A COMPARISON OF VECTORIZABLE DISCRETE SAMPLING METHODS IN MONTE CARLO APPLICATIONS

by

Riyanarto Sarno, Virendra C. Bhavsar and Esam M.A. Hussein

TR95-096, July 1995

Faculty of Computer Science
University of New Brunswick
Fredericton, N.B. E3B 5A3
Canada

July 1995

Phone: (506) 453-4566
Fax: (506) 453-3566

A COMPARISON OF VECTORIZABLE DISCRETE SAMPLING METHODS IN MONTE CARLO APPLICATIONS

RIYANARTO SARNO and VIRENDRA C. BHAVSAR

Faculty of Computer Science

ESAM M.A. HUSSEIN

Department of Mechanical Engineering

University of New Brunswick

Fredericton, N.B., Canada E3B 5A3

ABSTRACT

The performance of various vectorizable discrete random-sampling methods, along with the commonly used inverse sampling method, is assessed on a vector machine. Monte Carlo applications involving, one-dimensional, two-dimensional and multi-dimensional probability tables are used in the investigation. Various forms of the weighted sampling method and methods that transform the original probability table are examined. It is found that some form of weighted sampling is efficient, when the original probability distribution is not far from uniform or can be approximated analytically. Table transformation methods, though require additional memory storage, are best suited in applications where multi-dimensional tables are involved.

Keywords: Discrete sampling, Weighted sampling, Monte Carlo simulations, Vector processing.

1. Introduction. Monte Carlo applications often involve sampling based on probability tables. Most of the distributions based on experimental data are of this category. Sampling from a probability table, i.e. generation of an arbitrary random variable from the table, requires a considerable amount of computing time as it involves a table lookup computation.

The commonly used inverse method [4] samples from a cumulative probability table. It requires a DO-loop involving conditional IF and GOTO statements, which present an obstacle to high speed vector processing. Brown et al. [1] attempted to overcome this problem by devising a generation table with mass points transformed into equiprobable mixture of distributions. This is similar to the alias method [3] and requires large memory storage and a relatively complex procedure. In order to overcome these difficulties, we introduced the weighted sampling method [6]. This method was applied to sampling from one-dimensional tables, where it occasionally resulted in large sample variances.

The objective of this paper is to examine the performance of the above sampling methods in two- and multi-dimensional sampling problems on a vector computer. In addition, variants of the weighted sampling method are introduced to reduce the sample variance in some applications.

The paper is organized as follows. First, the weighted sampling method is briefly reviewed and two variants are introduced. In Section 3, the performance of the two variants is assessed for a one-dimensional table to examine their effect in reducing sample variance. The inverse and Brown et al. methods are chosen for comparison with the weighted sampling methods. Application of various sampling methods to a two-dimensional probability table encountered in the Monte Carlo solution of linear equations is presented in

Section 4. Section 5 considers the application of the sampling methods in a particle transport code, which involves multidimensional tables. All computational studies are carried out on an IBM 3090-180 mainframe computer with a vector facility.

2. Weighted Sampling Methods.

2.1. Direct Method. The weighted sampling method was introduced by the authors [6]. This method, is referred to here as direct weighted sampling. It utilizes a discrete uniform distribution to construct samples from a probability table. In direct weighted sampling (DWS), samples are constructed by randomly drawing mass points, x_j, from a probability table based on the associated probabilities p_j, for $j = 1, 2, \cdots, n$, with $\sum_{j=1}^{n} p_j = 1$. For the i-th sample, the mass point x_i, where $x_i \in \{x_j, j = 1, 2, \cdots, n\}$, is selected from the probability table according to uniform distribution in the interval $[1, n]$. Subsequently, the sampled mass point is multiplied by an *adjustment factor*, which is defined as np_i, where n is the table length and p_i is the probability of the selected mass point (p_i), where $p_i \in \{p_j, j = 1, 2, \cdots, n\}$. This method may not preserve the original mass points. It results in, however, unbiased estimates of the mean and variance of the original distribution.

It was previously shown [6] that the direct weighted sampling method requires the least scalar and vector computing times, in comparison to the inverse, alias and Brown et al. methods. This is due to the fact that the uniform random variables used in weighted sampling are faster to generate in scalar as well as in vector processing. However, this method may result in large sample variances in distributions that are far from being uniform [6, 7]. In order to reduce sample variance, two variants of weighted sampling are introduced below.

