Monte Carlo Integration

Ronald Kleiss[1]
HIMAPP
Radboud University, Nijmegen, the Netherlands

Preliminary version, February 2, 2010

[1] R.Kleiss@science.ru.nl

Contents

1 Introduction

Many problems in physical science involve integration in some form or another: sometimes this is directly evident, as in the computation of scattering cross sections in particle physics; alternatively, it can often be shown that the answer to a particular question is equivalent to some integral. Depending on cleverness, technique, and perseverance these integrals can sometimes be performed analytically, but only up to a point: almost invariably one ends up encountering integrals that can only be evaluated numerically *i.e.*, by computer[1]. The general field of numerical integration (or *quadrature*) has a long and distinguished history, and so does its Monte Carlo branch.

We may state: **any numerical procedure in which the outcome depends on at least one random variable is a Monte Carlo integral**. For, suppose that the numerical answer to a given problem can be written as $R(r_1, r_2, \ldots, r_n)$, where R is some (possibly extremely complicated) function involving random variables r_1,r_2, ...,r_n, each drawn from its own domain A_1,A_2,...,A_n, with a combined probability density $P(r_1, r_2, \ldots, r_n)$. Then, the *expected* answer is given by

$$\int_{A_1}\int_{A_2}\cdots\int_{A_n} R(x_1, x_2, \ldots, x_n)\; P(x_1, x_2, \ldots, x_n)\; dx_1\; dx_2 \cdots dx_n$$

A whole computer simulation is therefore conceptually equivalent to a multivariable integral. Of course, sometimes the number n of random variables involved is staggeringly huge[2], and then the picture of a single simulation run as the picking of one single point in a multissimidimensional space is not the most fertile one: but it must be kept in mind that in what follows the notion 'integral' encompasses essentially *all* numerical problems.

Notwithstanding the indeteminacy implied in the use of random numbers, Monte Carlo can be shown to be useful, since it has a solid base in the theory of probability and statistics, and that is the subject of section 2. In order to make Monte Carlo at all feasible we need a source of random numbers: ways to obtain these by computer algorithms are discussed in section 3, where we also deal with the problem of ascertaining how well the computer-generated *pseudo*-random numbers mimic the real stuff. Sections 4 and 5 refine these ideas into the notion of *discrepancy*, which naturally leads to the consideration of *quasi*-random numbers and Quasi-Monte Carlo. The important idea of *variance reduction*, and the general strategies of attaining it, are reviewed in the first half of section 6. This involves to the necessity of our being able to generate (pseudo)random numbers with a prescribed, *non*-uniform distribution, which fills the rest of that section. A very important application of the Monte Carlo

[1] Historically, Monte Carlo predates the advent of electronic computers, but not that of human ones. For examples, see [1])

[2] It can easily run into billions

method to particle phenomenology is the generation of points in *multiparticle phase space*: several algorithms to this end are exhibited in section 7. Some assorted mathematical details are collected in the Appendices.

2 Probability theory and Monte Carlo estimators

2.1 Random numbers and probability densities

The notion of random numbers plays a central rôle in probability theory and Monte Carlo integration. Strictly speaking, a single number is never random, but rather we can only discuss (potentially) infinite sequences of random numbers. In such a *stream*, we must be either be promised, or we must assume, that the relative frequency with which numbers fall into given intervals takes on definite values. Although the issue is much more subtle than might be expected[3], the basic property of randomness is that, no matter how much information we have about the number stream so far, we shall never be able to predict the outcome for the *next* number to better than that given by the relative frequency, the *probability.*[1].

For a random variable r we say that its probability density is $P(r)$ if the probability to find the variable in the interval $[x, x+dx]$ is $P(x)dx$ for infinitesimal dx. In these notes, we assume the variables to take values on the whole real axis. If the actual range of r is bounded, we impose this by simply putting $P(x) = 0$ if x is outside the range. Discrete random variables that take values in some enumerable set N of real numbers have a probability density consisting of Dirac delta functions[2]: if p_i is the probability of a discrete variable to take the precise value r_i, we have

$$P(x) = \sum_i p_i \, \delta(x - r_i) \quad . \tag{1}$$

For probability densities we have

$$P(x) \geq 0 \quad , \quad \int dx \, P(x) = 1 \quad . \tag{2}$$

We may also have combined probability densities $P(x_1, x_2, \ldots, x_k)$ of k real variables, by trivial extension. If this combined probability density factorizes, *i.e.* there are probability densities $P_j(x_j)$ such that

$$P(x_1, x_2, \ldots, x_k) = P_1(x_1)P_2(x_2)\cdots P_k(x_k) \quad , \tag{3}$$

the random variables $x_1, x_2, \ldots, x_k$ are independent: knowledge of the value of one does not improve our knowledge of the value of another variable. If $P_i(x) = P_j(x)$ for all i, j in $1, 2, \ldots, k$ the random variables are *independent, identically distributed*, or iid. A perfect source of random numbers is supposed

[1] This is the *frequentist* interpretation of probability. Beloved of physicists but not necessarily by mathematicians, nevertheless it is this interpretation that fits seamlessly on the use of very large sets of(pseudo)random numbers in Monte Carlo integration.

[2] This might lead to problems if we ever have to consider powers such as $P(x)^2$: fortunately we never have to.

to deliver iid variables.

A basic probability density is the *basic density*:

$$B(x) = \theta(0 \leq x \leq 1) \ , \tag{4}$$

where '$\leq$' might as well read '$<$' since the probability of having $x = 0$ or $x = 1$ precisely is vanishingly small[3]. Here and in the following, the *logical step function* $\theta(P)$ of a predicate P is one if the predicate is true, otherwise zero.

2.2 Expectation values

Let x be a random variable with density $P(x)$. The expectation value of a real-valued function $f(x)$ of x is defined as

$$\langle f(x) \rangle = \int dx \ P(x) \ f(x) \ . \tag{5}$$

In the usual, frequentist, interpretation of probability it corresponds to the average value of $f(x)$, sampled over very many values of x. The value $\langle f(x) \rangle$ is also called the *mean* of $f(x)$; another important quantity is the *variance* of $f(x)$:

$$\sigma \left(f(x) \right)^2 = \left\langle f(x)^2 \right\rangle - \langle f(x) \rangle^2 \ . \tag{6}$$

This nonnegative number indicates how the values of $f(x)$ are spread around the mean. Notice that $\sigma \left(f(x) + c \right)^2 = \sigma \left(f(x) \right)^2$ if c is a constant number. The *standard deviation* $\sigma(f(x))$ is physically more attractive than the variance since it has the same dimension as $f(x)$ itself.

The *moments* of the density $P(x)$ are the expectation values of powers of x: the n^{th} moment is defined by

$$M_n \equiv \langle x^n \rangle = \int dx \ P(x) \ x^n \quad (n = 0, 1, 2, 3, \ldots) \ . \tag{7}$$

The first moment, $\langle x \rangle$, is also sometimes called the 'mean of the density $P(x)$', although strictly speaking the mean of $P(x)$ is $\int dx \ P(x)^2$. Similarly, the 'variance of the density $P(x)$' is $\sigma \left(x \right)^2$. The moment M_0 is always equal to 1; the higher moments may or may not exist, depending on the density, although one usually discusses densities for which all moments exist. For such densities, one defines the *characteristic function* as

$$\phi(z) = \sum_{n \geq 0} \frac{1}{n!} (iz)^n M_n = \int dx \ P(x) \ e^{izx} \ . \tag{8}$$

Note that $\phi(0) = 1$, $\phi'(0) = iM_1$, and $\phi''(0) = -M_2$. Expectation values are always defined with respect to a given density $P(x)$. It is usually clear from the context which density is meant; if not, it must be explicitly specified.

[3] Any consideration or algorithm sensitive to the difference is dangerous and to be avoided.

2.3 Chebyshev-Bienaymé theorem

Consider a density $P(x)$ with finite mean m and variance σ^2. Let a be a positive number. We may then write the following series of inequalities:

$$\begin{aligned}\sigma^2 &= \int dx\ P(x)\ (x-m)^2 \ \geq \ \int dx\ P(x)\ (x-m)^2\ \theta(|x-m|\geq a) \\ &\geq \ a^2 \int dx\ P(x)\ \theta(|x-m|\geq a) \ = \ a^2 \mathrm{Prob}(|x-m|\geq a)\ ,\end{aligned} \tag{9}$$

which establishes the Chebyshev-Bienaymé inequality:

$$\mathrm{Prob}\left(|x-m|\leq a\right) \geq 1 - \frac{\sigma^2}{a^2}\ . \tag{10}$$

For a random variable with finite mean and variance, the probability that it is found further than k times its standard deviation away from its mean is less than $1/k^2$.

