Proceedings of the 1983
Winter Simulation Conference
S. Roberts, J. Banks, B. Schmeiser (eds.)

AN IMPORTANCE SAMPLING SCHEME FOR SIMULATING THE DEGRADATION AND FAILURE OF COMPLEX SYSTEMS DURING FINITE MISSIONS*

Leonidas C. Kioussis and Douglas R. Miller
Department of Operations Research
The George Washington University
Washington, D.C. 20052

The variance reduction technique of path-splitting is applied to reliability analysis of complex systems. Two approaches are given. In the first, paths are split into a fixed prespecified number of branches. In the second, a sequential procedure is used to estimate the number of branches which will give the maximum variance reduction. The method is illustrated on a simulation of a simplified model of a fault-tolerant computer system. The estimated variance reduction ranged from 36% to 61%.

1. INTRODUCTION

Reliability estimation for modern complex systems can be quite difficult and frequently analytically intractible. These systems often have redundancy and an associated capability for detecting component failures and reconfiguring themselves in order to continue functioning. Simulation is a natural approach to the problem of reliability estimation for these systems because of their complexity. However, the redundancy and capability for reconfiguration found in these systems are for achieving high reliability; thus we are typically faced with the problem of estimating system failure probabilities of .0001 or less. If a simulation approach is taken, it is clear that some variance reduction techniques must be used in order to make the approach feasible.

This paper looks at the importance sampling scheme of path-splitting for simulation of the degradation and failure of complex systems during missions of fixed finite duration. The idea of path-splitting dates from the early days of Monte Carlo analysis. The purpose of this paper is to make a preliminary assessment of the technique's value in reliability estimation. We shall use a model of a simplified fault-tolerant computing system as a test case. This system is described in Section 2. The basic idea of path-splitting is described in Section 3. The related statistical analysis is given in Section 4. A crucial design parameter in this sampling scheme is the number of branches into which a path splits.

*Research supported by National Aeronautics and Space Administration grant NAG-1-179.

In Section 5 a sequential procedure is described for dealing with this problem. Some of the details of how the simulation was implemented are presented in Section 6 (including some additional variance reduction). In Section 7 we show the performance of different sampling plans used in the simulation of four different cases of our fault-tolerant computer systems. We conclude with a summary in Section 8.

2. A FAULT-TOLERANT COMPUTER EXAMPLE

We shall illustrate the general technique of the path-splitting variance reduction technique applied to reliability estimation of complex multi-component systems over finite missions by considering a specific example of a fault-tolerant computing system. Fault-tolerance is used to achieve high reliability; see Hopkins, et al (1978) and Wensley, et al (1978) for discussion and examples. The following example is a simplified model of more realistic systems but it retains salient features which will test a variance reduction scheme.

The computer system consists of four processors. It is functioning in a real-time critical environment, repeating certain computational tasks every 10 seconds. The first 7 seconds of this 10 second cycle time is devoted to executing an application program. Three processors will execute this program, then compare their answers and, at the end of the 7 second application cycle, vote to determine the correct answer. If one processor gives the incorrect answer it is reconfigured out of the system. If two processors give incorrect answers the system fails.

While not executing the application program, the processors run diagnostic programs looking for faults. Thus, the remaining 3 seconds after the 7-second application run are devoted to diagnostics in each 10 second period. Also, if the fourth processor is still in the standby mode as a spare it will devote all 10 seconds to diagnostics. When all four processors have no detected faults the role of the spare is rotated, see Figure 1.

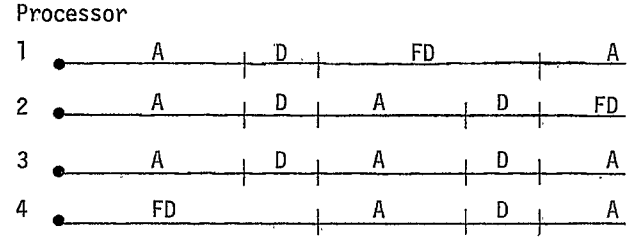

Figure 1: The pattern according to which the system operates. Where A = application program, D = diagnostic program, and FD = full-diagnostic program.

If the diagnostic program reveals a fault, the processor containing the fault is reconfigured out of the system.

We shall observe the behavior of the system during a mission of finite prespecified duration and observe mission outcome; success or failure. In our case the mission time is one hour. We assume that the processors fail independently and for each processor the time until a fault occurs is Exponentially distributed with mean 100 hours. We assume dependence in the error generator process: if a fault is present during execution of the application program, the probability of error generation is ε ; if faults are present in two processors involved with the application program then both will generate errors. If a single fault is detected by diagnostics or voting, the processor with the fault is reconfigured out of the system by the remaining processors, see Figures 2 and 3.

