OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

Technical Note

Mathematics and Computer Science Division
Computational Engineering and Energy Sciences Group

From: Stuart R. Slattery
Number: CSMD-00-000
Date: April 7, 2014

Subject: A Multilevel Monte Carlo Solver for Linear Systems (Rev. 1)

Executive Summary

Monte Carlo solvers for linear systems have been demonstrated to perform poorly for strongly elliptic problems. This poor performance is primarily due to the fact that both a large number of samples are required to obtain a good statistical error and that these samples require a large amount of time to compute. As a mechanism to reduce the computational complexity for such problems and thereby improve the figure of merit of the calculation, multilevel Monte Carlo has been introduced in finance problems, the solution of stochastic partial differential equations, and other algorithms that leverage Markov chain Monte Carlo. We adapt these ideas to form a multilevel Monte Carlo solver for linear systems which, in the form presented here, is effectively a stochastic realization of a multigrid solver. Numerical studies indicate that the new multilevel method can reduce the time required to achieve a certain statistical error in the solution by at least two orders of magnitude, thereby dramatically reducing the computational complexity of the problem. Furthermore, the general formulation and results presented here indicate that this methodology may be used with both geometric and algebraic formulations of the Galerkin mutlgrid method.

1 Introduction

Monte Carlo solvers for linear systems have been in existence for decades as a stochastic alternative to iterative methods [1–5]. However, these methods have failed to gain popularity both in the mathematics and applications community partly due to their slow convergence bound by the central limit theorem. Recent work has indicated that when used as an acceleration in the Monte Carlo Synthetic Acceleration (MCSA) method, exponential convergence rates may be achieved that are competitive with contemporary iterative methods [6–9]. However, in this recent work it was discovered that for physics problems that are largely elliptic (e.g. neutron transport in a light water reactor), convergence of the Monte Carlo method is extremely slow and prohibitive for the solution of larger systems. One avenue to improve the time to solution for these calculations and to enable the solution of more difficult problems is to study preconditioning strategies as in [9]. Another approach is to instead focus on improving the time complexity of the Monte Carlo sequence independent of the condition number of the linear problem.

Recent work in Monte Carlo methods for problems in finance and stochastic partial differential equations has indicated that the computationally complexity of the problem can be dramatically reduced by incorporating multigrid concepts into the solution scheme [10–12]. In this work, we adapt those ideas and apply them to the Monte Carlo problem for linear systems as a means of reducing the computational complexity of the algorithm. To begin, we first introduce the elliptic model problem for our numerical experiments and compare the spectral behavior of the Monte Carlo method to traditional iterative smoothers that would be used with the multigrid method. Next, we present the multilevel Monte Carlo method for linear systems

using a general algebraic formulation. Finally, we present results using the model problem the demonstrate the superiority of the multilevel method in terms of time to solution using both geometric and algebraic multigrid representations of the linear problem.

2 Monte Carlo Solver Fourier Analysis

We would first like to analyze the behavior of Monte Carlo solvers in the context of error modes for a given model problem with the numerical analysis presented here closely following that presented in [13]. For this analysis, we will use the following one-dimensional, homogeneous model problem:

$$\nabla^2 x = 0\,. \tag{1}$$

We discretize the problem into N discrete points where now $\mathbf{x} \in \mathbb{R}^N$ with boundary conditions:

$$\mathbf{x}_1 = 0, \quad \mathbf{x}_N = 0\,. \tag{2}$$

The Laplacian is discretized using a standard second-order finite difference with a grid spacing of one:

$$(\nabla \mathbf{u})_i = \mathbf{u}_{i-1} - 2\mathbf{u}_i + \mathbf{u}_{i+1}\,, \tag{3}$$

which then gives the following linear problem:

$$\mathbf{A}\mathbf{x} = \mathbf{0}\,. \tag{4}$$

To bound the spectral radius of the problem we will use a Jacobi preconditioner:

$$\mathbf{M} = diag(\mathbf{A})\,, \tag{5}$$

such that we instead solve the following linear problem:

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{0}\,. \tag{6}$$

To elucidate the effect of a given solution technique on a given error mode in the problem, we can assign an initial guess of $\mathbf{x}^0$ to be a chosen Fourier mode:

$$\mathbf{x}_i^0 = \sin\left(\frac{ik\pi}{N}\right), \tag{7}$$

where $\mathbf{x}_i^0$ is the i^{th} component of the initial guess and k is the wave number of the chosen Fourier mode.

