

6. Simulation Methods

It is well established that simulation techniques have proven their value especially for problems where the representation of the state function is associated with difficulties. Such cases are e.g. when the limit state function is not differentiable or when several design points contribute to the failure probability.

The basis for simulation techniques is well illustrated by rewriting the probability integral in equation

$$P_f = \int_{g(x) \leq 0} f_X(X) dX$$

by means of an indicator function as shown in Equation

$$P_f = \int_{g(x) \leq 0} f_X(X) dX = \int I[g(X) \leq 0] f_X(X) dX$$

where the integration domain is changed from the part of the sample space of the vector $X = (X_1, X_2, \dots, X_n)^T$ for which $g(x) \leq 0$ to the entire sample space of X and where $I[g(x) \leq 0]$ is an indicator function equal to 1 if $g(x) \leq 0$ and otherwise equal to zero. Above equation is in this way seen to yield the expected value of the indicator function $I[g(x) \leq 0]$.

Therefore if now N realizations of the vector X , i.e. $\hat{x}_j, j = 1, 2, \dots, N$ are sampled it follows from sample statistics that

$$P_f = \frac{1}{N} \sum_{j=1}^N I[g(x) \leq 0]$$

Is an unbiased estimator of the failure probability P_f .

6.1.1. Crude Monte Carlo Simulation

The principle of the crude Monte Carlo simulation technique rests directly on the application of above equation. A large number of realisations of the basic random variables X , i.e. $\hat{x}_j, j = 1, 2, \dots, N$ are generated (or simulated) and for each of the

outcomes \hat{x}_j , it is checked whether or not the limit state function taken in \hat{x}_j is positive. All the simulations for which this is not the case are counted (n_f) and after N simulations the failure probability p_f may be estimated through

$$p_f = \frac{n_f}{N}$$

which then may be considered a sample expected value of the probability of failure, In fact for $N \rightarrow \infty$, the estimate of the failure probability becomes exact. However, simulations are often costly in computation time and the uncertainty of estimate is thus of interest. It is easily realized that the coefficient of variation of the estimate is proportional to $1/\sqrt{n_f}$ meaning that if Monte Carlo simulation is pursued to estimate a probability in the order of 10^{-6} it must be expected that approximately 10^8 simulations are necessary to achieve an estimate with a coefficient of variance in the order of 10%. A large number of simulations are required using Monte Carlo simulation and all refinements of this crude techniques have the purpose of reducing the variance of the estimate. Such methods are for this reason often referred to as variance reduction methods.

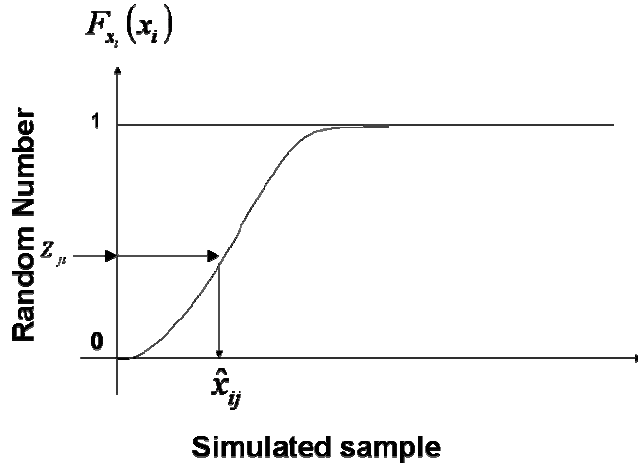
Simulation of the N outcomes of the joint density functions, in above equation principle simple and may be seen as consisting of two steps. Here we will illustrate the steps assuming that the n components of the random vector \mathbf{X} are independent.

In the first step a “pseudo random” number between 0 and 1 is generated for each of the components in \hat{x}_j . $\hat{x}_{ji}, j=1, \dots, n$ The generation of such numbers may be facilitated by build-in functions of basically all programming languages and spreadsheet software

In the second step the outcomes of the “pseudo random” numbers Z_{ji} are transformed to outcomes of \hat{x}_j by

$$\hat{x}_{ji} = F_{ji}^{-1}(Z_{ji})$$

where F_{x_i} is the probability distribution function for the random variable X_i , principle is also illustrated in Figure.



This process is continued until all components of the vector \hat{x}_j have been generated.