2.4 Central limit theorem

Let $x_1, x_2, \ldots, x_N$ be N iid random variables with distribution $P(x)$, for which both the mean m and the variance σ^2 are finite. Define their average value as

$$\xi = \frac{1}{N}\sum_{j=1}^{N} x_j\ . \tag{11}$$

Clearly this is also a random variable, and its density, $\Pi(\xi)$, is of course given by

$$\Pi(\xi) = \int dx_1 \cdots dx_N\ P(x_1)\cdots P(x_N)\ \delta\left(\xi - \frac{1}{N}(x_1+\cdots+x_N)\right)\ . \tag{12}$$

The characteristic function, $\Phi(z)$, is given by

$$\Phi(z) = \int d\xi\ \Pi(\xi)\ e^{iz\xi} = \left(\int dx\ P(x)\ e^{izx/N}\right)^N = \phi\left(\frac{z}{N}\right)^N\ , \tag{13}$$

where $\phi(z)$ is the characteristic function of any of the x_j. Now, if N is large, we may approximate as follows:

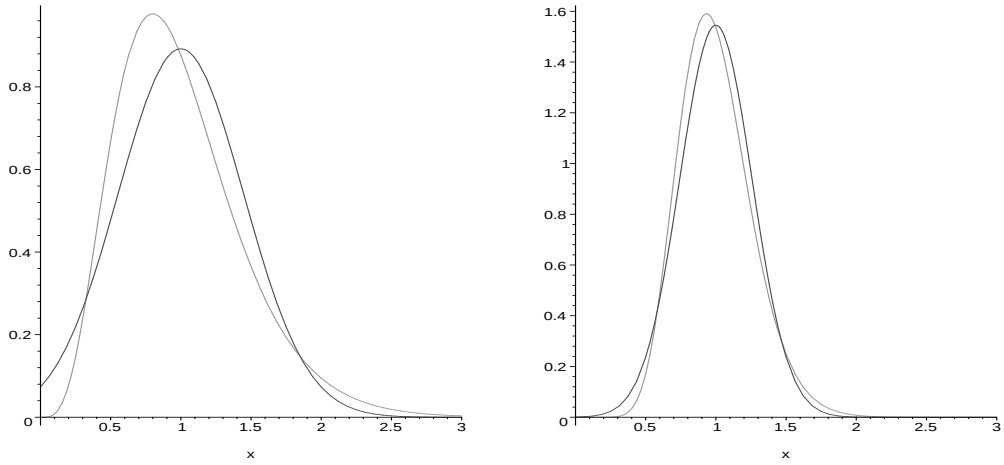
$$\begin{aligned}N\log\phi\left(\frac{z}{N}\right) &\approx N\log\left(1+\frac{iz}{N}M_1 - \frac{z^2}{2N^2}M_2+\cdots\right) \\ &\approx N\left(\frac{iz}{N}M_1 + \frac{z^2}{2N^2}M_1^2 - \frac{z^2}{2N^2}M_2+\cdots\right) \\ &\approx izm - \frac{z^2}{2N}\sigma^2+\cdots\ ,\end{aligned} \tag{14}$$

where the ellipsis denotes terms that are of higher order in $1/N$. In this approximation, the density $\Pi(\xi)$ is given by

$$\begin{aligned}\Pi(\xi) &\approx \frac{1}{2\pi}\int dz\ \exp\left(-iz\xi + izm - \frac{z^2\sigma^2}{2N}\right) \\ &= \sqrt{\frac{N}{2\pi\sigma^2}}\exp\left(-\frac{N}{2\sigma^2}(\xi-m)^2\right)\ . \end{aligned} \tag{15}$$

This is (a version of) the *central limit theorem*: under rather loose conditions, the average of a large number N of iid random variables with mean m and variance σ^2 is distributed as a Gaussian random variable with mean m and variance σ^2/N. The distribution of the average becomes increasingly sharply peaked as N increases.

The approach to the Gaussian limit is often quite rapid. As an example, if the density of the N iid random variables is the exponential density $P(x) = \exp(-x)\theta(x > 0)$, the density of ξ is $\Pi(\xi) = N^N\xi^{N-1}\exp(-N\xi)/\Gamma(N)$. Below we give $\Pi(\xi)$ and its central-limit approximation for $N = 5$ (left) and $N = 15$ (right), respectively. For densities with lower tails the approach is even faster.



Counterexamples to the central limit theorem are easily found. For instance, if the density of the iid random variables is the Cauchy density $P(x) = \pi^{-1}(1 + x^2)^{-2}$, then $\Pi(\xi) = P(\xi)$ for any N. Note, however, that in this case neither m nor σ^2 are well-defined.

2.5 Confidence levels

In Monte Carlo applications we shall have to estimate the reliability of our answers. This can be formulated in terms of confidence levels, that is, the probability that our numerically obtained estimate is within some distance from the actual, unknown, answer to the problem. Since the Monte Carlo result is a

random variable, we must estimate the variance of its distribution (see below). Having obtained σ^2, we may then use either the Gaussian, or the looser but more robust Chebyshev estimates to determine the probability that answer obtained, a, is within a given number k times the standard deviation σ from the actual answer A. Below we tabulate these confidence levels.

	Probability that $\lvert a - A\rvert \leq k\sigma$	
k	Chebyshev	Gaussian
0.5	≥ 0	0.384
1.0	≥ 0	0.684
1.5	≥ 0.556	0.866
2.0	≥ 0.750	0.955
2.5	≥ 0.840	0.988
3.0	≥ 0.889	0.997

By its nature, the Chebyshev result does not make any statement about deviations of less than one standard deviation. Note, however, that the Chebyshev result gives only a *lower limit* since it does not assume an explicit probability density. In most applications, the Gaussian confidence levels can be used with, yes, confidence.

2.6 Integral estimator: bias and convergence

The archetypical problem in Monte Carlo integration is the computation of the integral of a real-valued function $f(x)$ over the unit interval $[0, 1]$. To this end, let us denote

$$J_m = \int_0^1 dx\ f(x)^m \quad , \quad m = 1, 2, 3, \ldots \quad . \tag{16}$$

The desired integral is J_1. We shall estimate this integral using a stream of (at least, hopefully) iid random numbers x_j, $(j = 1, 2, 3, 4, \ldots)$ with the basic distribution $B(x)$. These have to be provided by a *random number generator*, about which more later; for the moment we simply assume such a source to exist. For each x that is delivered we compute $f(x)$, and after N points x we can compute the various sums

$$S_m = \sum_{j=1}^{N} (w_j)^m \quad , \quad w_j \equiv f(x_j) \ . \tag{17}$$

The w_j are colloquially known as the *weights*. Clearly, the various S_m, and any combination of them, are random numbers, for which we can compute expectation values. Note that

$$\langle (w_j)^m \rangle = \int dx\ B(x)\ f(x)^m = J_m \ . \tag{18}$$

The Monte Carlo estimator of the integral J_1 is

$$E_1 = \frac{1}{N} S_1 \quad . \tag{19}$$

The expectation value of the random variable E_1 is

$$\langle E_1 \rangle = \frac{1}{N} \langle S_1 \rangle = \frac{1}{N} \sum_{j=1}^{N} \langle w_j \rangle = J_1 \quad , \tag{20}$$

where we have used the fact that expectation values of linear combinations are linear combinations of expectation values. The Monte Carlo estimate is *unbiased* in the sense that $\langle E_1 \rangle = J_1$ strictly. Therefore, if the estimate converges at all, it will converge to the right answer. That it converges indeed appears if we compute the variance of E_1:

$$\sigma\left(E_1\right)^2 = \left\langle E_1^2 \right\rangle - \langle E_1 \rangle^2 = \frac{1}{N^2} \left\langle S_1^2 \right\rangle - J_1^2 = \frac{1}{N}(J_2 - J_1^2) \tag{21}$$

This computation is nontrivial in the sense that $\langle S_1^2 \rangle \neq \langle S_1 \rangle^2$ in general: in appendix 2 we show how perform such averages. The quantity $J_2 - J_1^2$ depends only on $f(x)$, and therefore the variance of the distribution of E_1 decreases as $1/N$: the Chebyshev inequality then tells us that for large N the probability of obtaining a value for E_1 far away from J_1 becomes very small, and the central limit theorem provides confidence levels. We conclude that Monte Carlo integration will converge as long as $f(x)$ is quadratically integrable. As numerical methods go, this is a very mild condition.

The convergence of the integration error as $1/\sqrt{N}$ is not very rapid, but the Monte Carlo method has a unique advantage over most other methods of numerical integration. In the above discussion, we have never actually used the fact that the integral was one-dimensional. Indeed, we may trivially extend it to the integration of a function $f(\vec{x})$ over a d-dimensional hypercube. If we transform our random number stream $x_1, x_2, x_3, \ldots$ into a stream of random d-dimensional points: $(x_1, x_2, \ldots, x_d), (x_{d+1}, x_{d+2}, \ldots, x_{2d}), \ldots$, then these points will be iid uniformly over the hypercube, and the result for the integral and the error is exactly the same (apart from the fact that for N d-dimensional points we have to use up Nd random numbers from the stream). Therefore, the error converges as $1/\sqrt{N}$ *in any dimension*, whereas many other methods quickly deteriorate in higher dimension. The underlying reason for this will be discussed later on.

2.7 First and second order error estimators

Knowing that Monte Carlo integration converges is not enough: we have to obtain the value of $J_2 - J_1^2$ in order to arrive at an error estimate. To this end we define

$$E_2 = \frac{1}{N^2} \left(S_2 - \frac{1}{N} S_1^2 \right) \quad , \tag{22}$$

where $N^{\underline{2}} = N(N-1)$: the general falling powers $N^{\underline{m}}$ are discussed in appendix 1. It is simple to prove (see appendix 2) that

$$\langle E_2 \rangle = \sigma\left(E_1\right)^2 \quad , \tag{23}$$

which provides the desired error estimate.

The error estimate contains, of course, its own numerical uncertainty. This is particularly relevant if we use the estimate to establish Gaussian confidence levels, since they vary quite rapidly with the number of standard deviations. We therefore need also to obtain knowledge about $\sigma\left(E_2\right)^2$. Straightforward algebra leads to

$$\sigma\left(E_2\right)^2 = \frac{1}{N^3}\left(J_4 - 4J_3J_1 + 3J_2^2\right) + \frac{1}{N^2}\left(\frac{N^{\underline{4}}}{(N^{\underline{2}})^2} - 1\right)\left(J_2 - J_1^2\right)^2 \quad , \tag{24}$$

which is $\mathcal{O}\left(N^{-3}\right)$ as it should[4]. With some effort we find that the estimator

$$E_4 = \frac{N^{\underline{2}}}{N^2 N^{\underline{4}}}\left(S_4 - \frac{4}{N}S_3S_1 + \frac{3}{N}S_2^2\right) + \left(\frac{1}{(N^{\underline{2}})^2} - \frac{1}{N^{\underline{4}}}\right)\left(S_2 - \frac{1}{N}S_1^2\right)^2 \tag{25}$$

has the desired property:

$$\langle E_4 \rangle = \sigma\left(E_2\right)^2 \quad . \tag{26}$$

The estimators E_1, E_2 and E_4 can easily be evaluated by bookkeeping of the sums S_1 to S_4. In most cases, N will be so large that terms of relative order $\mathcal{O}(1/N)$ can be neglected, and then we may use

$$\begin{aligned} E_2 &= \frac{1}{N^3}\left(NS_2 - S_1^2\right) \quad , \\ E_4 &= \frac{1}{N^7}\left(N^3S_4 - 4N^2S_3S_1 - N^2S_2^2 + 8NS_2S_1^2 - 4S_1^4\right) \quad . \end{aligned} \tag{27}$$

Of course, also the estimator E_4 has its variance. The corresponding estimator, E_8, and the estimator of *its* variance, E_{16} are known to leading order in N, but of little interest apart from the appealing combinatorial methods used in obtaining them[4].

