International Conference on Mathematics, Computational Methods & Reactor Physics (M&C 2009)
Saratoga Springs, New York, May 3-7, 2009, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2009)

AN HP-ADAPTIVITY APPROACH FOR MONTE CARLO TALLIES

Brian C. Franke and Ronald P. Kensek
Sandia National Laboratories
P.O. Box 5800, MS 1179
Albuquerque, NM 87185
bcfrank@sandia.gov; rpkense@sandia.gov

ABSTRACT

We present a scheme for adaptively refining both spatial binning and functional-expansion tallies within each spatial bin in a Monte Carlo calculation. Our approach attempts to minimize the degrees of freedom required to achieve a user-specified level of the 2-norm error of the representation. Other goals could be implemented. Such adaptivity could be applied to any distribution tally. We demonstrate the method for problems that are three-dimensional.

Key Words: adaptivity, Monte Carlo, functional-expansion tallies

1. INTRODUCTION

Adaptivity based on a posteriori error-estimation techniques has been extensively developed for finite-element methods, to automatically reduce solution error and improve code usability [1]. Adaptivity of tallies in Monte Carlo codes could provide similar improvements. Because the Monte Carlo transport of particles does not depend on the result tallies, adaptivity may be more robust than in finite-element methods, in which the results of one element affect the solution of the adjacent elements. This may make it possible to be more aggressive in the adaptive methods. Aggressive adaptivity is also desirable because of the greater expense of Monte Carlo calculations. Our method adaptively coarsens or refines both the spatial binning and the functional-expansion tallies within each spatial bin. We spatially divide the problem domain using a binary space-partitioning tree using arbitrarily positioned axis-aligned cutting planes. We use functional-expansion tallies with Legendre polynomial basis functions.

Functional-expansion tallies (FET) were first demonstrated to be more effective than binned tallies for some problems in the mid-1970's [2-3]. Recently, Griesheimer et al. explored FET in Monte Carlo calculations and compared the sources of error in FET and uniform binning refinement [4]. In previous work, we have extended FET error estimation to three-dimensional tallies and demonstrated an adaptivity approach [5]. Here, we investigate methods for including spatial adaptivity to achieve more efficient solution representations.

Adaptive Monte Carlo tallies were used by Booth in an effort to automate phase-space partitioning for arriving at a sufficiently detailed importance map [6]. That work used no tally moments and was entirely "h-adaptive." Adaptive functional tallies were used by Lai and Spanier for importance sampling to demonstrate geometric convergence of biasing in Monte Carlo [7]. Booth further demonstrated that adaptive importance sampling could achieve

geometric convergence by using both functional-expansion and spatial-partitioning adaptivity [8]. This work differs from those two previous efforts in several ways. The prior "adaptive" techniques used a fixed number of Monte Carlo tally moments and adapted the number of moments used to bias the forward calculation based on whether the moment tallies had achieved a low statistical uncertainty. We adapt the number of Monte Carlo tally moments based on error estimation and aim to achieve a specified degree of accuracy everywhere in the global solution representation. Booth [8] achieved spatial adaptivity by examining the effectiveness of the forward biasing and comparing point-wise estimates of importance against the functional-expansion representation. We achieve spatial adaptivity by determining regions of the problem where the functional expansion either converges too slowly or can be more efficient when spatial refinement is applied.

In Section 2, we present the error estimation technique that is used to drive the adaptivity algorithm. As discussed in Section 3, the functional-expansion error estimation forms the basis for both the h-adaptivity and p-adaptivity aspects of the algorithm. That section presents the details of our hp-adaptivity method and discusses what we perceive to be the key remaining deficiencies. In Section 4, we present some detailed results from using the hp-adaptivity method on a test problem. In Section 5, we compare the hp-adaptivity algorithm against several alternative approaches in terms of both the (reduced) memory usage and the (increased) computational expense. In Section 6, we offer a few concluding remarks.

2. FUNCTION-EXPANSION-TALLY (FET) ERROR METRICS

The error metrics presented here are an extension of the one-dimensional error metrics derived by Griesheimer et al. [4]. We use Legendre polynomials to form a three-dimensional functional expansion of a distribution:

$$f(x,y,z) = \sum_{n=0}^{\infty}\sum_{m=0}^{\infty}\sum_{l=0}^{\infty} b_{n,m,l} k_n k_m k_l P_n(x) P_m(y) P_l(z) , \tag{1}$$

where the normalization constants k_n, k_m, and k_l are defined as

$$k_n = \frac{2n+1}{2} \tag{2}$$

and the coefficients are defined as

$$b_{n,m,l} = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} f(x,y,z) P_n(x) P_m(y) P_l(z)\ dx\ dy\ dz . \tag{3}$$