TABLE 1

A probability table

x	10	20	30	40	50
p	0.40	0.20	0.30	0.08	0.02

2.2. Stretched Table Method. In this method the given probability table is stretched so that the adjustment factors exhibit low variability. The generation table is constructed by dividing bins with associated large probabilities such that the probabilities are close to being uniform, while keeping the memory requirements as low as possible. The sample variance of an estimated quantity may then be reduced. It should be noted that this method is in effect a table transformation method, not unlike that of Brown et al. [1]. This weighted sampling with a stretched table (WSST) employs the same sampling procedure as that of the direct weighted sampling method.

For example, for Table 1, the WSST generation table is shown in Table 2. This is constructed by subdividing the first four bins. Each of these bins is expanded into two bins, and the corresponding probability is divided evenly. The length of this generation table (n) is equal to 9. Array **w** stores the new mass points, which are equal to $x_j p_j n$, for $j = 1, 2, \cdots, n$. Since the stretched distribution is closer to a uniform distribution, the variation of the adjustment factors in weighted sampling is reduced.

2.3. Nonuniform Distribution Method. In this method, the time for constructing samples is minimized by employing a discrete nonuniform distribution, which has a closed form mathematical expression. Hence, the inverse function of the probability distribution function can be used to generate discrete random numbers in the interval [1,n], which are subsequently used to select mass points and associated probabilities from the probabil-

TABLE 2

The generation table of WSST for Table 1

x	10	10	20	20	30	30	40	40	50
p	0.20	0.20	0.10	0.10	0.15	0.15	0.04	0.04	0.02
w	18.0	18.0	18.0	18.0	40.5	40.5	14.4	14.4	9.0

ity table. Therefore, this weighted sampling with nonuniform distribution (WSNU) method does not involve any comparison. In order to reduce the sample variance, the probability mass function of the nonuniform distribution should have a similar shape to that of the $x_j p_j$ distribution. The closer the shape the smaller sample variance will be.

For the example of Table 1, the WSNU method may involve the use of a binomial distribution with the probability mass function given as

$$p_j = \begin{cases} \frac{n!}{j!(n-j)!} a^j (1-a)^{n-j} & \text{if } j \in \{0, 1, \cdots, n\}, \\ 0 & \text{otherwise,} \end{cases} \tag{2.1}$$

while a is the shape parameter. The value of a is chosen such that the binomial distribution closely approximates the original distribution.

Alternatively, the WSNU method may employ a geometric distribution. Its probability mass function is given as

$$p_j = \begin{cases} a(1-a)^j & \text{if } j \in \{0, 1, \cdots\}, \\ 0 & \text{otherwise,} \end{cases} \tag{2.2}$$

where a is the shape parameter. The geometrically distributed random numbers can be generated using the algorithm given on page 266 of reference [4].

The above three weighted sampling methods (DWS, WSST and WSNU) render themselves to vectorization. The programs for these methods are

TABLE 3

A probability table with large variance (mean = 87.431, variance = 555.99)

Mass points	100	90	70	50	20	15	10	5	2	1
Probabilities	.600	.200	.100	.030	.025	.016	.013	.010	.005	.001

TABLE 4

Generation table of WSST for Table 3

x	100(12×)	90(4×)	70(2×)	50	20	15	10	5	2	1
p	.05(12×)	.05(4×)	.05(2×)	.030	.025	.016	.013	.010	.005	.001

($m\times$) denotes that the corresponding mass point is repeated m times.

given in reference [7].

3. One-Dimensional Sampling. The methods discussed in the previous section are used to estimate the mean of an arbitrary distribution, given in Table 3. This distribution is designed so that it challenges the weighted sampling method by exhibiting a relatively large variance.

The inverse and Brown et al. methods are chosen for comparison with the DWS, WSST and WSNU methods. Scalar codes for the various sampling methods are developed for assessing the performance of corresponding vectorized codes. The various codes are referred to in the ensuing discussion as follows: INVER (inverse method), BROWN (Brown et al.), DWS (direct weighted sampling), WSST (weighted sampling with a stretched table), and WSNU (weighted sampling employing a geometric distribution). The details of INVER, BROWN and DWS are presented elsewhere [6]. For WSST, the generation table used is given in Table 4. The shape parameter (a) for the geometric distribution in Eq. (2.2) is chosen to be equal to 0.68, to closely approximate the original distribution.

