GENERATION OF DISCRETE RANDOM VARIABLES ON VECTOR COMPUTERS FOR MONTE CARLO SIMULATIONS

by

R. Sarno, V.C. Bhavsar, E.M.A. Hussein

TR90-051, August 1990
Revised October 1990

Faculty of Computer Science
University of New Brunswick
P.O. Box 4400
Fredericton, N.B. E3B 5A3

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

GENERATION OF DISCRETE RANDOM VARIABLES ON VECTOR COMPUTERS FOR MONTE CARLO SIMULATIONS

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 paper reviews existing methods for generating discrete random variables and their suitability for vector processing. A new method for generating discrete random variables for use in vectorized Monte Carlo simulations is presented. The method uses the concept of importance sampling and generates random variables by employing uniform distribution to speedup the computation. The sampled random variables are subsequently adjusted so that unbiased estimates are obtained. The method preserves both mean and variance of the original distribution. It is demonstrated that the method requires simpler coding and shorter execution time for both scalar and vector processing, when compared with other existing methods. The vectorization speedup of the method is demonstrated on an IBM 3090-180 machine with a vector facility.

Keywords: discrete random variables, importance sampling, Monte Carlo simulation, parallel processing, supercomputing, vector processing.

1. Introduction. Monte Carlo studies and simulation of stochastic systems often involve random variable generation from one or more distributions. Such studies usually involve statistical estimation of the mean, and sometimes the variance, of the underlying distribution. Probability tables are usually used to digitally represent the distributions as sets of mass points

with associated probabilities. The generation of arbitrary discrete random variables from probability tables can constitute a significant portion of the total simulation time.

Several methods are presently used to generate discrete random variables. The inverse method [8] is the most commonly used method. For arbitrary distributions in which the associated probabilities have been digitally stored, this method involves searching for intervals in the cummulative distribution in which the random numbers lie. The number of comparisons in a search is dependent on the number of mass points. The average number of comparisons in a sequential search, assuming that intervals are distributed uniformly, is $(n+1)/2$, where n is the number of mass points [10].

Since the searching process is time consuming attempts have been made to develop alternative methods which do not involve searching. Walker [12] has proposed the alias method, which requires only one comparison to obtain a mass point, regardless of the number of the mass points. Brown et al. [1] have proposed another method, which is essentially the same as the alias method; the only difference is in the procedure for setting up the generation table. A generation table, which is obtained from a probability table, is a set of arrays used in generating random variables. Some examples of generation tables are given in the next section.

With the advent of supercomputers and high performance workstations with parallel processing capabilities, it is necessary to investigate the vectorization and parallelization aspects of the process for generating discrete random variables in order to reduce the computing time of large scale Monte Carlo and stochastic simulation studies. The inverse method is not suitable for vectorization as it involves step-by-step searching. The alias method is not vectorizable completely since it requires IF-statement involving indirect addressing. Brown's method is more suitable for vectorization since it avoids the use of IF-statement.

In this paper we introduce a vectorizable method, which is called the *weighted sampling* method, for generating arbitrary discrete random variables from probability tables. This method is based on the concept of importance sampling [11] and may not preserve the mass points of the original distribution. It results in, however, unbiased mean and variance estimates of the original distribution. These estimates are the quantities of interest in engineering applications, such as particle transport problems [11]. In order to demonstrate the attractive features of the weighted sampling method, we begin by briefly reviewing the performance of existing sampling methods when they are vectorized.

2. Existing Methods. The objective of generating arbitrary discrete random variables from a probability table is to randomly select mass points,

x_j, according to the associated probabilities, p_j, for $j = 1, 2, \cdots, n$, with $\sum_{j=1}^{n} p_j = 1$. For illustration of the random variable generation process, we use the simple probability table shown in Table 1.

TABLE 1
An example of a probability table.

Mass points	10	20	30	40	50
Probabilities	0.40	0.20	0.30	0.08	0.02

2.1. The Inverse Method. The inverse method uses the generation table depicted in Table 2. This generation table is constructed by calculating the cummulative probability and storing it into array C. Array X contains the mass points. A scalar code of the inverse method, shown in Fig. 1, is to generate random variable RV, K times for a probability table of length N. The RNUNF function used in this code is an IMSL [6] routine, which generates uniform random numbers in the interval $(0, 1)$.