In order to work with normalized functions, so that our error metrics are independent of the magnitude of the distribution, we define normalized coefficients: $a_{n,m,l} = b_{n,m,l} / b_{0,0,0}$. Also for convenience in our analysis, we organize the terms by their order and perform an equivalent pair of summations of the expansion coefficients:

$$f(x,y,z) = b_{0,0,0} \sum_{j=0}^{\infty} \sum_{\left\{ \begin{smallmatrix} n,m,l \mid \\ n+m+l=j \end{smallmatrix} \right\}} a_{n,m,l} k_n k_m k_l\ P_n(x) P_m(y) P_l(z) . \tag{4}$$

In any practical application, we truncate the expansion to a finite order, J. We denote the truncated functional expansion as $\tilde{f}(x,y,z)$. We also estimate the expansion coefficients in a

Monte Carlo evaluation of Eq. (3). We denote the estimated coefficients as $\hat{a}_{n,m,l}$ with an estimated standard deviation of $\hat{\sigma}_{\hat{a}_{n,m,l}}$ yielding an approximate function $\hat{f}(x,y,z)$.

There are two sources of error in our approximation of the true distribution: truncation error in our expansion and statistical error in our Monte Carlo estimation of the coefficients. Using the functional expansions and the orthogonality of the Legendre polynomials, we find the volume-averaged 2-norm error of truncation is:

$$\left\|\tilde{E}_J\right\| = \sqrt{\frac{1}{8}\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1}\left(\frac{f(x,y,z)}{b_{0,0,0}/8} - \frac{\tilde{f}(x,y,z)}{b_{0,0,0}/8}\right)^2 dx\,dy\,dz} = \sqrt{8\sum_{j=J+1}^{\infty}\sum_{\left\{\substack{n,m,l \mid \\ n+m+l=j}\right\}} a_{n,m,l}^2 k_n k_m k_l}\,, \tag{5}$$

where *J* is the order at which the expansion is truncated. Factors of 1/8 are due to volume averaging, and $b_{0,0,0}/8$ is the volume averaged value of the function $f(x,y,z)$. Similarly, we find the volume-averaged 2-norm error due to statistics is:

$$\left\|\hat{E}_J\right\| = \sqrt{\frac{1}{8}\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1}\left(\frac{\tilde{f}(x,y,z)}{b_{0,0,0}/8} - \frac{\hat{f}(x,y,z)}{\hat{b}_{0,0,0}/8}\right)^2 dx\,dy\,dz} = \sqrt{8\sum_{j=0}^{J}\sum_{\left\{\substack{n,m,l \mid \\ n+m+l=j}\right\}} \left(a_{n,m,l} - \hat{a}_{n,m,l}\right)^2 k_n k_m k_l} \tag{6}$$

Following Griesheimer et al. [4], we approximate these as:

$$\left\|\tilde{E}_J\right\|^2 \approx 8\sum_{j=J+1}^{I}\sum_{\left\{\substack{n,m,l \mid \\ n+m+l=j}\right\}} \hat{a}_{n,m,l}^2 k_n k_m k_l\,, \tag{7}$$

where *I* is the maximum order for which data has been computed, and

$$\left\|\hat{E}_J\right\|^2 \approx 8\sum_{j=0}^{J}\sum_{\left\{\substack{n,m,l \mid \\ n+m+l=j}\right\}} \hat{\sigma}_{\hat{a}_{n,m,l}}^2 k_n k_m k_l\,. \tag{8}$$

The subject of using this data to determine the required expansion order and run time to achieve a specified error level is dealt with in greater detail in Ref. 5, but we will summarize the method here. If there is not sufficient statistically valid data, an estimate is made of the additional run time required. Provided enough statistically valid data, we can estimate the truncation error as a function of expansion order. For distributions with a finite algebraic index of convergence, we expect the truncation error to follow a power law. Thus, using a power-law fit to the low-order truncation error data, we can predict the order necessary to achieve a specified truncation error. Since statistical error increases in a predictable relation to expansion order (as the square root of order) and decreases as a function of Monte Carlo run time (inversely as the square root of the number of histories), we can estimate the run time required to a desired statistical error level at the necessary truncation order. In the next section we discuss how this estimate of a required expansion order (and run time) can be used to find an optimal spatial division to minimize the memory requirements for achieving the specified error level.

