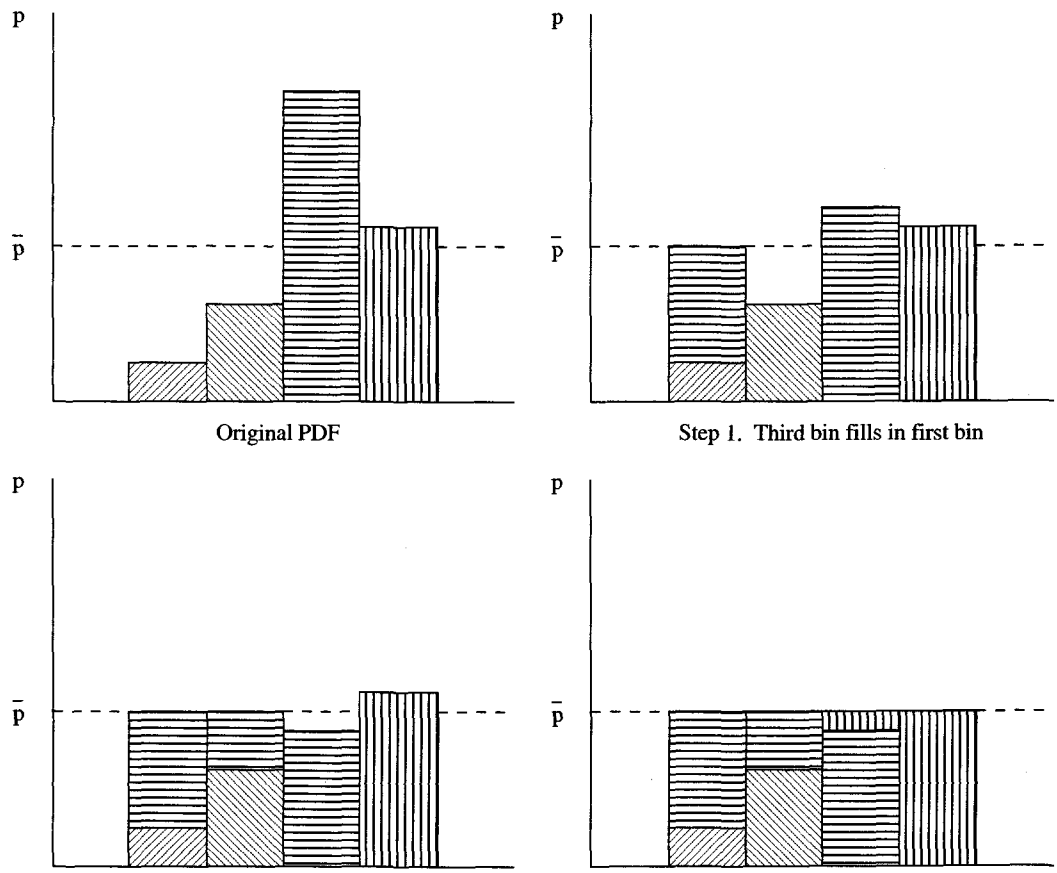

Fig. 2: Conversion of PDF to Aliased PDF.

2.1.3 *Forward vs. Adjoint*

No domain decomposition is employed in our algorithm which implies that every process has access to all of the problem information. This allows us to choose whether to solve the forward or adjoint problem. In the forward problem, pseudo-particles are born at locations with non-zero sources. They then propagate through the problem, and tallies are kept until the histories terminate. At that point, the tallies are applied to the element of the solution vector which corresponds to the location where the pseudo-particle terminated. Hence, to fully assemble a solution value for a given location, histories must be launched from, at least, all of the locations in the surrounding region with non-zero sources.

In the adjoint problem, as with the forward problem, pseudo-particles are born, propagate through the problem, and tallies are kept until the histories terminate. At that point, however, the tallies are applied to the element of the solution vector which corresponds to the location of the pseudo-particles' origin. Selecting which type of problem to solve is somewhat arbitrary since neither problem involves more computational rigor than the other. The adjoint problem, however, offers the additional option of solving for a single value of the solution vector while only starting histories from one location.

2.1.4 *Estimators*

Two types of tallies are used to estimate the solution; a path length estimator and a last event estimator. The adjoint path length estimator tallies the diagonally scaled source term s for the birth location and for each new location following a jump during the history. This tally continues for each particle history born at a specified location until all histories are complete. At that point the tallies are averaged by dividing by the number of histories (N). The resulting value is the Monte Carlo adjoint path length

estimate of the element of solution vector which corresponds to the birth location. Figure 3 and equation (7) describe the adjoint path length estimator.

$$PLE_{adjoint} = \frac{1}{N}\sum_{j=1}^{N}\sum_{i=0}^{n_j} s_i, \quad n_j = \text{number of jumps} \tag{7}$$

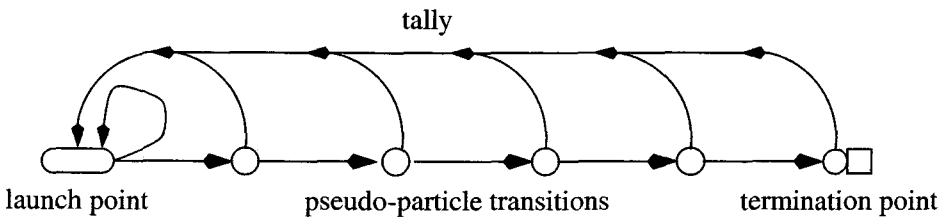

Fig. 3: The adjoint path length estimator.

The last event estimator, in the adjoint problem, tallies the diagonally scaled source term s only at the termination location. This value is normalized on each history by the termination probability. Like the path length estimator the solution is computed by dividing by the number of histories. The resulting value is the Monte Carlo last event estimate of the element of solution vector which corresponds to the birth location. Figure 4 and equation (8) describe the adjoint last event estimator.

$$LEE_{adjoint} = \frac{1}{N}\frac{s_{term}}{p_t} \tag{8}$$

In the forward problem, the tallies are not applied to birth location. The path length estimator tallies the diagonally scaled source term s associated with the birth location only. This value is tallied into the estimate for each element of the solution vector corresponding to the locations where the history tracked. The same normalization occurs over the number of histories as in the adjoint problem. After all histories

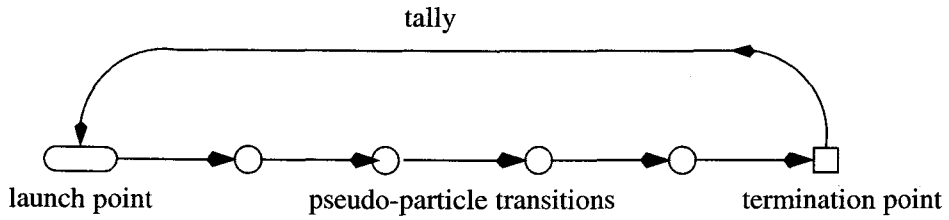

Fig. 4: The adjoint last event estimator.

are completed and the tally has been normalized, the forward path length estimator is complete. Figure 5 and equation (9) describe the forward path length estimator.

$$PLE_{forward_i} = \frac{1}{N} \sum_{j=1}^{M} \sum_{h=1}^{N_j} \sum_{k=1}^{K} s_i \delta_{ki}, \quad \begin{cases} M = \# \text{ of source locations} \\ N_j = \# \text{ of histories / source location} \\ K = \# \text{ of transitions} \end{cases} \tag{9}$$

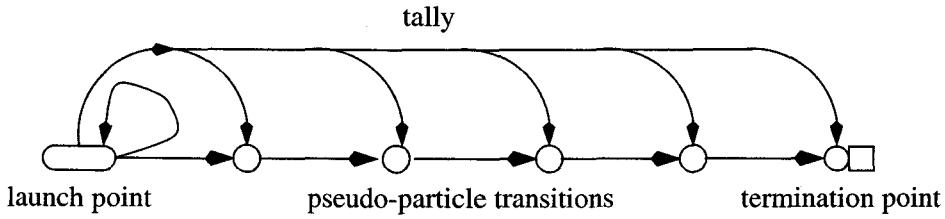

Fig. 5: The forward path length estimator.

The last event estimator, in the forward problem, tallies the diagonally scaled source term s from the birth location at the termination location. This value is normalized on each history by the termination probability and the total tally is normalized by the number of histories. The resulting value is the Monte Carlo last event

estimate of the element of the solution vector which corresponds to the termination location. Figure 6 and equation (10) describe the forward path length estimator.

$$LEE_{forward_i} = \frac{1}{N} \frac{s_{birth}}{p_{t_i}} \tag{10}$$

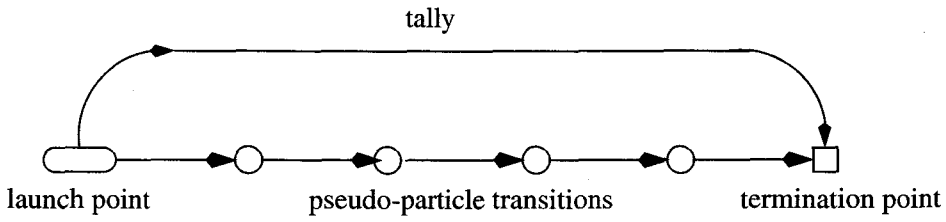

Fig. 6: The forward last event estimator.

2.1.5 Constraints

Our method is limited to matrices with negative or zero-valued off-diagonal elements. Positive off-diagonal elements would produce negative (and unphysical) transition probabilities. Problems with regions of very high scattering ratio are difficult for our method. A high scattering ratio indicates a low probability of absorption which translates to a low termination probability. Diagonally dominant (non-highly scattering) matrices can be solved very efficiently with our approach, but are also quickly solved by other linear solvers. In the extreme case of pure scattering, a pseudo-particle will theoretically jump around forever until it reaches a boundary. This is equivalent to a matrix with a large condition number, which is difficult for all linear solvers. Ideally, our problems have a significant absorption component.

Simply due to hardware considerations, there exist maxima in choosing problems to solve. The variable which typically limits our efforts is the number of unknowns, and the load on the system memory is generally the limiting parameter.