2.8 Examples

The simplest case is the integrand $f(x) = x_0$, the constant function. It is trivially seen that in that case the estimators E_2 and E_4 evaluate to zero. This simply checks the statement that a constant function is integrated with perfect accuracy for any $N \geq 1$.

A less trivial example is the class of functions of the form

$$f_\alpha(x) = (1+\alpha)\, x^\alpha\, \theta(0 < x \leq 1) \quad . \tag{28}$$

[4]The estimator E_2 is $\mathcal{O}\left(1/N\right)$ as is clear from $\langle E_2 \rangle$; and its variance, which is quadratic, is again multiplied by the standard Monte Carlo factor $1/N$.

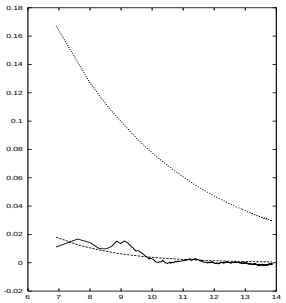
These functions are normalized: we have

$$J_m = \frac{(1+\alpha)^m}{1+m\alpha} \quad , \tag{29}$$

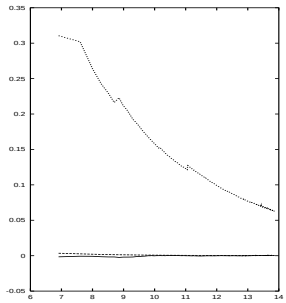
so that J_m is defined for $\alpha > -1/m$. The expectation values of the three estimators are, therefore

$$\begin{aligned}
\langle E_1 \rangle &= 1 \ , \\
\langle E_2 \rangle &= \frac{1}{N} \, \frac{\alpha^2}{1+2\alpha} \ , \\
\langle E_4 \rangle &= \frac{2}{N^2 N^2} \frac{\alpha^4 \left(N(4+4\alpha-6\alpha^2+6\alpha^3) + (-3+3\alpha+18\alpha^2-6\alpha^3) \right)}{(1+2\alpha)^2(1+3\alpha)(1+4\alpha)} \\
&\approx \frac{4}{N^3} \frac{\alpha^4(2+2\alpha-3\alpha^2+3\alpha^3)}{(1+2\alpha)^2(1+3\alpha)(1+4\alpha)} \ ,
\end{aligned} \tag{30}$$

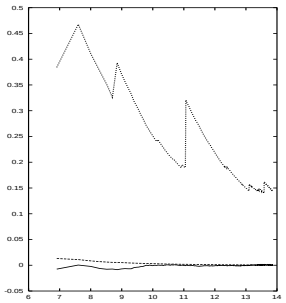
where the last approximation holds for large N. Below we present the estimates for various values of α in double-logarithmic plots, as a function of N up to 2^{20}. The random-number source used in these estimates, the algorithm `RCARRY`, will be discussed later on. The first- and second-order error estimates have been multiplied by appropriate powers of $\sqrt{N}$ so that, ideally, they approach constants.



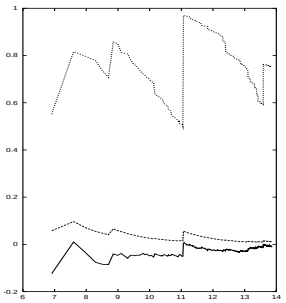
$\alpha = 1.0$



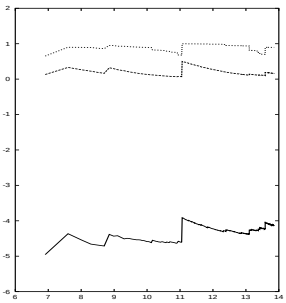
$\alpha = -0.1$



$\alpha = -0.3$



$\alpha = -0.7$



$\alpha = -0.999$

The results for the integral (lowest lines), first-order error estimate (middle lines) and second-order error estimates (upper lines). As α becomes more negative, the error estimates deteriorate.

Some numerical results are collected in the following table. The last column displays the actual integration error in units of the estimated one.

α	E_1	E_2 , $\langle E_2 \rangle$	E_2 , $\langle E_2 \rangle$	$\lvert 1 - E_1 \rvert / \sigma$
1	0.9990	3.178E-7 3.179E-7	7.716E-20 7.710E-20	1.858
-0.1	1.000	1.193E-8 1.192E-8	2.209E-21 2.281E-21	1.657
-0.3	1.000	2.114E-7 2.146E-7	1.918E-16 —	1.170
-0.7	0.9903	1.287E-4 —	5.260E-9 —	0.8474
-0.999	1.589E-2	6.354E-6 —	2.612E-11 —	6.203

For $\alpha < -0.25$, the integrand is not quartically integrable. This shows up as large jumps in the second-order error estimate: these are caused by incidental values of x that are very small, leading to a very large value of $f_\alpha(x)$. For $\alpha < -0.5$, the integrand is not quadratically integrable and the jump phenomenon starts to occur in the first-order error estimate as well. For α very close to -1 the integrand is almost not integrable, and the integral estimate itself becomes very bad. Note that the error estimators E_2 and E_4 are always finite, even if their expected values (the lower entries) are undefined. A lesson to be learnt from this excercise is that one should monitor the behaviour of the estimators as a function of the number of Monte Carlo points used, since often the smoothness of the integrand is obscure.

2.9 Comparison with other quadrature rules

The convergence of the error in Monte Carlo integration is, as we have seen, essentially as $1/\sqrt{N}$. This is, at first sight, not particularly fast. For example, a one-dimensional integral can be evaluated with the midpoint rule: J_1 is then approximated by Q_1, where

$$Q_1 = \frac{1}{N} \sum_{k=1}^{N} f\left(\frac{2k-1}{2N}\right) \quad , \tag{31}$$

and it is easy to show that this rule leads to an error term $|Q_1 - J_1| \approx C/N^2$, where C is an 'average' of the second derivative of $f(x)$. Indeed, in many cases one would not adopt Monte Carlo as the first choice to evaluate simple one-dimensional integrals. However, there may be drawbacks. For instance, the integrand $f(x)$ may be not very smooth: a discontinuity in $f(x)$ will lead to an infinite (first and) second derivative, so that the error bound becomes essentially useless: indeed, it is easy to see that in such cases the error will go as $1/N$ rather than as $1/N^2$. The real advantage of Monte Carlo however, appears in higher

dimensions. The d-dimensional analogue of the rule (31) is then given by

$$Q_1 = \frac{1}{N} \sum_{k_1=1}^{M} \sum_{k_2=1}^{M} \cdots \sum_{k_d=1}^{M} f\left(\frac{2k_1 - 1}{2N}, \frac{2k_2 - 1}{2N}, \ldots, \frac{2k_d - 1}{2N} \right) \quad , \tag{32}$$

where $M^d = N$; and the error will go as $1/M^2 = 1/N^{(2/d)}$. As mentioned before, he Monte Carlo estimate has the unique property that its error is expected to go as $1/\sqrt{N}$ *in any dimension*. Therefore, the midpoint rule will actually converge *more slowly* than Monte Carlo for dimensions larger than 4.

Of course, the midpoint rule is not the optimal one-dimensional quadrature rule. Simpson's rule will lead to a convergence rate of $1/N^4$ in one dimension, while the 'best' quadrature rule, an N-point Gauss' rule, will actually have an error of approximate order $1/N^{2N}$. Hence multidimensional quadrature with Gauss' rule *may* outperform Monte Carlo provided $N > (d/4)^d$ or so (still a reasonable 531441 for $d = 12$) but only *provided* that the integrand is in fact $2N$ times continuous differentiable. Any integrand that is less smooth than this leads to an error estimate that is certainly worse, and at any rate *unknown* – and this is what makes these very refined quadrature rules much less attractive for the typical ugly integrands one tends to encounter in practice, whose smoothness properties are often obscure.

In fact, the most cogent argument against fixed-rule quadrature may come from considering the practical application. Suppose that a numerical integration has been performed, but one decides to increase its precision. For a Gauss' rule or other quadrature rule, one then has to start all over again, with (presumably) increased M, and the function values computed so far are then essentially worthless[5]. For Monte Carlo on the other hand, one simply keeps adding points until the desired value of the error estimate has been reached.

It may be considered fair to conclude that Monte Carlo is essentially the most robust method of numerical integration, relying not on the differentiability properties of the the integrand bur rather on its integrability properties; and that it leads to an error estimate that is admittedly probabilistic but at least *known*.

[5]This is not strictly true for all rules. For instance, for the trapezoid rule where the points are parametrized by $(j-1)/(M-1)$, $j = 1, 2, \ldots, M$, the old points may be reused when M is replaced by $2M - 1$. Note, however, that for $d = 10$ this requires a computational increase by a factor of about 1000, which may be asking too much.

3 Pseudo-random number generators

3.1 Digital vs. analog methods

In generating random numbers, one may choose to employ some 'natural' process (radioactive decay, coin tossing or Bingo, noise or other fluctuations, ants or monkeys, ...) or a computer. The choice for a natural source may seem reasonable, but in practice it is hampered by a number of considerations.

- **Randomness** Even if a 'natural' process appears to give unpredictable results, it does not therefore follow that the results are random in the sense that they are uncorrelated. This would have to be 'proven' by empirical tests, which can only determine randomness up to a given level. In the end, if we suppose the randomness to arise out of the fact that we cannot understand the processes leading from one number to the next, this does not mean that a correlation does not exist: it may simply be too complicated for us to describe. Certainly, the numbers will not automatically be uniformly distributed. Note that if (for instance by coin tossing) a series of (0,1) bits is produced in which we are somehow guaranteed that the successive bits are uncorrelated, we may combine two or more bits into one so as to arrive at a strictly uniformly distributed set of bits. The trick is to take the incoming bits in pairs: if they are either (0,0) or (1,1), discard them: if they are (0,1), return 0; and if they are (1,0), return 1. Then 0 and 1 will have the same frequency.