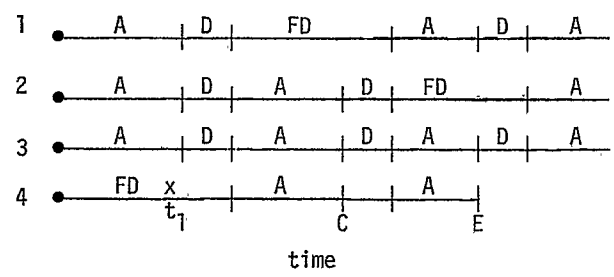

Figure 2: Component #4, failed at time t_1 . C = at the end of this application program, gives correct answer, E = at the end of this application program, gives erroneous answer, and by the voting mechanism is released.

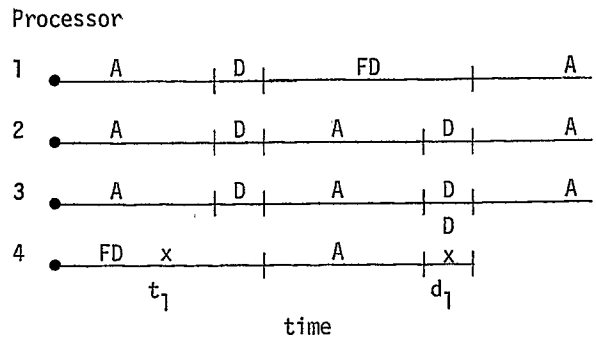

Figure 3: Component #4, failed at time t_1 , the fault is detected by the diagnostic program at time d_1 , and the system continues to operate with the rest of processors.

If two processors each give errors in the application program, the voting mechanism fails and we assume system failure, see Figure 4. We also assume system failure when an error occurs in a system which has been reduced to one or two processors or when a fault is detected in a system which has been reduced to one processor.

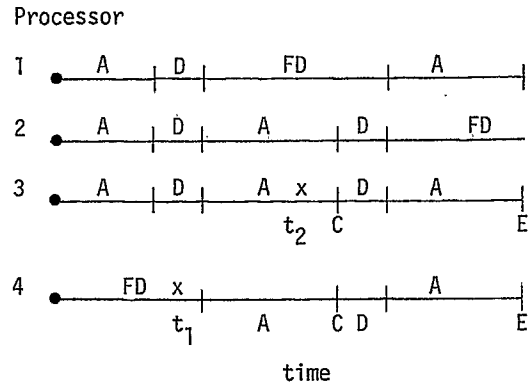

Figure 4: Component #4, failed at time t_1 , and component #3 failed at time t_2 . Correct answers are given at the end of the second application cycle, and at the end of the third an error is generated and system failure occurs.

3. IMPORTANCE SAMPLING FOR RELIABLE SYSTEMS

This paper looks at one approach to variance reduction for reliability estimation of systems which exhibit a certain amount of fault tolerance. These systems have the property that no single component failure can bring the system down, and furthermore that once a component fault has been detected, diagnosed, and isolated by reconfiguring the system, it can no longer cause system failure. Thus the main causes of system failure are (i) the presence of two or more undetected faults in the system and (ii) the depletion of standby redundant components through failures and reconfiguration. This type of system lends itself to an analysis using importance sampling procedures. Such procedures would bias the simulation sample so that missions during which the system is in jeopardy (e.g., the presence of simultaneous multiple undetected faults) are overrepresented.

The first step in a sampling plan with importance sampling is to avoid simulating any mission during which no component failures occur. If component failure is independent of application (as is often true for electronic system) it is possible to generate component failures during a mission conditional on at least one failure occurring. Furthermore, if redundant items are kept in a "warm standby" state, it is possible to generate component failures during a mission conditional on at least two failures occurring or conditional on other useful conditions. This rather obvious approach accounts for drastic savings in the simulation of systems with highly reliable components.

The second step in importance sampling, and the main topic of this paper, is to manipulate the sample paths for the system during missions where something interesting happens, e.g., two or more component failures. The success or failure of the system during these missions will depend on the timing and location of the component failures and the detection, diagnosis, and reconfiguration process, which has a good deal of randomness. To manipulate this sampling process we suggest using the techniques of "Russian roulette" and "splitting." These methods were used in the early days of Monte Carlo simulation to study neutron diffusion and nuclear shielding, see Hammersley and Handscomb (1964), Chapter 8, and references therein. Others have applied variations of the "splitting" technique to steady state analysis of stochastic service systems, sometimes successfully (Bayes 1970) and sometimes unsuccessfully (Hopmans and Kleijnen 1979). Carter and Ignall's (1975) idea of virtual measures is also related. The application of "splitting" to reliability estimation over finite missions involves terminating simulations and thus avoids some of the difficulties found by Hopmans and Kleijnen (1979).