TABLE 2
The generation table of the inverse method.

X	10	20	30	40	50
C	0.40	0.60	0.90	0.98	1.00

```
      DO 10 I = 1,K
        R1 = RNUNF()
        DO 20 J = 1,N
          IF(C(J) .GE. R1) THEN
            RV(I) = X(J)
            GO TO 10
          ENDIF
20      CONTINUE
10    CONTINUE
```

FIG. 1. *The scalar code of the inverse method.*

Under earlier version of the IBM VS FORTRAN, the scalar code was not directly vectorizable due to the presence of the GO TO-statement in the inner loop and the RNUNF function in the outer loop. VS FORTRAN Version 2 Release 4 [5] can however vectorize the inner loop, as shown in Fig. 2. Vector codes use an ESSL [4] routine, SURAND. ESSL (Engineering and Scientific Subroutine Library) contains highly tuned vector subprograms

```
      CALL SURAND(SEED1,K,RN1)
      DO 10 I = 1,K
        DO 20 J = 1,N
          IF(C(J) .GE. RN1(I)) THEN
            RV(I) = X(J)
            GO TO 10
          ENDIF
20      CONTINUE
10    CONTINUE
```

FIG. 2. *Vector code of the inverse method (the inner loop is vectorizable).*

```
      CALL SURAND(SEED1,K,RN1)
      DO 10 I = 1,K
        IF(RN1(I) .LE. C(1)) RV(I)=X(1)
        DO 20 J = 1,N-1
          IF(RN1(I) .GT. C(J)) RV(I)=X(J+1)
20      CONTINUE
10    CONTINUE
```

FIG. 3. *Vector code of the inverse method (the two loops are vectorizable).*

for IBM 3090-180VF. SURAND(SEED1,K,RN1) uses SEED1 as a seed to generate K random numbers which are stored in array RN1.

By restructuring the code, as shown in Fig. 3, the outer and inner loops become vectorizable. The compiler usually chooses the outer loop to vectorize in order to result in less processing time since K is usually much larger than N. In this scheme, however, the generation of every random variable requires N iterations of the inner loop regardless of the location of an interval in the cummulative distribution in which a random number lies.

2.2. The Alias Method. The generation table of the alias method is depicted in Table 3. The transformation from the probability table to the generation table is carried out systematically because of the fact that any discrete distribution with a finite number (n) of mass points can be expressed as an equiprobable mixture of n distributions, each of which has two mass points [7]. Array X stores the mass points of the first elements of the five two-point distributions, while array F stores the associated probabilities. Array A stores the mass points of the second elements of the five two-point distributions. The scalar and vector codes of the alias method are shown respectively in Fig. 4 and Fig. 5. IMSL also provides a scalar routine, RNGDA, which implements the alias method. The IF-statement in the vector code is not vectorizable since array A is used in conditionally executed

code and has non-inductive subscript expressions (indirect addressing).

TABLE 3
The generation table of the alias method.

X	10	20	30	40	50
A	10	20	10	30	10
F	0.0	0.0	0.9	0.4	0.1

```
      DO 10 I = 1,K
         R1    = RNUNF()
         R2    = RNUNF()
         J     = INT(R1 * N) + 1
         RV(I)= X(J)
         IF(R2 .GT. F(J)) RV(I) = A(J)
10    CONTINUE
```

FIG. 4. *The scalar code of the alias method.*

```
      CALL SURAND(SEED1,K,RN1)
      CALL SURAND(SEED2,K,RN2)
      DO 10 I = 1,K
         J(I)  = INT(RN1(I) * N) + 1
         RV(I) = X(J(I))
         IF(RN2(I) .GT. F(J(I))) RV(I) = A(J(I))
10    CONTINUE
```

FIG. 5. *The vector code of the alias method.*

2.3. Brown's Method. Table 4 shows the generation table for Brown's method. The construction of the generation table employs the same approach as the alias method; however, the procedure is different. Array Y stores the mass points of the first elements of the five two-point distributions, while array F stores the associated probabilities. Array B stores the mass points of the second elements of the five two-point distributions. In order to avoid the use of IF-statement the generation of random variables uses array D, which combines arrays Y and B. The first elements of arrays Y and B are stored into the first and second elements of array D. The second elements of arrays Y and B are stored into the third and fourth elements of array D, and so on.