TABLE 5

Results of 100,000 samples for Table 3

Code	Mean (%fsd)	Processing time, ms		Speedup w.r.t.	Performance index
		Scalar	Vector	Scalar INVER	(Speedup/fsd)
INVER	87.43(0.09)	504.30	90.94	5.545	61
BROWN	87.50(0.08)	491.25	54.53	9.248	116
DWS	87.49(0.65)	416.58	34.80	14.491	223
WSST	87.48(0.18)	416.58	34.80	14.491	81
WSNU	87.43(0.02)	691.00	179.93	2.803	150

Table 5 reports the estimates of the distribution mean and the associated standard deviation normalised with respect to the mean, the so-called percentage fractional standard deviation (%fsd). The estimated values by all sampling methods agree within the one standard deviation. For the selected sample size, the fsd produced by WSST is lower than that of DWS. This fsd is still higher than that of INVER and BROWN. However, WSNU gives the smallest fsd. Thus, for the example, the WSNU method reduces significantly the sample variance of the direct weighted sampling method. The least scalar processing time is obtained with the DWS and WSST codes. The difference in the processing time of DWS and WSST is of the order of microseconds and therefor is not seen in the table.

The processing times of different vector codes are also shown in Table 5. The DWS and WSST codes require the least processing time, which is only about 38 % of scalar INVER's processing time. In the scalar processing case, however, their processing time is about 83 % of that of scalar INVER. This indicates that the vectorizability of DWS and WSST is better than that of INVER. The vector processing time progressively increases for BROWN, INVER, and WSNU codes, respectively. This Table demonstrates that the vector codes of DWS and WSST entail the highest speedups, while WSNU attains the lowest speedup. Thus, vector processing together with the right choice of a sampling method can achieve significant processing speedups.

In order to fairly assess the performance of any of the above sampling methods, one must consider two factors: the vectorization speedup and the resulting fsd For this purpose, we define a performance index as the ratio of speedup to fsd Therefore, the higher the performance index, the better is a method. As Table 5 shows, WSNU has the highest performance index. This is not surprising given the careful choice of the matching geometry distribution, from which direct sampling was performed.

4. Linear Equations. The second application for assessing the performance of the various sampling methods considers two-dimensional tables. Such tables are encountered, for example, in the Monte Carlo solution of linear equations arising from finite-difference approximation of partial differential equations.

Consider a set of linear equations represented by the matrix equation $\mathbf{Ax}=\mathbf{b}$ with a solution given by $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. By introducing $\mathbf{H} = \mathbf{I} - \mathbf{A}$, where $\mathbf{I}$ is an identity matrix, one can write

$$\mathbf{x} = \mathbf{Hx} + \mathbf{b}. \tag{4.1}$$

When the spectral radius of $\mathbf{H}$ is less than one ($\rho(\mathbf{H}) < 1$), the Neumann series expansion

$$\begin{aligned} \mathbf{x} &= (\mathbf{I} - \mathbf{H})^{-1}\mathbf{b} \\ &= (\mathbf{I} + \mathbf{H} + \mathbf{H}^2 + \cdots + \mathbf{H}^m + \cdots)\mathbf{b} \\ &= \sum_{m=0}^{\infty} \mathbf{H}^m \mathbf{b} \end{aligned} \tag{4.2}$$

is absolutely convergent.

There are two approaches to solving this problem using the Monte Carlo method. The first, suggested in reference [8], employs an absorbing Markov chain which has an absorbing state, at which a random walk is terminated. Another approach to solving Eq. (4.2) employs an ergodic Markov chain [9], where the random walks are not terminated by an absorbing state, but their lengths are predetermined. The implementation details of these two methods are given in reference [7]. In each method, a state-transition probability matrix is constructed to facilitate the random walk process.

We examine here the Monte Carlo solution of the linear equations associated with the finite-difference approximation of a Laplace's equation. It should be noted that many efficient techniques are available for solving this problem. We are considering the problem only as an example to demonstrate the speedup that can be achieved by employing vetorizable sampling methods.