One possible general formulation of the simulation with roulette and splitting is the following. Let $\{X(t), 0 \le t \le t_m\}$ be a stochastic process on a state space S, where t_m is the deterministic (or random) mission duration. Let $G_1, G_2, \ldots, G_g$ and $B_1, B_2, \ldots, B_b$ be disjoint subsets of S. The sets B_i are "bad" in the sense that they are more likely to lead to system failure. The sets G_i are "good" in the sense that passage into one of these sets signifies an improvement in the system, making system failure appear less likely. The "Russian roulette" aspect of the simulation occurs when a sample path enters one of the "good" subsets: If G_i is entered at time t, the path is terminated with probability $p_i(t)$ and continued with probability $1 - p_i(t)$; if the path is continued its weight is increased by a factor of $(1 - p_i(t))^{-1}$. The "splitting" aspect of the simulation occurs when a sample path enters one of the "bad" subsets: If B_j is entered at time t, it is split into $s_j(t)$ paths, each of which continues in time independently of the others (conditional on the common history up to time t); the weight of each path is decreased by a factor of $1/s_j(t)$. At time t_m, the mission failure probability is estimated using the sum of the weight of all surviving paths. By replicating, confidence intervals can be computed. The statistical design of this experiment amounts to choosing the G_i's, B_j's, $p_i(\cdot)$'s, $s_j(\cdot)$'s, and the number of independent paths to start with at time 0. (Even more general formulations of the problem are possible; for example, splitting may occur at a certain probabilistic rate while in a certain subset rather than only when entering the subset. Or perhaps, roulette may only be allowed for paths which have previously split and it is not allowed to terminate all branches of one original path.) Kahn (1956) derives some formulae for determining optimal design parameters for experiments involving bivariate observations, i.e. stochastic processes indexed by a set with two elements.

In this paper we shall restrict the analysis to "splitting" and use a single class B. Namely, let B consist of all states where two active (undetected) faults exist. Furthermore, we assume that the sample path splits into b branches upon hitting B for the first time independent of the time of hitting. Figure 5 depicts a sample path which never hits B and therefore does not split. Figure 6 depicts a sample path which hits B at time t_B whereupon it splits into four continuations. In the next section we discuss the statistical analysis of this special case.

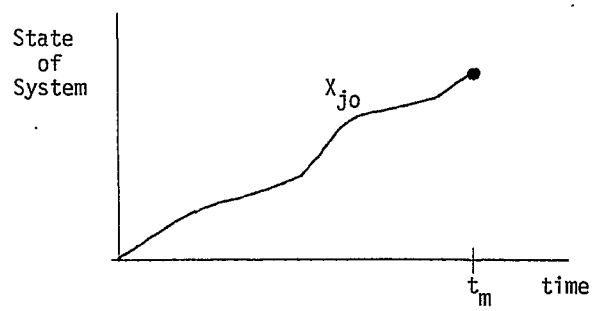

Figure 5: Typical sample path during the jth replication, where the branching point is not hit and the mission was successful.

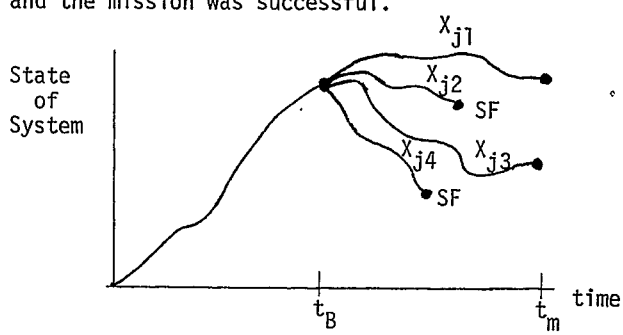

Figure 6: Sample path during the jth replication, where the branching point was hit at time t_B. Here, branches 1 and 3 led to a successful mission, whereas branches 2 and 4 experienced a system failure.

4. STATISTICAL ANALYSES OF PROCEDURE

For the purpose of comparison, we first describe the standard sampling scheme with no path-splitting and its statistical analysis. Suppose r independent replicates are observed. Define

$$x_j = \begin{cases} 1 \text{ , if failure occurs on replicate } j \\ 0 \text{ , otherwise} \end{cases}$$

and

$$t_j(F|F) = \begin{cases} \text{failure time on replicate } j \text{ , if} \\ \text{failure occurs.} \\ 0 \text{ , otherwise} \end{cases}$$