- **Speed** Natural processes tend to be much too slow for useful application. If some natural process (like noise in an electric circuit) is sampled very rapidly, the numbers are likely to be correlated since any natural process has some internal time scale. Therefore, in a modern type of simulation which may use up to 10^{12} random numbers per second, it will be hard to find a natural process with this kind of output. Of course this might be circumvented by laboriously collecting random numbers and storing them: but then the storage requirements easily run into terabytes.

- **Reproducibility** The essence of a natural process is that its series of numbers *never* repeats exactly. Actually this may be a liability rather than an asset. Suppose a simulation code has to be debugged since it behaves strangely for some random input: in that case it is crucial that the same input can be applied again and again, and with a 'natural' source this is sadly impossible.

The consensus nowadays is that it is much preferable to use not 'natural' random numbers, but rather numbers produced by a simple, repeatable, computer algorithm. Of course, being generated by an algorithm, such number streams are not truly random. The idea is that in an actual application the requirement of true randomness is much more than is actually needed: for instance, in a simple integration it is sufficient if the numbers are simply uniformly distributed.

Such computer-generated number streams are called *psuedo-random*[1]; they only pretend to be random enough for practical purposes. The search for algorithms to generate sequences that are acceptably pseudo-random is one of the major topics in the field.

3.2 The ensemble of random-number algorithms

It is illustrative to discuss the ensemble of all algorithms for 'random number' streams. Suppose that we want to generate streams of integers that take values in $(1, 2, 3, \ldots, M)$. Typically, M is a large number, such as $2^{\text{'large'}}$: many actual random number generators work internally with integers, and in the output scale them to the interval $(0, 1)$ by dividing by M. This has the advantage that no rounding errors occur internally, and the precision of the outputted numbers is constant over the interval[2].

We shall first consider algorithms that produce a new number in the stream using only the last one produced: that is, there is some function $f(n)$, and $n_{j+1} = f(n_j)$, $j = 1, 2, \ldots$. Now, no matter how the algorithm $f(n)$ is programmed, it can completely and uniquely be specified by giving the set of its results:

$$f(n) \quad \leftrightarrow \quad \{f(1), f(2), f(3), \ldots, f(M-1), f(M)\}$$

There are, therefore, exactly M^M different algorithms in the ensemble.

Consider the proceedings of an algorithm in the ensemble. One starts by picking a starting value, n_1, and succesively computes $n_2 = f(n_1)$, $n_3 = f(n_2)$, and so on, thus generating a stream of numbers. Now, as soon as a number occurs that was already produced before, the sequence of numbers will start to cycle, simply repeating the numbers between the two repeats, and from that point on the stream becomes useless as a model for 'natural' random numbers. We call the length of the stream up to the first repetition the *lifetime* of the algorithm: M is the maximal possible lifetime of any algorithm [3]. Clearly, a long lifetime is the most basic property that an acceptable pseudo-random number algorithm must have, since a stream of 'real' random numbers should not exhibit any periodicity.

We now decide on a starting value n_1, and choose an algorithm at random from the ensemble. The probability that $n_2 \neq n_1$ is clearly $(1 - 1/M)$; the probability that, in addition $n_3 \neq n_1$ and $n_3 \neq n_2$ is $(1 - 1/M)(1 - 2/M)$, and so on. Suppose that the numbers up to n_p are all different, but that n_{p+1} is a number already encoutered before in the stream. This algorithm, then, has a lifetime of p, and the probability to pick such an algorithm is

$$Q(p) = \frac{M!}{(M-p)!} \frac{p}{M^{p+1}} \quad . \tag{33}$$

[1] From the Greek, $\psi\epsilon\upsilon\delta\epsilon\iota\nu$ = 'to lie'.

[2] For floating-point numbers, small numbers have a higher absolute precision than large ones.

[3] We do not call it the 'period' since only part of it will be repeated periodically in general. The period is equal to the lifetime if the first repeated number is actually the first generated number.

We can write this as

$$Q(p) = A(p) - A(p+1) \quad , \quad A(p) = \frac{M^{\underline{p}}}{M^p} \ , \tag{34}$$

so that $A(p)$ is the probability to pick an algorithm with a lifetime of *at least* p; indeed, $\sum_p Q(p) = A(1) = 1$.

We can compute the expected lifetime of an algorithm as follows:

$$\begin{aligned} \langle p \rangle &= \sum_{p=1}^{M} p\, Q(p) = \sum_{p=1}^{M} A(p) \\ &= \int_0^\infty dx\, e^{-x} \sum_{p=1}^{M} \frac{x^p}{p!} A(p) = \int_0^\infty e^{-x} \left(\left(1 + \frac{x}{M}\right)^M - 1 \right) \\ &= \sqrt{\frac{\pi M}{2}} - \frac{1}{3} + \mathcal{O}\left(\frac{1}{\sqrt{M}}\right) \ . \end{aligned} \tag{35}$$

This justifies a Stirling approximation in $Q(p)$: we find

$$Q(p) \approx \frac{p}{M} \exp\left(-\frac{p^2}{2M}\right) \ , \tag{36}$$

so that, apart from the normalization, the probabilities are a function of $p/\sqrt{M}$: the lifetime of a typical, arbitrarily chosen algorithm is much smaller than M.

The desirable algorithms are, of course, those with $p = M$: the probability to hit upon such an algorithm by coincidence is

$$Q(M) = \frac{M!}{M^M} \approx e^{-M}\sqrt{2\pi M} \ , \tag{37}$$

which is very unlikely indeed. The message from the above discussion is that pseudo-random number algorithms must be chosen very carefully; the fact that an algorithm looks complicated does *not* imply that the resulting stream is acceptably random (for a horrible example, see [3]). Indeed, no matter how complicated the code, the probability that an algorithm chosen at random contains one or more numbers that are transformed *into themselves* is $1 - (1 - 1/N)^N \approx 1 - 1/e$, or about 63%!

We may also consider algorithms that employ not only the current number to obtain the next one, but use, say, the numbers n_{j-1}, n_{j-2},...,n_{j-k} to arrive at n_j. It is not necessary that all numbers between n_{j-1} and n_{j-k} are actually used. Note that after n_j has been generated, the new set of numbers to be kept in memory is n_j,n_{j-1},...,n_{j-k+1}, and therefore these are called *shift-register* algorithms. The analysis is now precisely the same as before since we may realize that a set of k numbers with values in $(1, \ldots, M)$ is completely equivalent with a single ('large') number taking values in $(1, \ldots, M^k)$. We may therefore simply replace M by M^k, and the number n_j in the stream is, in

this picture, a rounded-off version of the 'large' number. This influences the absolute precision of the result (which one can certainly afford if M is large), but not the lifetime of the algorithm nor the distribution of its numbers (up to details of order $\mathcal{O}(1/M)$). The maximal possible period will now be M^k, which can be huge. One might adopt the point of view that the average lifetime of a randomly chosen algorithm, $\sqrt{\pi M^k/2}$, is also quite sufficient, but for such suboptimal algorithms there is no guarantee that the resulting numbers will be uniformly distributed.

Before finishing this section an interesting observation is in order. Suppose we use the random number stream in a simulation. If we only look at the state of the random number generator before and after an event has been simulated, we see that the simulation program can be interpreted as just another algorithm for updating the state of the generator: in fact it is itself a would-be random number generator with an algorithm that is usually very complicated (for instance, often the number of random numbers used inside is not fixed but subject to random choices). This is precisely the type of 'random algorithm' discussed here. Therefore, if the random number algorithm is optimal so that it contains M possible states, the simulation program can be expected to yield *exactly the same event* after only $\mathcal{O}(\sqrt{M})$ steps. This 'duplication of events' has, indeed been observed[**?**]. Of course it can only occur if the simulation exhausts the period of the generator several or many times over.

The lesson from this discussion is twofold: first, a pseudo-random number algorithm should be sufficiently simple that it can be analyzed; and, second, a Monte Carlo study should *not* come close to exhausting the period of the algorithm.

3.3 Obsolete algorithms

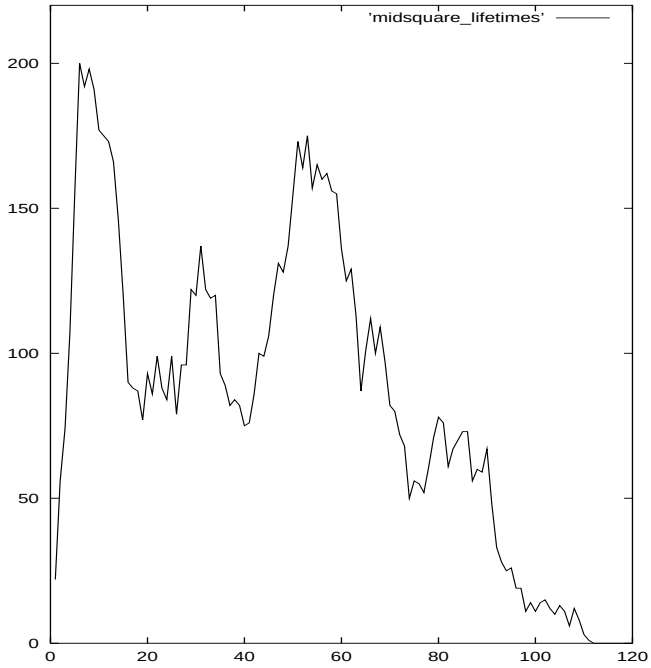
3.3.1 The mid-square method

One of the oldest methods for generating pseudo-random number streams is the mid-square method, which consists of the following: take a number consisting of an even number of digits (or bits), square it, and truncate to the middle digits by chopping off the leading and least leading digits; so, if one deals with decimal numbers of $2k$ digits,

$$n_{j+1} = \left\lfloor \frac{n_j^2}{10^k} \right\rfloor \mod 10^k \quad . \tag{38}$$

This rests on the idea that of the square of a given number, both the most leading and the least leading digits can simply be predicted, while the 'middle part' depends on the previous number in a more complicated way. This algorithm is a typical example of a 'random algorithm'; it is difficult to analyze since the resulting stream depends crucially on the starting value, and performs poorly as can be seen from the lifetimes, of which the distribution as a function the of

starting values for 4-digit decimal numbers is shown.