Using the cartesian coordinate system, the problem for a continuous function u in x and y is defined as:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0. \tag{4.3}$$

The boundary conditions for a region $0 \leq x \leq 10$, and $0 \leq y \leq 10$ are chosen as $u(x,0) = u(x,10) = u(10,y) = -10$ and $u(0,y) = 10$.

The finite difference method using the five point formula [2] is used to approximate the original problem. The x as well as y axis is discretized into 33 intervals. This discretization results in 1024 internal points. Each point represents an unknown in the linear equation. Thus, the linear system contains 1024 linear equations.

The transition probability matrices are chosen based on the resulting set of linear equations. For the ergodic Monte Carlo method, we chose a uniform distribution for the transition probability matrix; that is, each point in a row has the same probability. In the absorbing Monte Carlo method, the probability for each point is 0.25. Since the sum of the probabilities for a row has to be equal to one, rows having two and three points have absorbing probabilities 0.50 and 0.25, respectively.

The average values of the solution estimates for all unknowns and the corresponding errors are presented in Table 6. The error is calculated here as the mean-squared difference between Gauss-Siedel solutions (after 80 iterations) and the obtained estimates, for all unknowns. In this table, in identifying the codes, the first three letters represent the ergodic (ERG) or absorbing (ABS) Markov chain, while the last letters designate the sampling methods: inverse (IN), Brown et al. (BR) and direct weighted sampling (DWS). For estimating each unknown, 1,000 samples are used. For each sample size, the random walk length in the ergodic Markov chain is chosen to be equal to 125, which also represents the maximum random walk length in the absorbing Markov chain. As shown in this table, the Monte Carlo method employing an absorbing Markov chain results in smaller errors. The scalar and vector codes of various methods are given in reference [7]. The vectorized codes employing ergodic Markov chains require simpler codes than those using absorbing markov chains since their random walk lengths for all samples are predetermined. Vectoriziation of the absorbing Monte Carlo codes requires a stack processing scheme to efficiently vectorize the processing of random walks of varying length[11].

As shown in Table 6, the ergodic algorithms result in larger errors than the absorbing ones. This is caused by the fixed lengths of the random walks for estimating the solutions of all unknowns. When we choose a random walk of length 50, good solutions of the boundary points are obtained, while poor solutions results for the central points [7]. However, if the random walk of length is 125, the solutions of the central points improve, but the solutions

TABLE 6

Average values of all solution estimates for Laplace's equation, (Gauss-Siedel (GS) average = −2.8664 after 80 iterations)

Code	Solution	Error	Processing time, s		Speedup	Performance index
		w.r.t. GS	Scalar	Vector	Scalar/Vector	(Speedup/Error)
ERGIN	−2.3394	1.3967	618.42	329.25	1.88	1.3
ERGBR	−2.3394	1.3967	621.02	141.43	4.39	3.1
ERGDWS	−2.3394	1.3967	485.23	86.03	5.64	4.0
ABSIN	−2.7563	0.5505	644.35	63.40	3.94	7.2
ABSBR	−2.9611	0.5855	778.78	1178.48	4.36	7.4
ABSDWS	−2.7583	0.5587	601.56	147.21	4.09	7.3

of the boundary points become unacceptable. In conclusion, the absorbing codes are more suitable for estimating all unknowns.

The scalar and vector processing times of Monte Carlo solutions for all unknowns are also depicted in Table 6. In scalar processing, the ergodic and absorbing codes incorporating the direct weighted sampling method (ERGDWS and ABSDWS) require about 20 % less computing time than those of the codes implementing the inverse method (ERGIN and ABSIN). The ergodic and absorbing codes implementing Brown et al. method (ERGBR and ABSBR) require larger processing time than those incorporating the inverse method. The performance of the inverse method is comparatively good since the maximum number of required comparisons (conditional IF's) is only four; given the fact that there are at most four non-zero probabilities in a row of the transition probability matrix. In the vector processing, Table 6 shows that ERGIN requires four times more computing time than that of ERGDWS. This means that weighted sampling can enhance the vectorization of the ergodic algorithm; however, it is less successful in the absorbing algorithm. The maximum speedup is achieved by using the vector ERGDWS code. The ERGDWS can achieve the best speedup, because the uniform sampling contains no loops and the random walk length is predetermined. Thus, Monte Carlo codes employing the weighted sampling methods result in faster execution in scalar as well as vector processing modes.