3. FET-BASED HP-ADAPTIVITY

The h-adaptivity of our algorithm is built on a binary space-partitioning tree. The p-adaptivity of our algorithm is built on the functional-expansion moments. Adaptivity in the functional expansion is driven by the error metrics described in the previous section. It would be highly desirable to have similar error metrics to guide adaptivity in spatial partitioning. Such metrics allow predictive extrapolation and enable more aggressive adaption. As far as we know such information has only been derived within the constraints of a uniform-binning approach [4]. Instead, we have adopted a trial-and-error approach to comparing the spatial-partitioning error for limited numbers of possible tree cuts. This is less aggressive but more memory efficient.

For simplicity, we constrain the tree cut planes to be axis aligned, which results in a "kd-tree." We require the root of the tree to be an axis-aligned box, and since all cut planes are axis aligned, all nodes of the tree are also axis-aligned boxes. Because we use a binary tree, each node containing a cut plane is divided into two nodes. A node that is not cut is a "leaf" of the tree. Because we use a spatial-partitioning tree, each point of space within the root box is uniquely contained within a single leaf of the tree.

To obtain sufficient data on which to base our partitioning decisions, we calculate distribution data as discontinuous functional expansions on a regular grid within each tree leaf. Each regular grid is defined as an N□N□N division of the leaf's box into spatial "cells." A 4th order functional expansion (or higher order specified by the user) is calculated within each cell of the grid.

A Monte Carlo tally at any point in space is only performed for a single grid cell functional expansion. In the adaptivity algorithm, when we wish to evaluate the functional-expansion error-convergence rate for a prospective spatial division, we perform three-dimensional quadrature integration [9] across the functional data stored on grid cells. Thus, these alternative spatial divsions may arbitrarily encompass any portion of one or more grid cells and may span one or more leaves.

The goal of our adaptivity algorithm is to minimize the number of variables required to achieve a user-specified level for a volume-averaged 2-norm error criterion. To achieve this we calculate Monte Carlo estimates of functional-expansion coefficients in each cell of the regular grid on each leaf of the tree. The tree may begin as a single leaf. Based on the grid data, there are four ways in which the tree may be adapted:

1. We will attempt to isolate any grid cell where the estimated order-of-representation required to meet the error criterion is greater than a maximum limit. This maximum limit is used due to the difficulty of accurately projecting error to very high orders (short of calculating moments to very high orders) and the dubious comparisons of such extrapolations that would result. The algorithm for doing this attempts to identify tree cutting planes along grid boundaries that will most effectively partition space to isolate regions requiring high-order representations.

2. We estimate the required order-of-representation for the unmodified leaf (by performing numerical integration across the Monte-Carlo-calculated grid-cell data and determining an error-convergence rate) and compare the number of degrees of freedom against the results for a selection of possible tree cuts (by similar numerical integrations and analysis). We choose the representation that yields the minimum degrees of freedom from this set of alternatives. If the order-of-representation required for the unmodified leaf is greater than the maximum limit, the leaf will be divided, regardless of the outcome of the degree-of-freedom comparisons.

3. All tree cuts are re-evaluated to determine whether eliminating cuts will reduce the degrees of freedom required to meet the user-specified error criterion (while not exceeding the maximum order-of-representation limit). This re-evaluation is done by performing numerical integration over the data stored in the tree leaves encompassed by each higher level node in the tree. The degrees of freedom required for the child nodes are compared with the requirements of the parent node at each level of the tree to decide whether a tree cut should be eliminated.

4. If a tree leaf is not being modified, the functional-expansion order is adjusted to the estimated optimum.

Continuing to use the same grid-cell data, steps 1 and 2 of this process may be applied repeatedly to further partition space. When the process of refining and coarsening the spatial partitioning and functional expansions has completed, another Monte Carlo calculation is performed to repopulate the tree with data. Iteration between tree refinement and Monte Carlo calculation continues until the user is satisfied that the results have converged.

There are many improvements that could be applied to our implementation to reduce the cost of the Monte Carlo calculations and to make the adaptivity algorithm more robust. First, the last step of the adaptivity algorithm has not been integrated into the process. That is, p-adaptivity of tree leaves is only being performed when h-adaptivity has been halted; only the estimation of p-adaptivity error is currently being performed while spatial partitioning cuts are being made. As will be seen later, this is why we allow some leaves to adapt to representation orders beyond the "maximum limit." There have been a few cases where this has resulted in a leaf that needed an excessively high-order representation and the existing algorithm was clearly deficient. Second, we have not attempted to integrate the adaptivity algorithm into the Monte Carlo calculation. This could be done in a variety of ways, with differing degrees of implementation cost and potential efficiency-improvement payoffs. The simplest improvement would be to carry over Monte Carlo result data. When a tree leaf is not modified, the functional-expansion tallies remain valid. And increasing the order of representation does not invalidate the low-order tallies. There is no need to reset the tallies, as long as the number of histories contributing to each tally is properly recorded. A more tightly integrated approach could embed the adaptivity within the Monte Carlo calculation to assess whether cell refinement is warranted after some fixed number of incremental tallies. Third, the algorithm for determining the optimal tree cut is currently only evaluating one possible cut at a time. More complicated algorithms could determine an optimal series of cuts. Fourth, the p-adaptivity method is constrained to adapting to a single order of representation (see Eq. (4)). A more efficient representation might be achieved

in some cases by having a high order representation for one or two axial components of the functional expansion and a low-order expansion for the other component(s).