6.1.2. Importance of Sampling Simulation Method

As already mentioned the problem in using above equation is that the sampling function $f(x)$ typically is located in a region far away from the region where the indicator function $I[g(x) \leq 0]$ attains contributions. The success rate in the performed simulations are thus low. In practical reliability assessment problems where typical failure probabilities are in the order of 10^{-3} to 10^{-6} this in turn leads to the effect that the variance of the estimate of failure probability will be rather large unless a substantial amount of simulations are performed.

To overcome this problem different variance reduction techniques have been proposed aiming at, with the same number of simulations to reduce the variance of the probability estimate.

The importance sampling method takes basis in the utilization of prior information about the domain contribution to the probability integral, i.e. the region that contributes to the indicator function. Let us first assume that we know which point in the sample space x^* contributes the most to the failure probability. Then by centering the simulations on this point, the important point, we would obtain a higher success rate in the simulations and

the variance of the estimated failure probability would be reduced. Sampling centered on an important point may be accomplished by rewriting above equation 1 in the following way

$$P_f = \int I[g(x) \leq 0] f_x(X) dX = \int I[g(x) \leq 0] \frac{f_x(X)}{f_s(X)} f_s(X) dX$$

which $f_s(x)$ is denoted the importance sampling density function. It is seen the integral in above equation represents the expected value of the term $I[g(x) \leq 0] \frac{f_x(X)}{f_s(X)}$ where the

components of s are distributed according to $f_v(x)$. The question in regard to the choice of an appropriate importance sampling function $f_s(x)$, however, remains open

One approach to the selection of an importance sampling density function $f_s(x)$ to select a n -dimensional joint Normal probability density function with uncorrelated components, mean values equal to the design point as obtained from FORM analysis, i.e. $\mu_s = x^*$ and standard deviations standard deviation of the component of X i.e. $\sigma_s = \sigma_x$. In this case the above equation may written as

$$P_f = \int I[g(x) \leq 0] \frac{f_x(X)}{f_s(X)} f_s(X) dX = \int I[g[x] \leq 0] \frac{f_x(X)}{\phi(X)} \phi(X) dX$$

In the equivalence of the equation leading to

$$P_f = \frac{1}{N} \sum_{j=1}^N I[g(s) \leq 0] \frac{f_x(s)}{\phi(s)}$$

which may be assessed by sampling over realizations of s as described in the above.

Application of these equations greatly enhances efficiency of the simulations. If the limit state function is not too non-linear around the design point x^* the success rate of the simulations will be close to 50%. If the design point is known in advance in a reliability problem where the probability of failure is in the order of 10^{-6} the number of simulations required to achieve a coefficient of variance in the order of 10% is thus around 200. This number stands in strong contrast to the 10^8 required using the crude Monte Method discussed before, but of course also requires knowledge about the design point.

6.2. GENERATION OF RANDOM NUMBERS

6.2.1. Random Outcomes from Standard Uniform Variates

The probability integral transform indicates that generation of uniform (0, 1) random numbers is the basic generation process used to derive the outcomes from a variate with known probability distribution. Current methods of generating standard uniform variates are deterministic, in the sense that systematic procedures are used after one or more initial values are randomly selected. For example, system-supplied *random number generators* in most digital computers are almost always linear congruent generators. This algorithm is based on recursive calculation of sequence of integers k_1, k_2, k_3, \dots , each between 0 and $m-1$ (a large number) from a linear transformation:

$$k_{i+1} = (ak_i + c) \pmod{m}$$

Here a and c are positive integers called the *multiplier* and the *increment* respectively, and the notation (modulo m) signifies that k_i is the remainder obtained after dividing $(ak_i + c)$ by m where m denotes a (large) positive integer, hence denoting $\eta_i = \text{Int}[(ak_i + c)m]$ the corresponding residual is defined as

$$k_{i+1} = ak_i + c - m\eta_i$$

Hence,

$$u_{i+1} = k_{i+1} / m = (ak_i + c) / m = \text{Int}[(ak_i + c)m] / m$$

where the u_i are uniform(0,1). Because these numbers are repeated with a given period, they are usually called pseudo-random numbers. The quality of the results depends on the magnitudes of the constants a, c and m and their relationships, but the particular computer used will impose constraints. Because the period of the cycle is not greater than m and it increases with m , the main criterion is that the periods after which the original numbers are unavoidably repeated should be as long as possible. In practice, m is set equal to the word length that is, the number of bits retained as a unit in the computer. Moreover, the constants c and m should not have any common factors, and the value of a should be sufficiently high. Because all possible integers between 0 and $m-1$ occur after some interval of time, regardless of the generator used, any initial choice of the seed k_0 is as good as any other.