An estimate of system failure probability $P(F)$ is then

$$\hat{p}(F) = \frac{1}{r} \sum_{j=1}^{r} x_j$$

This is an unbiased estimator with variance

$$\mathrm{Var}(\hat{P}(F)) = P(F)(1-P(F))/r \ .$$

This variance can be estimated by

$$\hat{v}/r = \hat{p}(F)(1-\hat{p}(F))/r$$

The expected simulation time per replicate is

$$E(S) = P(F)E(T_F|F) + P(\overline{F})t_m$$

which can be estimated by

$$\hat{s} = \hat{p}(F)\overline{t}(F|F) + \hat{p}(\overline{F})t_m$$

where

$$\overline{t}(F|F) = \sum_{j=1}^{r} t_j(F|F) \Big/ \sum_{j=1}^{r} x_j$$

is an estimate of $E(T_F|F)$. The efficiency measure of this design, $r\mathrm{Var}(\hat{P}(F))E(S)$ can be estimated by $\hat{v}\hat{s}$. We shall compute this quantity and compare it to the similar quantity for other designs.

Now consider the path-splitting case. As before, r independent replicates are observed. If the sample path of a replicate hits the branch point, there will be b continuations of the process after the branch point. Let B denote the event that the branch point is reached and F the event of system failure. We observe the following statistics

$$x_{j,0} = \begin{cases} 1 \text{ , if } (\overline{B},F) \text{ occurs on replicate } j \\ 0 \text{ , otherwise} \end{cases}$$

$$x_{j,k} = \begin{cases} 1 \text{ , if } B \text{ occurs on replicate } j \\ \quad \text{and then } F \text{ occurs on branch } k \\ 0 \text{ , otherwise} \end{cases}$$

n = number of replicates which branch

$\overline{n}$ = number of replicates which do not branch

$$t_j(B|B) = \begin{cases} \text{time of branch on replicate } j \text{ ,} \\ \text{if } B \text{ occurs.} \\ 0 \text{ , otherwise} \end{cases}$$

$$t_{j,k}(F|B,F) = \begin{cases} \text{time of failure on } k\text{th branch of} \\ \text{replicate } j \text{ , if } (B,F) \text{ occurs} \\ 0 \text{ , otherwise} \end{cases}$$

$$t_j(F|\overline{B},F) = \begin{cases} \text{time of failure on replicate } j \text{ ,} \\ \text{if } (\overline{B},F) \text{ occurs} \\ 0 \text{ , otherwise} \end{cases}$$

From this data we compute the estimates of $P(B)$, $P(\overline{B})$, $P(F|B)$, $P(F|\overline{B})$ and $P(F)$:

$$\hat{p}(B) = n/r$$

$$\hat{p}(\overline{B}) = \overline{n}/r$$

$$\hat{p}(F|B) = \frac{1}{n} \sum_{j=1}^{r} \frac{1}{b} \sum_{k=1}^{b} x_{j,k}$$

$$\hat{p}(F|\overline{B}) = \frac{1}{\overline{n}} \sum_{j=1}^{r} x_{j,0}$$

$$\hat{p}(F) = \hat{p}(F|B)\hat{p}(B) + \hat{p}(F|\overline{B})p(\overline{B})$$

$$= \frac{1}{r} \sum_{j=1}^{r} \left(x_{j,0} + \frac{1}{b} \sum_{k=1}^{b} x_{j,k}\right)$$

The estimator $\hat{p}(F)$ is unbiased and its variance can be computed. Let

$$V(b) = \mathrm{Var}\left(X_{1,0} + \frac{1}{b} \sum_{k=1}^{b} X_{1,k}\right)$$

$$= (P(F|B) - P(F|B)^2)P(B)/b$$

$$+ (P(FF|B) - P(F|B)^2)P(B)(b-1)/b$$

$$+ P(F|B)^2P(B) + P(F|\overline{B})P(\overline{B}) - P(F)^2 \ ,$$

where $P(FF|B)$ equals the probability that both of a pair of branches lead to system failure for a replicate which hits the branch point; it can be estimated by

$$p(FF|B) = \frac{1}{n} \sum_{j=1}^{r} \frac{\left(\sum_{k=1}^{b} x_{j,k}\right)\left(\sum_{k=1}^{b} x_{j,k} - 1\right)}{b(b-1)} \ .$$

An estimate $\hat{v}(b)$ for $V(b)$ can be obtained by substituting the appropriate probability estimate into the equation for $V(b)$, then $\hat{v}(b)/r$ is an estimate for $\mathrm{Var}(\hat{P}(F))$.

The expected simulation time per replicate is

$$E(S(b)) = P(B)E(T_B|B) + P(B)b(P(F|B)E(T_F|B,F) +$$

$$P(\overline{F}|B)t_m - E(T_B|B)) + P(\overline{B})(P(\overline{F}|\overline{B})t_m +$$

$P(F|\overline{B})E(T_F|\overline{B}))$.