Lifetimes of the mid-square algorithm for decimal 4-difit numbers. The longest lifetime is a disappointing 111 for starting value 6239, not even the 'expected' 125 for $M = 10,000$.

3.3.2 The logistic map

It is frequently proposed to employ the phenomenon of 'chaos' to produce pseudo-random numbers. As an example, the fully-chaotic logistic map

$$x_{j+1} = 4x_j(1 - x_j) \tag{39}$$

is supposed to produce a really chaotic sequence of values. In finite-precision arithmetic, however, it performs poorly as a pseudo-random number source. The reason for the mediocre performance of this algorithm becomes evident if we realize that if we define

$$x_j = (\sin\theta_j)^2 \quad , \quad 0 \leq \theta_j \leq \pi/2 \quad , \tag{40}$$

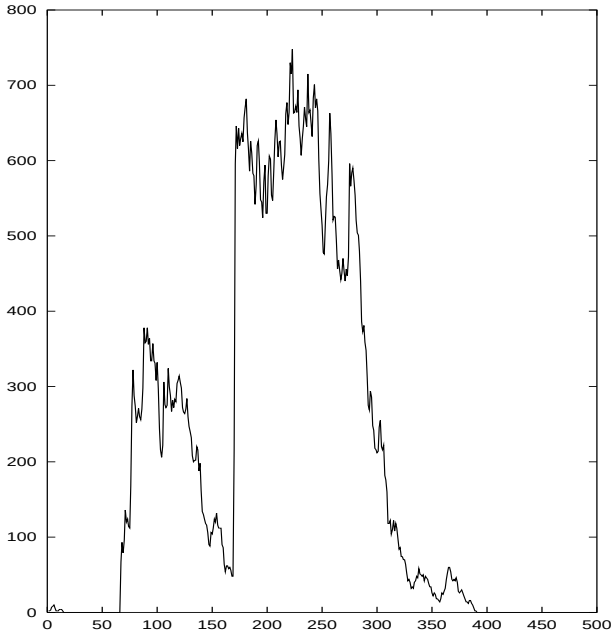
the logistic map is equivalent to

$$\theta_{j+1} = 2\theta_j \bmod (\pi/2) \ . \tag{41}$$

After a few iterations, therefore, the algorithm will completely obliterate[4] the information in the starting value θ_1, and from then on it is governed by the rounding errors in the computer. This makes it, essentially, a random algorithm in the sense discussed above, and it behaves accordingly.

[4]Because the leading bits keep getting chopped off.

The distribution of lifetimes of the logistic map algorithm 39 is given here as a function of the starting value. The numbers have been truncated to an accuracy of 10^{-5} in the algorithm, so that the maximum lifetime is potentially 10^5; different truncations produce essentially the same plot, with a maximum lifetime of order of the square root of the maximum.



T

3.4 Linear congruential generators

An old but robust algorithm that turns an integer into another integer is the *linear congruential* algorithm:

$$x_{n+1} = (ax_n + c) \bmod m \ , \tag{42}$$

where a, c and m are predetermined integers. The shift register consists, in this case, of only the last generated number. Pseudo-random floating-point numbers in the unit interval are obtained by simply using x_n/m. The simplicity of the linear congruential algorithm makes it possible (but nor trivial!) to analyze it. Unsurprisingly, the properties of the random number stream are heavily dependent on the number-theoretical properties of a, c and m. A wealth of information about this can be found in [3]. An important simple result is that the random numer stream has the maximum possible period, m, iff

1. c is relatively prime to m;
2. every prime factor of m also divides $b \equiv a - 1$;
3. if m is a multiple of 4, b is also a multiple of 4.

If $c = 0$ is chosen, the maximal period is smaller than m, but can be made quite large [3]. If the period is m, all integers $0, 1, 2, \ldots, m - 1$ will occur precisely once in a period, and the random floating-point numbers are spaced perfectly

regularly in the unit interval. Since this doesn't look random at all, such a situation is usually to be avoided: typically, one aims for an m so large that a full period will never be exhausted.

Finding good values for the parameters is to some extent an art form: see [3]. Bad choices can often be identified by analytic means. As an example, the *potency* of a period-m linear congruential algorithm is defined to be the smallest integer s such that b^s is a multiple of m. Taking $x_0 = 0$ for simplicity, we then see that

$$\begin{aligned} x_n &= \left(\frac{a^n - 1}{b} c\right) \bmod m \\ &= \left(c\left(n + \binom{n}{2} b + \binom{n}{3} b^2 + \cdots + \binom{n}{s} b^{s-1}\right)\right) \bmod m \end{aligned} \quad (43)$$

If the potency is s, the values of x_n are governed by a polynomial in n of degree s. A potency of at least 5 appears to be a minimal requirement.

Another theoretical result can be obtained approximately, by switching to a real-number model of computation. Assume m to be very large, and c to be smaller than ma. Moreover, assume that the period is actually m. The numbers in the stream fill the unit interval densely, and uniformly if we consider the whole period. We then have an approximate real-number relation between the floating-point pseudorandom numbers:

$$x_{n+1} = a x_n + \delta - k \ , \quad (44)$$

where k is a natural number such that

$$\frac{k - \delta}{a} \le x_n < \frac{k + 1 - \delta}{a} \ , \quad (45)$$

and $\delta = c/m$. A sum over the whole period is equivalent to an integration over x_n, so that we have, approximately,

$$\langle x_n \rangle \approx \frac{1}{2} \quad , \quad \left\langle x_n^2 \right\rangle \approx \frac{1}{3} \ . \quad (46)$$

The average value of $x_n x_{n+1}$ is now given by

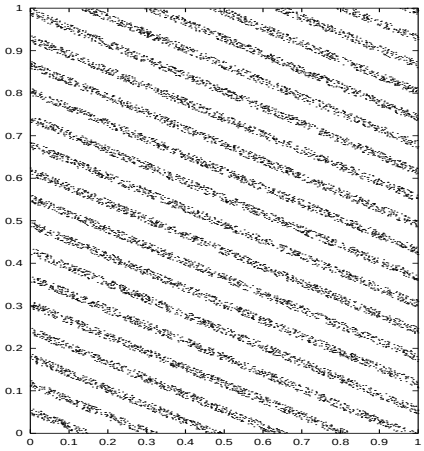
$$\begin{aligned} \langle x_n x_{n+1} \rangle &\approx \int_0^{(1-\delta)/a} dx \; x(ax + \delta) + \int_{1-\delta/a}^{1} dx \; x(ax + \delta - a) \\ &\quad + \sum_{k=1}^{a-1} \int_{(k-\delta)/a}^{(k+1-\delta)/a} dx \; x(ax + \delta - k) \\ &= \frac{1}{4} + \frac{1}{12a}(1 - 6\delta(1 - \delta)) \ , \end{aligned} \quad (47)$$

so that the *serial correlation coefficient* is given by

$$\frac{\langle x_n x_{n+1} \rangle - \langle x_n \rangle \langle x_{n+1} \rangle}{\langle x_n^2 \rangle - \langle x_n \rangle^2} \approx \frac{1 - 6\delta(1 - \delta)}{a} \ . \quad (48)$$

The error made here is in approximating a discrete set of points by a continuum, and is therefore[5] of order $\mathcal{O}(a/m)$. It is seen that the serial correlation can be expected to be small if a is large. From the uncertainty of $\mathcal{O}(a/m)$ it might be concluded that a ought to be $\mathcal{O}(\sqrt{m})$, but not so: in fact the serial correlation is usually quite small [3].

A final remark is in order on the 'lattice structure' of linear congruential generators. We may read the k-tuples $(x_n, x_{n+1}, x_{n+2}, \ldots, x_{n+k})$ as points in a k-dimensional unit hypercube. These points will then be restricted to a number of regularly spaced parallel hyperplanes. For high k the number of planes can actually be quite small, and the generator is then not useful as a model of a random point stream. That *some* lattice structure must occur is trivial: for instance, in the real-number model for computation used above, all 2-tuples fall on the single straight line given by $f(x) = ax + \delta$ if we disregard the $\bmod 1$ operation: and this operation just translates the various pieces of this line to the unit square. The surprise, rather, is that the actually observed number of planes is usually quite small. The so-called *spectral test* allows for a determination of the distance between the hyperplanes, and can indicate good choices for a in a given dimension.



Distribution of 3-tuples (x_n, x_{n+1}, x_{n+2}), for the linear congruential generator with $m = 2^{20} + 7$, $a = 1021$, $c = 1$ and $x_0 = 512$. Both m and a are prime numbers, so the period is maximal. The plot shows the projection of the slice $x_{n+2} < 0.3$, for the first 40,000 3-tuples.

3.5 Modern forms

The straightforward implementation of the linear congruential algorithm is limited by the computer wordsize for integers, usually 2^{31}. For many applications this is not sufficient, in particular the period is often too small. From our discussion of 'random algorithms' it is seen that by using a larger register a much larger effective m can be obtained. As a useful example, we may discuss the `RCARRY` algorithm [5]. The register consists of the last r generated numbers, $(x_{n-1}, x_{n-2}, \ldots, x_{n-r})$ (these can be stored as integers in the range $(0, 1, 2, \ldots, B-1)$ and the so-called *carry bit* c_{n-1}, associated with x_{n-1}. The

[5]This can be proven by a more rigorous treatment.

algorithm is as follows:

$$\begin{aligned} y &\leftarrow x_{n-s} - x_{n-r} - c_{n-1} \\ \text{if } y \geq 0 \text{ then } x_n &\leftarrow y \quad , \quad c_n \leftarrow 0 \\ \text{if } y < 0 \text{ then } x_n &\leftarrow y + B \quad , \quad c_n \leftarrow 1 \ , \end{aligned}$$

after which the register is shifted. The new number x_n is again in the range from 0 to $B-1$. The recommended choice is $s = 10$, $r = 24$, $B = 2^{24}$.The floating-point output is obtained by division by B as usual. On most machines, floating-point representation is exact if we take $B = 2^{24}$, so that in that case the numbers x_j can even be stored as rational fractions with denominator 2^{24} without risk of rounding errors that can wreck the performance. The first r numbers have to be stored by hand and can be anything except either all zeroes and $c_{n-1} = 0$, or all $B-1$ and $c_{n-1} = 1$.