Monte Carlo solvers are effectively a stochastic realization of Richardson's iteration and therefore we will first look at the performance of Richardson's iteration as a smoother:

$$\mathbf{x}^{k+1} = (\mathbf{I} - \mathbf{M}^{-1}\mathbf{A})\mathbf{x}^k\,, \tag{8}$$

where k is the iteration index and which is equivalently a Jacobi iteration when Jacobi preconditioning is used. Figure 1 gives infinity norm of the error in the solution vector[1] as a function of iteration for wave numbers of 1, 5, and 10 on a grid with $J = 100$. Immediately we note that the larger the wave number the better Richardson's iteration performs. Per the spectral analysis in [13], this iteration sequence performs better for higher wave numbers as the eigenvalue spectrum spans a smaller space then less oscillatory modes. This behavior motivates the multigrid approach where moving a smooth mode to a coarser grid makes that mode appear more oscillatory relative to that coarser grid, thus improving convergence for that particular mode.

[1] The solution to the homogeneous problem is zero and therefore $||\mathbf{e}||_\infty = ||\mathbf{x}||_\infty$.

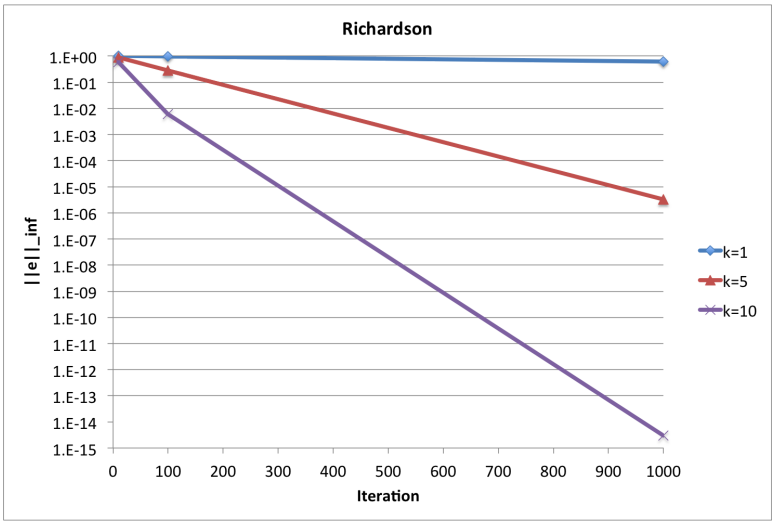


Figure 1: **Convergence of Richardson's iteration as a function of iteration for a grid of size $J = 100$.** *Richardson's iteration performs better for larger wave numbers and therefore more oscillatory modes.*

Next we perform the same calculations using the adjoint Monte Carlo solver presented in [8]. Before doing this however, we must first modify the linear problem as the Monte Carlo solver is a direct method and therefore a homogeneous problem with a right hand side of $\mathbf{0}$ will yield no samples. Instead, we will solve the residual problem:

$$\mathbf{A}\mathbf{d} = \mathbf{r}\,, \tag{9}$$

where the residual of the homogeneous problem is:

$$\mathbf{r} = -\mathbf{A}\mathbf{x}^0\,, \tag{10}$$

and the solution is computed as:

$$\mathbf{x} = \mathbf{x}^0 + \mathbf{d}\,. \tag{11}$$

Forming the problem in this way lets us directly apply the adjoint Monte Carlo method to the homogeneous problem and then apply the effective correction, $\mathbf{d}$, to the initial guess to give the solution. This approach is also equivalent to performing a single iteration of Halton's method [3]. Using this formulation we can again solve the model problem with wave numbers of 1, 5 and 10 on a grid of size $J = 100$ but this time we vary the number of histories used to compute the solution instead of the number of iterations. Figure 2 gives the results of these calculations. Surprisingly, the Monte Carlo method performs better for smooth modes than more oscillatory modes[2]. The results of these calculations are counterintuitive given the fact that the Monte Carlo solver is effectively a stochastic realization of Richardson's iteration and therefore one should expect the same spectral behavior from the results.