4. HP-ADAPTIVITY RESULTS

The test function used by Griesheimer et al. was

$$g(x)=\begin{cases}\cos(x)\exp(2x+1), & -1.0\le x<-0.5\\ \cos(x), & -0.5<x<0.5\\ \cos(x)\exp\left[-(2x+1)/4\right], & 0.5<x\le 1.0\end{cases}. \tag{7}$$

For our test function, we simply extend this to a three-dimensional distribution as

$$f(x,y,z)=g(x)g(y)g(z). \tag{8}$$

The distribution is C^0 continuous but contains planar surfaces of C^1 discontinuities (i.e., discontinuities in the derivative) at -0.5 and 0.5 in each of the coordinate axes. We use samples from this distribution to perform Monte Carlo integration of the coefficients defined by Eq. (3). We restrict evaluation to the range (-1.0, 0.9) in all three axes in order to prevent the algorithm from being capable of exactly determining that tree cuts should be placed on the C^1 discontinuities. A cut-away of the distribution is shown in Figure 1.

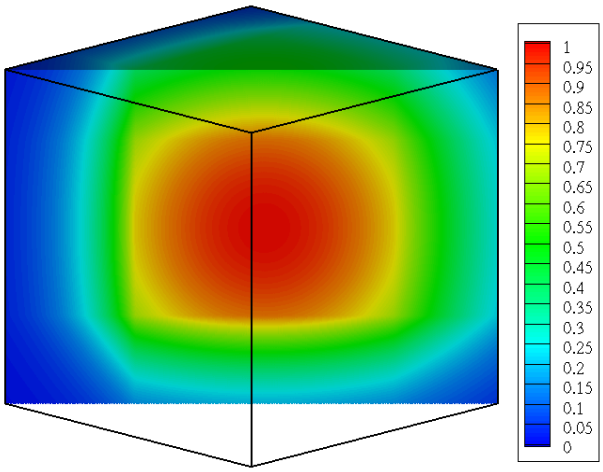

Figure 1. The test-function distribution.

Two sets of results were generated by allowing the adaptivity algorithm to iterate on our test function until converged to a 0.005 and 0.001 volume-averaged 2-norm error criterion. The tree-cut boundaries and the functional-expansion orders for these two cases are shown in Figure 2. The maximum-order limit was 10 in this test. As previously noted, functional-expansion orders are allowed to settle to an expansion order beyond what was desired. We note that tree cuts were placed very near C^1 discontinuities or were placed repeatedly on either side in an attempt to isolate them. Many more cuts are required for the more stringent error criterion.

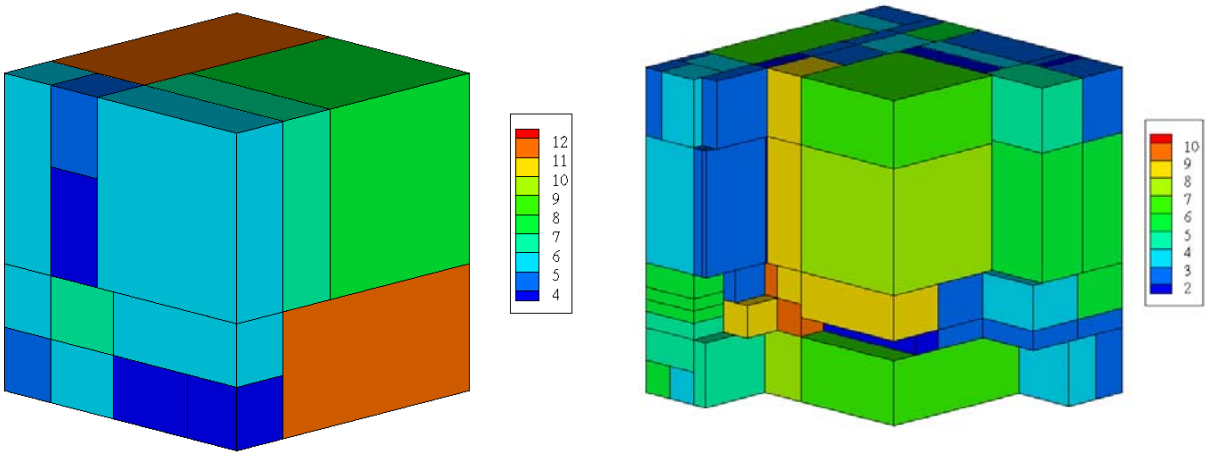

Figure 2. The spatial partitioning and functional-expansion order for converged results for a 0.005 volume-averaged 2-norm error criterion on the left and a 0.001 error criterion on the right.

