8 STOCHASTIC SIMULATION

Whereas in optimization we seek a set of parameters $\vec{x}$ to minimize a cost, or to maximize a reward function $J(\vec{x})$, here we pose a related but different question. Given a system S, it is desired to understand how *variations in the defining parameters $\vec{x}$ lead to variations in the system output*. We will focus on the case where $\vec{x}$ is a set of random variables, that can be considered unchanging - they are static. In the context of robotic systems, these unknown parameters could be masses, stiffness, or geometric attributes. How does the system behavior depend on variations in these physical parameters? Such a calculation is immensely useful because real systems have to be robust against modeling errors.

At the core of this question, the random parameters x_i in our discussion are described by distributions; for example each could have a pdf $p(x_i)$. If the variable is known to be normal or uniformly distributed, then of course it suffices to specify the mean and variance, but in the general case, more information may be needed.

8.1 Monte Carlo Simulation

Suppose that we make N simulations, each time drawing the needed random parameters x_i from a random number "black box" (about which we will give more details in the next section). We define the high-level output of our system S to be $g(\vec{x})$. For simplicity, we will say that $g(\vec{x})$ is a scalar. $g(\vec{x})$ can be virtually any output of interest, for example: the value of one state at a given time after an impulsive input, or the integral over time of the trajectory of one of the outputs, with a given input. In what follows, will drop the vector notation on x for clarity.

Let the *estimator* G of $g(x)$ be defined as

$$G = \frac{1}{N} \sum_{j=1}^{N} g(x_j).$$

You recognize this as a straight average. Indeed, taking the expectation on both sides,

$$E(G) = \frac{1}{N} \sum_{j=1}^{N} E(g(x_j)),$$

it is clear that $E(G) = E(g)$. At the same time, however, we do not know $E(g)$; we calculate G understanding that with a very large number of trials, G should approach $E(g)$. Now let's look at the variance of the estimator. This conceptually results from an infinite number of estimator trials, each one of which involves N evaluations of g according to the above definition. It is important to keep in mind that such a variance involves samples of the estimator (each involving N evaluations) - not the underlying function $g(x)$. We have

$$\begin{aligned}
\sigma^2(G) &= \sigma^2\left[\frac{1}{N}\sum_{j=1}^{N} g(x_j)\right] \\
&= \frac{1}{N^2}\sigma^2\left[\sum_{j=1}^{N} g(x_j)\right] \\
&= \frac{1}{N^2}\sum_{j=1}^{N}\sigma^2(g) \\
&= \frac{1}{N}\sigma^2(g).
\end{aligned}$$

This relation is key. The first equality follows from the fact that $\sigma^2(nx) = n^2\sigma^2(x)$, if n is a constant. The second equality is true because $\sigma^2(x+y) = \sigma^2(x) + \sigma^2(y)$, where x and y are random variables. The major result is that $\sigma^2(G) = \sigma^2(g)$ if only one-sample trials are considered, but that $\sigma^2(G) \to 0$ as $N \to \infty$. Hence with a large enough N, we can indeed expect that our G will be very close to $E(g)$.

Let us take this a bit further, to get an explicit estimate for the error in G as we go to large N. Define a nondimensional estimator error

$$\begin{aligned}
q &= \frac{G - E(g)}{\sigma(G)} \\
&= \frac{(G - E(g))\sqrt{N}}{\sigma(g)},
\end{aligned}$$

where the second equality comes from the result above. We call the factor $\sigma(g)/\sqrt{N}$ the standard error. Invoking the central limit theorem, which guarantees that the distribution of G becomes Gaussian for large enough N, we have

$$\begin{aligned}
\lim_{N\to\infty} \text{prob}(a < q < b) &= \int_a^b \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt \\
&= F(a) - F(b),
\end{aligned}$$

where $F(x)$ is the cumulative probability function of the standard Gaussian variable:

$$F(a) = \int_{-\infty}^{a} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt$$

Looking up some values for $F(x)$, we see that the nondimensional error is less than one in 68.3% of trials; it is less than two in 95.4% of trials, and less than three in 99.7% of trials. The 99.7% confidence interval corresponds with

$$\begin{aligned}
-3 \leq \quad & (G - E(g))\sqrt{N}/\sigma(g) && \leq 3 \to \\
-3\sigma(g)/\sqrt{N} \leq \quad & G - E(g) && \leq 3\sigma(g)/\sqrt{N}.
\end{aligned}$$

In general, quadrupling the number of trials improves the error by a factor of two.