Looking at the timing results in Table 1, we see that the behavior of the Monte Carlo solver is in fact consistent with these expectations. Timing results show that the average time required to compute an entire history in the Monte Carlo solver decreases as a function of wave number. We showed analytically in [14] that the length of the random walk is equivalent to the number of Richardson iterations that would be required to achieve a given convergence criteria. In Figure 1, we see that fewer iterations are required to converge larger wave numbers and therefore we should also expect shorter random walks in the Monte Carlo solver and therefore a faster time to solution as observed in Table 1. However, this does not indicate why

[2]This behavior was also observed for the forward Monte Carlo method presented in [8].

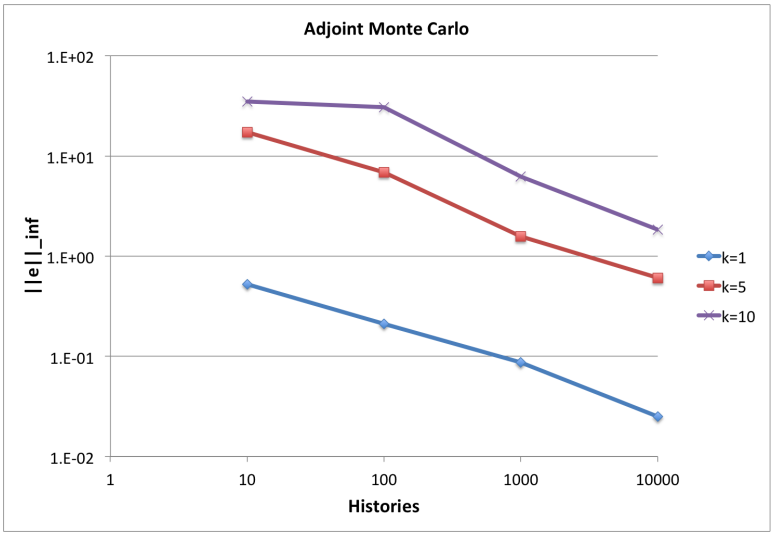


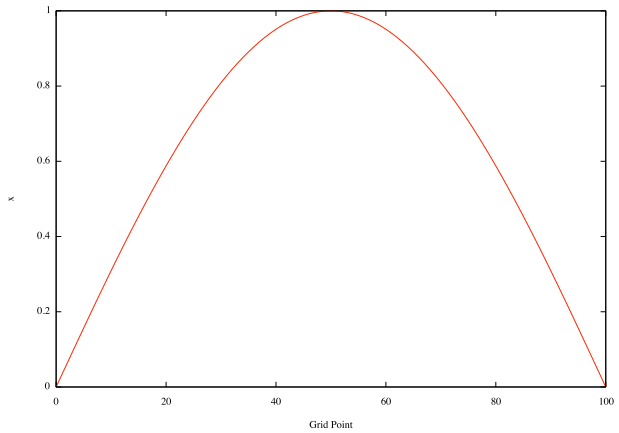
Figure 2: **Convergence of the adjoint Monte Carlo method as a function of sampled histories for a grid of size** $J = 100$**.** *Adjoint Monte Carlo performs better for smaller wave numbers and therefore smoother modes.*