Cut-aways of the point-wise relative error are shown in Figure 3. The error in the plot is measured point-wise, whereas the algorithm is based on a volume-averaged 2-norm error. The highest errors are near C^1 discontinuities and at corners of leaves. Throughout much of the problem the error is near the desired level.

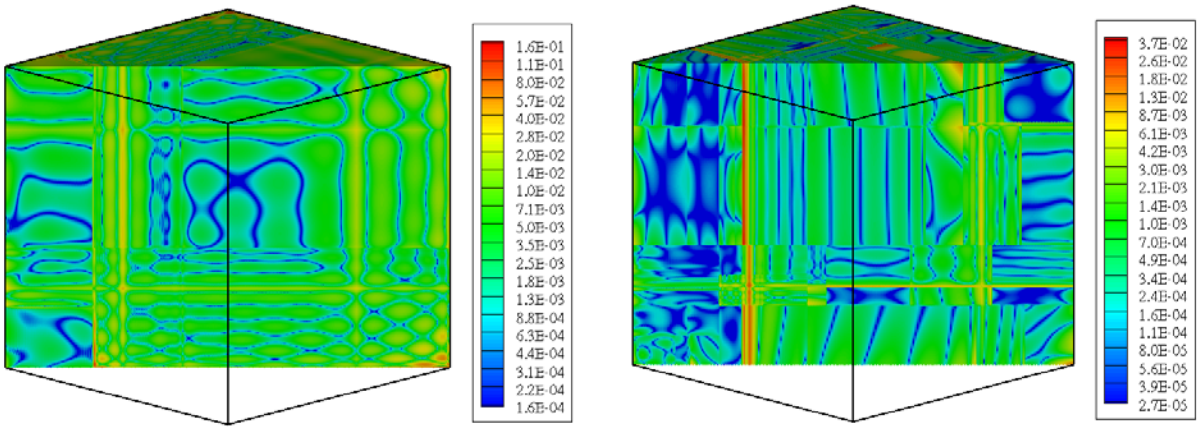

Figure 3. Absolute point-wise relative error of the converged functional distribution, $\left|1-\hat{f}/f\right|$, for a 0.005 volume-averaged 2-norm error criterion on the left and a 0.001 error criterion on the right.

5. HP-ADAPTIVITY VERSUS ALTERNATIVE APPROACHES

The previous section presented some evidence that the hp-adaptivity method could be used to achieve a desired level of error. In this section we address how this approach compares with some alternatives. We present evidence that our approach generally uses memory as efficiently as (or more efficiently than) either h-adaptivity or p-adaptivity alone. Our objective was to decrease memory usage and improve ease-of-use, but since this is accomplished by iterating on Monte Carlo calculations, we also present some estimates of the added computational expense.

This may allow an assessment of whether the trade-off is acceptable for some applications. To limit the run-time requirements in performing the following studies to compare memory requirements, these results were acquired using deterministic methods (i.e., using three-dimensional quadrature integration) for all aspects of the analysis and neglecting the effects of statistical uncertainty that would be present with Monte Carlo calculations.

Here we consider three test functions. The first was described by Eqs. (7) and (8). The second test function is similar to the first, but with C^0 discontinuities included:

$$g(x)=\begin{cases}9\cos(x)/10, & 0\le|x|<0.25\\ \cos(x), & 0.25\le|x|<0.5\\ \cos(x)\exp\left[-(2x+1)/4\right]/2, & 0.5\le|x|\le 0.75\\ 11\cos(x)\exp\left[-(2x+1)/4\right]/20, & 0.75\le|x|\le 1.0\end{cases}. \tag{9}$$

Again, we extend this to a three-dimensional distribution by Eq. (8).