In Table 6, the performance is defined as the ratio of the vectorization speedup to the error (with respect to the Gauss-Siedel method). As the table shows, among the ergodic codes, direct weighted sampling has the highest performance index. However, the Brown et al. method becomes competitive with weighted sampling for the absorbing codes.

5. Particle Transport Problems. A more complex application of random sampling is encountered in the solution of the particle transport (Boltzmann) equation. Here, multidimensional (in space, energy and angle) probability tables are often involved. The MORSE [10] Monte Carlo code, is one of the common codes used for performing neutron and gamma transport problems. MORSE requires extensive table lookup, particularly to sample the outgoing neutron energy and direction at every collision. This code uses the inverse sampling method, but we modified it to include the Brown et al. (BROWN), direct weighted sampling (DWS) method and a hybrid of DWS with the inverse method, referred to as INWS. In INWS, the inverse method is used for the first few mass points, and the weighed sampling method is utilized for the rest, where the probability table tended to be more uniform for the chosen problems discussed below. It should be noted that due to the involvement of many different sampling tables with various characteristics, the use of the WSST and WSNU methods is not considered to be practical, as they require some knowledge of the nature of the distributions involved and they are many of such distributions in transport problems.

Three particle transport problems with different physical characteristics are examined. In Problem I, the neutron fluence is calculated at a radius of 200 meters from a point isotropic fission source in an infinite air medium. A cross-section (probability table) library consisting of thirteen neutron energy groups with five Legendre coefficients (utilized in determining the angular distribution) is employed. The neutron fluence is estimated using a boundary crossing estimator.

In Problem II, a fast neutron beam is directed towards a cylindrical body of liquid (water), 200 mm in diameter and 250 kg/m^3 in density. A neutron cross section consisting of 33 energy groups with nine Legendre coefficients is used. Two point detectors are located perpendicular to both the neutron beam and the axis of the cylindrical body. The next event estimator is used to estimate the detector response. This estimator evaluates the probability of the next collision being at the detector site, without altering th original particle track. Problem III is identical to the second one, except that the number of collisions is increased by raising artificially the liquid density to 1000 kg/m^3. Further details of these problems are given in reference [7].

The sequential search in the probability table for air involves a small number of energy groups, since the outgoing energy does not differ significantly from the incoming energy. The search process for water may however involve the entire energy range due to hydrogen scattering (the hydrogen nucleus has a mass that is almost equal that of the neutron).

Table 7 shows the estimated fluence and other relevant information for different sampling methods, using 60,000 source particles. It is seen that the estimates, for a given problem, overlap each other within the confidence intervals.

Only the time required for executing the sampling routines is measured

TABLE 7

Results for three neutron transport problems

Problem	Method	Flux ± 99 % Confidence (neutrons/cm^2)	Processing time, s		Speedup	Performance index (Speedup/Confidence)
			Scalar	Vector		
I	INVER	$(4.077\pm 0.155)10^{-10}$	3.79	1.10	3.5	23
	BROWN	$(4.016\pm 0.167)10^{-10}$	4.00	0.45	8.4	50
	DWS	$(3.707\pm 0.537)10^{-10}$	3.76	0.41	9.2	17
	INWS	$(4.151\pm 0.175)10^{-10}$	3.76	1.11	3.4	19
II	INVER	$(9.439\pm0.258)10^{-5}$	5.16	1.57	3.3	13
	BROWN	$(9.230\pm0.259)10^{-5}$	5.14	0.58	8.9	34
	DWS	$(9.970\pm3.678)10^{-5}$	4.84	0.50	10.3	3
	INWS	$(9.143\pm0.259)10^{-5}$	5.14	1.17	4.4	17
III	INVER	$(2.901\pm0.120)10^{-4}$	5.16	1.57	3.3	28
	BROWN	$(2.901\pm0.119)10^{-4}$	5.14	0.58	8.9	75
	DWS	$(2.428\pm0.463)10^{-4}$	4.84	0.50	10.3	22
	INWS	$(2.906\pm0.118)10^{-4}$	5.14	1.17	4.4	37

to avoid the additional overhead of other routines in the MORSE code. The vectorization speedup is calculated relative to the total computing time of the scalar INVER routine.