Wave Number	Time per History (s)
1	1
5	0.85
10	0.83

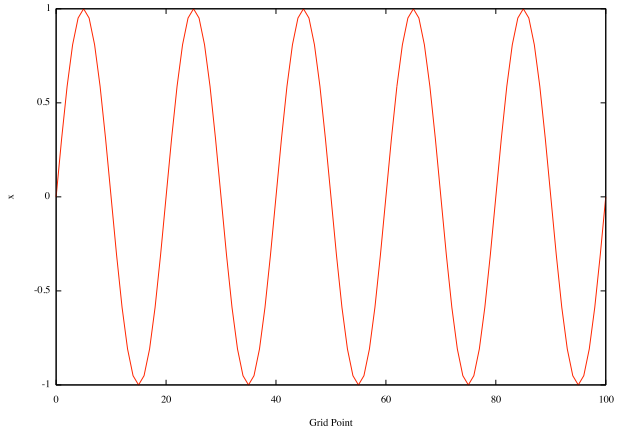
Table 1: **Normalized average CPU time per history.**

the Monte Carlo solvers perform better for smoother modes rather than oscillatory modes. Fortunately, the explanation for this behavior is simple. Consider the plots for the Fourier modes with $k = 1$ and $k = 10$ given in Figure 3. The higher the wave number is the more oscillatory the mode appears. To resolve these fine structures in the error we therefore naturally require more Monte Carlo samples to distinguish them from one another. In other words, statistical noise in the solution must be reduced by more samples in order to resolve fine structures in the error that would otherwise be buried in the noise.

Based on this, it may not seem likely that applying Monte Carlo in a multigrid context will be useful considering that error modes appear more oscillatory as the grid is coarsened. This would indicate that in order to achieve the same amount of error on a coarse grid representation of the problem, one would be required to compute many more samples to resolve the resulting finer structures. However, we may gain a computational advantage for two reasons. First, consider the timing results in Table 1. As the error structure becomes more oscillatory, the time it takes to compute a sample decreases. Second, if we coarsen the grid the time it takes to compute a sample will also decrease due to the decreased problem size. Therefore, it is plausible that we might observe an improvement in run times for the Monte Carlo problem by using a multigrid approach.



(a) **Fourier mode with $k = 1$.**



(b) **Fourier mode with $k = 10$.**

Figure 3: *The more oscillatory the mode the finer the structure of the error. Finer structures require more samples to resolve them.*

3 Multilevel Monte Carlo Algorithm

A potential improvement in computational complexity motivates applying techniques in multilevel Markov chain Monte Carlo recently developed by Heinrich [10] and Giles [11]. We start first with the standard Monte Carlo estimator for the solution vector:

$$\hat{\mathbf{x}} = \frac{1}{N}\sum_{m=1}^{N} x^m \,, \tag{12}$$

where x^m is the m^{th} observation. We next consider representing the problem in multiple levels designated by l with each one finer than the next such that there are L total levels and $l = L$ is the coarsest level. We can combine expectation values of the estimates from each of these levels using the linearity of expectation such that:

$$E(\mathbf{x}_0) = E(\mathbf{x}_L) + E(\mathbf{x}_{L-1} - \mathbf{x}_L) + E(\mathbf{x}_{L-2} - \mathbf{x}_{L-1}) + \cdots + E(\mathbf{x}_0 - \mathbf{x}_1) \,, \tag{13}$$

where $E(\mathbf{x}_L)$ is the expectation value of the solution vector at the coarsest level and $E(\mathbf{x}_0)$ the expectation at the finest level. If we rewrite this summation as:

$$E(\mathbf{x}_0) = E(\mathbf{x}_L) + \sum_{l=1}^{L} E(\mathbf{x}_l - \mathbf{x}_{l+1}) \,, \tag{14}$$

we can then build a new *multilevel estimator* for the l^{th} level as:

$$\hat{\mathbf{y}}_l = \frac{1}{N_l}\sum_{m=1}^{N_l} (x_l^m - x_{l+1}^m) \,, \tag{15}$$

where N_l are the number of samples computed at the level and $x_{l+1}^m = 0$ for $l = L$. Estimates from all levels can then be combined to give the final solution as:

$$\hat{\mathbf{x}} = \sum_{l=0}^{L} \hat{\mathbf{y}}_l \,. \tag{16}$$

It is critical to note here that the term $(x_l^m - x_{l+1}^m)$ in Eq (15) consists of observations from the same Markov chain computed over the space of the l^{th} level.