An estimate $\hat{s}(b)$ for $E(S(b))$ can be obtained by substituting estimates for the probabilities and expected times into the equation for $E(S(b))$; $E(T_B|B)$, $E(T_F|B,F)$ and $E(T_F|\overline{B})$ can be estimated by

$$\overline{t}(B|B) = \frac{1}{n}\sum_{j=1}^{r} t_j(B|B)$$

$$\overline{t}(F|B,F) = \frac{1}{f_B}\sum_{j=1}^{r}\sum_{k=1}^{b} t_{j,k}(F|B,F)$$

$$\overline{t}(F|\overline{B}) = \frac{1}{f_0}\sum_{j=1}^{r} t_j(F|\overline{B},F)$$

where

$$f_B = \sum_{j=1}^{r}\sum_{k=1}^{b} x_{j,k} , \quad f_0 = \sum_{j=1}^{r} x_{j,0} .$$

The efficiency measure of this procedure equals $V(b)E(S(b))$; it can be estimated by $\hat{v}(b)\hat{s}(b)$. We shall compare estimates of this efficiency measure for different designs to make comparisons.

5. A SEQUENTIAL PROCEDURE

It is impossible to know in advance the number of branches b into which to split a path when it hits the branch point. We would like to use the value which minimizes the efficiency measure $V(b)E(S(b))$. In Section 4 we assumed a fixed value of b , which the simulation analyst must guess before starting the simulation. In this section we propose a sequential procedure which allows the value of b to change from the initial guess.

Suppose the simulation is split up into ℓ blocks. Each block consists of r replicates. Let $b(i)$ equal the number of branches into which a replicate is split when it hits a branch point. For the ith block we shall collect the same data and compute exactly the same probability and time estimates as before. We add a subscript "i" to denote ith block. The data are:

$$x_{i,j,0} \; ; \; t_{i,j}(F|\overline{B},F) \; ;$$

$$x_{i,j,k} \; ; \; t_{i,j}(B|B) \; ; \; t_{i,j,k}(F|B,F) \; ;$$

$$n(i) \; ; \; \overline{n}(i) \; ;$$

$$f_{i,0} = \sum_{j=1}^{r} x_{i,j,0} \; ; \; f_{i,B} = \sum_{j=1}^{r}\sum_{k=1}^{b(i)} x_{i,j,k} .$$

The estimates of $P(B)$, etc. for the ith block are:

$$\hat{p}_i(B) , \hat{p}_i(\overline{B}) , \hat{p}_i(F|B) , \hat{p}_i(F|\overline{B}) , \hat{p}_i(FF|B) ,$$

$$\overline{t}_i(B|B) , \overline{t}_i(F|B,F) , \overline{t}_i(F|\overline{B},F) ,$$

$$\hat{p}_i(F) = \hat{p}_i(F|B)\hat{p}_i(B) + \hat{p}_i(F|\overline{B})\hat{p}_i(\overline{B}) .$$

As before this estimator of $P(F)$ is unbiased and its variance $V(b(i))/r$ can be computed using the formula from Section 4.

At the end of each block we estimate the efficiency measure per replicate for different values of b , $V(b)E(S(b))$, $b = 1,2,\ldots$. The minimizing value of b is then used as the branch number in the simulation of the next block of replicates. In order to estimate $V(b)E(S(b))$ we need estimates of $P(B)$, $P(F|B)$, $P(FF|B)$, $P(F|B)$, $E(T_B|B)$, $E(T_F|B,F)$, and $E(T_F|B,F)$.

If we have completed m blocks we will estimate these quantities by pooling the estimates from the individual blocks. The pooled estimates are

$$\hat{p}^*_m(B) = \frac{1}{m}\sum_{i=1}^{m} \hat{p}_i(B)$$

$$\hat{p}^*_m(F|B) = \sum_{i=1}^{m} n(i)b(i)\hat{p}_i(F|B) \Big/ \sum_{i=1}^{m} n(i)b(i)$$

$$\hat{p}^*_m(FF|B) = \frac{\sum_{i=1}^{m} n(i)b(i)(b(i)-1)\,\hat{p}_i(FF|B)}{\sum_{i=1}^{m} n(i)b(i)(b(i)-1)}$$

$$\hat{p}^*_m(F|\overline{B}) = \sum_{i=1}^{m} \overline{n}(i)\hat{p}_i(F|\overline{B}) \Big/ \sum_{i=1}^{m} \overline{n}(i)$$

$$\overline{t}^*_m(B|B) = \sum_{i=1}^{m} n(i)\overline{t}_i(B|B) \Big/ \sum_{i=1}^{m} n(i)$$

$$\overline{t}^*_m(F|B,F) = \sum_{i=1}^{m} f_{i,B}\overline{t}_i(F|B,F) \Big/ \sum_{i=1}^{m} f_{i,B}$$

$$\overline{t}^*_m(F|\overline{B},F) = \sum_{i=1}^{m} f_{i,0}\overline{t}_i(F|\overline{B},F) \Big/ \sum_{i=1}^{m} f_{i,0} .$$

