An Introduction to Efficient Simulation Techniques

Gary C. Hill

Signals Analysis Discipline
Communications Division
DSTO

1 Introduction

Many physical and algorithmic processes have outcomes that cannot be determined analytically, and therefore must be solved using numerical methods. When a problem contains elements that have a random behaviour, the system is typically modelled and the outcome determined using Monte Carlo simulation techniques. Each random element in the simulation is described by a probability distribution function, which expresses the likelihood of any given realisation of that element occurring. The Monte Carlo method consists of sampling events from the underlying probability density functions and determining their effect on the final outcome. By taking many samples, the distribution of the final outcome can be inferred. In many applications of the Monte Carlo method, the most interesting outcomes are induced by the rare random events in the simulation. Obtaining a good estimate of the outcomes due to rare events by sampling from the original probability distribution functions is sometimes not possible, as not enough of the rare events will be selected over the course of the simulation to obtain a statistically significant result. Simulations can however be made more efficient by artificially increasing the rate at which the rare events occur in the simulation, and weighting the outcomes accordingly. This paper discusses techniques for making Monte Carlo simulations more efficient, thereby allowing the final solutions to be obtained with significant reduction in computational time and effort.

2 Methods of sampling from distributions

In order to perform a simulation we must be able to sample deviates that are distributed according to various probability density functions. Two methods are described here.

2.1 Integration method

We start with a probability density function $f(x)$, which satisfies

$$\int_{-\infty}^{+\infty} f(x)dx = 1 \tag{1}$$

$f(x)dx$ represents the probability that the value x lies in the infinitesimal interval $(x, x + \delta x)$. To choose a random value x_r from the interval (x_1, x_2) consistent with the distribution $f(x)dx$, we solve the following equation for x_r

$$\int_{x_1}^{x_r} f(x)dx = r \int_{x_1}^{x_2} f(x)dx \tag{2}$$

which equates the area between x_1 and x_r with the fraction r of the total area in the interval from x_1 and x_2. As an example, consider the distribution $f(x)dx = 1/x$. Integrating yields

$$\ln(x_r) - \ln(x_1) = r(\ln(x_2) - \ln(x_1)) \tag{3}$$

and solving for x_r gives

$$x_r = \exp(r \ln x_2 + (1 - r) \ln x_1) \tag{4}$$

This maps the uniformly distributed random numbers in the interval $(0, 1)$ to the function $f(x)$ in the interval (x_1, x_2).

2.2 Comparison function/rejection method

Sampling using the integration method relies on the probability density function being easily integrable. If the probability density function $f(x)$ cannot be simply integrated to yield a simple form as in equation 4, but if $f(x)$ is smaller than some other easily integrated function $g(x)$ (the comparison function) for all x in the region (x_1, x_2), then the comparison function/rejection method may be applied. Two random numbers x_1 and x_2 are chosen from the interval $(0, 1)$. The first is used to choose a value x_r from the distribution $g(x)$, the second to choose a corresponding value $y = r_2 g(x_r)$ This in effect samples values (x_r, y) uniformly distributed over the area under the comparison function $g(x)$. The x values of those coordinates (x, y) that also lie in the area bounded by the function $f(x)$ are random samples from the distribution $f(x)$. If the coordinate (x_r, y), selected from the area bounded by $g(x)$ satisfies $y < f(x_r)$, then we *accept* the value x_r. If $y > f(x_r)$, then we *reject* x_r.

3 Examples of increased simulation efficiency

In this section we examine some numerical simulations where substantial increases in efficiency are possible. Firstly we examine a physical example (neutrino interactions in the earth), isolate the areas of inefficiency in the model, and show how such inefficiencies may be reduced. Next we look more formally at the theory of *importance samplimg*, using Monte Carlo integration as an example.

3.1 Neutrino interactions

Neutrinos are elementary particles that are believed to be produced in various astrophysical settings (eg binary X-ray systems, active galaxies) as the secondary products of the interactions of high energy particles with other matter and with radiation fields (photons). Once produced, the neutrinos propagate in straight lines. Various astrophysical models have been used to estimate the energies of the neutrinos that arrive at earth. The energy of a high energy particle is usually expressed in electon-volts (eV), where one eV is the energy gained by an electron when accelerated through a potential of one Volt. The estimated spectrum of neutrinos $F(E)$ typically follows a power law like

$$F(E) = CE^{-\gamma}\mathrm{cm}^{-2}\mathrm{sec}^{-1}\mathrm{steradian}^{-1} \tag{5}$$

with a value of γ of about $2 - 3$, where C is a constant.