Although multigrid methods for linear algebra problems indicate that a coarsening parameter of $M = 2$ is optimal such that every coarse level is half the size of the previous finer level. However, Giles reports that values of 4 and larger give optimal performance for these particular estimators. In addition, as mentioned in § 2, because coarsening the problem makes the modes appear more oscillatory, we must compute more samples at coarser levels to balance the error amongst the levels. For this work we will use the result reported by Heinrich for computing the number of samples at each level:

$$N_l = M^{-3(L-l)/2} N \,, \tag{17}$$

which provides a larger number of samples at the coarse levels. It should be noted that Giles does not compute this quantity a priori but instead first estimates the variance at each level and then computes N_l based on that variance.

Using the estimator given by Eq (16) we can now define a multilevel Monte Carlo method for linear systems based on the adjoint Monte Carlo solver given in [8]. To do this, we must consider how to compute the expectation value $E(\mathbf{x}_l - \mathbf{x}_{l+1})$ at each level in the problem. We are required to construct observations on both grids from the same Markov chain. Computing observations on the l^{th} grid in this case is simply the procedure for solving the adjoint Monte Carlo problem. We could modify this procedure such that we tally observations on the grid for the $l + 1$ level through the use of a weight preserving linear interpolation operator such that a tally in some state of level l is also a tally in some state of level $l + 1$ with that state

determined by the operator. If this is the case, then we can actually avoid tallying observations of the coarse grid altogether and simply apply the linear interpolation operator the fine grid results once all samples have been computed instead of modifying the Monte Carlo sequence.

Using this idea we then define a *prolongation operator*, $\mathbf{P}_l$, which maps a vector defined on grid $l+1$ to a vector defined on grid l and a *restriction operator*, $\mathbf{R}_l$, which maps a vector defined on grid l to a vector defined on grid $l+1$. We can then define the following equivalent expectation value for a given level l:

$$E(\mathbf{x}_l - \mathbf{x}_{l+1}) = \Big(\mathbf{I} - \mathbf{P}_l\mathbf{R}_l\Big)\hat{\mathbf{x}}_l \,, \tag{18}$$

with $\hat{\mathbf{x}}_l$ given by Eq (12) as the solution on grid l. In this estimator, the action of the restriction operator maps the tally to the coarse level to compute the observations on that grid and the action of the prolongation operator on the coarse tally maps the result back to the fine level. Using this estimator, Algorithm 1 gives the multilevel Monte Carlo algorithm for linear systems now with an arbitrary right hand side vector, $\mathbf{b}$, and an initial solution guess, $\mathbf{x}^0$. This algorithm permits each of the levels to be computed independently and then

Algorithm 1 Multilevel Monte Carlo Method

```
for l = 0...L do
    P_l = P(A_l)                    ▷ Build the prolongation and restriction operators for the l^th level.
    R_l = c P_l^T
    r_l = b_l − A_l x_l^0           ▷ Build the l^th level residual.
    d_l = Â_l^{-1} r_l              ▷ Solve the l^th level problem with adjoint Monte Carlo
    if l != L then
        d_l = (I − P_l R_l) d_l     ▷ Apply the multilevel tally
        A_{l+1} = R_l A_l P_l       ▷ Construct the next level.
        x_{l+1}^0 = R_l x_l^0
        b_{l+1} = R_l b_l
    end if
end for
for l = L...1 do
    d_{l−1} = (I + P_l) d_l         ▷ Collapse the tallies to the finest grid
end for
x = x^0 + d_0
```

combined appropriately with the prolongation and restriction operators. Like other multigrid algorithms for linear systems, we are required to produce a representation of the linear operator on each level, $\mathbf{A}_l$, as well as the right hand side, $\mathbf{b}_l$. The formulation of Algorithm 1 places no restrictions on the form of $\mathbf{A}_l$, $\mathbf{b}_l$, $\mathbf{P}_l$, or $\mathbf{R}_l$ and therefore could be used with any multigrid scheme including those arising from algebraic multigrid formulations. It should be noted, however, that work to date has only tested the algorithm for the Galerkin formulation presented in Algorithm 1.