Substituting the above estimates into the equation for $E(S(b))$ we obtain an estimate for this quantity which we shall denote $\hat{s}^*_m(b)$. Similarly, we estimate $V(b)$ by substituting the above estimates into the equation for $V(b)$. A slight difficulty can arise here: the covariance term $P(FF|B) - P(F|B)^2$ may be estimated as negative. We know that this is impossible because we are dealing with mixtures of bivariate independent Bernoulli random variables which in general have non-negative covariance. If the covariance is incorrectly estimated as negative, there are instances where the estimate of $V(b)E(S(b))$, denoted $\hat{v}^*_m(b)\hat{s}^*_m(b)$ will achieve its minimum when $b = \infty$. Thus we estimate the

covariance term of the equation for $V(b)$ by $\max(0, \hat{p}^*_m(FF|B) - \hat{p}^*_m(F|B)^2)$. Thus gives us an estimate $\hat{v}^*_m(b)\hat{s}^*_m(b)$ of the efficiency measure $V(b)E(S(b))$. Letting $b(m+1)$ equal the value of b which minimizes the value of $\hat{v}^*_m(b)\hat{s}^*_m(b)$ gives us the branching number for next block of replicates.

After simulating the desired number (ℓ) of blocks of replicates, we compute a final estimate of $P(F)$. There are two choices:

$$\hat{p}^*_\ell(F) = \hat{p}^*_\ell(F|B)\hat{p}^*_\ell(B) + \hat{p}^*_\ell(F|\bar{B})\hat{p}^*_\ell(\bar{B})$$

or

$$\hat{p}^\dagger_\ell(F) = \frac{1}{\ell}\sum_{i=1}^{\ell} \hat{p}_\ell(F) .$$

The estimate $\hat{p}^\dagger_\ell(F)$ has the advantage that its variance is easily computed and then estimated:

$$\mathrm{Var}(\hat{p}^\dagger_\ell(F)) = \sum_{i=1}^{\ell} v(b(i)) \Big/ r\ell .$$

The variance of $p^*_\ell(F)$ appears difficult, if not impossible, to compute; however, there is a strong possibility that it has a smaller variance. If the values of $b(i)$ converge quickly, it seems reasonable that the two estimators will give virtually identical values. For the purpose of estimating the efficiency measure of the method we use an estimate based on the above variance multiplied by the total simulation time.

6. SIMULATION OF SYSTEM

The first thing to note for the above example is that most missions did not have two or more component failures and thus system failure was impossible. Letting T_1, T_2, T_3 and T_4 be ordered fault occurrence times, for the above example $P(T_2 < t_m) = .00058618$ (where t_m = 1 hour, and the component lifetimes are exponentially distributed with MTBF = 100 hours). Thus it is unnecessary to simulate 99.94% of the paths. We just generate T_1, T_2, T_3 and T_4 conditional on $\{T_2 < t_m\}$, as follows

$$P(T_2 < t_m) = P(\text{at least 2 component failures in } [0,t_m])$$

$$= 1 - (e^{-\lambda t_m})^4 - 4(1 - e^{-\lambda t_m})(e^{-\lambda t_m})^3.$$

$$P(t_2 < T_2 < t_2 + dt \mid T_2 < t_m)$$

$$= 12(1 - e^{-\lambda t_2})\lambda e^{-\lambda t_2} dt (e^{-\lambda t_2})^2 \Big/ P(T_2 < t_m)$$

We use the rejection method to generate the random variate T_2 from the above density. After generating T_2, we generate the random variate T_1 conditional on the observed value of T_2. The conditional cdf is

$$P(T_1 \le t_1 \mid T_2 = t_2) = \frac{1 - e^{-\lambda t_1}}{1 - e^{-\lambda t_2}}$$

We used the inverse cdf method to generate the random variate T_1 from the above cdf. Once the second component failure occurs, the subsequent T_3 and T_4 can be generated easily as $T_3 - T_2$ and $T_4 - T_3$ are independent exponential variates with rates 2λ and λ respectively. The aforementioned technique gives a tremendous savings: a reduction of approximately 99.9% in total simulation time.

The second point is that the behavior of the system is deterministic up to T_1 and then there is no need to simulate it during $[0,T_1]$. We start simulating at T_1. This gives an additional savings of approximately 33% in simulation time. (In general this can't be done, so our conclusions concerning simulation time are based on the assumption that the process was simulated during the whole period $[0,t_m]$.)