The third test function is:

$$g_1(x)=\begin{cases}1+\cos(10\pi x)\exp(-2x-3.5), & abs(x)\ge 0.1\\ 1+\cos(10\pi x)\exp(-2x-3.5)-0.2\cos(5\pi x), & abs(x)<0.1\end{cases}$$
$$g_2(y)=1+\cos(3\pi y)\exp(-2y-3.5), \tag{9}$$
$$g_3(z)=0.75+0.25\cos(\pi z)$$
$$f(x,y,z)=g_1(x)g_2(y)g_3(z)$$

This contains only two C^1 discontinuities.

These three test functions were evaluated using four methods: the hp-adaptivity method previously described, uniform h-refinement, uniform p-refinement, and an h-adaptivity method. The h-adaptivity method was based on a simple algorithm of comparing neighboring leaves of a tree. If the ratio of the average values of two neighbors is greater than a threshold value, then both are divided in half parallel to the plane of their shared face. For each method, a parameter was varied that was expected to improve the accuracy, and the 2-norm of the relative error was recorded as function of the number of distribution variables required for the representation. For uniform h-refinement, the parameter was the depth to which the binary tree was uniformly divided. For uniform p-refinement, the parameter was the functional-expansion order. For h-adaptivity, the parameter was the threshold value for determining whether neighboring leaves were to be divided. For hp-adaptivity, the parameter was the requested volume-averaged 2-norm error level. The plots of error versus degrees of freedom are shown in Figures 4-6.

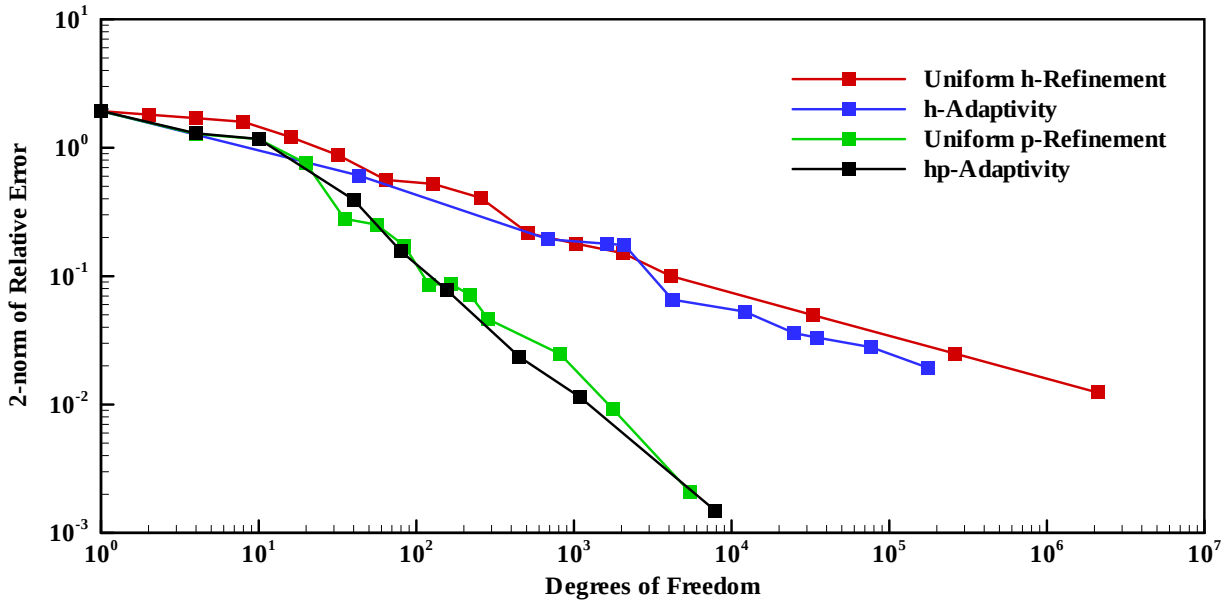

Figure 4. Error versus degrees of freedom for the first test function.