Let us define, for $n \geq r$, the numbers z_n through

$$z_n = \sum_{j=0}^{r-1} B^j x_{n-r+j} - \sum_{j=0}^{s-1} B^j x_{n-s+j} + c_{n-1} \ . \tag{49}$$

A single step of the algorithm is then equivalent to

$$B z_{n+1} - z_n - m x_n = 0 \quad , \quad m \equiv B^r - B^s + 1 \ , \tag{50}$$

so that, in essence, we have a linear congruential generator for the numbers z:

$$z_{n+1} = a z_n \bmod m \quad , \quad a = m - \frac{m-1}{B} \ \text{ so that } \ aB \bmod m = 1 \ . \tag{51}$$

The choices recommended above rely on the fact that

$$m = 2^{576} - 2^{240} + 1 \ \text{ is prime } \ , \tag{52}$$

so that the period will be very large: not equal to m, since $c = 0$, but still a respectable $(m-1)/48 \approx 5 \times 10^{171}$. The multiplier a is[6]

$$a = 2^{576} - 2^{552} - 2^{240} + 2^{216} + 1 \ . \tag{53}$$

In spite of its long period, the `RCARRY` algorithm is known to show some nonrandomness. In [6] it is shown that there are correlations between nearby numbers, and it is suggested to skip a number of points in the final output stream, thus

[6] For the record: $m =$ 24733040147310453406050252101964719003513134910121183991406305609289722510653186717031640106124304498783082436123775500976806753356383269414006225822627420979500057085607936 1.
The multiplier is $a =$ 24733038673106381210135661382607460804970599395698832266234263274834136477206248282598494759981052476260126375768920671440398509175301416716677335617826706568514290466160640 1.
The period is 5152716697356344459593802521242649792398569772941913331542980335268692189719413899381591688775896770579808840859119896036834740282579847794584630379714046037395845226168320.

avoiding correlation between the final output numbers. The recommendation is to produce a batch of up to 24 numbers, then skip 223, and then produce the next batch. In view of the large available period, this is certainly useful, although it slows the generator down; but in many cases the most time-consuming part of a Monte Carlo calculation is not spent in the pseudorandom-number code.

3.6 Algorithm testing: general strategy

The aim of using (pseudo-)random numbers in Monte Carlo is, essentially, the calculation of an unknown integeral. Apart from the error estimators (whose results may or may not be meaningful, see section 1), it is of course not known precisely how well the numbers perform. Therefore, one needs to build up some confidence that the pseudorandom number stream does actually look like a truly random number stream, by subjecting it to testing. The test may be theoretical, such as the spectral test or the determination of the potency, but most theoretical tests only address the whole period and are therefore not guaranteed to be meaningful for the recommended shorter streams. A number of empirical tests exist, that claim to look for various aspects of (non)randomness of a given stream. Invariably, they consist of the calculation of a certain number using the given stream, and comparing the result with that which an equally long stream of truly random numbers would give. Since this last result is stochastic, we need its mean, variance, and so on to assign a confidence level to the pseudorandom result. As a simple example, consider a stream of N pseudorandom numbers $x_1, x_2, x_3, \ldots, x_n$ in the unit interval. If the algorithm is any good, this stream ought to look just like a set of N truly random numbers drawn from $B(x)$. We may therefore divide the unit interval into, say, M bins, and count the occupancies of these bins. If the bins are equally sized, the occupancy numbers are expected to fluctuate around N/M, and we have to decide whether the fluctuations are suspiciously small, suspiciously large, or just about right. To this end, we may calculate the χ^2 of the occupancies. This quantity is discussed below.

There is a philosophical glitch here. If we perform any statistical test, a stream of truly random numbers will fail at the one-σ level in about one out of three cases. A truly random-looking stream of pseudorandom numbers ought to do the same. So, if a given test is failed, what does one conclude? In practice, of course, one usually performs many different tests. At any rate, the literature abounds with proposed generators for which it is claimed that they pass all tests performed. The conclusion then should be that such a generator is *not* a good model for truly random numbers!

3.7 Chi-square distribution

Assume that N truly random iid uniform numbers ('events') fall each into any one of M bins. Let the probability for any event to fall in bin i be $p_i > 0$, and

let the actual number in bin i be n_i. We have

$$\sum_{i=1}^{M} p_i = 1 \quad , \quad \sum_{i=1}^{M} n_i = N \quad . \tag{54}$$

The combined probability for occupation $(n_1, n_2, \ldots, n_M)$ is

$$P(n_1, n_2, \ldots, n_m) = \frac{N!}{n_1! n_2! \cdots n_m!} p_1^{n_1} p_2^{n_2} \cdots p_M^{n_M} \quad , \tag{55}$$

leading to expectation values

$$\langle n_{j_1}^{\underline{m_1}} n_{j_2}^{\underline{m_2}} \cdots n_{j_k}^{\underline{m_k}} \rangle = N^{\underline{m_1+m_2+\cdots+m_k}} p_{j_1}^{m_1} p_{j_2}^{m_2} \cdots p_{j_k}^{m_k} \quad , \tag{56}$$

where the labels $j_1, \ldots, j_k$ are different. In particular, $\langle n_i \rangle = N p_i$. The χ^2 statistic tests how well the actual occupations approach the expected ones: it is defined by

$$\chi^2 = \sum_{i=1}^{M} \frac{(n_i - N p_i)^2}{N p_i} = \sum_{i=1}^{M} \frac{n_i^2}{N p_i} - N \quad . \tag{57}$$

Straightforward algebra gives

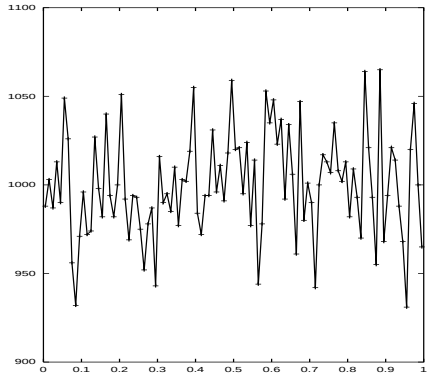
$$\begin{aligned} \left\langle \chi^2 \right\rangle &= M - 1 \quad , \\ \sigma\left(\chi^2\right)^2 &= 2(M-1) + \frac{1}{N}\left(\sum_{i=1}^{M} \frac{1}{p_i} - M^2 - 2M + 2 \right) \quad . \end{aligned} \tag{58}$$

The variance has its minimum, $2(M-1)(1-1/N)$, if $p_i = 1/M$ for all i. The expectation value of χ^2 for truly random points is independent of the probabilities p_i and the total number of points N, which makes it particularly attractive[7]; however, for the variance to be useful, $\sum (p_i N)^{-1}$ should not be too large. A rule of thumb is that $N p_i$ should be at least 5 or more in every bin. Therefore, if very 'small' bins (with low p_i) are present, N must be large. If χ^2 is either very large or very close to 0, the fluctuations, and hence the stream of pseudorandom numbers, are suspect. The number $\chi^2/(M-1)$, called the χ^2 per degree of freedom, should be of $\mathcal{O}(1)$.

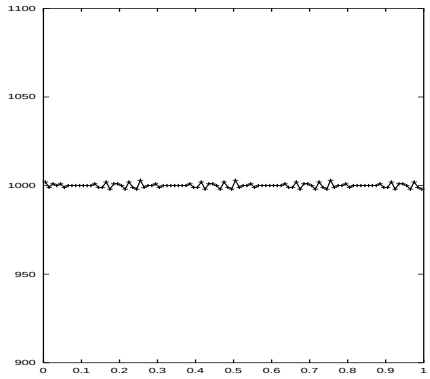
3.8 An example

We divide the unit interval in 100 equal-sized bins, and divide a stream of 10^5 pseudorandom points into it. The expected χ^2 is, then, 99, and its standard deviation is about 14.07. We consider one stream generated by the `RCARRY` algorithm, and one generated by the `CORPUT` algorithm (which is discussed later).

[7] The fact that it is equal to $M-1$ rather than M reflectes the fact that the occupancies of the bins are not independent, since $\sum n_i = N$.



The plot shows the occupancies of the 100 bins for 100,000 points generated by `RCARRY`, connected by a solid line to guide the eye. The average occupancy is of course 1000, with fluctuations of the expected $\mathcal{O}\left(\sqrt{1000}\right)$. The computed value of χ^2 is 85.234, or 0.86 per degree of freedom, or 0.98 standard deviations. As far as this simple test is concerned, the stream looks acceptably random.



The plot shows the occupancies of the 100 bins for 100,000 points generated by `CORPUT`, connected by a solid line to guide the eye. The average occupancy is of course 1000, but the fluctuations are tiny. The computed value of χ^2 is 0.148, or 0.0015 per degree of freedom, off the mark by 7 standard deviations. We conclude that the stream generated by `CORPUT` is much too uniform to be considered random.

3.9 Non-binning tests

Many of the empirical tests of pseudorandom number streams rely on binning n-tuples of numbers in some way or another. For instance, the permutation test takes n-tuples and determines which of the $n!$ possible permutations is required to put the n elements into a given order, leading to $n!$ bins that allow for a χ^2 determination. Time-honoured examples include also the collision test, the poker test, the coupon-collector's test, the run test, the gap test, and more: for an in-depth description, see [3]. Other tests, such as the spectral test, consider instead the actual distribution of the n-tuples in an n-dimensional hypercube. The test characteristic computed in that case is almost invariably a *discrepancy*, a measure of nonuniformity of real-number n-dimensional vectors. Since discrepancies play a part of their own, we defer a discussion to the next section.