2.2 Application to Discretized Diffusion Equations

2.2.1 Derivation of Equations

We begin with the one-group, fixed source, diffusion equation in two dimensions

$$-\frac{\partial}{\partial x}\left(D\frac{\partial}{\partial x}\phi(x,y)\right)-\frac{\partial}{\partial y}\left(D\frac{\partial}{\partial y}\phi(x,y)\right)+\Sigma_a(x,y)\phi(x,y)=Q(x,y) \qquad (11)$$

and integrate over a control volume (12). The control volumes differ in geometry depending on the location of the unknown as shown in Figure 7.

$$-\int_{y_{j-1}}^{y_j}\int_{x_{i-1}}^{x_i}\frac{\partial}{\partial x}\left(D\frac{\partial}{\partial x}\phi(x,y)\right)dxdy-\int_{y_{j-1}}^{y_j}\int_{x_{i-1}}^{x_i}\frac{\partial}{\partial y}\left(D\frac{\partial}{\partial y}\phi(x,y)\right)dxdy$$
$$+\int_{y_{j-1}}^{y_j}\int_{x_{i-1}}^{x_i}\Sigma_a(x,y)\phi(x,y)dxdy=\int_{y_{j-1}}^{y_j}\int_{x_{i-1}}^{x_i}Q(x,y)dxdy \qquad (12)$$

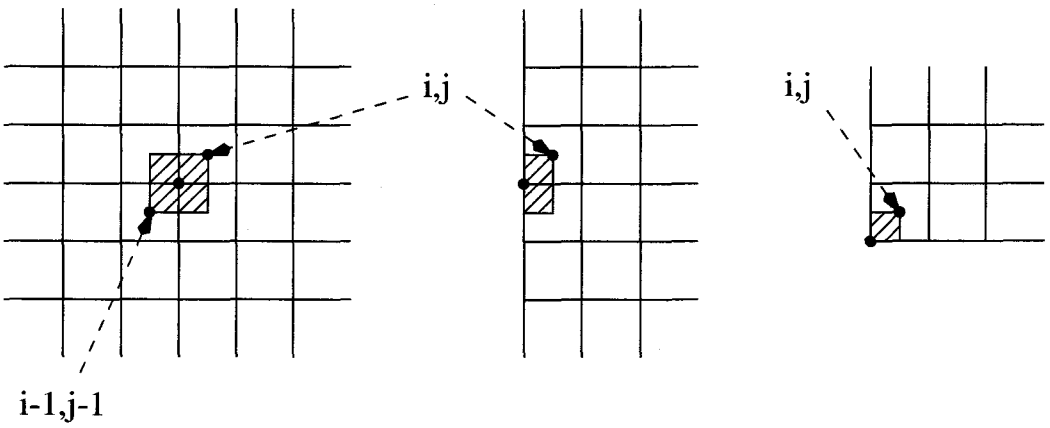

Fig. 7: Representative control volumes.

We define the node averaged flux (13) to solve the absorption term in (12).

$$\phi_{i-1/2,j-1/2} = \frac{\int_{y_{j-1}}^{y_j}\int_{x_{i-1}}^{x_i}\phi(x,y)dxdy}{\int_{y_{j-1}}^{y_j}\int_{x_{i-1}}^{x_i}dxdy} \tag{13}$$

Using (13) we rewrite the absorption term as (14). As the material properties are constant in each cell, (14) is then simplified to (15).

$$\int_{y_{j-1}}^{y_j}\int_{x_{i-1}}^{x_i}\Sigma_a(x,y)\phi(x,y)dxdy = \phi_{i-1/2,j-1/2}\int_{y_{j-1}}^{y_j}\int_{x_{i-1}}^{x_i}\Sigma_a(x,y)dxdy \tag{14}$$

$$\begin{aligned}\int_{y_{j-1}}^{y_j}\int_{x_{i-1}}^{x_i}\Sigma_a(x,y)\phi(x,y)dxdy = \phi_{i-1/2,j-1/2}\Bigg[&\Sigma_{a_{i,j}}\frac{\Delta x_i\Delta y_j}{4}\\ &+\Sigma_{a_{i,j-1}}\frac{\Delta x_i\Delta y_{j-1}}{4}+\Sigma_{a_{i-1,j}}\frac{\Delta x_{i-1}\Delta y_j}{4}+\Sigma_{a_{i-1,j-1}}\frac{\Delta x_{i-1}\Delta y_{j-1}}{4}\Bigg]\end{aligned} \tag{15}$$

Integrating the first two (leakage) terms, we are left with gradient terms that must be eliminated. An example is shown in (16).

$$\int_{y_{j-1}}^{y_j}\int_{x_{i-1}}^{x_i}-\frac{\partial}{\partial x}\left(D\frac{\partial}{\partial x}\phi(x,y)\right)dxdy = -\frac{\Delta y}{2}\left[D\frac{\partial}{\partial x}\phi\right]_{x_{i-1}}^{x_i} \tag{16}$$

We resolve this problem by using a finite difference theorem approximation (17) to the gradient term.

$$\left.\frac{\partial\phi}{\partial x}\right|_{x_{i-1}}^{x_i} = \frac{\phi_{x_i}-\phi_{x_{i-1}}}{\Delta x} \tag{17}$$

Equation (17) replaces the partial derivative terms and facilitates spatial discretization. Using our definitions, we arrive at the 2-D discretized diffusion equations (18) in the problem interior.

$$
\begin{aligned}
&-\left(\frac{D_{i,j}}{2}\frac{\Delta x_i}{\Delta y_j}+\frac{D_{i-1,j}}{2}\frac{\Delta x_{i-1}}{\Delta y_j}\right)\phi_{i-1/2,j+1/2}\\
&-\left(\frac{D_{i,j-1}}{2}\frac{\Delta x_i}{\Delta y_{j-1}}+\frac{D_{i-1,j-1}}{2}\frac{\Delta x_{i-1}}{\Delta y_{j-1}}\right)\phi_{i-1/2,j-3/2}\\
&-\left(\frac{D_{i,j}}{2}\frac{\Delta y_j}{\Delta x_i}+\frac{D_{i,j-1}}{2}\frac{\Delta y_{j-1}}{\Delta x_i}\right)\phi_{i+1/2,j-1/2}\\
&-\left(\frac{D_{i-1,j}}{2}\frac{\Delta y_j}{\Delta x_{i-1}}+\frac{D_{i-1,j-1}}{2}\frac{\Delta y_{j-1}}{\Delta x_{i-1}}\right)\phi_{i-3/2,j-1/2}\\
&+\left(\frac{D_{i,j}}{2}\frac{\Delta x_i}{\Delta y_j}+\frac{D_{i-1,j}}{2}\frac{\Delta x_{i-1}}{\Delta y_j}+\frac{D_{i,j-1}}{2}\frac{\Delta x_i}{\Delta y_{j-1}}+\frac{D_{i-1,j-1}}{2}\frac{\Delta x_{i-1}}{\Delta y_{j-1}}\right.\\
&+\frac{D_{i,j}}{2}\frac{\Delta y_j}{\Delta x_i}+\frac{D_{i,j-1}}{2}\frac{\Delta y_{j-1}}{\Delta x_i}+\frac{D_{i-1,j}}{2}\frac{\Delta y_j}{\Delta x_{i-1}}+\frac{D_{i-1,j-1}}{2}\frac{\Delta y_{j-1}}{\Delta x_{i-1}}\\
&+\Sigma_{a_{i,j}}\frac{\Delta x_i\Delta y_j}{4}+\Sigma_{a_{i,j-1}}\frac{\Delta x_i\Delta y_{j-1}}{4}+\Sigma_{a_{i-1,j}}\frac{\Delta x_{i-1}\Delta y_j}{4}\\
&\left.+\Sigma_{a_{i-1,j-1}}\frac{\Delta x_{i-1}\Delta y_{j-1}}{4}\right)\phi_{i-1/2,j-1/2}=Q_{i,j}\frac{\Delta x_i\Delta y_j}{4}\\
&+Q_{i,j-1}\frac{\Delta x_i\Delta y_{j-1}}{4}+Q_{i-1,j}\frac{\Delta x_{i-1}\Delta y_j}{4}+Q_{i-1,j-1}\frac{\Delta x_{i-1}\Delta y_{j-1}}{4}
\end{aligned}
\tag{18}
$$

For boundary conditions we make the following observations:

- Reflecting Boundary

$$J\left(x_{i-1/2}, y\right)\big) = -D_{i-1/2,y} \left.\frac{\partial \phi}{\partial x}\right|_{x_{i-1/2}} = 0 \tag{19}$$

- Vacuum Boundary

$$D_{i-1/2,y} \left.\frac{\partial \phi}{\partial x}\right|_{x_{i-1/2}} = \frac{\phi_{i-1/2,y}}{2} \tag{20}$$

- Incident Boundary

$$J^- = \frac{\phi}{4} - \frac{D_{i-1/2,y}}{2} \left.\frac{\partial \phi}{\partial x}\right|_{x_{i-1/2}} \quad \text{or} \quad D_{i-1/2,y} \left.\frac{\partial \phi}{\partial x}\right|_{x_{i-1/2}} = \frac{\phi_{i-1/2,y}}{2} - 2J^- \tag{21}$$

If we consider the left edge boundary as illustrated in Figure 7, our boundary condition observations allow us to write the governing equations as follows:

$$\begin{aligned}
&- \left(\frac{D_{i,j}}{2}\frac{\Delta x_i}{\Delta y_j}\right)\phi_{i-1/2,j+1/2} - \left(\frac{D_{i,j-1}}{2}\frac{\Delta x_i}{\Delta y_{j-1}}\right)\phi_{i-1/2,j-3/2} \\
&- \left(\frac{D_{i,j}}{2}\frac{\Delta y_j}{\Delta x_i} + \frac{D_{i,j-1}}{2}\frac{\Delta y_{j-1}}{\Delta x_i}\right)\phi_{i+1/2,j-1/2} \\
&+ \left(\frac{D_{i,j}}{2}\frac{\Delta x_i}{\Delta y_j} + \frac{D_{i,j-1}}{2}\frac{\Delta x_i}{\Delta y_{j-1}} + \frac{D_{i,j}}{2}\frac{\Delta y_j}{\Delta x_i} + \frac{D_{i,j-1}}{2}\frac{\Delta y_{j-1}}{\Delta x_i}\right. \\
&\left. + \Sigma_{a_{i,j-1}}\frac{\Delta x_i \Delta y_{j-1}}{4} + \Sigma_{a_{i,j}}\frac{\Delta x_i \Delta y_j}{4} + \alpha\left(\frac{\Delta y_j}{4} + \frac{\Delta y_{j-1}}{4}\right)\right)\phi_{i-1/2,j-1/2} \\
&= Q_{i,j-1}\frac{\Delta x_i \Delta y_{j-1}}{4} + Q_{i,j}\frac{\Delta x_i \Delta y_j}{4} + \beta 2J^-\left(\frac{\Delta y_j}{2} + \frac{\Delta y_{j-1}}{2}\right)
\end{aligned} \tag{22}$$

where the α and β parameters vary as shown in Table 1 for different boundary conditions.