4 Geometric Multigrid Results

To demonstrate the multilevel Monte Carlo algorithm we solve the Poisson problem presented in § 2, this time with a grid of size $J = 1024$. For the coarsening parameter we will first use a value of $M = 2$. For simplicity, we will assume a basic geometric multigrid formulation of the linear problem per [13] with the the following prolongation operator for $M = 2$:

$$x^h_{2j} = x^{2h}_j \tag{19a}$$

$$x^h_{2j+1} = \frac{1}{2}(x^{2h}_j + x^{2h}_{j+1}) \,, \tag{19b}$$

for $0 \leq j \leq \frac{J}{2} - 1$ and the restriction operator:

$$x_j^{2h} = \frac{1}{4}(x_{2j-1}^h + 2x_{2j}^{2h} + x_{2j+1}^{2h}) \tag{20}$$

for $1 \leq j \leq \frac{J}{2} - 1$.

To measure performance of the algorithm, we will introduce the following standard figure of merit metric to assess the quality of the Monte Carlo solver:

$$FOM = \frac{1}{||\mathbf{e}||_\infty^2 T}\,, \tag{21}$$

where $||\mathbf{e}||_\infty$ is a measure of the variance of the calculation and T is the time required to solve the problem. The larger the figure of merit, the more useful the solution scheme is. For example, if two solutions schemes arrive at the same error but the first calculation takes twice as long as the second, the figure of merit will be twice as large for the second calculation.

We again perform calculations with $N = 10,000$ and initial guesses of wave numbers 1, 5 and 10 and vary the number of levels used in the calculation. Tables 2, 3, and 4 giving the results of these calculations. In these tables, the RFOM column indicates the relative figure of merit which has been normalized to the single level calculation. The results indicate the performance of the multilevel method compared to the original adjoint Monte Carlo method is excellent, offering a dramatic improvement in time to solution and figure of merit values 10-100 times that of the single level solver.

Levels	**Samples**	$\|\|\mathbf{e}\|\|_\infty$	**Time (s)**	**RFOM**
1	10,000	0.022	119.1	1
2	13,535	0.026	72.0	1.14
3	14,785	0.035	33.2	1.38
4	15,226	0.039	13.7	2.63
5	15,382	0.027	5.5	13.86
6	15,437	0.036	2.3	19.21
7	15,456	0.045	0.98	28.00
8	15,462	0.081	0.41	20.66
9	15,464	0.267	0.17	11.72

Table 2: **Geometric Multilevel Monte Carlo results for $k = 1$ with $M = 2$, $N = 10,000$ and run times reported in seconds.**

Next we perform the calculations with $M = 4$ using the following prolongation operator:

$$x_{4j}^h = x_j^{4h} \tag{22a}$$

$$x_{4j+1}^h = \frac{3}{4}x_j^{4h} + \frac{1}{4}x_{j+1}^{4h} \tag{22b}$$

$$x_{4j+2}^h = \frac{1}{2}x_j^{4h} + \frac{1}{2}x_{j+1}^{4h} \tag{22c}$$

$$x_{4j+3}^h = \frac{1}{4}x_j^{4h} + \frac{3}{4}x_{j+1}^{4h}\,, \tag{22d}$$

for $0 \leq j \leq \frac{J}{4} - 1$ and the restriction operator:

$$x_j^{4h} = \frac{1}{10}(x_{4j-2}^h + 2x_{4j-1}^h + 4x_{4j}^{4h} + 2x_{4j+1}^{4h} + x_{4j+2}^{4h}) \tag{23}$$

Levels	Samples	$\|\|e\|\|_\infty$	Time (s)	RFOM
1	10,000	0.668	101.6	1
2	13,535	0.381	60.0	5.21
3	14,785	0.396	28.0	10.34
4	15,226	0.599	11.9	10.64
5	15,382	0.638	4.7	23.60
6	15,437	1.052	1.8	23.15
7	15,456	1.070	0.67	59.16
8	15,462	1.180	0.23	144.10
9	15,464	2.130	0.10	99.95

Table 3: **Geometric Multilevel Monte Carlo results for $k = 5$ with $M = 2$, $N = 10,000$ and run times reported in seconds.**