The scalar and vector codes of Brown's method are shown respectively in Fig. 6 and Fig. 7. This method uses only one random number generator

since the second generator can be determined from the first one. Since this method relies on the randomness of the lower order digits of the uniform random number generator, it is not recommended for large n, see reference [3, page 108].

TABLE 4
The generation table of Brown's method.

		Y	10	30	10	10	20			
		B	50	40	30	10	20			
		F	0.9	0.6	0.1	0.0	0.0			
D	10	50	30	40	10	30	10	10	20	20

```
      DO 10 I = 1,K
         R1    = RNUNF()
         R     = R1 * N + 1.0
         IR    = INT(R)
         J     = IR + R - F(IR)
         RV(I)= D(J)
10    CONTINUE
```

FIG. 6. *The scalar code of Brown's method.*

```
      CALL SURAND(SEED1,K,RN1)
      DO 10 I = 1,K
         R     = RN1(I) * N + 1.0
         IR    = INT(R)
         J     = IR + R - F(IR)
         RV(I)= D(J)
10    CONTINUE
```

FIG. 7. *The vector code of Brown's method.*

3. Proposed Method. Importance sampling is often used in Monte Carlo simulations to speedup the computation. The objective is to minimize the statistical error associated with the estimated quantity. This is achieved by altering the original distribution such that an unbiased estimate of the quantity of interest is obtained with minimum statistical error. In a similar way, the proposed method alters the given distribution by employing uniform distribution, which requires smaller execution time both with scalar and vector processing. In this method, the sampled quantities are adjusted using the associated probabilities in order to obtain unbiased estimates. In

this method, the mass points are fetched directly from a discrete random variable uniformly distributed in the interval $[1, n]$, where n is the number of the mass points. The number of mass points (n) multiplied by the probability of the selected mass point (p_i), where $p_i \in \{p_j, j = 1, 2, \cdots, n\}$, is used as an adjustment factor of the sampled mass point. The random variable generated is equal to $x_i p_i n$, where $x_i \in \{x_j, j = 1, 2, \cdots, n\}$. The multiplication operation can be eliminated by storing the values of $x_j p_j n$, for $j = 1, 2, \cdots, n$, into an array W, as shown in Table 5. It should be mentioned that zero probabilities are excluded in this method and the table length is accordingly adjusted. Note that the method produces the random variables 5, 16, 20 and 45, which are not mass points of the original distribution; whereas the original mass points 10, 20, 30, 40 and 50 will never be generated. However, the method preserves the mean and variance of the original distribution (as shown in Section 4) which are the only important quantities in many Monte Carlo simulations.

TABLE 5
The generation table of the weighted sampling method.

X	10	20	30	40	50
P	0.40	0.20	0.30	0.08	0.02
W	20	20	45	16	5

The scalar and vector codes of the weighted sampling method are shown, respectively, in Fig. 8 and Fig. 9. The vectorization of the weighted sampling method is straightforward and contains no logical IF-statement. The loop is vectorizable since there are no data dependencies in the fetch operations. The scalar and vector codes of the weighted sampling method are simpler than those of the other methods. Also, the weighted sampling method can make direct use of the original probability tables without any alternations. Therefore, this method can be incorporated easily into large simulation codes. The initialization time is also much less than those of other methods.

```
      DO 10 I = 1,K
         R1  = RNUNF()
         J   = INT(R1 * N) + 1
         RV(I)=W(J)
10    CONTINUE
```

FIG. 8. *The scalar code of the weighted sampling method.*

```
      CALL SURAND(SEED1,K,RN1)
      DO 10 I = 1,K
         J    = INT(RN1 * N) + 1
         RV(I) = W(J)
10    CONTINUE
```

FIG. 9. *The vector code of the weighted sampling method.*

4. Statistical Analysis. This section proves that the weighted sampling method provides unbiased estimators of the distribution mean and variance. Unbiased estimators for higher order moments can also be developed in a similar fashion. The subscript w is used for the weighted sampling estimators in order to distinguish them from the corresponding estimators which employ the original distribution. The sample mean and variance estimated directly from the original distribution are $\overline{X}$ and S^2, respectively; while, the weighted sampling method provides $\overline{X}_w$ and S_w^2. The notations for mass points and the associated probabilities respectively are x_j, and p_j, for $j = 1, 2, \cdots, n$, where $\sum_{j=1}^{n} p_j = 1$.