Table 1: Side node multipliers

Boundary Condition	α	β
Reflecting	0	0
Vacuum	1	0
Incident	1	1

Similarly, if we consider the lower left corner boundary as illustrated in Figure 7 and the boundary condition observations we may write the governing equations as follows:

$$
\begin{aligned}
&- \left[\frac{D_{i,j}}{2}\frac{\Delta y_j}{\Delta x_i}\right]\phi_{i+1/2,j-1/2} - \left[\frac{D_{i,j}}{2}\frac{\Delta x_i}{\Delta y_j}\right]\phi_{i-1/2,j+1/2} \\
&+ \left[\frac{D_{i,j}}{2}\frac{\Delta y_j}{\Delta x_i} + \frac{D_{i,j}}{2}\frac{\Delta x_i}{\Delta y_j} + \Sigma_{a_{i,j}}\frac{\Delta x_i \Delta y_j}{4} + \gamma_x\frac{\Delta x_i}{4} + \gamma_y\frac{\Delta y_j}{4}\right]\phi_{i-1/2,j-1/2} \\
&= Q_{i,j}\frac{\Delta x_i \Delta y_j}{4} + \rho_y 2J^{-}\frac{\Delta y_j}{2} + \rho_x 2J^{-}\frac{\Delta x_i}{2}
\end{aligned}
\tag{23}
$$

where the α, β, and ρ parameters are given in Table 2 for the different combinations of boundary conditions.

The same type of analysis may be applied to the remaining edge and corner boundaries with similar results.

2.2.2 Algorithm

For an arbitrarily selected pseudo-particle history at an arbitrary location, the algorithm calculates the jump and termination probabilities, in essence building a

Table 2: Corner node multipliers

Left BC	Bottom BC	γ_x	γ_y	ρ_x	ρ_y
Reflecting	Reflecting	0	0	0	0
Vacuum	Vacuum	1	1	0	0
Incident	Incident	1	1	1	1
Reflecting	Vacuum	1	0	0	0
Reflecting	Incident	1	0	1	0
Incident	Reflecting	0	1	0	1
Incident	Vacuum	1	1	0	1
Vacuum	Reflecting	0	1	0	0
Vacuum	Incident	1	1	1	0

probability distribution function. This PDF is converted to an aliased PDF that is randomly sampled to select a bin, and the bin is randomly sampled to select the original or alias event for the pseudo-particle. A tally is made and the process repeats for each new location until the history terminates.

The algorithm contains distinct serial and parallel portions. The serial portion has two basic functions: it reads the problem data and initializes the message-passing interface. Parallelization is performed by simply defining the span in the loop over unknowns to be equal to the number of processes employed. Each process is tasked with finding the Monte Carlo solution for a particular unknown by introducing and tracking a specified number of histories from that location and collecting the tallies. At the completion of all histories, the tallies are normalized, stored, and the process is given a new unknown. This continues until all of the unknowns are treated.

2.3 Parallelization

2.3.1 Message-Passing Interface

We parallelize our algorithm using the MPI [Gro 94] message-passing interface. MPI is a library that has been created over time, collecting subroutines and calling sequences which represent the best features of existing message-passing systems. It allows setup of parallel processes and communication between them. We used the Portable, Extensible Toolkit for Scientific Computing (PETSc) [Bal 99] as a benchmark for evaluating our Monte Carlo technique. PETSc is a suite of parallelized linear solvers developed and maintained by Argonne National Laboratory. Results from PETSc were used a measure of the accuracy and efficiency of the Monte Carlo algorithm with the understanding that PETSc does not calculate an exact solution.

2.3.2 Machine Architecture

We implemented our algorithm in FORTRAN 90 on the "Compass" cluster of the Lawrence Livermore National Laboratory open computing facility. Compass is a group of Compaq AlphaServer 8400 model 5/440 systems. This cluster has a total of 8 nodes, 80 processors and MIMD architecture with 56 GB of shared memory.

3 RESULTS

3.1 Accuracy

The accuracy of the Monte Carlo algorithm was tested in a two part process. The first part of the process was to analyze the behavior of the algorithm in two special cases of problems with known analytic solutions. These problems model a homogeneous, fixed-source, infinite medium and a source-free, purely absorbing medium. It is well documented [And 84] that a node-centered diffusion discretization is second order accurate as $\Delta x \to 0$, so the mesh spacing was chosen to be on the order of 0.1 mean free paths. The same two problems were also solved by the PETSc algorithm.

In the homogeneous infinite medium case, the leakage terms are eliminated and the solution reduces analytically to

$$\phi(x) = \frac{Q}{\Sigma_a} \tag{24}$$

In the test problem, we set $Q = 4.50cm^{-3}s^{-1}$ and $\Sigma_a = 3.00cm^{-1}$ and expect a constant solution of $\phi = 1.50cm^{-2}s^{-1}$. The results of the infinite medium benchmark problem are shown in Figures 8 and 9 for the Monte Carlo algorithm and Figure 10 for PETSc. The Monte Carlo estimators and PETSc all calculated the solution exactly or within expected statistical noise.

In the source free pure absorber case, (11) reduces to a second order, homogenous, differential equation with an exponential analytic solution as shown in (25).

$$\phi = Ce^{-\frac{x}{L}} \quad \text{where} \quad L = \sqrt{\frac{D}{\Sigma_a}} \quad \text{and} \quad C = J^{-}\left(\frac{1}{4} + \frac{D}{2L}\right)^{-1} \tag{25}$$

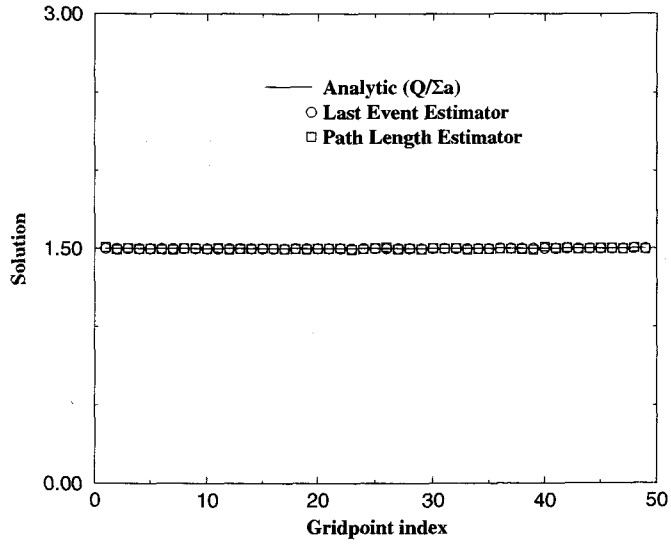

Fig. 8: Monte Carlo: Infinite, Homogeneous Medium, Adjoint Estimators.

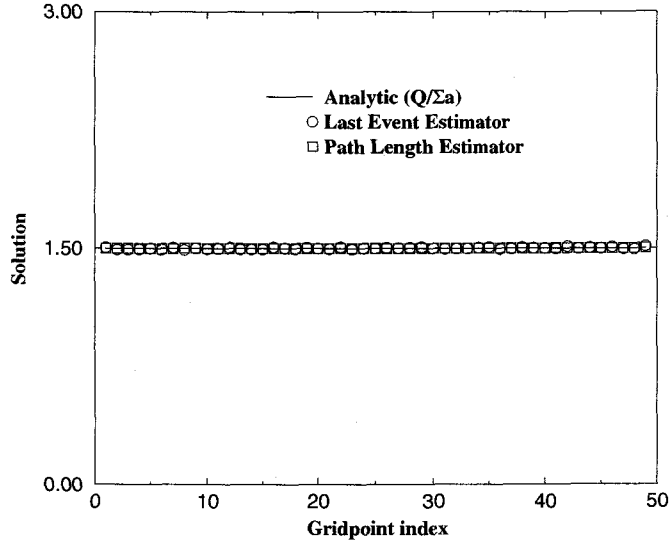

Fig. 9: Monte Carlo: Infinite, Homogeneous Medium, Forward Estimators.

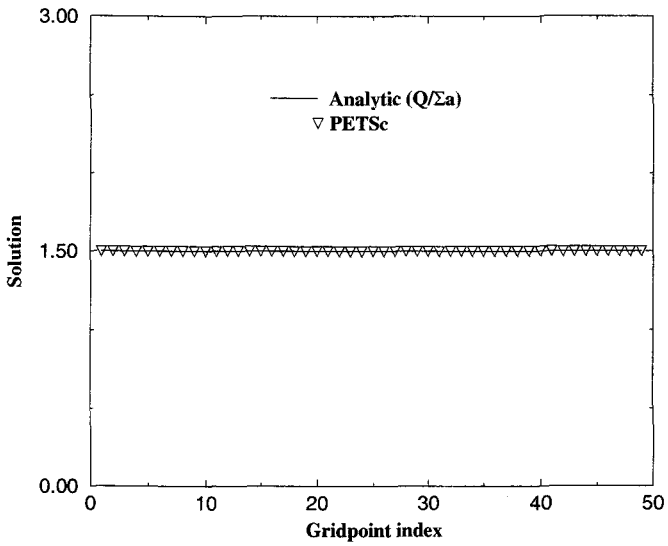

Fig. 10: PETSc: Infinite, Homogeneous Medium.

In the test problem, we set $Q = 0.0$ and defined the material properties to model the medium as a pure absorber. We then introduced an incident boundary condition on the left edge, a vacuum boundary condition on the right edge, and reflecting boundaries on the top and bottom. This configuration mimics a 1-D geometry. The results of the source free, pure absorber benchmark problem are shown in Figures 11 and 12 for the Monte Carlo algorithm and Figure 13 for PETSc. Both Monte Carlo adjoint estimators and PETSc closely approximated the solution to this problem. The Monte Carlo forward estimators calculated the boundary values very closely, but were approximately 10% low in the problem interior.