Linear congruential algorithm

Suppose we assume low values for the constants in equation, $a = 5$, $c = 1$ and $m = 8$. Let $k_0 = 1$ be the seed for generating a sequence of random integers k_i $i=1,2,3,\dots$. For $i=1$. One has

$$\begin{aligned} k_1 &= ak_0 + c - m \text{Int}[(ak_0 + c)/m] = 5 \times 1 + 1 - 8 \times \text{Int}[(5 \times 1 + 1)/8] \\ &= 5 + 1 - 8 \times \text{Int}(0.75) = 5 + 1 - 8 \times 0 = 6 \end{aligned}$$

and from equation

$$u_1 = k_1/m = 6/8 = 0.75$$

The second iteration yields

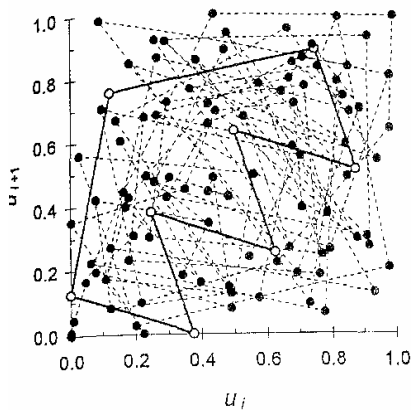
$$\begin{aligned} k_2 &= ak_1 + c - m \text{Int}[(ak_1 + c)/m] = 5 \times 6 + 1 - 8 \times \text{Int}[(5 \times 6 + 1)/8] \\ &= 30 + 1 - 8 \times \text{Int}(3.875) = 30 + 1 - 8 \times 3 = 7 \end{aligned}$$

$$u_2 = k_2/m = 7/8 = 0.875$$

The subsequent iterations yield the following sequence

0.5, 0.625, 0.25, 0.375, 0, 0.125, 0.75, 0.875, 0.5, 0.625, 0.25, 0.375, 0, 0.125, 0.75, 0.875, 0.5, 0.625, 0.25, 0.375, 0, 0.125, 0.75, 0.875, 0.5, 0.625, 0.25, 0.375, 0, 0.125

which is seen to be cyclic with a period of 8, because the underlined sequence of 8 values is repeated indefinitely. This is clearly shown by plotting u_{i+1} against u_i in



Trajectory of 100 sequentially generated standard uniform random numbers with $a = 5$, $c = 1$, and $m = 8$ (solid line) and with $a = 2^7 + 1$, $c = 1$ and $m = 2^{35}$ (dotted line)

Also shown in figure are results from the generator $a = 2^7 + 1$, $c = 1$ and $m = 2^{35}$, which yields a much larger period of cyclicity. This choice gives satisfactory results for binary computers: and $a = 101$, $c = 1$, and $m = 2^b$ for a decimal computer with a word length b .

The advantage of the linear congruencies method when applied through a digital computer is the speed of implementation. Because only a few operations are required each time, its use has become widespread. A disadvantage is that once the seed is specified, the entire series is predictable.

The pseudorandom numbers generated by these procedures can be tested for uniform distribution and for statistical independence. Goodness-of-fit tests, such as the chi-squared and the Kolmogorov-Smirnov tests can be used to verify that these numbers are uniformly distributed. Both parametric and nonparametric methods, such as the runs test, can be used to check for randomness between successive numbers in a sequence in spite of the fact that these procedures are essentially deterministic. Pseudo-random numbers generated with large m and accurate choices of a and c generally appear to be uniformly distributed, and stochastically independent, so that they can be properly used to perform Monte Carlo simulations. Algorithms to generate pseudo-random numbers which closely approximate mutually independent standard uniform variates are a standard feature in statistical software. Standard uniform random numbers are available as a system-supplied function in digital computers, as well as in most customary computational and data management facilities such as spreadsheets and data bases. Now a days, software tools MATLAB, MS Excel and Mathcad have built in options to generate random numbers and evaluation of probability of failure. Problems solved using Hasofer –Lind approach and FORM methods earlier can be reworked using simulation procedures.