The table shows that INVER results in the lowest speedups, since the execution of the IF-statements is slow and the vectorization of its innermost loop is not effective. DWS achieves the highest speedups, as it required only simple calculations and is fully vectorizable. The speedups achieved by INWS are lower than those of DWS. However, INWS compensates this by its lower sample variance. Brown et al. method (BROWN) is fully vectorizable and entailed high speedups. However, a significant amount of memory and a complex procedure are required.

The performance index, reported in Table 7, is found by dividing the vectorization speedup by the relative error (defined as the confidence interval normalized with the estimated flux). It seen that Brown et al. method has the highest performance index for all three problems. This comes however at the expense of increased computer memory storage needed to accommodate the transformed tables. In Brown et al. method, for a probability table of n mass points, $3n$ extra storage locations are required [1].

The results of this work indicate that in problems involving a small number of energy groups, the inverse method should be employed. A hybrid of the weighted sampling and inverse methods is most effective in problems with high probabilities for the first few points. The weighted sampling method is most useful when a large number of collisions occur at low energy.

6. Conclusions. We have examined the performance of various discrete sampling methods for one-, two- and multi-dimensional probability tables. In the one-dimensional case, direct and stretched-table weighted sampling methods achieve the highest vectorization speedups at the cost of a relatively higher statistical variance. However, weighted-sampling with a matching analytical distribution achieves the highest performance, taking into account both speedup and variance. The weighted sampling methods are also found to speedup the computations of the two-dimensional probability tables arising in Monte Carlo solutions of Laplace's equation, without significantly affecting the estimates. In Monte Carlo applications involving multi-dimensional tables, direct weighted sampling provides the highest vectorization speedup. However, in terms of overall performance, Brown et al. method is superior, as it results in smaller variances. If memory requirements are of concern, then a combination of inverse and weighted sampling is an alternative. In conclusion, to take advantage of vector processing, one should utilize either table transformation methods, such as that of Brown et al., or a weighted sampling method. Though weighted sampling tends to increase the statistical variance of the estimates, however, if devised carefully, weighted sampling can provide attractive performance.

REFERENCES

[1] F.B. BROWN, W.R. MARTIN and D.A. CALAHAN, A Discrete Sampling Method for Vectorized Monte Carlo Calculations. *Trans. Am. Nuclear Soc.,* Vol. 38, pp. 354–355, 1981.

[2] CHENEY, W. and KINCAID, D., *Numerical Mathematics and Computing.* 2nd edition. California: Brooks / Cole Publishing Co., 1985.

[3] R.A. KRONMAL and A.V. PETERSON, JR., On the Alias Method for Generating Random Variables from a Discrete Distribution. *The American Statistician.* Vol. 33, No. 4, pp. 214–218, 1979.

[4] A.M. LAW and W.D. KELTON, *Simulation Modeling and Analysis.* McGraw-Hill Book Company, New York, 1982.

[5] G. MARSAGLIA, *Generating Discrete Random Variables in a Computer.* Comm. ACM, Vol. 6, No. 1, pp. 37–38, 1963.

[6] R. SARNO, V.C. BHAVSAR and E.M.A. HUSSEIN, Generation of Discrete Random Variables on Vector Computers For Monte Carlo Simulations. *International Journal Of High Speed Computing.* Vol. 2, No. 4, pp. 335–350, 1990.

[7] R. SARNO, *Discrete Sampling Methods For Vector Monte Carlo Codes.* PhD Thesis. The Faculty of Computer Science, University of New Brunswick, Fredericton, N.B., Canada, 1992.

[8] FORSYTHE G.E. and LEIBLER, R.A., Matrix Inversion by a Monte Carlo Method. *Math. Tables Other Aids Comput.* Vol. 4, pp. 127–129, 1950.

[9] WASOW, W.R., A Note on the Inversion of Matrices by Random Walks. *Math. Tables Other Aids Comput.* Vol. 6, pp. 78–81, 1952.

[10] MORSE-CG, *General Purpose Monte Carlo Multigroup Transport Code with Combinatorial Geometry,* CCC-203, RSIC, Oak Ridge National

Laboratory, 1971.

[11] F.B. BROWN and W.R. MARTIN, Status of Vectorized Monte Carlo for Particle Transport Analysis, *Int. J. Supercomputer Applications,* Vol. 1, No. 2, pp. 11–32, 1987.