The second part of the accuracy analysis was to observe the behavior of the algorithm for two test problems which were designed to employ a range of material properties and boundary conditions. For these tests, PETSc was used to benchmark the Monte Carlo algorithm. The first problem models a crude cluster of reactor fuel assemblies in a water medium with vacuum boundaries. The second problem models

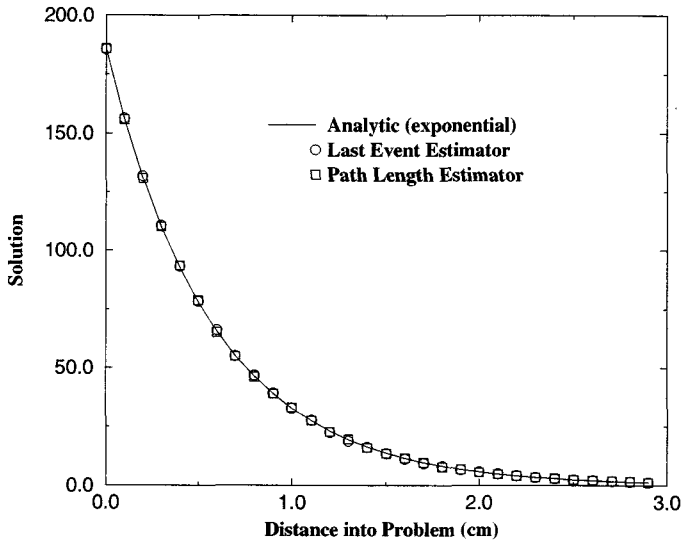

Fig. 11: Monte Carlo: Source-Free, Pure Absorber, Adjoint Estimators.

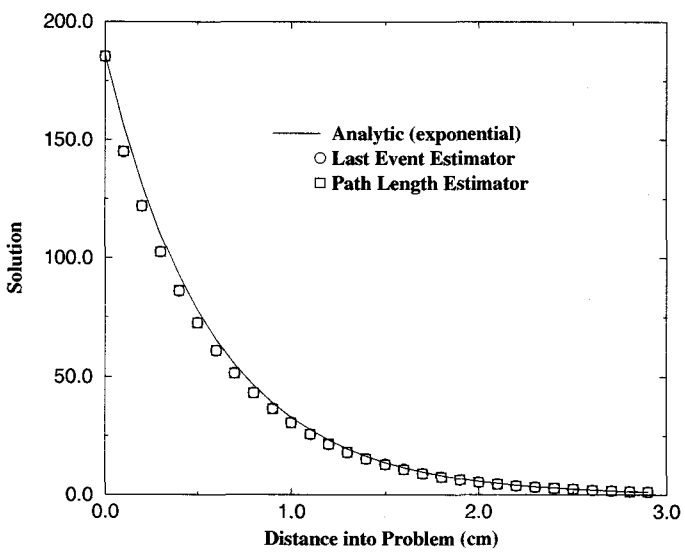

Fig. 12: Monte Carlo: Source-Free, Pure Absorber, Forward Estimators.

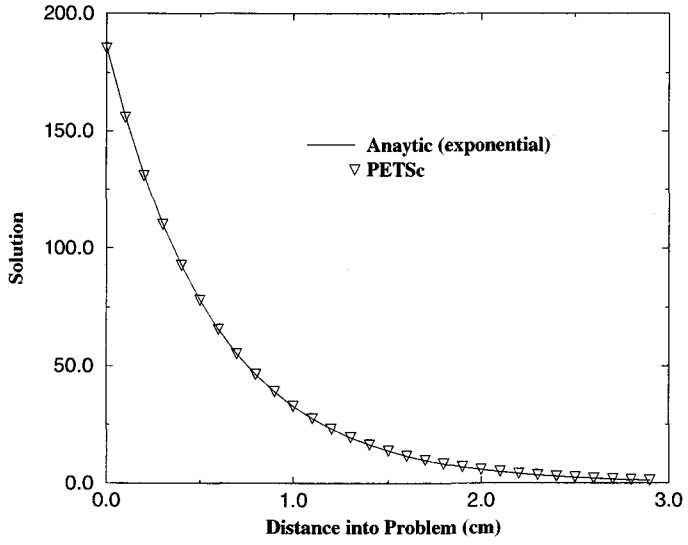

Fig. 13: PETSc: Source-Free, Pure Absorber.

an arbitrarily selected source and shield configuation with one incident boundary condition and reflecting boundaries elsewhere. In both model problems, grid spacing was chosen to be on the order of 0.1 mean free paths to minimize error caused by discretization.

3.1.1 Test Problem 1: Reactor Fuel Assemblies

This test problem models twelve reactor fuel assemblies. The scale of the model is small, only about 38cm across in x and y, and the fuel pins inside the assemblies are not resolved. Creating a full scale model with complete resolution would not have contributed significantly to our objective here, which is simply to demonstrate that the Monte Carlo algorithm arrives at the same solution as the PETSc benchmark. Figures 14, 15, and 16 show the two-dimensional results for the Monte Carlo estimators and the PETSc solution. The contour legend for the Monte Carlo estimators has been matched to the PETSc solution to give a graphical indication

of differences in the solutions. PETSc and the adjoint Monte Carlo estimators are practically identical. The solution calculated by the forward Monte Carlo estimators showed an abrupt change at the material boundaries. This was neither expected nor exhibited by PETSc or the adjoint estimators.

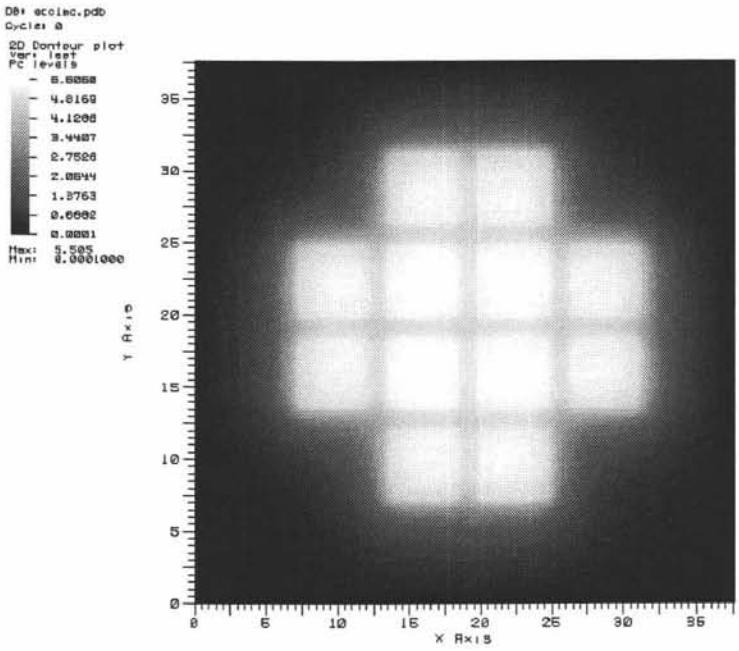

Fig. 14: Test Problem 1: Monte Carlo, Adjoint Estimator.

3.1.2 Test Problem 2: Source/Shield Assembly

This test problem models a diffusive region with a fixed source occupying a portion of the space. A highly absorbing shield is located in the diffusive region to illustrate the ability of the Monte Carlo algorithm to handle problems with mixed diffuse and strongly absorbing regions. Figures 17, 18, and 19 show the two-dimensional results for the Monte Carlo estimators and the PETSc solver. Again, the scale for the Monte Carlo solutions have been matched to the PETSc solution to give a graphical indication of differences in the solutions. Again, PETSc and the adjoint Monte Carlo estimators matched very well, but the forward estimators were noticeably in error.

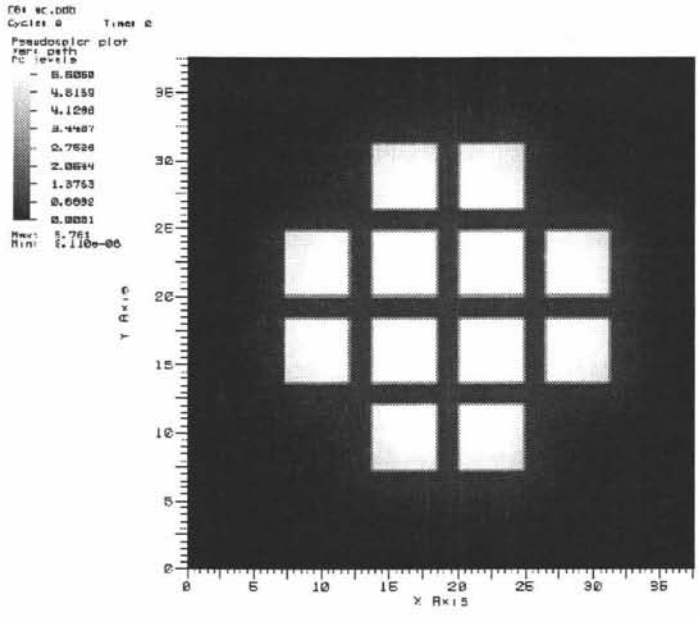

Fig. 15: Test Problem 1: Monte Carlo, Forward Estimator.

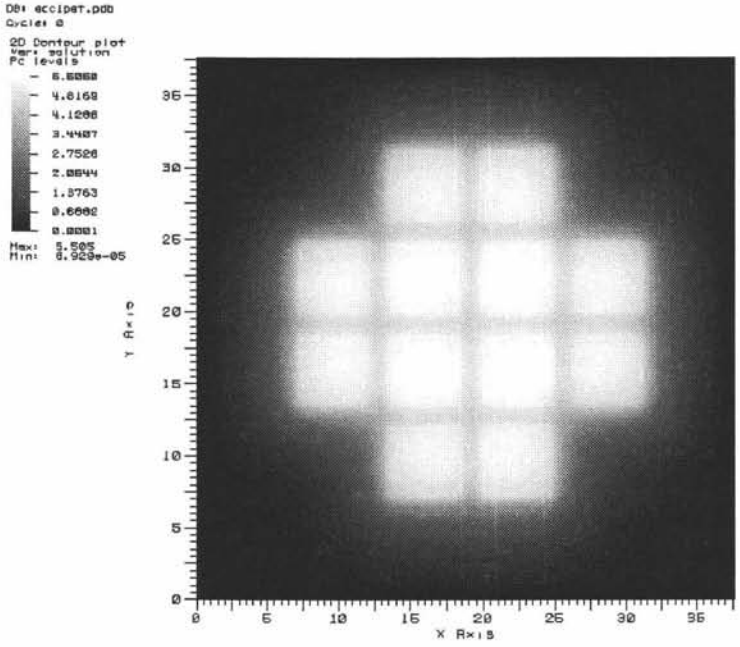

Fig. 16: Test Problem 1: PETSc solution.