THEOREM 1. *The sample mean* $\overline{X}_w = \frac{1}{k}\sum_{i=1}^{k} p_i n x_i$ *is an unbiased estimator of the distribution mean* μ *whenever the latter exists, where* $p_i \in \{p_j, j = 1, 2, \cdots, n\}$, $x_i \in \{x_j, j = 1, 2, \cdots, n\}$ *and* k *is the sample size.*

Proof. The expected value of $\overline{X}_w$ can be expressed as

$$E[\overline{X}_w] = E\left[\frac{\sum_{i=1}^{k} p_i n x_i}{k}\right] = \frac{\sum_{i=1}^{k} E[p_i n x_i]}{k}. \tag{4.1}$$

Since the random variables are generated uniformly and $x_i, i = 1, 2, \cdots, k$, are independent and identically distributed then for every i, the expected value of $p_i n x_i$ can be expressed as

$$E[p_i n x_i] = \sum_{j=1}^{n} \frac{1}{n}(p_j n x_j) = \mu. \tag{4.2}$$

The expected value of $\overline{X}_w$ then is as follows.

$$E[\overline{X}_w] = \frac{k\mu}{k} = \mu. \tag{4.3}$$

Thus, the expected value of $\overline{X}_w$ is equal to the mean of the distribution, μ, and $\overline{X}_w$ is therefore an unbiased estimator of μ. ∎

THEOREM 2. *The* $S_w^2 = \frac{1}{k}\sum_{i=1}^{k} p_i n(x_i - \overline{X}_w)^2$ *is an unbiased estimator of the distribution variance* σ^2 *whenever the latter exists, where* $p_i \in \{p_j, j = 1, 2, \cdots, n\}$, $x_i \in \{x_j, j = 1, 2, \cdots, n\}$ *and* k *is the sample size.*

Proof. According to Theorem 1, $\overline{X}_w$ approaches μ, for a large k. The expected value of S_w^2 can therefore be approximated as

$$E[S_w^2] = \frac{n}{k}\sum_{i=1}^{k} E[p_i(x_i - \mu)^2], \text{ for a large } k. \tag{4.4}$$

Since the random variables are generated uniformly, Eq. (4.4) can then be expressed as

$$\begin{aligned} E[S_w^2] &= \frac{n}{k}\sum_{i=1}^{k}\sum_{j=1}^{n}\frac{1}{n}(p_j(x_j - \mu)^2) \\ &= \frac{1}{k}\sum_{i=1}^{k}\sigma^2 \\ &= \sigma^2. \end{aligned} \tag{4.5}$$

Thus, the expected value of S_w^2 is equal to the variance of the distribution; hence, S_w^2 is an unbiased estimator of σ^2. ∎

5. Results and Discussion. The inverse method, the alias method, Brown's method and the weighted sampling method have been implemented on the IBM 3090-180 Vector Facility computer of the University of New Brunswick to estimate the mean and variance of an arbitrary distribution. VS FORTRAN Version 2 Release 4 was used to code the methods. The programs were compiled using optimization OPTION(3), the highest optimization option for scalar and vector processing. Notations for the scalar codes are as follows.

1. INVER: scalar code of the inverse method,
2. ALIAS: scalar code of the alias method,
3. DISCS: scalar code of Brown's method,
4. IMSL: employs the IMSL routine RNGDA, in which the alias method is implemented, and
5. WGHTS: scalar code of the weighted sampling method.

The following notations are for the vector codes.

1. INVR1: vector code of the inverse method, shown in Fig. 2,

2. INVR2: vector code of the inverse method, shown in Fig. 3,
3. ALIAS: vector code of the alias method,
4. DISCS: vector code of Brown's method, and
5. WGHTS: vector code of the weighted sampling method.

5.1. Estimation of Distribution Mean. Table 6 shows the probability table used in the computation of the sample mean and the sample variance. This probability table was chosen as it results in a relatively large variance; which enables one to verify the validity of the estimators.