4 Discrepancies and Quasi-Monte Carlo

4.1 The notion of non-uniformity of point sets

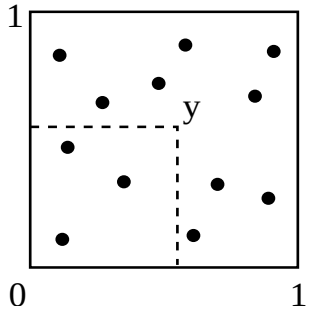
As discussed above, Monte Carlo integration error has a rather slow $1/\sqrt{N}$ convergence, to be contrasted with other rules such as the midpoint rule, with a $1/N^2$ convergence. The difference in convergence between the two methods is explained by the greater irregularity exhibited by random integration points. It is therefore useful to discuss *measures of non-uniformity of point sets* with an eye to how well the point sets integrate certain functions. Such measures are called *discrepancies*.

4.2 Star discrepancies

One of the oldest and most beloved discrepancies[1] is the 'star'-discrepancy, arising from the following arguments. Consider the fact that a point set that integrates arbitrary multidimensional step functions[2] very well, will integrate any Riemann-integrable function quite well. Thus, we alight on the following notions. Take some given D-dimensional point set $X = (\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N)$, random, regular, or otherwise. Define a step function as follows: by $\theta(\vec{y} > \vec{x})$ we denote the D-fold product $\prod_{\mu=1}^{D} \theta(0 \leq x^\mu \leq y^\mu)$, so that this step function equals 1 if $\vec{x}$ is inside the rectangle spanned by the origin and $\vec{y}$, otherwise zero. The quality of performance of the point set X in integrating the step function $\theta(\vec{y} > \vec{x})$ can be defined by

$$L^*(\vec{y}) = \frac{1}{N} \sum_{j=1}^{N} \theta(\vec{y} > \vec{x}) - \prod_{\mu=1}^{D} y^\mu \ , \tag{59}$$

and this is called the *local discrepancy* at $\vec{y}$.



This picture llustrates the concept of local discpreancy for $D = 2$. The point $\vec{y}$ is at $(0.55, 0.55)$ and the rectangle spanned by the origin and $\vec{y}$ contains 3 out of the 12 points. The local discrepancy is therefore 0.0525.

[1] The term 'discrepancy' used in these notes covers more than is usually meant in the mathematics literature. There, discrepancy in the strict sense deals with all rectangles with sides parallel to the coordinate axes in the hypercube, and *discrepancy restricts itself to those rectangles with one vertex at the origin. Here, we call *any* measure of non-uniformity a discrepancy.

[2] Also called *characteristic functions*.

If the rectangle between the origin and $\vec{y}$ receives its 'proper' share of integration points, the local *discrepancy will be zero, if not it will differ from zero. Some commonly used measures of *global* discrepancy are

$$L_\infty^* \equiv \sup_{\vec{y}} |L^*(\vec{y})| \quad , \quad L_2^* \equiv \int d^D\vec{y} \ (L^*(\vec{y}))^2 \quad . \tag{60}$$

The first of these is the *extreme* *discrepancy[3], the second one the *quadratic* *discrepancy[4]. The extreme *discrepancy is intuitively attractive, but the quadratic one is usually easier to handle. Of course, other variations are also possible. The L_∞^* and L_2^* discrepancies are, of course, closely correlated[5], and a point set with a very low discrepancy of the one type will usually have a low discrepancy of the other type as well. For the L_2^* discpreancy we can perform the $\vec{y}$ integral to get

$$L_2^* = \frac{1}{N^2} \sum_{i,j=1}^{N} \prod_{\mu=1}^{D} \left(1 - \max(x_i^\mu, x_j^\mu)\right) - \frac{2}{2^D N} \sum_{i=1}^{N} \prod_{\mu=1}^{D} \left(1 - (x_i^\mu)^2\right) + \left(\frac{1}{3}\right)^D \quad , \tag{61}$$

where $\mu = 1, 2, \ldots, D$ labels the coordinates of the points.

The notion of discrepancy allows to discriminate between uniform and non-uniform point sets. For instance, for a set of N iid uniform random points, we have

$$\langle L_2^* \rangle = \frac{1}{N} \left(2^{-D} - 3^{-D}\right) \quad , \tag{62}$$

and the expected value of L_∞^* is also roughly of $\mathcal{O}\left(1/\sqrt{N}\right)$. For the regular set of N points used in a D-dimensional M-point midpoint-trapezoid integration rule:

$$x_j^\mu = \frac{2k_\mu - 1}{2M} \quad , \quad j = 1 + \sum_{\mu=1}^{D} (k_\mu - 1) M^{\mu-1} \quad , \quad k_\mu \in \{1, 2, \ldots, M\} \ , \tag{63}$$

so that j runs from 1 to $N = M^D$, we find

$$L_2^* = \left(\frac{1}{3}\right)^D \left[\left(1 + \frac{1}{2M^2}\right)^D - 2\left(1 + \frac{1}{8M^2}\right)^D + 1\right] \approx \frac{D}{4M^2 3^D} \ , \tag{64}$$

where the approximation holds for large M. Finally, the 'most non-uniform' point set, in which all N points are identical to a given point $\vec{x}$, is independent of N:

$$L_2^* = \prod_{\mu=1}^{D} (1 - x^\mu) - 2 \prod_{\mu=1}^{D} \left(\frac{1 - (x^\mu)^2}{2}\right) + \left(\frac{1}{3}\right)^D \quad . \tag{65}$$

[3] In statisticians' terms, the Kolmogorov statistic.
[4] In statisticians' terms, the Kramér-von Mises statistic.
[5] Since L_2^* must fall between 0 and $(L_\infty^*)^2$.

In this case we have

$$(1-a)^D - 2\left(\frac{1-a^2}{2}\right)^D + 3^{-D} \leq L_2^* \leq 1 - 2^{1-D} + 3^{-D} \ , \tag{66}$$

where a is the unique number in $(0,1)$ for which $a(1+a)^{D-1} = 2^{D-2}$.

We see that L_2^* can reach values indpendent of N for the worst possible point set; for random point sets we expect a value of $\mathcal{O}(1/N)$, while for the regular grid we have $\mathcal{O}\left(1/N^{2/D}\right)$. Surprisingly, the regular grid is, under this characterization of nonuniformity, *less* uniform than a typical random point set in dimensions larger than two. The underlying reason is that, whereas regular grids and random point sets are both translation-invariant to a good degree, the random point set is also rotation-invariant, while the grid isn't; and in high dimensions rotational invariance becomes important[6].

4.3 Koksma-Hlawka inequalities

A number of theorems exist that illuminate the rôle of discrepancy in numerical integration. The best-known and simplest of these is the Koksma inequality. To see what it means, let us first describe the notion of the *variation* of a function in one dimension. Consider an arbitrary partition of the unit interval:

$$\mathcal{P} : (0 \equiv z_0 < z_1 < z_2 < z_3 < \cdots < z_{m-1} < z_m \equiv 1) \ ,$$

where m can take any positive integer value, and form the sum of the absolute jumps in the function if we step from one partition point to the next:

$$W(\mathcal{P}) = \sum_{j=1}^{m} |f(z_j) - f(z_{j-1})| \ . \tag{67}$$

The supremum of this over all possible partitions, including the values of m, is called the variation in the sense of Vitali:

$$\mathrm{Var}[f] = \sup_{\mathcal{P}} \ W(\mathcal{P}) \ . \tag{68}$$

Consider the estimate of the integral of f with *any* given point set X consisting of points x_1,x_2,...,x_N. The Koksma inequality then reads that the integration error has a strict upper bound:

$$\left| \frac{1}{N} \sum_{j=1}^{N} f(x_j) - \int_0^1 dx \ f(x) \right| \leq \mathrm{Var}[f] \ \times \ L_\infty^*(X) \ . \tag{69}$$

The multidimensional generalization of this theorem, the so-called Koksma-Hlawka inequality, can simply be imagined. For instance, in two dimensions the partition points are given by a grid of $m_1 \times m_2$ points, with

$$(\vec{z}_{j_1,j_2})^1 = z_{j_1} \ , \ (\vec{z}_{j_1,j_2})^2 = z'_{j_2} \ ,$$

[6] Apparently!

and the sum runs over absolute values of

$$f(z_{j_1,j_2}) - f(z_{j_1-1,j_2}) - f(z_{j_1,j_2-1}) + f(z_{j_1-1,j_2-1}) \quad .$$

In higher dimensions we have analogous definitions. The appropriate definition of the variation of the integrand f is then obtained by taking its variation in the sense of Vitali in the interior of the hypercube as well as the same variation on its lower-dimensional boundaries (for $D = 3$, for example, inside the cube, on the sides, and on the edges): this is called the variation in the sense of Hardy and Krause. With this definition, the same inequality for the integration error holds. Other definitions of the 'amount of oscillation' of an integrand can be given, together with other notions of discrepancy, such that similar absolute error bounds exist.

The Koksma-Hlawka inequalities would be useful to find absolute error bounds rather than (as in Monte Carlo) probabilistic ones, but for a few reasons. In the first place, finding the variation of an integrand is much harder than finding its integral: in fact, if we could find the variation we would immediately have the Riemann integral anyway. In the second place, it is easy to show that for bounded but discontinuous functions in more than one dimension the variation can be infinitely large, which makes the inequalities true but trivial.

We must conclude that the Koksma-Hlawka inequalities are useless from the point of practical computation, but they do show that the notion of non-uniformity of point sets, as embodied in discrepancy, is relevant in establishing a more manageable way of talking about uniformity.

4.4 The Woźniakowski Lemma

The *discrepancy is useful in the formulation of integration error bounds à la Koksma-Hlawka, but as a characteristic measure of non-uniformity it has drawbacks. In the first place, all the rectangles considered are cornered at the origin, thus destroying translational invariance. Indeed, if we consider the case $N = 1$ in which the point set consists of a single point $\vec{x}$, the extreme discrepancy L^*_∞ is minimal if $\prod_{\mu=1}^{D} x^\mu = 1/2$. Such 'optimal points' therefore lie on a hyperboloid $(D-1)$-dimensional hypersurface that moves in the direction of the point $(1, 1, \ldots, 1)$ as D increases; whereas one would expect single-point sets to be equally uniform wherever the point is located, or if not, an 'optimal point' in the center of the unit hypercube would seem most reasonable. Another drawback is the lack of rotational invariance; by slightly tilting a regular grid the *discrepancy can be drastically altered.