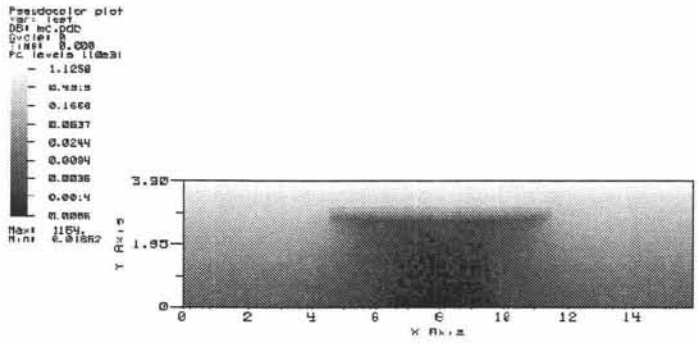

Fig. 17: Test Problem 2: Monte Carlo, Adjoint Estimator.

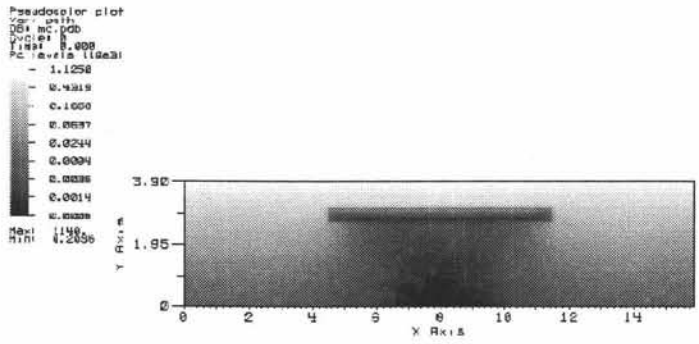

Fig. 18: Test Problem 2: Monte Carlo, Forward Estimator.

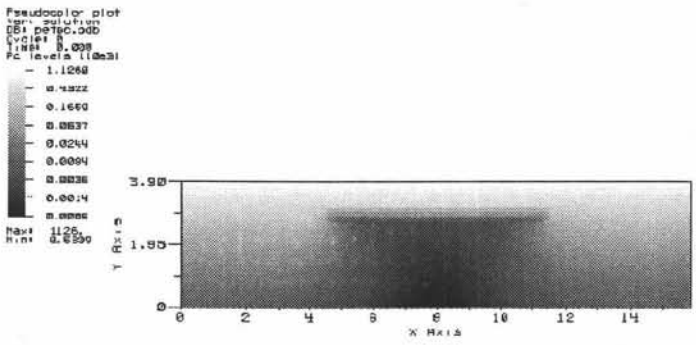

Fig. 19: Test Problem 2: PETSc solution.

3.1.3 *Test Problem 3: Relative Error*

A final test problem was formulated which represents a broad range of material properties and mixed boundary conditions. The PETSc algorithm was used to benchmark the Monte Carlo results. A statistical analysis was performed in which the L_2 norm

$$\mathrm{L_2\ Norm} = \sqrt{\sum_i \left| \frac{\mathrm{PETSc}_i - \mathrm{MC}_i}{\mathrm{PETSc}_i} \right|^2} \tag{26}$$

of the relative error was calculated for each test run. The number of histories per node was varied from 1 to 100,000 by orders of magnitude. Figure 20 and 21 show the convergence characteristics of the convergence test problem for the adjoint and forward last event and path length estimators, respectively. The adjoint estimators exhibit a steady downward trend indicating that up to 100,000 histories they converge to the PETSc solution. The forward estimators plateau at a fixed L_2 norm value. This indicates that they converge to a solution slightly different than PETSc. This is more evidence that the forward estimators are slightly inaccurate. However, we can make no assumptions about the trends in either type of estimator since PETSc is not an exact solution.

3.2 Convergence

The convergence of the Monte Carlo algorithm was tested by a controlled variation in the number of histories. In theory, the standard deviation should decrease as the square root of the number of histories [Lew 93] or

$$\epsilon = \frac{1}{N^{1/2}}. \tag{27}$$

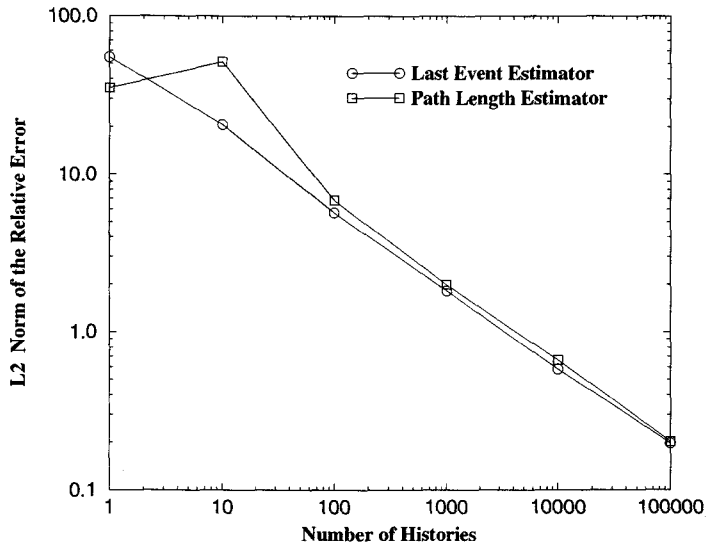

Fig. 20: L_2 norm of the Relative Error, Adjoint Estimators.

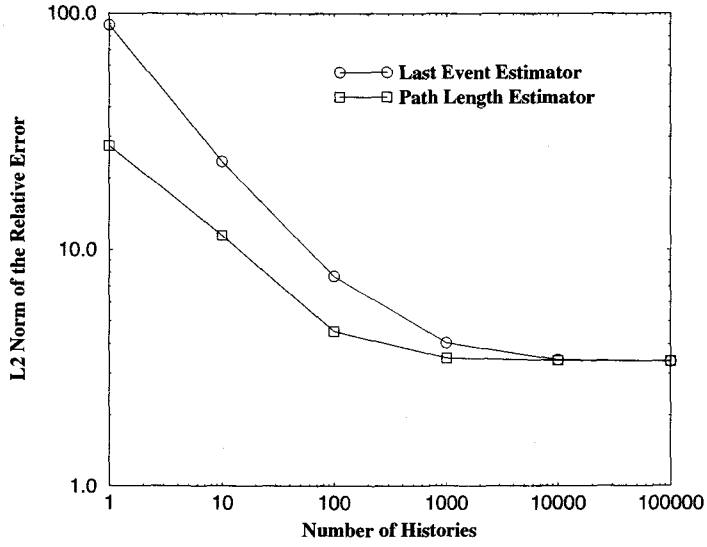

Fig. 21: L_2 norm of the Relative Error, Forward Estimators.

A simple test problem was explored to determine how well our estimators conform to (27). The number of histories was varied from 100 to 100,000 in orders of magnitude. Figures 22 and 23 show the standard deviation about the sample mean for each of the unknowns in the problem, at each number of histories, for the adjoint last event and path length estimators, respectively. Figures 24 and 25 show the same information for the forward estimators.

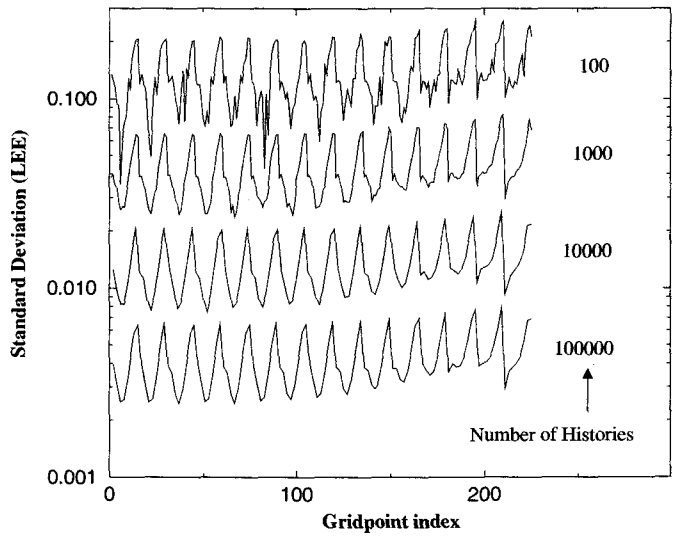

Fig. 22: Standard Deviation, Adjoint Last Event Estimator.

Figure 26 and 27 consolidate the convergence data into average values of the standard deviation about the mean. Both figures show very close adherence to the expectation of (27).

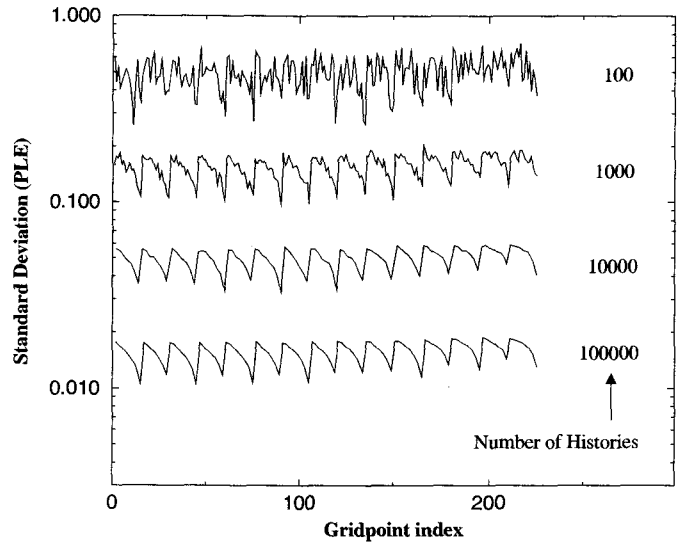

Fig. 23: Standard Deviation, Adjoint Path Length Estimator.