Neutrinos can only be detected by looking for the secondary particles they produce when they interact with a target. The best candidate for detection is the muon, which in many ways may be considered to be a heavier version of the electron. The produced muons basically retain the original neutrino direction, meaning that if the muon direction can be determined, then the neutrino direction can be found, allowing a map of the neutrino sky to built up. Muon detection is achieved by looking for the optical Cerenkov radiation they emit as they pass through a body of water (eg a lake or the ocean). For various reasons (eg greater signal, less backgrounds) the detectors are designed to look for upward going muons that resulted from interactions in the earth of neutrinos that arrived on other side of the earth. The probability of a neutrino producing a muon is determined using the Electro-weak/Quantum Chromodynamic framework which describes the interaction of neutrinos with the constituent quarks in protons and neutrons. It will suffice to say that, the higher the neutrino energy, the more likely it is to produce a secondary muon. The muons lose energy as they travel through the earth with the muon range being described by a probability distribution due to the stochastic nature of the various energy loss processes. In general, the higher the muon energy, the greater the range. The highest energy neutrinos produce muons that have a range in the earth of 10's of kilometres, so the detector will not see all the muons that are originally produced with trajectories that would take them through the detector.

The most basic Monte Carlo that could be implemented using the above physical model would go as follows.

1. Use the integration method (section 2.1) to select neutrino energies from the power law in equation 5.

2. Calculate the position in the earth at which the neutrino interacts (ie choose an interaction depth x randomly from an exponential distibution - $p(x) = 1/\lambda \exp(-x/\lambda)$ where λ is the energy dependent *mean free path*).

3. Start the muon off at the determined interaction depth. Choose the muon range from the appropriate distribution. See if the muon survives to the detector.

This is an extremely inefficent algorithm and thus will serve as an excellent example of how to apply weighting techniques to make a Monte Carlo more efficient.

Suppose that the power γ in equation 5 equals 3 and the neutrino energies run from 10^3GeV up to 10^8GeV. An E^{-3} power law means that as we go up one power of 10 in neutrino energy, the flux falls off by a factor of 1000. The selected energies will be predominantly around the lower limit, 10^3GeV. If 100000 neutrinos fall in the vicinity of 10^3GeV, then around 100 will fall in the vicinity of 10^4GeV. It is clear that many iterations will be needed to sample from the higher energy regions of the spectrum.

The mean free path for a 10^3GeV neutrino is around 20 earth thicknesses, meaning that only $1 - \exp(-1/20) = 0.05$ of the neutrinos will interact in the earth (the rest will have random interaction depths longer than an earth thickness). The range of the muons at these low energies is only a few kilometres, so unless the muon was produced in the last kilometres of earth, it will not survive to the detector. If we assume for a moment that the interaction depths are uniformly distributed over one earth thickness, then the probability of muon production in the last 3 km is about $3/r_{earth}$, which is roughly $3/12700 = 2 \times 10^{-4}$. Then, of the original 100000 neutrinos, we see $100000 \times 0.05 \times 2 \times 10^{-4} = 1$ muon for our effort, hardly enough to draw any conclusions as to the final energy spectrum of the observed muons!

We will tackle these inefficiencies step by step. The problem of the power law selection is important as higher energy neutrinos firstly have a greater chance of producing a muon, and secondly, these muons will be more energetic and will be more likely to reach the detector. We must weight our neutrino energy selections to give more emphasis to higher energies than the simple scheme based on the integration selection. A very simple method is to select equal numbers from each logarithmic energy bin, ie equal numbers from the intervals $10^3 - 10^4$GeV, $10^4 - 10^5$GeV,....,$10^7 - 10^8$GeV. The numbers of neutrinos in each logarithmic bin fall off by a factor of 100 each time (factor of 1000 drop in flux offset by an increase in bin width by factor 10). Each set of results generated is assigned a weight based on the based on the relative contribution of that energy bin to the total spectrum. The bins contain respectively 0.99, 0.0099, 0.000099, etc of the total and so each set of results is weighted by that number.