So far we have been describing a single estimator G, which recovers the mean. The mean, however, is in fact an integral over the random domain:

$$E(g) = \int_{x \epsilon X} p(x)g(x)dx,$$

where $p(x)$ is the pdf of random variable x. So the Monte Carlo estimator G is in fact an integrator:

$$G \simeq \int_{x \epsilon X} p(x)g(x)dx.$$

We can just as easily define estimators of statistical moments:

$$G_n = \frac{1}{N} \sum_{j=1}^{N} x_j^n g(x_j) \simeq \int_{x \exists X} x^n p(x)g(x)dx,$$

which will follow the same basic convergence trends of the mean estimator G. These moments can be calculated all using the same N evaluations of $g(x)$.

The above equation gives another point of view to understand how the Monte Carlo approach works: the effect of the probability density function in the integral is replaced by the fact that random variables in MC are drawn from the same distribution. In other words, a high $p(x)$ in a given area of the domain X amplifies $g(x)$ there. MC does the same thing, because there are in fact more x drawn from this area, in making the N evaluations.

8.2 Making Random Numbers

The Monte Carlo method requires that we fire into our evaluation $g(x)$ a group of N random numbers (or sets of random numbers), drawn from a distribution (or a set of distributions for more than one element in x). Here we describe how to generate such data from simple distributions.

Note that both the normal and the uniform distributions are captured in standard MATLAB commands.

We describe in particular how to generate samples of a given distribution, from random numbers taken from an underlying *uniform* distribution. First, we say that the cumulative probability function of the uniform distribution is

$$P(w) = \begin{cases} 0, & w \leq 0 \\ w, & 0 < w < 1 \\ 1, & w \geq 1 \end{cases}$$

If $x = r(w)$ where r is the transformation we seek, recall that the cumulative probabilities are

$$P(x) = P(r(w)) = P(w) = w,$$

and the result we need is that

$$w = P(x) = \int_{-\infty}^{x} p(x)dx.$$

Our task is to come up with an x that goes with the uniformly distributed w - it is not as hard as it would seem. As an example, suppose we want to generate a normal variable x (zero mean, unity variance). We have

$$\begin{aligned} P(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt &= w \\ F(x) &= w, \text{ or} \\ x &= F^{-1}(w), \end{aligned}$$

where $F(x)$ is the cumulative probability function of a standard Gaussian variable (zero mean, unity variance), and can be looked up or calculated with standard routines. Note $F(x)$ is related within a scale factor to the error function (erf).

As another example, consider the exponential distribution

$$p(x) = \lambda e^{-\lambda x};$$

this distribution is often used to describe the time of failure in complex systems. We have

$$\begin{aligned} P(x) = \int_{0}^{x} \lambda e^{-\lambda t} dt &= w \\ 1 - e^{-\lambda x} &= w \text{ or} \\ x &= -\frac{\log(1-w)}{\lambda}. \end{aligned}$$

Similarly, this procedure applied to the Rayleigh distribution

$$p(x) = xe^{-x^2/2}$$

gives $x = \sqrt{-2\log(1-w)}$. In these formulas, we can replace $(w-1)$ with w throughout; w is uniformly distributed on the interval [0,1], so they are equivalent.