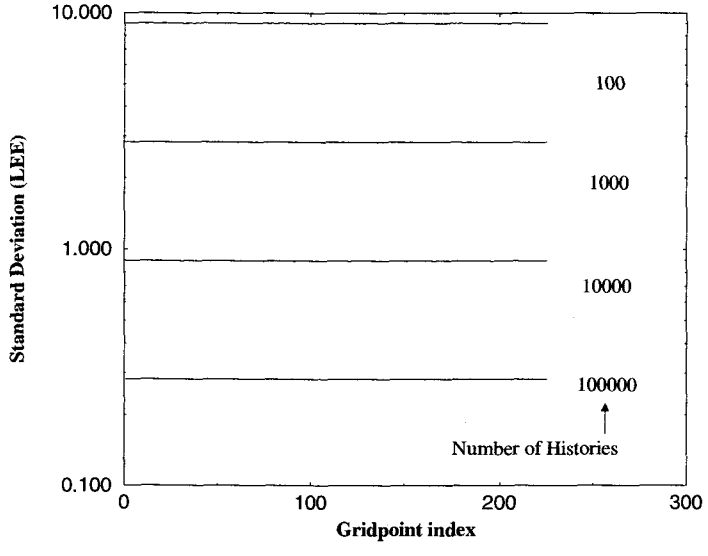

Fig. 24: Standard Deviation, Forward Last Event Estimator.

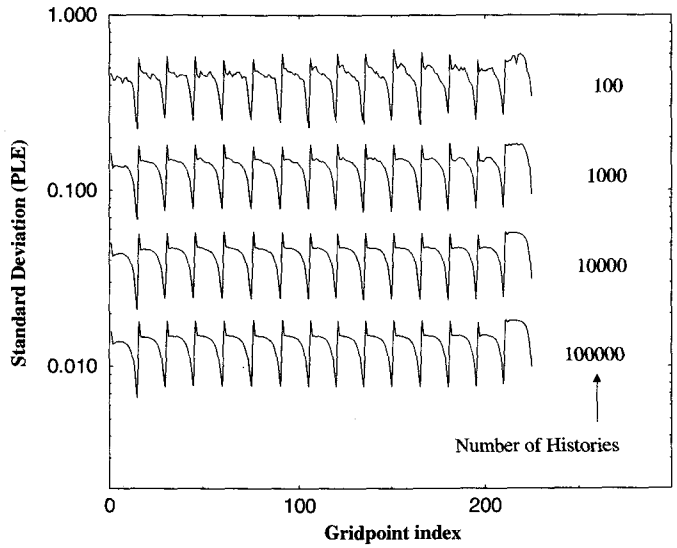

Fig. 25: Standard Deviation, Forward Path Length Estimator.

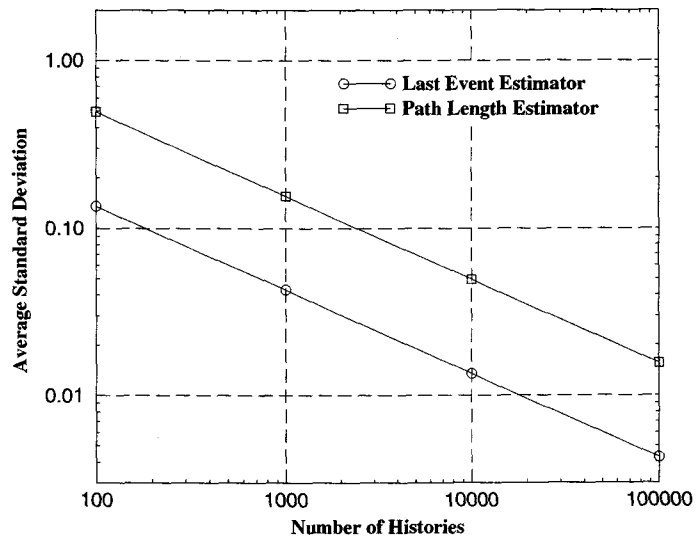

Fig. 26: Average Standard Deviation, Both Adjoint Estimators.

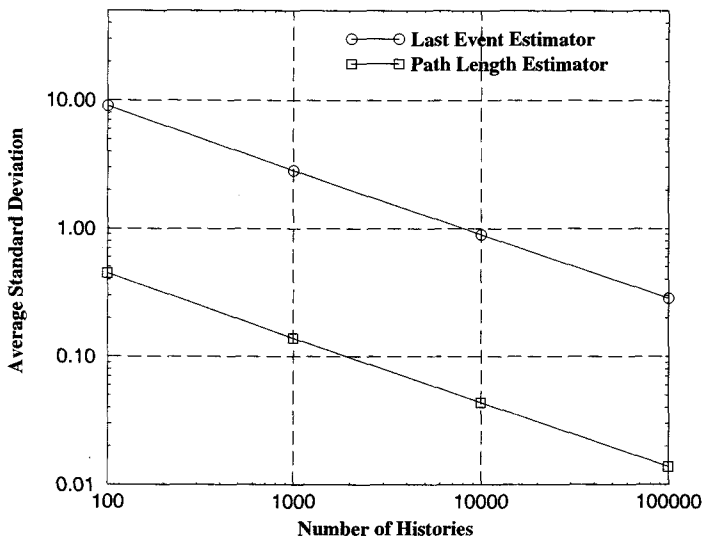

Fig. 27: Average Standard Deviation, Both Forward Estimators.

3.3 Efficiency

Efficiency of the Monte Carlo algorithm was tested by varying the number of processes used to solve a suite of test problems. The key parameter that determines the speed at which Monte Carlo can solve a problem is the amount of "scattering" that occurs. This varies in a heterogeneous problem. For our purposes, heterogeneity would obscure the relevance of the timing results. The first three test problems were chosen to eliminate this complexity while exploring how variation in the scattering ratio effects the Monte Carlo performance. The fourth test problem contains multiple regions with a range of scattering ratios. It also has mixed boundary conditions, and regions with fixed sources. This last test problem was included to show that the Monte Carlo algorithm can be efficient for problems with complicated geometries. The results that follow show only data for the adjoint Monte Carlo estimators. Since the random walk is essentially the same for forward or adjoint problems with a given number of total histories, inclusion of data for both would be redundant.

In the first three test problems we look only at the speedup. In the final test problem we will look both speedup and CPU time. Speedup is defined here as the ratio of the time required to solve the problem using N processes to the time required to solve the problem using one process. In theory this speedup should be linear for the Monte Carlo technique and less than linear for PETSc due to communication overhead. With limitations on resources, it is likely that PETSc will typically solve most problems in less CPU time than the Monte Carlo algorithm. However, the results that follow indicate that availability of more massively parallel machines will make the Monte Carlo algorithm more competitive.

3.3.1 Test Problem 1

Test problem 1 employs a scattering ratio of $c = 0.9$. Figure 28 is a graph of the speedup as a function of the number of processes.

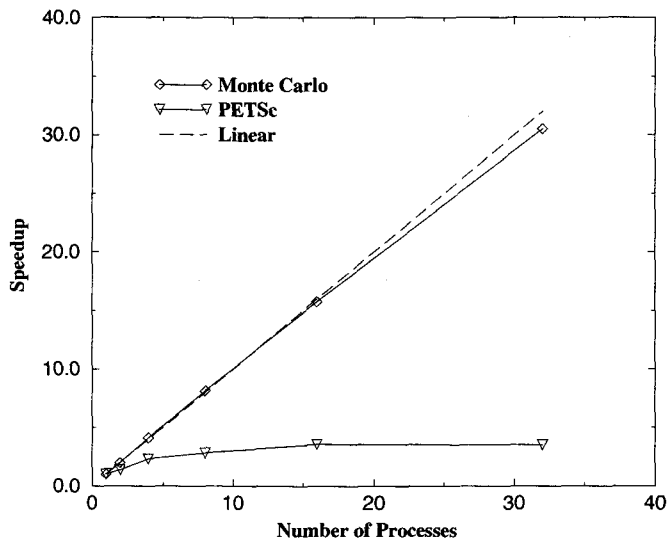

Fig. 28: Efficiency Test Problem 1 Speedup: Scattering ratio $c = 0.9$.

3.3.2 Test Problem 2

Test problem 2 employs a scattering ratio of $c = 0.5$. Figure 29 is a graph of the speedup as a function of the number of processes.

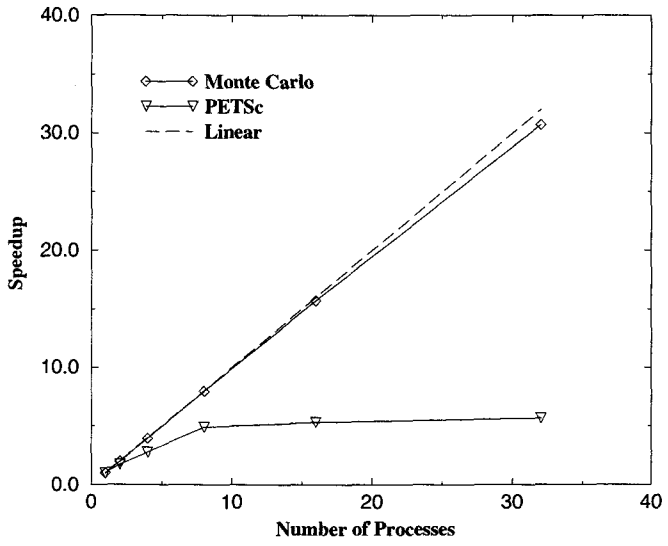

Fig. 29: Efficiency Test Problem 2 Speedup: Scattering ratio $c = 0.5$.

3.3.3 *Test Problem 3*

Test problem 3 employs a scattering ratio of $c = 0.1$. Figure 30 is a graph of the speedup as a function of the number of processes.

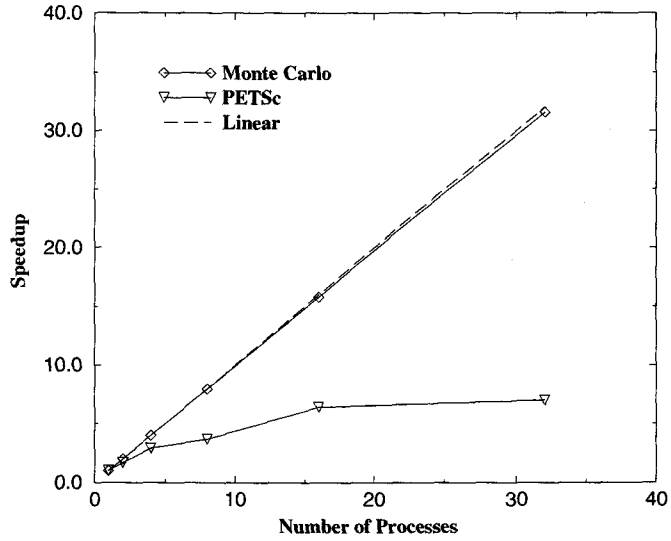

Fig. 30: Efficiency Test Problem 3 Speedup: Scattering ratio $c = 0.1$.