Levels	Samples	$\|\|e\|\|_\infty$	Time (s)	RFOM
1	10,000	1.81	98.6	1
2	13,535	1.67	59.6	1.94
3	14,785	2.41	27.1	2.05
4	15,226	3.20	11.5	2.74
5	15,382	1.85	4.93	19.14
6	15,437	3.89	1.95	10.95
7	15,456	3.79	0.78	28.68
8	15,462	7.08	0.30	21.62
9	15,464	11.7	0.11	21.45

Table 4: **Geometric Multilevel Monte Carlo results for $k = 10$ with $M = 2$, $N = 10,000$ and run times reported in seconds.**

for $1 \leq j \leq \frac{J}{4} - 1$. For these calculations we fix $k = 10$ and vary N to study performance for the more difficult larger wave numbers. Tables 5, 6, and 7 give the results for $N = 10,000$, $N = 100,000$, and $N = 1,000,000$ respectively. Because of the time required to compute problems with a small number of levels and large numbers of samples, single level calculations were not performed for the last two cases. For these cases, relative figure of merit values were computed relative to the single level case with $N = 10,000$.

Although Giles reports that $M \geq 4$ provides good performance in the context of his calculations, the

Levels	Samples	$\|\|e\|\|_\infty$	Time (s)	RFOM
1	10,000	1.81	98.6	1
2	11,250	2.64	18.2	2.53
3	11,406	2.61	3.1	15.15
4	11,425	4.56	0.47	33.04
5	11,427	16.5	0.07	16.95

Table 5: **Geometric Multilevel Monte Carlo results for $k = 10$ with $M = 4$, $N = 10,000$ and run times reported in seconds.**

improvement is not as dramatic for these cases. A coarser discretization of $M = 4$ does provide a small improvement in the time required to compute a single sample and therefore enables slightly higher figures

Levels	Samples	$\|\|e\|\|_\infty$	Time (s)	RFOM
3	114,062	0.76	28.3	19.63
4	114,275	1.53	4.4	31.43
5	114,281	4.60	0.67	22.78

Table 6: **Geometric Multilevel Monte Carlo results for $k = 10$ with $M = 4$, $N = 100,000$ and run times reported in seconds.**

Levels	Samples	$\|\|e\|\|_\infty$	Time (s)	RFOM
4	1,142,758	0.56	41.8	25.00
5	1,142,822	1.42	6.49	24.68

Table 7: **Geometric Multilevel Monte Carlo results for $k = 10$ with $M = 4$, $N = 1,000,000$ and run times reported in seconds.**

of merit. However, these results do not rule out using $M = 2$. This is an important result in that both geometric and algebraic multigrid methods utilize $M = 2$ and therefore we may leverage the interpolation operators and other infrastructure developed for those methods within the multilevel Monte Carlo scheme presented here without a dramatic increase in computational complexity.

5 Algebraic Multigrid Results

We next solve the same model problem, this time using an algebraic multigrid formulation of the Galerkin problem. For these calculations we used the ML multigrid library to construct $\mathbf{A}$, $\mathbf{P}$, and $\mathbf{R}$ on each level with $\mathbf{M} \approx 3$ [15]. Tables 8, 9, and 10 give the results of these calculations. In general, the results show similar

Levels	Samples	$\|\|e\|\|_\infty$	Time (s)	RFOM
1	10,000	0.022	119.1	1
2	11,924	0.037	31.9	1.28
3	12,294	0.055	7.6	2.40
4	12,365	0.051	1.8	11.76
5	12,378	0.094	0.5	12.01

Table 8: **Algebraic Multilevel Monte Carlo results for $k = 1$ with $M = 3$, $N = 10,000$ and run times reported in seconds.**

performance to the geometric multigrid method, however, gains in figure of merit were not as significant which may be in part due to the simplified selection of sample size at each level. In addition, the Dirichlet conditions for the model problem were explicitly included in the geometric interpolation operators while the level operators generated by the algebraic scheme did not. Most importantly, these results indicate that Algorithm 1 can be used with general Galerkin formulations of the multigrid problem.