If one or more undetected faults are present, we simulate error generation by conducting a Bernoulli experiment with parameter p equal to the error generation probability at the end of each 7 second application cycle. If the outcome is "success" an error occurs in all the processors which contain faults, otherwise all processors give the correct answer. If one processor generates an error it is reconfigured out of the system. If two or more generate errors, system failure occurs. Meanwhile, the diagnostic program is being executed during the diagnostic cycle, and we assume that the time to diagnose a fault is exponentially distributed with some diagnostic rate.

We decided to split the sample path when the process enters states consisting of "two active faults present" for the first time. We arbitrarily chose to split the path into five branches $(b = 5)$ for the fixed path-splitting scheme. When the sample path hits the branching point, we save the following information: (1) state of the system, (2) time of occurrence and (3) future

event list. This is easy to do with the Discrete Event World View. (The programs are written in SIMSCRIPT II.5)

7. COMPUTATIONAL EXPERIENCE

We ran four different cases, with error generation probability = .002 and .01 , and diagnostic rate = .002 and .001 . Table 1 presents the point estimates for the standard sampling scheme.

Table 1. Standard Sampling Scheme

Error Gen. Probability	Diagnostic Rate	Number of Replicates	Point Estimate of P{sys. fail|2}	Estimate of Estimation Error σ	Simulation Time	Efficiency Measure
.002	.002	400	.01000000	.00497494	963,527	23.85
.002	.002	2000	.01600000	.00280571	4,748,745	37.38
.002	.001	400	.03500000	.00918898	351,447	80.34
.002	.001	2000	.03400000	.00405240	4,708,722	77.33
.01	.002	400	.04250000	.01008634	944,368	96.07
.01	.002	2000	.04600000	.00468422	4,656,700	102.18
.01	.001	400	.08250000	.01375625	933,110	176.58
.01	.001	2000	.07500000	.00588861	4,594,670	159.38

Table 2 presents the point estimates for the case where the splitting technique was applied with a fixed number of branches (b = 5) .

Table 2: Path-splitting Variance Reduction Scheme Fixed number of branches = 5

Error Gen. Rate	Diagnostic Rate	Number of Replicates	Point Estimate of P{sys. fail|2}	Estimate of Estimation Error σ	Simulation Time	Efficiency Measure
.002	.002	400	.01199999	.00252982	1,869,938	11.96
.002	.002	2000	.01449999	.00125698	9,235,245	14.64
.002	.001	400	.03749999	.00451110	2,013,718	40.97
.002	.001	2000	.03379997	.00196214	10,426,922	40.13
.01	.002	400	.04949985	.00648999	1,503,742	63.34
.01	.002	2000	.04789997	.00265330	7,729,047	54.43
.01	.001	400	.08949989	.00840952	1,638,789	115.90
.01	.001	2000	.07649994	.00343220	8,086,885	95.23

Table 3 presents the point estimates obtained using the sequential scheme with blocks of 100 replicates (so, we had 4 blocks of 100 replicates, and 20 blocks of 100 replicates for each case). In this table the number of branches used in the last block is shown. From our experience, the convergence of the optimization scheme was fast in all cases. (The specific sequences of b(i)'s were: (i) for (ε = .002, r = .002) , (b(i), i = 1, 2, ...,20) = (5, 14, 10, 10, 11, 11, 12, 12, 13, 13, 13, 13, 14, 14, 13, 13, 14, 14,

13, 13) ; (ii) for (ε = .002, r = .001) , (5, 11, 12, 11, 11, 10, 11, 11, 10, 10, 10, 10, 11, 10, 10, 10, 11, 11, 10, 10) ; (iii) for (ε = .01, r = .002) , (5, 8, 7, 7, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7) ; (iv) for (ε = .01, r = .001) , (5, 6, 5, 5, 5, 5, 5, 5, 5, 5,...,5) . Even though convergence of the b(i)'s was quite fast and also stable, we observed that the objective function was quite flat. In Table 3, the final point estimate is

$\hat{p}^{*}_{20}(F)$, but the "Estimate of Estimation Error"

is the square root of the estimate of $Var(\hat{P}^{\dagger}_{20}(F))$

as discussed in Section 5.