3.3.4 *Test Problem 4*

Test problem 4 is diagrammed in Figure 31. It considers a combination of reflecting, vacuum, and incident boundary conditions and three types of materials. These materials are defined to introduce a range of optical thicknesses and scattering ratios. Figure 32 is a graph of the speedup as a function of the number of processes for PETSc and our Monte Carlo algorithm. It is clear from this graph that speedup is not affected by problem complexity.

In all of the efficiency test problems the speedup profiles are very nearly linear. This is important in asserting that the Monte Carlo algorithm is robust within its limitations as discussed in section 2.1.5.

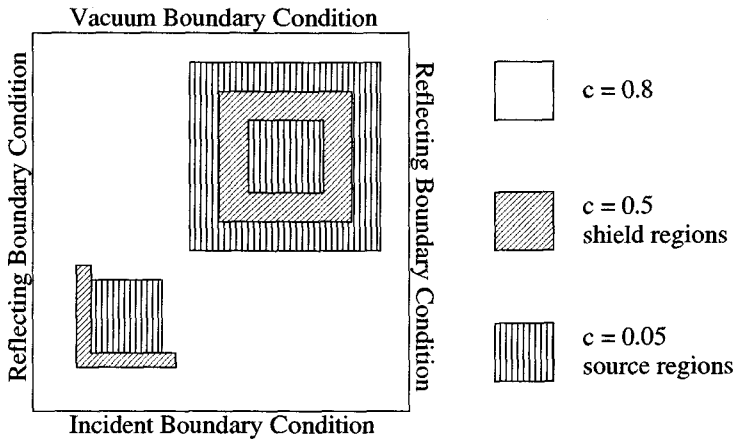

Fig. 31: Efficiency Test Problem 4: Layout of Broad Range Problem.

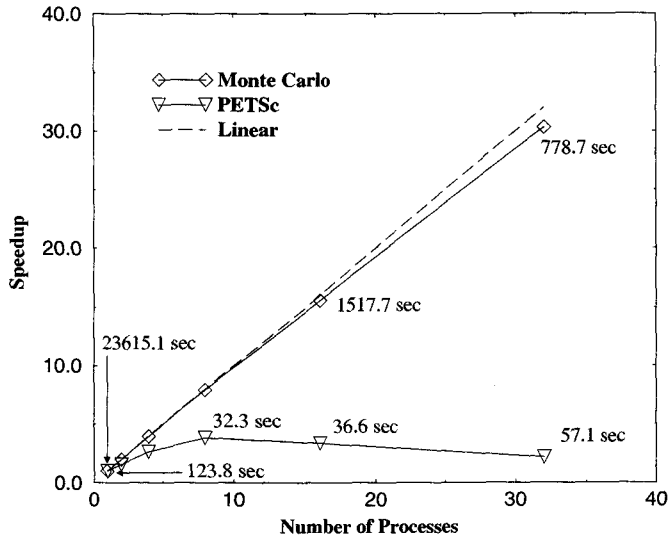

Fig. 32: Efficiency Test Problem 4 Speedup: Broad Range Problem.

3.4 On-the-Fly versus Standard Monte Carlo

Up to this point we have discussed the standard Monte Carlo algorithm. While the previous section shows its efficiency follows expectation, it does so at a cost. Every additional unknown in the physical problem increases the size of the matrix. In the standard Monte Carlo algorithm discussed thus far, the significant elements of this matrix are all calculated and stored in arrays. This information is held in memory for subsequent calculations. This is very taxing on the resources of the computer system. A less memory-intensive way of making these calculations is to calculate only the matrix elements needed for a particular event at the time they are needed. With less information stored during the algorithm run-time, memory load is reduced. We call this method calculating the elements *on-the-fly*. The motivation for such a technique is that larger problems can be attempted.

There are limitations to the effectiveness of this technique, however. Storage is still required for some information, particularly the solution vector and the mesh information. Efficiency is reduced in terms of run-time but speedup is not affected. The point to be considered is whether or not the reduction in memory load justifies the longer run-time.

We developed an on-the-fly algorithm to compare to standard Monte Carlo. We used memory load and CPU time as the metrics of comparison. A simple test problem was devised and all parameters held constant except the number of unknowns. We solved this problem with the standard Monte Carlo and the on-the-fly algorithm for 1000 histories per unknown and collected CPU time and memory load data.

Figure 33 shows a comparison of the CPU times as a function of the number of unknowns. Clearly the standard Monte Carlo algorithm solves the problem faster. Figure 34 is a comparison of the memory load for each algorithm as a function of the number of unknowns. Memory load is divided into two categories. The amount of

memory allocated is the total memory that the process is allowed to use at any given moment during run-time. Resident memory is the amount of memory in use. Here the expectation is met with the on-the-fly algorithm showing a marked reduction in memory load over standard Monte Carlo.

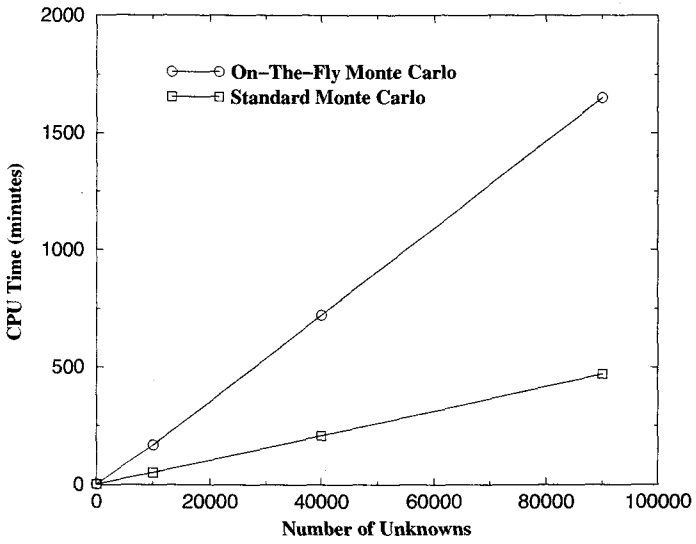

Fig. 33: CPU Time: On-the-Fly Monte Carlo vs. Standard Monte Carlo.

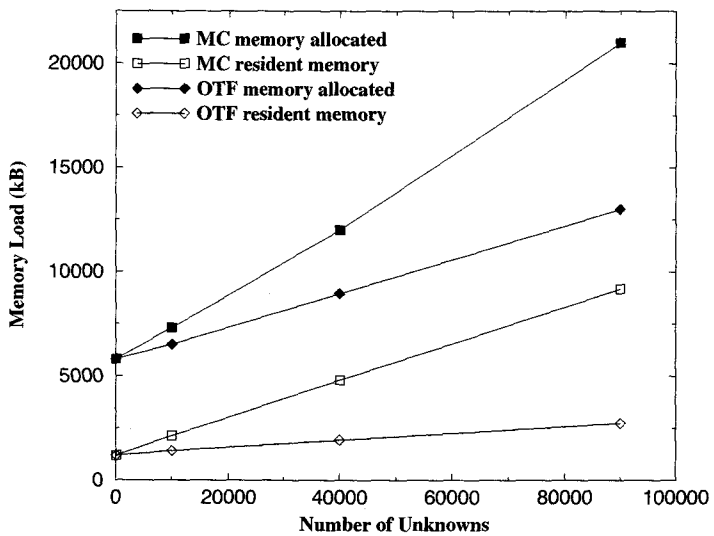

Fig. 34: Memory Load: On-the-Fly Monte Carlo vs. Standard Monte Carlo.

4 CONCLUSION

4.1 Summary of Results

We have developed and investigated the performance of a Monte Carlo algorithm for solving linear systems with application to discretized diffusion equations. We measured performance with regard to accuracy, convergence, and efficiency. Our benchmark was the Portable, Extensible Toolkit for Scientific Computing (PETSc), a code package of parallelized linear system solution routines developed and maintained by Argonne National Laboratory. In this chapter we will discuss the results and consider some possibilities for future work.

Our first metric of performance was accuracy. We presented two benchmark problems with known solutions. We used Monte Carlo and PETSc to solve each problem and compared these numerical solutions to the analytic results. PETSc and both the forward and adjoint Monte Carlo estimators compared very well for the infinite, homogeneous medium problem. The source-free, pure absorber identified a potential weakness in the forward Monte Carlo estimators. The solutions were approximately 10% below the analytic result, while the adjoint estimators and PETSc results were excellent. We then considered two representative problems: a group of reactor fuel assemblies in a basic configuration, and a shielding problem. Monte Carlo and PETSc were both tasked with solving these problems and the results were practically identical for PETSc and adjoint Monte Carlo. Again, the forward Monte Carlo estimators exhibited poor results, in this case at material boundaries. Finally, a test problem was formulated to collect statistical information between the Monte Carlo and PETSc techniques. An L_2 norm of the relative error was calculated and presented which showed that the adjoint estimators behaved very well, displaying a continuous decrease in the relative error as the number of histories was increased. The forward estimators, on the other hand, reached a plateau in the relative error. The error noted

in the solutions resulted in a L_2 norm value which could not be reduced by higher numbers of histories.

The second performance parameter that we looked at was convergence. Here we developed a simple test problem and looked at how the standard deviation varied with increasing numbers of histories. The standard deviation decreased as the square root of the number of histories as predicted for both the adjoint and forward Monte Carlo estimators. This result indicates that all of the Monte Carlo estimators are converging.

Our third and final metric for comparison was efficiency. CPU time data was collected and presented for a varying number of processes. The Monte Carlo algorithm showed a very nearly linear speedup profile. This was expected due to the inherently parallel nature of Monte Carlo in general. PETSc did not fair as well. There is significant interprocess communication during a parallel PETSc calculation. This communication overhead hinders speedup and given enough processes will reduce the performance of PETSc. With the data collected in this study, it is apparent that an ideal number of processes exists for PETSc given a specific problem.

We developed an on-the-fly version of the Monte Carlo algorithm to investigate the potential savings in memory load and what effect that would have on CPU time. One test problem was studied in which the number of unknowns in the problem was increased while all other parameters were held constant. The data collected indicated an approximate three-fold reduction in memory load with a three-fold increase in CPU time.