The problem of the low probability of interaction in the final kilometres of earth before the detector (rather than not interacting or interacting too far from the detector) is solved by artificially making *every* neutrino interact in the region where the muon has a high survival probability (final kms) and weighting each final result by the probability of that interaction actually occuring.

Given a neutrino of energy E, we select the muon energy from the appropriate distribution to give E_μ. Now there is a distribution of muon ranges for E_μ, and a maximum range r_{max} for that energy. If the muon is produced any further than r_{max} from the detector, it will not survive, and that iteration is wasted. So we force the interaction to take place within r_{max} of the detector, and choose the actual point from an exponential distribution. The muon range is then selected from the appropriate distribution and we see if the muon survives to the detector and note its energy. Now the weight assigned to the muon is just the original neutrino weight multiplied by the probability P that the muon was actually produced in the region $0 \to r_{max}$. P is the product of two probabilities

$$P = P_{\text{survive}} \; P_{\text{interact}} \tag{6}$$

where P_{survive} is the probability that the neutrino survived the thickness of the earth and $P_{interact}$ is the probability the neutrino then interacted in the thickness r_{max}. Now $P_{\text{survive}} = \exp(-(r_e - r_{max})/\lambda)$ and $P_{interact} = 1 - \exp(-r_{max}/\lambda)$ (where λ is again the energy dependent mean free path) yielding

$$P = \exp(-(r_e - r_{max})/\lambda)(1 - \exp(-r_{max}/\lambda)) \tag{7}$$

This, in combination with the original neutrino weight, is used to weight the muon in the final detected spectrum.

Now we have a highly efficient algorithm where every iteration has a high chance of contributing to the final spectrum. In practice, choosing 100000 neutrinos from each logarithmic energy bin is sufficient to yield good final results for the energy spectrum as a function of arrival angle.

3.2 Monte Carlo Integration

As an introduction to *importance sampling* techniques, we examine the numerical calculation of certain integrals, and techniques to reduce the variance in the computed integral.

Consider a function $f(x)$, which we wish to integrate over the interval (a, b):

$$I_f = \int_a^b f(x)dx \tag{8}$$

The strategy of Monte Carlo integration is to randomly sample N points $(x_1, ..., x_N)$ uniformly in the interval (a, b), and compute the average value of the function, $\langle f \rangle$, over these points

$$\langle f \rangle = \frac{1}{N} \sum_{i=1}^{N} f(x_i) \tag{9}$$

The integral is then approximated by this average value, multiplied by the length of the interval

$$I_f = \int_a^b f(x)dx \simeq (b-a)\langle f \rangle \tag{10}$$

An estimate of the error comes from the variance in the values $(b-a)f(x_i)$,

$$\begin{aligned} \sigma_f^2 &= \frac{1}{N} \sum_{i=1}^{N} ((b-a)(f(x_i) - \langle f \rangle))^2 \\ &= (b-a)^2(\langle f^2 \rangle - \langle f \rangle^2) \end{aligned} \tag{11}$$

where we have defined

$$\langle f^2 \rangle = \frac{1}{N} \sum_{i=1}^{N} f(x_i)^2 \tag{12}$$

If we compute I_f using N random deviates $(x_1, ..., x_N)$, then repeat the calculation using the next N deviates generated by the random number generator, we will not get the same answer and indeed, both answers will differ from the true I_f value. The spread in the values of I_f obtained by repeated calculation using sets of N iterations gives us an estimate of the error in the mean value obtained for I_f. The Central Limit Theorem relates the variance from a single calculation σ_f^2 to the variance for the averages over many calculations $(\sigma_{\bar{f}}^2)$ in the following way

$$\sigma_{\bar{f}}^2 = \frac{\sigma_f^2}{N} \tag{13}$$

Then the integral is estimated by

$$\int_a^b f(x)dx \simeq (b-a)\langle f\rangle \pm (b-a)\sqrt{\frac{\langle f^2\rangle - \langle f\rangle^2}{N}} \tag{14}$$

where the $\pm$ term is the *standard error*, describing the spread in the averages derived from many independent sets of N random x_i points.