TABLE 6
The probability table used in computation.

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

Estimates of the distribution mean and the associated errors evaluated using the different codes are summarized in Table 7. The distribution mean, according to the probability table shown in Table 6, is 87.431. The error of the sample mean is defined as the fractional standard deviation of the sample mean multiplied by 100%. It is found, as shown in Table 7, that all of the sample means computed using various methods are unbiased, i.e. they estimate a value of the mean almost equal to that of the distribution mean. For this particular example, the weighted sampling method produces the largest error, which means that the sample variance is the largest. The errors of the other methods are relatively equal. It can be noted that if a distribution such as that depicted in Table 1 is used, the weighted sampling method produces less sample variance than the other methods. In general, the variance of the weighted sampling method may be smaller or larger than the other methods, depending on the distribution.

TABLE 7
Sample means and their errors for various methods.

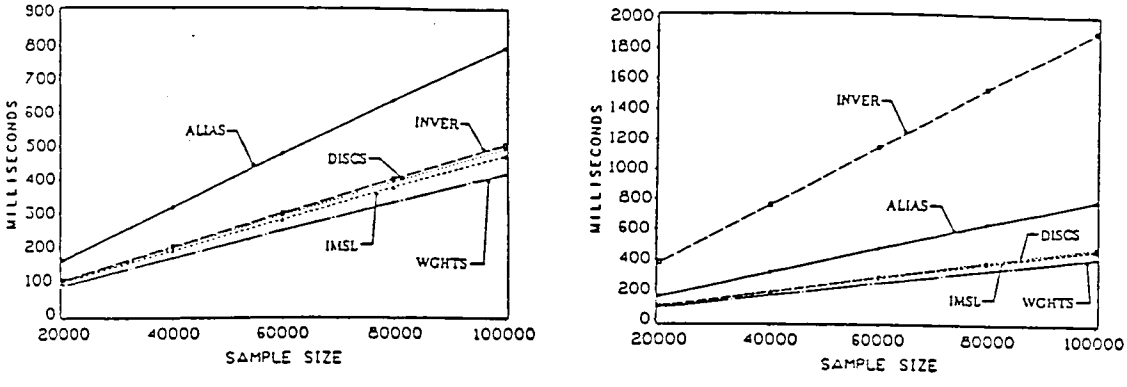
Sample size	INVER	ALIAS	DISCS	IMSL	WGHTS
	$\bar{X}$(Error)	$\bar{X}$(Error)	$\bar{X}$(Error)	$\bar{X}$(Error)	$\bar{X}_w$(Error)
20000	87.44(0.19)	87.40(0.19)	87.40(0.19)	87.39(0.19)	87.76(1.44)
40000	87.42(0.14)	87.58(0.13)	87.58(0.13)	87.57(0.13)	87.36(1.02)
60000	87.41(0.11)	87.56(0.11)	87.56(0.11)	87.56(0.11)	87.54(0.84)
80000	87.39(0.10)	87.57(0.10)	87.57(0.09)	87.55(0.09)	87.47(0.73)
100000	87.43(0.09)	87.50(0.09)	87.50(0.08)	87.50(0.08)	87.49(0.65)

5.2. Estimation of Distribution Variance. Table 8 summarizes the distribution variances estimated using the different methods. The distribution variance, according to the probability table shown in Table 6, is 555.99. As Table 8 shows, all variances evaluated by the different methods are almost equal to the distribution variance. The estimates of the variance are therefore unbiased.

TABLE 8
Sample variances and their errors for various methods.

Sample size	INVER S^2(Error)	ALIAS S^2(Error)	DISCS S^2(Error)	IMSL S^2(Error)	WGHTS S_w^2(Error)
20000	557.9(1.8)	550.8(1.8)	550.1(1.8)	549.6(1.8)	554.3(0.5)
40000	559.8(1.3)	543.4(1.3)	539.9(1.3)	539.8(1.3)	555.5(0.3)
60000	558.0(1.0)	547.4(1.0)	541.1(1.0)	541.0(1.0)	556.6(0.3)
80000	558.9(0.9)	548.9(0.9)	541.1(0.9)	540.9(0.9)	556.5(0.2)
100000	557.1(0.8)	552.4(0.8)	543.8(0.8)	543.8(0.8)	556.6(0.2)

5.3. Processing Time. Distributions of different number of mass points ($n = 10$ and $n = 200$) are used for evaluating the processing time of scalar and vector codes. The sample size ranges from 20,000 to 100,000.