4.2 Discussion

The most notable problem with the Monte Carlo algorithms we considered was the poor degree of accuracy achieved by the forward estimators. However, the results

obtained for the convergence test problem is particularly important in determining the source of the error. It eliminates the random walk as the source of the error and places the tally formulation in question. At the time of this publication, work is still underway to correct this problem.

The adjoint Monte Carlo algorithm performed well for all of the metrics of comparison. The one significant deficiency was CPU time. For test problem 4 of section 3.3.4 actual CPU times were included in the speedup graph. These numbers indicate that our Monte Carlo algorithm is not under these experimental conditions going to be the method of choice. However, PETSc reached its peak performance at about 8 processes with a CPU time of 32.3 seconds. Given the linear speedup profile of the Monte Carlo method we can extrapolate that by applying more processes, the CPU times for Monte Carlo could become competitive.

In a more general sense, CPU time is highly dependent on the diagonal dominance of the coefficient matrix. As the condition number of the matrix approaches a value of 1, the CPU time increases rapidly. In our diffusion application, this is analogous to a purely scattering problem. Also, because of the discretization scheme, as mesh spacing becomes fine the absorption terms in the diffusion equation approach zero as seen in (18). This, too, causes CPU times to increase dramatically. Here we see a paradox. Refining the mesh spacing causes problems for the Monte Carlo technique. However, as mentioned in section 3.1, a node-centered diffusion discretization is second-order accurate as the mesh spacing is reduced. We must conclude that for certain diffusion problems, and more generally for certain matrices, our Monte Carlo technique breaks down.

In our on-the-fly test problem we saw a three-fold memory load reduction in exchange for a three-fold increase in CPU time. This should not be viewed as a general expectation for any linear system. The amount of CPU time required to solve a problem is highly sensitive to the elements of the matrix. Making generalizations

about the comparitive performance between the standard Monte Carlo algorithm and the on-the-fly Monte Carlo algorithm would be unfounded.

4.3 Suggestions for Future Work

As indicated in the preceeding discussion, work on the forward Monte Carlo estimators is ongoing

The on-the-fly variant of the Monte Carlo algorithm is not fully developed. Our treatment is adequate for proof of principle, however, significant work remains. The algorithm is not fully optimized for memory reduction. Certain information, such as the solution are still stored in memory. In addition, consideration could be given to developing a more efficient method for calculating the matrix and driving term vector elements.

The Monte Carlo algorithm discussed is restricted to probability matrices with positive off-diagonal elements. Negative off-diagonal elements result in negative probabilities. Consideration should be given to developing a method to account for positive off-diagonal elements. Our treatment of diffusion considers only a node-centered finite volume discretization. The technique could be reevaluated for its application other types of discretization schemes.

The algorithm currently starts a set number of histories. To obtain good statistical results this number is typically set quite high. For problems which converge for a relatively small number of histories, this implies wasted work and unnecessary CPU time expense. An adaptive convergence algorithm could be developed to eliminate this waste. This algorithm would routinely test for convergence as the problem was being solved. The algorithm would be terminated when it reached the convergence criteria, eliminating the excess work.

BIBLIOGRAPHY

[And 84] Anderson, D., Tannehill, J., Pletcher, R., *Computational Fluid Mechanics and Heat Transfer*, Hemisphere Publishing Corp., New York, NY, 1984.

[Bal 99] Balay, S., Gropp, W., Curfman McInnes, L., Smith, B.F., *PETSc 2.0 Users Manual (ANL-95/11 - Revision 2.0.24)*, Argonne National Laboratory, 1999.

[Del 85] Delves, L.M., "Monte Carlo Calculations of Neutron Diffusion on the ICL DAP," *Comput. Phys. Comm.* **37**, (1985), 295-301.

[Edw 91] Edwards, A.L., Rathkopf, J.A., Smidt, R.K., "Extending the Alias Monte Carlo Sampling Method to General Distributions," *Advances in Mathematics, Computations, and Reactor Physics*, ANS International Topical Meeting, volume 1, (1991), session 2.1, 3.1-3.10.

[Fin 88] Finnemenn, H., Brehm, J., Michel, E., Volkert, J., "Solution of the Neutron Diffusion Equation through Multigrid Methods Implemented on a Memory-Coupled 25-processor System," *Parallel Computing* **8**, (1988), 391-398.

[Gro 94] Gropp, W., Lusk, E., Skjellum, A., *Using MPI: Portable Parallel Programming with the Message-Passing Interface*, MIT Press, Cambridge, MA, 1994.

[HaHa 64] Hammersley, J.M., Handscomb, D.C., *Monte Carlo Methods*, Methuen & Co., LTD, London, England, 1964.

[Jen 88] Jenkins, T.M., Nelson, W.R., Rindi, A., (editors), *Monte Carlo Transport of Electrons and Photons*, Plenum Press, New York, NY, 1988.

[Lan 97] Landau, R.H., Paez, M.J., *Computational Phsics: Problem Solving with Computers*, John Wiley & Sons, Inc., New York, NY, 1997.

[Lew 93] Lewis, E.E., Miller, W.F., *Computational Methods of Neutron Transport*, American Nuclear Society, Inc., La Grange Park, IL, 1993.

[Mar 91] Martin, W.R., "Monte Carlo Methods on Advanced Computer Architectures," *Advances in Nuclear Science and Technology* **22**, (1991), 105-164.

[Mar 92] Martin, W.R., Rathkopf, J.A., Brown, F.B. "The Impact of Advances in Computer Technology on Particle Transport Monte Carlo," LLNL, UCRL-JC-109525, 1992.

[McK 88] McKerrell, A., Delves, L.M., "Monte Carlo Simulation of Neutron Diffusion on SIMD Architectures," *Parallel Computing* **8**, (1988), 363-370.

[Saa 96] Saad, Y., *Iterative Methods for Sparse Linear Systems*, PWS Publishing Co., Boston, MA, 1996.

APPENDIX A

This appendix is included to illustrate that our random walk scheme solves the linear system $\mathbf{A}x = b$. As shown in the discussion culminating with equation (3), the diagonally scaled form of the linear system can be written as

$$x = s + \mathbf{H}x.$$

If we iteratively expand this equation to find the i^{th} element of the solution vector, we obtain

$$\begin{aligned} x_i &= s_i + (\mathbf{H}(s + \mathbf{H}x))_i \\ x_i &= s_i + (\mathbf{H}s)_i + \left(\mathbf{H}^2(s + \mathbf{H}x)\right)_i \\ x_i &= s_i + (\mathbf{H}s)_i + \left(\mathbf{H}^2 s\right)_i + \left(\mathbf{H}^3(\ldots)\right)_i . \end{aligned} \tag{28}$$

We may now look at a simple problem that involves a 2x2 linear system. If we expand the first several terms of the matrix form in (28) we obtain the form shown in (30).

$$\begin{aligned} x_i = s_i &+ \left(\begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \right)_i \\ &+ \left(\begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix}^2 \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \right)_i + \left(\begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix}^3 \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \right)_i + \ldots \end{aligned} \tag{29}$$

$$
\begin{aligned}
x_i = s_i &+ \left(\begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \right)_i \\
&+ \left(\begin{bmatrix} h_{1,1}^2 + h_{1,2}h_{2,1} & h_{1,1}h_{1,2} + h_{1,2}h_{2,2} \\ h_{2,1}h_{1,1} + h_{2,2}h_{2,1} & h_{2,1}h_{1,2} + h_{2,2}^2 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \right)_i \\
&+ \left(\begin{bmatrix} (\mathbf{H}^3)_{1,1} & (\mathbf{H}^3)_{1,2} \\ (\mathbf{H}^3)_{2,1} & (\mathbf{H}^3)_{2,2} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \right)_i + \dots
\end{aligned} \tag{30}
$$

where:

$$
\begin{aligned}
\left(\mathbf{H}^3\right)_{1,1} &= h_{1,1}^3 + h_{1,1}h_{1,2}h_{2,1} + h_{1,1}h_{1,2}h_{2,1} + h_{1,2}h_{2,1}h_{2,2} \\
\left(\mathbf{H}^3\right)_{1,2} &= h_{1,1}^2 h_{1,2} + h_{1,1}h_{1,2}h_{2,2} + h_{1,2}^2 h_{2,1} + h_{1,2}h_{2,2}^2 \\
\left(\mathbf{H}^3\right)_{2,1} &= h_{1,1}^2 h_{2,1} + h_{2,1}^2 h_{1,2} + h_{1,1}h_{2,1}h_{2,2} + h_{2,2}^2 h_{2,1} \\
\left(\mathbf{H}^3\right)_{2,2} &= h_{2,2}^3 + h_{1,1}h_{1,2}h_{2,1} + h_{1,2}h_{2,1}h_{2,2} + h_{1,2}h_{2,1}h_{2,2}
\end{aligned}
$$

If we now select $i = 1$ for instance, (30) becomes (31).

$$
\begin{aligned}
x_1 = s_1 &+ (h_{1,1}s_1 + h_{1,2}s_2) \\
&+ (h_{1,1}h_{1,1} + h_{1,2}h_{2,1})\, s_1 + (h_{1,1}h_{1,2} + h_{1,2}h_{2,2})\, s_2 \\
&+ (h_{1,1}h_{1,1}h_{1,1} + h_{1,1}h_{1,2}h_{2,1} + h_{1,2}h_{2,1}h_{1,1} + h_{1,2}h_{2,2}h_{2,1})\, s_1 \\
&+ (h_{1,1}h_{1,1}h_{1,2} + h_{1,1}h_{1,2}h_{2,2} + h_{1,2}h_{2,1}h_{1,2} + h_{1,2}h_{2,2}h_{2,2})\, s_2 + \dots
\end{aligned} \tag{31}
$$

We can draw from this a general, infinite summation form.

$$x_i = s_i + \sum_{j=1}^{n} h_{i,j} s_j + \sum_{j=1}^{n} \left(\sum_{k=1}^{n} h_{i,k} h_{k,j} \right) s_j + \sum_{j=1}^{n} \left(\sum_{l=1}^{n} h_{i,l}, \sum_{k=1}^{n} h_{i,k} h_{k,j} \right) s_j + \ldots \tag{32}$$

Hammersley and Handscomb have shown [HaHa 64] that (32) is equivalent to the expectation value of the Monte Carlo estimator for a given history. From this, we may assert that our Monte Carlo technique is a suitable estimator for the linear system $\mathbf{A}x = b$.