As an example, consider the numerical integration of the function $1/x^2$ in the interval $(1, 10)$ which can be solved analytically to yield

$$I_f = \int_1^{10} \frac{1}{x^2} dx = 0.9 \tag{15}$$

If we compute I_f many times using independent sets $(x_1, ..., x_N)$ of deviates, and calculate the overall mean $(\bar{I}_f)$; of $(I_1, I_2, I_3, ...)$ and variance of the $I's$ we obtain $\bar{I}_f \sim 0.9$ and $\sigma^2_{\bar{I}_f} \sim 0.02$. Each calculation of the single run variance (equations 11 and 12) gives an answer about 2.0 $(= 100 \times 0.02)$ as expected.

How do we reduce the variance in the final result? Examination of equation 14 suggests that a larger value of N will reduce the error term. Alternatively, we want to find a way to reduce the variance of the final result whilst retaining the original value of N. The function $1/x^2$ takes higher values for lower x, suggesting that we concentrate our points in this region. We do this in a systematic way by defining a normalised sampling density, $p(x)$, which we can define in any way we like but must satisfy

$$\int_a^b p(x)dx = 1 \tag{16}$$

Then we can write the integral as the expectation of the function $f(x)/p(x)$ with respect to the probability distribution $p(x)$

$$I_f \equiv \left\langle \frac{f}{p} \right\rangle = \int_a^b \frac{f(x)}{p(x)} p(x)dx \tag{17}$$

which we interpret as sampling the function $f(x)/p(x)$ with x values distributed according to $p(x)$. The variance of I_f will be

$$\sigma^2 = \left\langle \frac{f^2}{p^2} \right\rangle - \left\langle \frac{f}{p} \right\rangle^2 = \int_a^b \frac{f^2(x)}{p^2(x)} p(x)dx - \left(\int_a^b \frac{f(x)}{p(x)} p(x)dx \right)^2 \tag{18}$$

In the previous example, the normalised sampling distribution was $p(x) = 1/9$. Using this in equation 18 yields 2.997 for the first term, and 0.81 for the second, for an overall variance of 2.187.

Now we consider a non-uniform sampling density $p(x) = 1/(x \ln 10)$, (normalised over the interval (1,10)), which concentrates more points nearer the lower region of the interval of integration, where $f(x)$ takes larger values.

To illustrate how we actually sample from the density function $p(x) = 1/(x \ln 10)$ in the calculation of

$$I_f = \int_1^{10} \left(\frac{\frac{1}{x^2}}{\frac{1}{x \ln 10}} \right) \frac{1}{x \ln 10} dx \tag{19}$$

consider a change of variable to $dz = 1/(x \ln 10)dx$, or $z = \ln x / \ln 10$. Compute new limits of integration,

$$\begin{aligned} x = 10 \quad &\Rightarrow \quad z = 1 \\ x = 1 \quad &\Rightarrow \quad z = 0 \end{aligned} \tag{20}$$

Then

$$I_f = \int_1^{10} \left(\frac{\frac{1}{x^2}}{\frac{1}{x \ln 10}} \right) \frac{1}{x \ln 10} dx \equiv \int_0^1 \frac{\ln 10}{\exp(z \ln 10)} dz \equiv \frac{1}{N} \sum_{i=1}^{N} \frac{\ln 10}{\exp(z_i \ln 10)} \tag{21}$$

for $z_i s$ chosen uniformly in the interval (0,1). It is clear that the subsequent value of I_f will be 0.9 as before, but what happens to the variance for this sampling scheme? The value of the expectation $\langle f^2/p^2 \rangle$ in equation 18

$$\begin{aligned} \left\langle \frac{f^2}{p^2} \right\rangle &= \int_a^b \frac{f^2(x)}{p^2(x)} p(x) dx \\ &= \int_1^{10} \left(\frac{\frac{1}{x^2}}{\frac{1}{x \ln 10}} \right)^2 \frac{1}{x \ln 10} dx \\ &= \frac{1}{N} \sum_{i=1}^{N} \frac{(\ln 10)^2}{(\exp(z_i \ln 10))^2} \end{aligned} \tag{22}$$

yields a variance of 0.33 - a significant reduction from the previous value of 2.19 obtained with uniform sampling. Repeating the calculation for a sampling function $p(x) = 1/(x^{1.5})$ gives a variance of 0.07286 - an even more significant reduction.