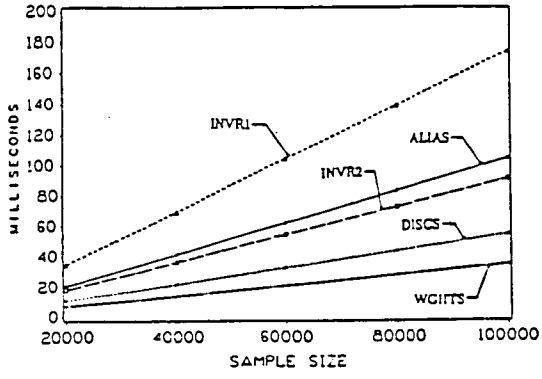


(a) $n = 10$ (b) $n = 200$

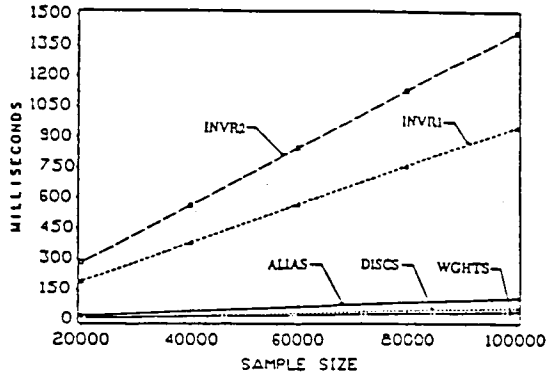
FIG. 10. *Processing time of the scalar codes.*

Fig. 10.a and Fig. 10.b show the processing time of the scalar codes for $n = 10$ and $n = 200$, respectively. For $n = 10$, the least processing time is for the WGHTS code, followed by the IMSL code; then the processing time

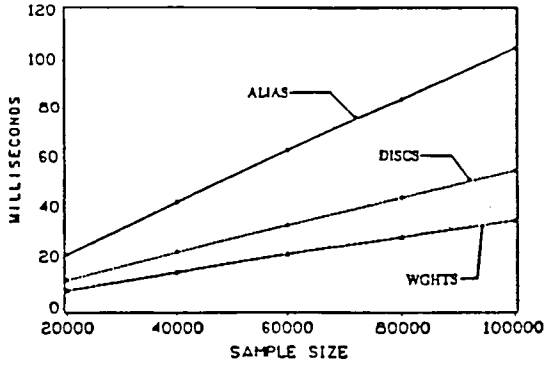
increases respectively for the DISCS, INVER and ALIAS codes. For $n = 200$, the least processing time is still for the WGHTS code, followed by the IMSL, DISCS and ALIAS codes. The INVER code requires the largest processing time. The processing time of the WGHTS, IMSL, DISCS and ALIAS codes for $n = 10$ and $n = 200$ are almost equal; whereas, the processing time of the INVER code depends on the value of n.



(a) $n = 10$



(b) $n = 200$



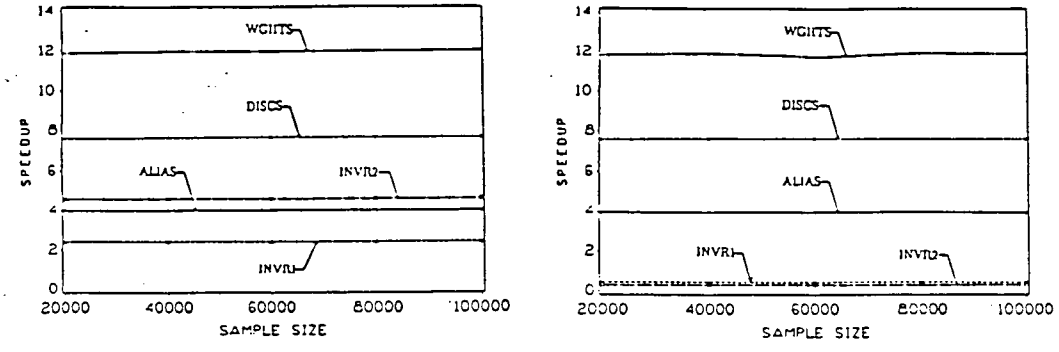
(c) $n = 200$ (without INVR1 and INVR2).