Levels	**Samples**	$\|\|\mathbf{e}\|\|_\infty$	**Time (s)**	**RFOM**
1	10,000	0.668	101.6	1
2	11,924	0.834	30.8	2.12
3	12,294	1.140	7.5	4.63
4	12,365	0.975	1.6	29.09
5	12,378	2.240	0.4	25.10

Table 9: **Algebraic Multilevel Monte Carlo results for $k = 5$ with $M = 3$, $N = 10,000$ and run times reported in seconds.**

Levels	**Samples**	$\|\|\mathbf{e}\|\|_\infty$	**Time (s)**	**RFOM**
1	10,000	1.81	98.6	1
2	11,924	2.33	30.0	1.99
3	12,294	3.19	7.6	4.20
4	12,365	3.31	1.8	16.29
5	12,378	9.98	0.5	7.21

Table 10: **Algebraic Multilevel Monte Carlo results for $k = 10$ with $M = 3$, $N = 10,000$ and run times reported in seconds.**

References

[1] G. E. Forsythe and R. A. Leibler, "Matrix inversion by a Monte Carlo method," *Mathematical Tables and Other Aids to Computation*, vol. 4, pp. 127–129, July 1950. ArticleType: research-article / Full publication date: Jul., 1950 / Copyright 1950 American Mathematical Society.

[2] W. Wasow, "A note on the inversion of matrices by random walks," *Mathematical Tables and Other Aids to Computation*, vol. 6, pp. 78–81, Apr. 1952.

[3] J. H. Halton, "Sequential Monte Carlo," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 58, no. 01, pp. 57–78, 1962.

[4] J. M. Hammersley and D. C. Handscomb, *Monte Carlo Methods*. Methuen, 1964.

[5] J. Spanier and E. M. Gelbard, *Monte Carlo Principles and Neutron Transport Problems*. New York: Dover Publications, 1969.

[6] T. Evans, T. Urbatsch, H. Lichtenstein, and Morel, "A residual Monte Carlo method for discrete thermal radiative diffusion," *Journal of Computational Physics*, vol. 189, pp. 539–556, Aug. 2003.

[7] T. Evans and S. Mosher, "A Monte Carlo synthetic acceleration method for the non-linear, time-dependent diffusion equation," *American Nuclear Society - International Conference on Mathematics, Computational Methods and Reactor Physics 2009*, 2009.

[8] T. Evans, S. Mosher, S. Slattery, and S. Hamilton, "A Monte Carlo synthetic-acceleration method for solving the thermal radiation diffusion equation," *Journal of Computational Physics*, vol. 258, pp. 338–358, 2013.

[9] S. R. Slattery, *Parallel Monte Carlo Synethetic Acceleration Methods for Discrete Transport Problems*. PhD nuclear engineering and engineering physics, University of Wisconsin-Madison, Madison, WI, Sept. 2013.

[10] S. Heinrich, "Multilevel Monte Carlo methods," *ICLSSC*, vol. LNCS 2179, pp. 58–67, 2001.

[11] M. Giles, "Multilevel Monte Carlo path simulation," *Operations Research*, vol. 56, no. 3, pp. 607–617, 2008.

[12] K. Cliffe, M. Giles, R. Scheichl, and A. Teckentrup, "Multilevel Monte Carlo methods and applications to elliptic pdes with random coefficients," *Comput Visual Sci*, vol. 14, pp. 3–15, 2011.

[13] W. L. Briggs, *A Multigrid Tutorial*. Pennsylvania: SIAM, 1987.

[14] S. Slattery, P. Wilson, and T. Evans, "A spectral analysis of the domain decomposed Monte Carlo method for linear systems," *American Nuclear Society - International Conference on Mathematics, Computational Methods and Reactor Physics 2013*, 2013.

[15] M. W. Gee, C. Siefert, J. Hu, R. Tuminaro, and M. Sala, "ML 5.0 smoothed aggregation user's guide," Technical Report SAND2006-2649, Sandia National Laboratories, Feb. 2007.

Distribution

Stuart Slattery <slatterysr@ornl.gov>
Tom Evans <evanstm@ornl.gov>
Steven Hamilton <hamiltonsp@ornl.gov>