These results of these calculations lead to the following questions. What is the optimal sampling distribution? How small can we make the variance? The trend that has emerged from these numerical examples is that the variance is reduced as $f(x)/p(x)$ tends to a constant value. What happens if we take $p(x)$ as a normalised version of $f(x)$? Then equation 17 becomes, after a change of variable to $dz = 1/(0.9x^2)dx$

$$\begin{aligned} I_f &= \int_1^{10} \left(\frac{\frac{1}{x^2}}{\frac{1}{0.9x^2}} \right) \frac{1}{0.9x^2} dx \\ &= \int_{-10/9}^{-1/9} 0.9 dz = \frac{1}{N} \sum_{i=1}^{N} 0.9 \end{aligned} \tag{23}$$

and the exact result is obtainable using only a single iteration. Following the same procedure with equation 18 yields (as we now suspect) a variance of zero. The explanation for this result is that in order to normalise and sample from $f(x)$, we had to know how to analytically integrate it in the first place. However, we have seen that in order to reduce the variance of a Monte Carlo integration, we should choose our (easily integrable) sampling density function $p(x)$ to be as close to the normalised version of $f(x)$ as we can get.

The techniques here described are the basis for the adaptive Monte Carlo integration algorithm VEGAS, described in "Numerical Recipes".

3.3 Gaussian processes

Many simulations involve Gaussian noise processes, where the most important effects are due to contributions from the density function many standard deviations from the mean value. The probability of getting a Gaussian fluctuation greater than 3 standard deviations from the mean is around 10^{-3}, ie once in every 1000 trials. If such fluctuations are important contributors to the final outcome, then the simulations may benefit from applying techniques that artificially increase the rate at which such events occur and appropriately weight the final results, leading to reduced computational time. To quantify the benefits obtained by a particular weighting scheme, we need some measure of the "closeness" of a Monte Carlo simulation result to the actual result. If a n iteration Monte Carlo is performed, producing a single valued outcome p_1, then a further simulation (using the next sequence of independent random numbers) will produce an outcome p_2, assuming that n is not great enough that the simulation will always converge to the correct answer within the limits of the available precision.

After running N simulations, the best estimate of the outcome p is the mean of the N outcomes

$$\bar{p} = \frac{1}{N}\sum_{i=1}^{N} p_i \tag{24}$$

An estimate of the uncertainty in $\bar{p}$ is the standard deviation

$$\sigma = \sqrt{\sum_{i=1}^{N}\frac{(\bar{p}-p_i)^2}{N-1}} \tag{25}$$

If the p_is follow a Gaussian distribution, then about 68% of the computed $p_i^{'}$ s will fall within $\pm 1\sigma$ of the mean value, $\bar{p}$.

Ideally, one designs a Monte Carlo that reduces the uncertainty in the value $\bar{p}$, so that if we only run the simulation for a single run of n iterations, then the value obtained, p_i, will be sufficiently close to $\bar{p}$ for whatever purpose we desire.

Increasing the number of iterations in the simulation will decrease the value of σ. The objective of weighting techniques is to reduce the uncertainty in $\bar{p}$ for a specified value of n.

To demonstrate some weighting techniques we examine a trivial example, where the expected outcome can be exactly calculated.

Consider a random process with a Gaussian probability density function

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{(x-\bar{x})}{\sigma}\right)^2} \tag{26}$$

where $\bar{x}$ is the mean and σ is the standard deviation.

To select deviates that follow a Gaussian distribution we use the Box-Muller method (**?**), firstly select two random deviates r_1 and r_2 from the interval (0,1), and calculate the quantity

$$x_g = \sqrt{-2\ln(r_1)}\sin(2\pi r_2)\sigma + \bar{x} \tag{27}$$

The sequence of x_g s selected in this fashion are sampled from a Gaussian with mean $\bar{x}$ and standard deviation σ.