FIG. 11. *Processing time of the vectorized codes.*

Fig. 11 shows the processing times of the vector codes for $n = 10$ and $n = 200$. For $n = 10$ (Fig. 11.a), the least processing time is for the

WGHTS code, followed by the DISCS code; then the processing time increases respectively for the INVR2, ALIAS and INVR1 codes. For $n = 200$, the least processing time is still for the WGHTS code, followed by the DISCS and ALIAS codes. However, the INVR1 code requires less processing time than the INVR2 code. The INVR1 code requires smaller processing time than INVR2 when $n = 200$ since in the INVR1 code the inner loop gets vectorized and the overhead for the INVR2 code is larger than the benefit of the vectorization of its outer loop. INVR2 involves an overhead processing time since the inner loop is always iterated n times regardless of the location of an interval in the cummulative distribution in which a random number lies.

The WGHTS codes, as shown in Fig. 11.a and Fig. 11.c, are about 1.6 times faster than the DISCS codes for both $n = 10$ and $n = 200$.



(a) $n = 10$ (b) $n = 200$

FIG. 12. *Speedups of the vector codes relative to the scalar WGHTS code.*

Fig. 12.a and Fig. 12.b, respectively, show the speedups of the vector codes relative to the scalar WGHTS code, which exhibits the smallest scalar execution time. The speedups of the ALIAS, DISCS and the WGHTS codes are about 4, 7.7 and 12, respectively for both $n = 10$ and $n = 200$. For $n = 10$, the speedup of the INVR1 is 2.4, while INVR2 is 4.6. For $n = 200$, the speedup of the INVR1 is 0.4, while INVR2 is 0.3. The speedups of the INVR1 and INVR2 codes are the same (0.6, when $n = 100$). Thus the inverse method is not suited for vector processing, while the proposed method achieves the best vectorization speedup.

6. Conclusion. The proposed weighted sampling method requires the least processing time in scalar as well as in vector processing, compared to other methods. It is amenable for vectorization without special effort since there is no form of data dependency in the computational process which inhibits vectorization. The weighted sampling method satisfies therefore the criteria for a good random variable generator, as stated by Devroye [3, page 8]; namely speed, initialization time, length of the compiled code, portability, simplicity and readability.

Acknowledgements. The first author would like to thank the Government of Indonesia for the partial financial support for studies at the University of New Brunswick and World University Service of Canada for administering the program. This research is partially supported by the Natural Sciences and Engineering Research Council of Canada grant no. OGP0089.

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] F.B. BROWN, W.R. MARTIN and D.A. CALAHAN, *A Computer Program for A Discrete Sampling Method for Vectorized Monte Carlo Calculations*, Personal Communication, 1981.

[3] L. DEVROYE, *Non-Uniform Random Variate Generation*, Springer-Verlag, New York, 1986.

[4] IBM, *Engineering and Scientific Subroutine Library, Guide and Reference Release 4*, Fifth Edition, IBM Corp., Kingston, New York, March 1990.

[5] IBM, *VS FORTRAN Version 2 Release 4, Language and Library Reference*, Fifth Edition, IBM Corp., Kingston, New York, August 1989.

[6] IMSL, *Fortran Subroutines for Statistical Analysis*, Version 1.0, IMSL Inc., Houston, Texas, August 1989.

[7] 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.

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

[9] R. SARNO, V.C. BHAVSAR and E.M.A. HUSSEIN, *Performance of Discrete Random Variable Generators on IBM 3090-180VF*, Proceedings of APICS Annual Computer Science Conference, pp. 114–125, Fredericton, N.B., November 1989.

[10] R. Sedgewick, *Algorithms*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1983.

[11] J. Spanier and E.M. Gelbard, *Monte Carlo Principles and Neutron Transport Problems*, Addison–Wesley Publishing Company, 1969.

[12] A.J. WALKER, *An Efficient Method for Generating Discrete Random Variables with General Distributions*, ACM Trans. on Mathematical Software, Vol. 3, No. 3, pp. 253–256, 1977.