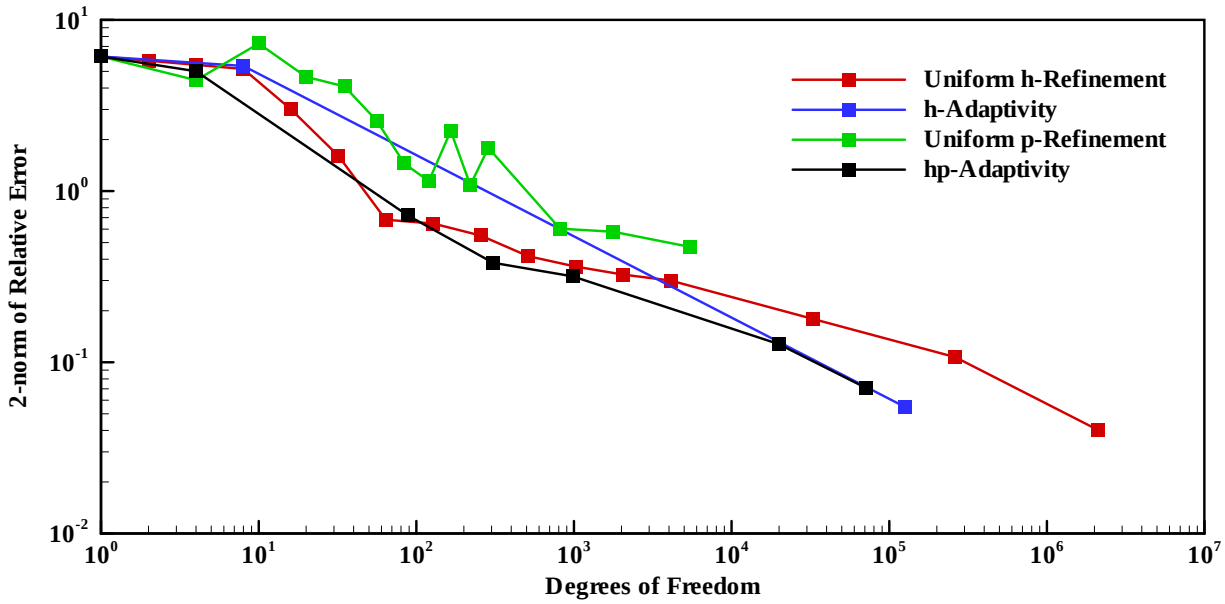

Figure 5. Error versus degrees of freedom for the second test function.

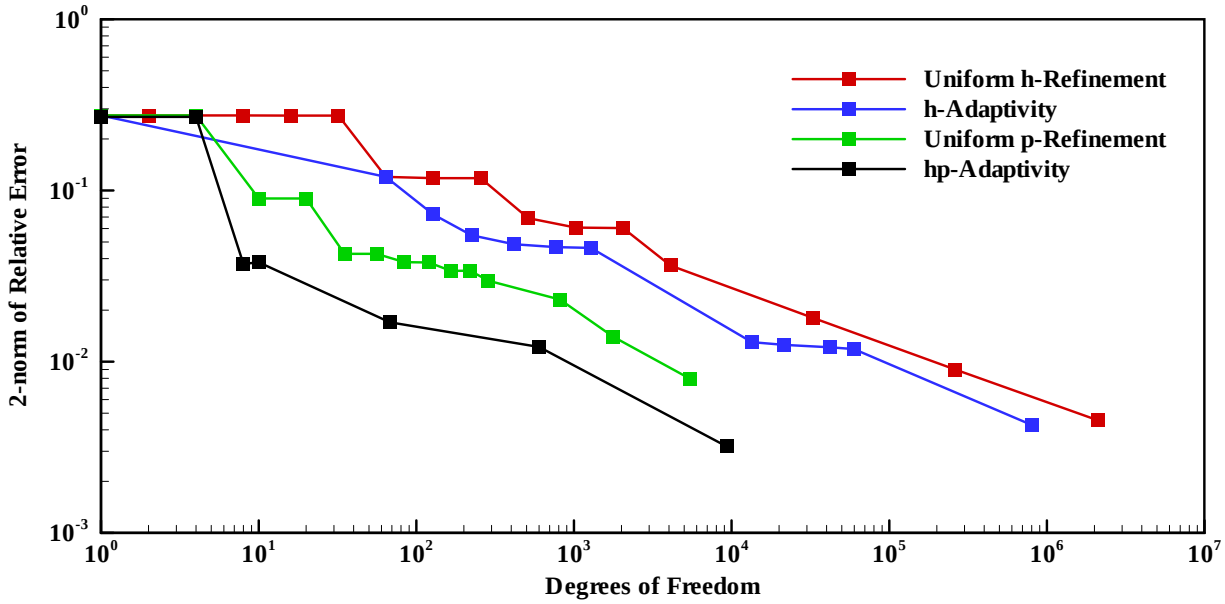

Figure 6. Error versus degrees of freedom for the third test function.

By these measures, hp-adaptivity is almost always performing as well or better than the best alternative. The h-adaptivity generally performs slightly better than the uniform h-refinement (but these test problems were not selected to highlight the difference between them). The h-adaptivity method employed was not appropriate for the second test problem, because it is not suited to problems with C^0 discontinuities; that is, when the discontinuity in the distribution exceeds the threshold for dividing neighboring leaves, the algorithm will make divisions that do not improve the representation and may attempt to make divisions interminably.