In our example, we sum two Gaussian deviates and compute the number of times the sum exceeds a given threshold. This could correspond to calculating the bit error rate for a digital communication channel corrupted by noise. Suppose that a sequence of digital data is transmitted by sending voltage pulses, either zero Volts (0 sent) or some finite positive level (1 sent). We assume that Gaussian noise adds onto these levels. To determine which symbol was sent, suppose we sample the transmitted bit twice within the bit duration, average the result and compare to a threshold. If equal numbers of 1's and 0's are sent, then the overall signal averages to 0.5. If we sample at two points in a transmitted bit, and the sum exceeds 2×0.5, then we decide that a 1 was received. If the sum does not exceed 1, then a 0 was received. Knowing which symbol was actually sent in the simulation allows us to count up how often an error is made due to the addition of the noise and thus determine the bit error rate.

The case considered here is simple enough that the bit error rate can be calculated exactly. For simplicity, assume that a 0 was sent. Then to calculate the error rate in the presence of Gaussian noise of variance σ^2, we just need to know the number of times that two Gaussian deviates with variance σ^2 add to be greater than 1. This is found from the variance of the sum (equal to the sum of the variances). For example, if the noise has a variance $\sigma^2 = 1$, and a zero is sent, then an error is made if the sum of the two samples exceeds 1.0, which, for a summed variance of 2, occurs with probability 0.2398. In a real simulation, this average will be approached for many iterations. The measure of the convergence of a simulation is the variance in the mean values obtained from repeated sets of simulations. Suppose we calculate the mean for 1000 random pairs of Gaussian deviates, resulting in a mean error rate r_1. Repeating the procedure for further sets of 1000 iterations gives $r_2, r_3, ...r_n$. The variance in the r_is (which we denote $\sigma_{\bar{r}}^2$) gives us the uncertainty in the mean. For the simulation just described, $\sigma_{\bar{r}}^2 = 1.8 \times 10^{-4}$ giving an overall answer ($\bar{r} \pm \sqrt{\sigma_{\bar{r}}^2}$) 0.240 ± 0.013.

Next suppose that the error rate is significantly reduced by increasing the underlying real signal level to 6.0. Then if a zero was sent, an error occurs if the two samples of the signal sum to be greater than 6.0 (the threshold above which we decide a 1 was sent). Calculating the mean and variance of 5000 means (each derived from 1000 iterations at a time) gives an answer of $(1.04 \pm 10.3) \times 10^{-5}$. The error in the mean is so great because the overall mean is so low - only 1 in 100 of the 5000 simulation runs actually contained an error count.

Cleary, a lot of the computational effort has been wasted. Let us bias the underlying Gaussian error density, $f(x)$, so that more errors are made to occur in the simulation. Of course, we then will have to unbias the final error rate. Suppose that instead of sampling the two values to be summed (x_1 and x_2) from the distribution $f(x)$, we sample them from another distribution $f^*(x)$. Each x_i is then given a weight $w_i = f(x_i)/f^*(x_i)$, reflecting the actual likelihood of that value occurring (if x_i is 10 times more likely to occur in $f^*(x)$ than in $f(x)$, then the weight assigned is 1/10 th). The values x_1 and x_2 are summed, and compared to the threshold. The weight assigned to the sum, $x_1 + x_2$, is the product of the weights, $w(x_1)w(x_2)$. If the sum exceeds the threshold, then the error count is incremented, not by 1, but by $1 \times w(x_1)w(x_2)$. For appropriate choices of $f^*(x)$, the variance of the final error rate will be reduced. We first consider changing the variance of the underlying Gaussian error distribution to induce more errors. Suppose we use a variance of 3.0, instead of 1.0. Then the overall error rate comes out to $(1.11 \pm 0.35) \times 10^{-5}$, which is a significant improvement on the original result using the unbiased error density with $\sigma^2 = 1.0$. However, increasing the variance also increases the occurance of x_is in the negative tails - selections which are still wasted. Intuitively we would expect that most of the x_is that contribute to sums exceeding the threshold would cluster about $x = 3.0$, so instead of changing the variance of the underlying probability distribution, we change the mean from $\bar{x} = 0.0$ to $\bar{x} = 3.0$, to emphasise selections in the important region. This gives an overall simulation result of $(1.103 \pm 0.076) \times 10^{-5}$). The exact error rate comes from the area under a Gaussian of variance $\sigma^2 = 2$, for $x > 6$, which is 1.105×10^{-5}.