Table 3: Sequential (Replicates split into blocks of 100)

Error Gen. Probability	Diagnostic Rate	Number of Replicates	Final Optional Number of Branches	Point Estimate of P{sys.fail |2}	Estimate of Estimation Error σ	Simulation Time	Efficiency Measure
.002	.002	400	10	.01731585	.00245357	2,954,350	17.78
.002	.002	2000	13	.01545740	.00095394	17,528,016	15.90
.002	.001	400	11	.03233641	.00333317	3,359,782	37.33
.002	.001	2000	10	.03214544	.00144568	18,259,600	38.21
.01	.002	400	7	.04819386	.00545802	1,899,295	56.59
.01	.002	2000	7	.04575009	.00247184	9,017,191	55.13
.01	.001	400	5	.07783008	.00779295	1,776,468	107.89
.01	.001	2000	5	.07350981	.00348134	8,479,483	102.75

In Table 4 we present point estimates of a run using the splitting technique with fixed number of branches again, but where this number of branches is the number of branches used in the 20th block of the sequential case. Note that in Tables 1 through 4 we estimate the conditional probability of system failure given at least two processor faults during $[0,t_m]$; in order

to obtain the unconditional probability of system failure. These numbers must be multiplied by .00058618 .

Table 4: Path-splitting Variance Reduction Scheme Fixed Number of Branches = Optimal Number Estimated

Error Gen. Probabiliy	Diagnostic Rate	Number of Replicates	Number of Branches	Point Estimate of P{sys.fail |2}	Estimate of Estimation Error σ	Simulation Time	Efficiency Measure
.002	.002	400	10	.01791666	.00227816	3,321,968	17.23
.002	.002	2000	13	.01624999	.00089443	19,354,112	15.44
.002	.001	400	11	.03727269	.00340735	3,703,299	42.99
.002	.001	2000	10	.03559995	.00155885	17,716.464	42.97
.01	.002	400	7	.04285712	.00529717	1,864,769	52.33
.01	.002	2000	7	.04135712	.00223159	9,167,203	45.62
.01	.001	400	5	.08949989	.00840952	1,638,789	115.90
.01	.001	2000	5	.07649994	.00343220	8,086,885	95.29

In Table 5 the estimated amount of variance reduction is summarized. The efficiency measure is the estimate of the variation in one replicate times the expected simulation time per replicate.

Table 5. Estimates of Variance Reduction

Error Gen. Probability	Diagnostic Rate	Number of Replicates	Efficiency Measures			Variance Reduction	
			Standard	Fixed(5)	Sequential	Fixed(5)	Sequential
.002	.002	2000	37.38	14.64	15.90	60%	57%
.002	.001	2000	77.33	40.13	38.21	48%	51%
.010	.002	2000	102.18	54.43	55.13	47%	46%
.010	.001	2000	159.38	95.29	102.75	40%	36%

8. CONCLUSIONS & SUMMARY

Importance sampling was used for variance reduction. The "splitting" method requires more simulation time, so we used a measure of efficiency equal to the product of the estimated variance and simulation time. Percentage decreases of approximately 40 to 60 percent were achieved in this measure from the standard to the splitting case (with fixed number of branches = 5) .

The sequential scheme converges rapidly to the optimal number of branches. But, we observed cases where this "optimal number" estimated, actually was not optimal. In order words, the percentage reduction for the scheme with the arbitrarily chosen number of branches equal to 5 was greater than that achieved with the sequential method and also that achieved with a fixed number of branches equal to the last value in the sequential method (Table 4). There is considerable noise in the system which may be the source of this apparent contradiction. It would be necessary to simulate more cases or examples to get a clearer picture of this aspect of the performance of the sequential method. Nevertheless, in the runs of 2000 replications, if the sequential scheme is not better (in the sense that it doesn't give the best estimate of reduction in efficiency measure) it appears close to optimal.

This preliminary study shows that the "splitting" technique is a promising method but that more study is necessary to understand its properties more fully.

REFERENCES

Bayes AJ, (1970), "Statistical Techniques for Simulation Models," Australian Computer Journal, Vol. 2, pp. 180-184.

Carter G, Ignall E, (1975) "Virtual Measures: A Variance Reduction Technique for Simulation," Management Science, Vol. 21, pp. 607-616.

Hammersley JM, Handscomb DC, (1964) Monte Carlo Methods, Methuen, London.

Hopkins AH, Smith III TB, Lala JH, (1978) "FTMP--A Highly Reliable Fault-tolerant Multiprocessor for Aircraft Control," Proceedings of the IEEE, Vol. 66, pp. 1221-1239.

Hopmans ACM, Kleijnen JPC, (1979), "Importance Sampling in Systems Simulation: A Practical Failure?," Mathematics and Computers in Simulation, Vol. 21, pp. 209-220.

Kahn H, (1956), "Use of Different Monte Carlo Sampling Techniques," Symposium on Monte Carlo Methods, Meyer HA (ed), pp. 146-190, New York: Wiley.

Wensley JH, Lanport L, Goldberg J, Green MW, Levitt KN, Melliar-Smith MP, Shostak RE, Weinstock CB, (1978), "SIFT: Design and Analysis of a Fault-tolerant Computer for Aircraft Control," Proceedings of the IEEE, Vol. 66, pp. 1240-1255.