To estimate the run-time requirements for each method, we selected an error level and performed the Monte Carlo integrations to include the contributions of statistical error. To achieve a 2-norm error level of approximately 0.02 with the first test function, we found the run-time requirement for hp-adaptivity to be about 10,000 processor-seconds. This compared with an estimated run-time requirement for h-refinement of about 2900 processor-seconds and p-refinement of about 120 processor-seconds. For a 2-norm error level of approximately 0.5 with the second test function, we found the run-time requirement for hp-adaptivity to be about 440 processor-seconds. This compared with an estimated run-time requirement for h-refinement of about 0.4 processor-seconds and p-refinement of about 40 processor-seconds. For a 2-norm error level of approximately 0.02 with the third test function, we found the run-time requirement for hp-adaptivity to be about 6000 processor-seconds. This compared with an estimated run-time requirement for h-refinement of about 210 processor-seconds and p-refinement of about 60 processor-seconds.

The run times provided are coarse estimations, intended only to give the reader a sense of the cost, and there are several important caveats. The run times cited for the h-refinement and p-refinement are based only on running a calculation at the refinement level already known to give the desired error level; the expense of Monte Carlo calculations and analysis for adaptively finding the refinement needed to yield a desired error level is not included. Our expectation was

that the ratio of run time between a p-refinement calculation and an h-refinement calculation would be approximately proportional to the square of the ratio of the degrees of freedom; the h-refinement calculations consistently outperformed this. We suspect that the p-refinement calculation was less efficient than expected because the Monte Carlo calculation expense consists almost entirely of the tally expense and the moment tallies are more expensive than the spatially binned tallies (as implemented). In a realistic Monte Carlo calculation, where run time is dominated by the physics of the simulation, we would expect moment tallies to perform better than the results shown here. The time required for the hp-adaptivity to converge can vary depending upon the cuts that are chosen, which can be affected by the statistical variation of the data. In all cases, we aimed for having 20% of the error due to statistical uncertainty and 80% of the error due to truncation error, but we did not iterate exhaustively to achieve this precise balance at the precise error level. Other balances between the error terms could affect the relative run times for the different approaches.

6. CONCLUSIONS

We have presented a method for adaptively refining Monte Carlo tallies in both spatial division and functional representation. Adaptive tallies offer the potential to make Monte Carlo codes still more user friendly, at the expense of increased computational cost. Other adaptivity goals (such as minimizing run time to achieve an error level) and error criteria (such as an infinity-norm) appear to be possible. Memory requirements could be further reduced by determining an optimal expansion order for each axial component; in regions where a distribution is primarily one-dimensional, significant memory savings could result.

ACKNOWLEDGMENTS

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000. The authors wish to acknowledge Martin Crawford and Tim Tautges for helpful suggestions.

REFERENCES

1. M. Ainsworth, J. T. Oden, *A Posteriori Error Estimation in Finite Element Analysis*, Wiley-Interscience, New York (2000).
2. W. L. Chadsey, C. W. Wilson, V. W. Pine, "X-ray Photoemission Calculations," *IEEE Trans. Nucl. Sci.*, **22** (6), pp. 2345–2350 (1975).
3. B. L. Beers, V. W. Pine, "Functional Expansion Technique for Monte Carlo Electron Transport Calculations," *IEEE Trans. Nucl. Sci.*, **23** (6), pp. 1850–1856 (1976).
4. D. P. Griesheimer, W. R. Martin, J. P. Holloway, "Convergence properties of Monte Carlo functional expansion tallies," *J. Comput. Phys.*, **211**, pp. 129–153 (2006).
5. B. C. Franke, R. P. Kensek, "Adaptive Three-Dimensional Monte Carlo Functional-Expansion Tallies," submitted to *Nucl. Sci. Eng.*, (2008).

6. T. E. Booth, “Intelligent Monte Carlo Phase-Space Division and Importance Estimation,” *Trans. Am. Nucl. Soc.*, **60**, pp. 358-360 (1989).
7. Y. Lai, J. Spanier, “Adaptive Importance Sampling Algorithms for Transport Problems,” *Monte Carlo and Quasi-Monte Carlo Methods 1998*, *Lecture Notes in Computational Science and Engineering*, H. Niederreiter and J. Spanier, Eds., Springer-Verlag, New York, 273 (1999).
8. T. E. Booth, “Adaptive Importance Sampling with a Rapidly Varying Importance Function,” *Nucl. Sci. Eng.*, **136**, pp. 399-408 (2000).
9. W. Press, S. Teukolsky, W. Vetterling, B. Flannery, *Numerical Recipes in C*, Cambridge University Press, Cambridge, UK (1992).