8.3 Grid-Based Techniques

As noted above, moment calculations on the output variable are essentially an integral over the domain of the random variables. Given this fact, an obvious approach is simply to focus on a high-quality integration routine that uses some *fixed* points x - in the Monte Carlo method, these were chosen at random. In one dimension, we have the standard trapezoid rule:

$$\begin{aligned}
\int_a^b g(x)dx &\approx \sum_{i=1}^{n} w_i g(x_i), \text{ with} \\
w_1 &= (b-a)/2(n-1) \\
w_n &= (b-a)/2(n-1) \\
w_2, \cdots, w_{n-1} &= (b-a)/(n-1) \\
x_i &= a + (i-1)(b-a)/(n-1).
\end{aligned}$$

Here the w_i are simply weights, and g is to be evaluated at the different abscissas x_i. This rule has error of order $1/n^2$, meaning that a doubling in the number of points n gives a fourfold improvement in the error. To make an integration in two dimensions we take the tensor product of two single-dimension calculations:

$$\int_a^b \int_c^d g(x)dx \approx \sum_{i=0}^{n_1} \sum_{j=0}^{n_2} w_i w_j g(x_{ij}).$$

Here the abscissas have two elements, taken according to the grids in the two directions. In many applications, the trapezoid rule in multi-dimensions works quite well and can give a good understanding of the various moments of the given system. Extensions such as Romberg integration, with the tensor product, can be applied to improve results.

A particularly powerful technique involving orthogonal polynomials is also available and it gives truly remarkable accuracy for smooth functions. For our development here, we will focus on normal distributions of random variables. These pertain to a particular set of orthogonal polynomials known as the Hermite (pronounced "hermit") polynomials. These are:

$$\begin{aligned}
h_0(x) &= 1 \\
h_1(x) &= x \\
h_2(x) &= x^2 - 1 \\
&\cdots \\
h_{n+1}(x) &= x h_n(x) - n h_{n-1}(x) \text{ (recurrence relation).}
\end{aligned}$$

The defining feature of this set of polynomials is that

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} h_i(x) h_j(x) dx = \begin{cases} 0, & \text{if and only if } i \neq j \\ 1, & \text{if and only if } i = j. \end{cases}$$

Note that we have chosen a scaling so that the inner product comes out to be exactly one - some textbook definitions of the Hermite polynomials will not match this. Note also that the inner product is taken with respect to the Gaussian exponential weighting function, which we equate below with the Gaussian pdf. Now the magic comes from the following[2]: a $(2n-1)$'th order polynomial $g(x)$ can be written as

$$g(x) = h_n(x)[a_{n-1}h_{n-1}(x) + a_{n-2}h_{n-2}(x) + \cdots + a_0h_0(x)] + b_{n-1}h_{n-1}(x) + \cdots + b_0h_0(x).$$

This formula, with the $2n$ coefficients a and b, covers the $(2n-1)$'th-order term with $a_{n-1}h_n(x)h_{n-1}(x)$, and the zero'th order term with $b_0h_0(x)$. It can be shown that all the products are linearly independent, so that indeed *all* polynomials up to order $2n-1$ are accounted for.

Returning to the integration problem, for the determination of moments, recall the definition that

$$E(g(x)) = \int_{x \epsilon X} p(x)g(x)dx,$$

where $p(x)$ is the probability density function of random variable x. Employing specifically the Gaussian pdf in $p(x)$ and the Hermite polynomials so as to achieve orthogonality, we can integrate our $g(x)$ as follows:

$$\int_{-\infty}^{\infty} p(x)g(x)dx = b_0\int_{-\infty}^{\infty} p(x)h_0(x)dx = b_0.$$

Thus, we have only to find b_0! To do this, we cleverly select the abscissas to be the zeros (or roots) of $h_n(x)$ - let's call them $x_1, \cdots, x_n$. We obtain a linear system:

$$\begin{aligned} \left\{ \begin{array}{c} g(x_1) \\ \vdots \\ g(x_N) \end{array} \right\} &= \left[\begin{array}{ccc} h_{n-1}(x_1) & \cdots & h_0(x_1) \\ \vdots & & \vdots \\ h_{n-1}(x_n) & \cdots & h_0(x_n) \end{array} \right] \left\{ \begin{array}{c} b_{n-1} \\ \vdots \\ b_0 \end{array} \right\}, \text{ or} \\ \vec{g} &= H\vec{b} \end{aligned}$$