We have demonstrated the savings from importance sampling in terms of the variance reduction for the two different schemes, ie variance changing and translation of the mean of the probability distributions. Another approach is define an accuracy level required, then calculate the number of iterations needed to reach that level as a function of the various biasing schemes. This gives a more direct estimate of the reduction in computational burden.

A single simulation run of N iterations produces an error rate, r, and a variance, σ_r^2. As a measure of convergence, we take N to be the number of iterations to produce an error of 10%, ie the standard error of the mean is less than 10% of r:

$$\frac{\sigma_r}{\sqrt{N}} < 0.1r \tag{28}$$

This relation is tested over many independent simulation runs to obtain an average for N. Table 1 shows the results for the double sampling case just considered.

Mean translation method		Variance increase method	
Biased density parameters	N_{mt}	Biased density parameters	N_{vi}
$\sigma^2 = 1.0 \ \ \bar{x} = 0.0$	9×10^6	$\bar{x} = 0.0 \ \ \sigma^2 = 1.0$	9×10^6
$\sigma^2 = 1.0 \ \ \bar{x} = 0.5$	450000	$\bar{x} = 0.0 \ \ \sigma^2 = 2.25$	70600
$\sigma^2 = 1.0 \ \ \bar{x} = 1.0$	46000	$\bar{x} = 0.0 \ \ \sigma^2 = 4.0$	19000
$\sigma^2 = 1.0 \ \ \bar{x} = 2.0$	1770	$\bar{x} = 0.0 \ \ \sigma^2 = 6.25$	12000
$\sigma^2 = 1.0 \ \ \bar{x} = 3.0$	477	$\bar{x} = 0.0 \ \ \sigma^2 = 9.0$	10600
$\sigma^2 = 1.0 \ \ \bar{x} = 4.0$	1230	$\bar{x} = 0.0 \ \ \sigma^2 = 12.25$	11000
$\sigma^2 = 1.0 \ \ \bar{x} = 5.0$	23700	$\bar{x} = 0.0 \ \ \sigma^2 = 16.0$	12000

Table 1: Mean number of iterations required to achieve 10% error for simulation using 2 samples per received symbol.

Although both importance sampling schemes produce orders of magnitude improvement over the non-importance sampling scheme, the optimal mean translation case is much more effective than the optimal variance increase case. In order to compare the two sampling schemes, we look at the ratio of iterations required (N_{mt}/N_{vi}) to obtain 10% accuracy for the optimal choice of parameters for each scheme.

If we consider the one-sample case, (table 2), we see that the ratio N_{mt}/N_{vi} of iterations required to reach 10% accuracy for both biasing schemes is 0.15 for the optimal choice of biasing parameters. For the double-sampling case, the ratio $N_{mt}/N_{vi} = 0.045$. For the case where 5 samples per symbol are used, (table 3), $N_{mt}/N_{vi} = 0.001$. It is clear that as the sample rate (system dimensionality) increases, the mean translation technique becomes far superior to the variance increase technique. This is because all the biasing of mean translation is in a 'good' direction (more likely to produce the desired error counts), whereas much of the biasing in the variance increase scheme pushes probability density in directions less likely to produce the desired error counts, an effect which gets worse as the dimensionality of the system increases.

Mean translation method		Variance increase method	
Biased density parameters	N_{mt}	Biased density parameters	N_{vi}
$\sigma^2 = 1.0 \;\; \bar{x} = 0.0$	74000	$\bar{x} = 0.0 \;\; \sigma^2 = 1.0$	74000
$\sigma^2 = 1.0 \;\; \bar{x} = 1.0$	4600	$\bar{x} = 0.0 \;\; \sigma^2 = 4.0$	2900
$\sigma^2 = 1.0 \;\; \bar{x} = 2.0$	770	$\bar{x} = 0.0 \;\; \sigma^2 = 6.25$	2200
$\sigma^2 = 1.0 \;\; \bar{x} = 3.0$	335	$\bar{x} = 0.0 \;\; \sigma^2 = 9.0$	2250
$\sigma^2 = 1.0 \;\; \bar{x} = 4.0$	520	$\bar{x} = 0.0 \;\; \sigma^2 = 12.25$	2900

Table 2: Mean number of iterations required to achieve 10% error for simulation using a single sample per received symbol.