Notice that the a coefficients do not appear here because the x_i are taken at the roots of $h_{n-1}(x)$. The linear system can be solved easily as $\vec{b} = H^{-1}\vec{g}$, and we have a special interest in the last row, which of course has the solution for b_0. Indeed, the bottom row of H^{-1} is the set of weights w_i, in complete analogy with the weights we defined above for the trapezoid rule:

$$b_0 = \int_{-\infty}^{\infty} p(x)g(x)dx = \sum_{i=1}^{n} w_i g(x_i).$$

[2]Parts of this derivation follow J.R. Hockenberry and B.C. Lesieutre (2004), IEEE Transactions on Power Systems, 19:1483-1491.

8.4 Issues of Cost and Accuracy

The balance of cost versus error for the methods above are summarized as:

- The Monte-Carlo estimator has a variance that decreases with $1/N$, where N is the number of evaluations. There is no dependence on the random dimension d or on the form of the function.
- The trapezoidal rule in one dimension has an error that decreases with n^{-2}.
- Gauss-Hermite quadrature in one dimension matches exactly all polynomials $g(x)$ of order less than or equal to $2n-1$, and makes the best fit possible when it is of higher order. The error is zero if $g(x)$ is a polynomial of order $2n-1$ or less, but if not then the error goes as n^{-r}, where r is the "smoothness" of the function. The smoothness is the same as the number of derivatives that can be taken everywhere in the domain.

Some simple but important scaling laws show that the Monte Carlo will ultimately outperform any quadrature rule, in high enough dimensions. Recall that the MC estimator G has error - or standard deviation of error - that scales as

$$\epsilon = O(1/\sqrt{N})$$

Now, in contrast, the error of a quadrature rule in one dimension scales as:

$$\epsilon = O(n^{-k}),$$

where k is a parameter of the method used and the function itself, two for the trapezoid rule we described above, and the smoothness r for Gaussian quadrature. Considering the multi-dimensional quadrature case, the error stays the same but now the total number of evaluations N includes *all* the grid points, i.e., $N = n^d$ (assuming $N_1 = N_2 = \cdots = N$). Hence, here we have

$$\epsilon = O((N^{1/d})^{-k}) = O(N^{-k/d})$$

Error in the quadrature rules is affected dramatically by d: Even in two dimensions, the n^{-2} convergence of the trapezoid rule is degraded to n^{-1}! In four dimensions, the error rate of $1/\sqrt{N}$ is the same as for Monte Carlo. The Gaussian quadrature may do better for some functions, but there are plenty of them for which $r = 2$, for example the innocuous $g(x) = x^2|x|$!

Another major factor to consider is that the quadrature rules are inherently trying to make a polynomial approximation of the function, whereas the Monte Carlo technique has no such intention. Discontinuous and otherwise non-smooth functions in particular can cause serious problems for the quadratures, which must have many points in order to cover the sharp spots accurately.

Summarizing, we recommend that you keep the two major classes of integration tools handy - grid-less and grid-based. For lower dimensions and smoother functions, Gaussian quadratures can provide exceptional results, whereas the Monte-Carlo workhorse always comes out on top for high-dimension, difficult functions. The differences are trivial for very cheap evaluations of g, but become very compelling when each evaluation takes a lot of computer time, or involves an actual experiment.

There are substantial extensions to multi-dimensional quadrature rules and to Monte Carlo methods, some of which are available in the following references:

M.H. Kalos and P.A. Whitlock, 1986, *Monte Carlo methods, volume 1: basics*, New York: Wiley.

W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, 1992, *Numerical recipes in C*, Cambridge, UK: Cambridge University Press.