Mean translation method		Variance increase method	
Biased density parameters	N_{mt}	Biased density parameters	N_{vi}
$\sigma^2 = 1.0 \;\; \bar{x} = 1.5$	335000	$\bar{x} = 0.0 \;\; \sigma^2 = 4.0$	2.7×10^6
$\sigma^2 = 1.0 \;\; \bar{x} = 2.0$	12400	$\bar{x} = 0.0 \;\; \sigma^2 = 9.0$	0.8×10^6
$\sigma^2 = 1.0 \;\; \bar{x} = 2.5$	1670	$\bar{x} = 0.0 \;\; \sigma^2 = 16.0$	1.0×10^6
$\sigma^2 = 1.0 \;\; \bar{x} = 3.0$	770	$\bar{x} = 0.0 \;\; \sigma^2 = 25.0$	1.8×10^6
$\sigma^2 = 1.0 \;\; \bar{x} = 3.5$	1400	$\bar{x} = 0.0 \;\; \sigma^2 = 36.0$	2.8×10^6

Table 3: Mean number of iterations required to achieve 10% error for simulation using 5 samples per received symbol.

References

(n.d.).

Al-Qaq, W. & Devetsikiotis, M. (1993), 'Importance Sampling Methodologies for Simulation of Communication Systems with Time-Varying Channels and Adaptive Equalizers', *IEEE Journal on Selected Areas in Communications* **11**(3), 317.

Al-Qaq, W., Devetsikiotis, M. & Townsend, J. (1996), 'Stochastic Gradient Optimization of Importance Sampling for the Efficient Simulation of Digital Communication Systems', *IEEE Transactions on Communications* **43**(12), 2975.

Chen, J.-C. & Sadowsky, J. (1993), 'On Importance Sampling in Digital Communications-Part II: Trellis-Coded Modulation', *IEEE Journal on Selected Areas in Communications* **11**(3), 300.

Chen, J.-C., Lu, D., Sadowsky, J. & Yao, K. (1993), 'On Importance Sampling in Digital Communications-Part I: Fundamentals', *IEEE Journal on Selected Areas in Communications* **11**(3), 289.

Davis, B. (1986), 'An Improved Importance Sampling Method for Digital Communication System Simulations', *IEEE Transactions on Communications* **COM-34**(7), 715.

Hahn, P. & Jeruchim, M. (1987), 'Developments in the Theory and Application of Importance Sampling', *IEEE Transactions on Communications* **COM-35**(7), 706.

Jeruchim, M. (1984*a*), 'On the Application of Importance Sampling to the Simulation of Digital Satellite and Multihop Links', *IEEE Transactions on Communications* **COM-32**(10), 1088.

Jeruchim, M. (1984*b*), 'Techniques for Estimating the Bit Error Rate in the Simulation of Digital Communication Systems', *IEEE Journal on Selected Areas in Communications* **SAC-2**(1), 153.

Letaief, K. & Muhammad, K. (1995), 'An Efficient New Technique for Accurate Bit Error Probability Estimation of ZJ Decoders', *IEEE Transactions on Communications* **43**(6), 2020.

Lu, D. & Yao, K. (1988), 'Improved Importance Sampling Technique for Efficient Simulation of Digital Communication Systems', *IEEE Journal on Selected Areas in Communications* **6**(1), 67.

Shanmugam, K. & Balaban, P. (1980), 'A Modified Monte-Carlo Simulation Technique for the Evaluation of Error Rate in Digital Communication Systems', *IEEE Transactions on Communications* **COM-28**(11), 1916.

Stadler, J. & Roy, S. (1993), 'Adaptive Importance Sampling', *IEEE Journal on Selected Areas in Communications* **11**(3), 309.

Stadler, J. & Roy, S. (1995), 'Importance Sampling for Detection of Scale Problems', *IEEE Transactions on Communications* **43**(12), 2862.