The search for more useful notions of discrepancy is helped by the following insight, first formulated in [7]. Consider the Wiener measure that assigns probability densities to functions. It is defined for functions of a single variable in $(0, 1)$ to be a Gaussian measure such that

$$f(0) = 0 \quad , \quad \langle f(x) \rangle = 0 \quad , \quad \langle f'(x)\, f'(y) \rangle = \delta\,(x - y) \quad , \tag{70}$$

where the average is over the whole class of functions. A typical function $f(x)$ drawn from this distribution describes Brownian motion, where $f(x)$ is the displacement at time x. A generalization to D dimensions is the Wiener sheet measure, defined by

$$f(\vec{0}) = 0 \quad , \quad \langle f(x) \rangle = 0 \quad , \quad \left\langle f^{(D)}(\vec{x}) f^{(D)}(\vec{y}) \right\rangle = \delta^D (\vec{x} - \vec{y}) \quad ,$$
$$f^{(D)}(\vec{x}) \equiv \frac{\partial^D}{\partial x_1 \partial x_2 \cdots \partial x_D} f(\vec{x}) \ . \tag{71}$$

Let us estimate the integral of a function $f(\vec{x})$ drawn from the Wiener sheet measure using a given point set $X = (\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N)$. The integral has zero expectation value since the measure is Gaussian; its numerical estimate will have an error which we denote by η. Taking the average of η^2 over the Wiener (sheet) measure, we arrive[7] at the Wózniakowski Lemma:

$$\left\langle \eta^2 \right\rangle = L_2^*(X) \ . \tag{72}$$

We see that the discrepancy measures how well the given point set integrates an average integrand drawn from a certain class of integrands: it is called the *average-case complexity*. Every function class for which we can describe the probability density is seen to lead to its own notion of discrepancy, and this we shall employ in the following.

4.5 Diaphonies

The Wiener sheet measure is mathematically interesting, but not well suited to typical integration problems encountered in particle phenomenology. This is most clear in the one-dimensional case: a typical integrand in particle physics does *not* look like Brownian motion, in which $f(x)$ is nowhere differentiable[8]. The *discrepancy may therefore be comparatively useless. A possibly better class of integrands can be constructed as follows. Consider a D-dimensional integrand that has a Fourier decomposition:

$$f(\vec{x}) = \sum_{\vec{n} \neq \vec{0}} \alpha_{\vec{n}} \exp(2i\pi \vec{n} \cdot \vec{x}) \ , \tag{73}$$

where the vectors $\vec{n} = (n_1, n_2, \ldots, n_D)$ have integer components, and we discard the zero mode $\vec{0}$ since it represents the constant part of $f(\vec{x})$, which is integrated with zero error anyway. For simplicity, we assume all the $\alpha_{\vec{n}}$ to be real[9]. Let us describe the class of functions by the probability distributions of the modes:

$$\mathcal{D}(f) = \prod_{\vec{n} \neq \vec{0}} d\alpha_{\vec{n}} \ \frac{1}{\sqrt{2\pi\sigma_{\vec{n}}^2}} \exp\left(-\frac{\alpha_{\vec{n}}}{2\sigma_{\vec{n}}^2} \right) \ , \tag{74}$$

[7] Under a slight redefinition of the measure, such that the function is not fixed at $\vec{0}$ but at the opposite point $(1, 1, \ldots, 1)$.

[8] The Woźniakowski Lemma was formulated in the context of financial mathematics; and stock markets *do* behave in a Brownian-motion-like way.

[9] The class of functions is a bit larger than necessary, since not every $f(x)$ is real: however, for Gaussian measures with zero mean the *average* integrand is real.

We find the expectation value

$$\langle \alpha_{\vec{n}} \alpha_{\vec{n}'} \rangle = \int \mathcal{D}(f) \alpha_{\vec{n}} \alpha_{\vec{n}'} = \sigma_{\vec{n}}^2 \theta(\vec{n} = \vec{n}') \ . \tag{75}$$

We may call the positive number $\sigma_{\vec{n}}^2$ the *strength* of the Fourier mode $\vec{n}$. All integrands in this class have zero integral, and the integration error made by using a given point set $X = (\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N)$ is therefore

$$\eta = \frac{1}{N} \sum_{\vec{n} \neq \vec{0}} \alpha_{\vec{n}} \sum_{j=1}^{N} \exp(2i\pi \vec{n} \cdot \vec{x}_j) \ . \tag{76}$$

The squared absolute value of the integration error, averaged over this ensemble of functions is

$$\left\langle |\eta|^2 \right\rangle = \int \mathcal{D}(f) \left| \frac{1}{N} \sum_{j=1}^{N} f(\vec{x}_j) \right|^2 = \frac{1}{N^2} \sum_{\vec{n} \neq \vec{0}} \sigma_{\vec{n}}^2 \sum_{j,k=1}^{N} \exp\left(2i\pi \vec{n} \cdot (\vec{x}_j - \vec{x}_k)\right) \ . \tag{77}$$

Taking out a factor $1/N$ which is expected for the squared error for truly random points, we arrive at the following notion for discrepancy, which is called a *diaphony*:

$$T(X) = \frac{1}{N} \sum_{j,k=1}^{N} \beta(\vec{x}_j - \vec{x}_k) \quad , \quad \beta(\vec{x}) = \sum_{\vec{n} \neq \vec{0}} \sigma_{\vec{n}}^2 \exp(2i\pi \vec{n} \cdot \vec{x}) \ . \tag{78}$$

By construction, the diaphony is invariant under translations $\bmod 1$ of the point set, an attractive and reasonable property. The quadratic nature evinces itself in the sum over *pairs* of points. Further properties of the diaphony depend on the distribution of the strengths $\sigma_{\vec{n}}^2$.

4.6 Examples of diaphonies

By its nature, the function $\beta(\vec{x})$ integrates to zero. For random point sets, therefore, the average diaphony is

$$\langle T(X) \rangle = \frac{1}{N} \sum_{j,k=1}^{N} \int d\vec{x}_j \ d\vec{x}_k \ \beta(\vec{x}_j - \vec{x}_k) = \beta(\vec{0}) = \sum_{\vec{n} \neq \vec{0}} \sigma_{\vec{n}}^2 \ . \tag{79}$$

For purposes of standardization, it is useful to choose the strengths so that the expectation value of the diaphony is one:

$$\sum_{\vec{n} \neq \vec{0}} \sigma_{\vec{n}}^2 = 1 \ . \tag{80}$$

In one dimension, a mathematically simple choice of the strengths is

$$\sigma_n^2 = \frac{3}{\pi^2} \ \frac{1}{n^2} \quad , \quad n \neq 0 \ . \tag{81}$$

The corresponding function $\beta(x)$ reads

$$\beta(x) = \sum_{n\geq 1} \frac{6}{\pi^2 n^2} \cos(2\pi n x) = 1 - 6\{x\}(1-\{x\}) \quad , \quad \{x\} = x \bmod 1 \ . \tag{82}$$

The most straightforward generalization to D dimensions is

$$\sigma^2_{\vec{n}} = \prod_{\mu=1}^{D} c_D \tau(n^\mu) \quad , \quad \tau(n) = \max(1,n)^{-2} \quad , \quad c_D = \left(\left(1 + \frac{\pi^2}{3} \right)^D - 1 \right)^{-1} \ , \tag{83}$$

and the corresponding β is

$$\beta(\vec{x}) = c_D \ \left\{ -1 + \prod_{\mu=1}^{D} \left(1 + \frac{\pi^2}{3} \left(1 - 6\{x^\mu\}(1-\{x^\mu\}) \right) \right) \right\} \ . \tag{84}$$

This diaphony has the nice property that it can be evaluated in a simple manner; but a drawback is its lack of even approximate rotational invariance. For instance, consider two Fourier modes in $D = k^2$ dimensions, one with wave vector $\vec{n}_1 = (1,1,\ldots,1)$, and the other one with wave vector $\vec{n}_2 = (k,0,0,\ldots,0)$. These two modes have the same wavelength, since $|\vec{n}_1|^2 = |\vec{n}_2|^2 = k^2$, but their strengths are $\sigma^2_{\vec{n}_1} = 1$ and $\sigma^2_{\vec{n}_2} = 1/k^2$, respectively, so that the above diaphony is much more sensitive to $\vec{n}_1$ than to $\vec{n}_2$.

In order to keep translational invariance and introduce as much of rotational invariance as possible, the strengths should depend on $|\vec{n}|^2$ alone. We may therefore choose the strengths to be

$$\sigma^2_{\vec{n}} = \frac{1}{K(v)} \exp(-v\vec{n}^2) \quad \forall\, \vec{n} \quad , \quad K(v) = -1 + \left(\sum_{n=-\infty}^{\infty} e^{-vn^2} \right)^D \ , \tag{85}$$

with $v > 0$ a real parameter: as v decreases, more and more modes are important for the diaphony. We may call this choice the *Jacobi diaphony*. The β function is in this case given by

$$\beta(\vec{x}) = \frac{1}{K(v)} \left\{ -1 + \prod_{\mu=1}^{D} \phi(v, x^\mu) \right\} \ , \tag{86}$$

where

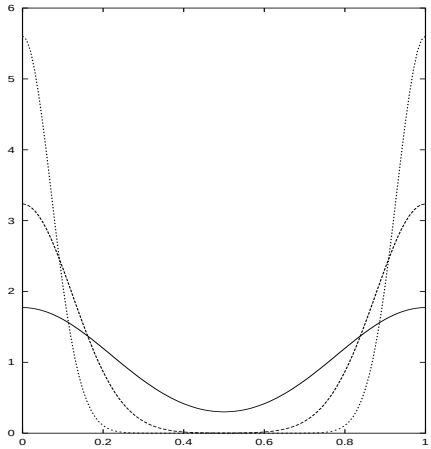
$$\phi(v,x) = \sum_n \exp(-vn^2 + 2i\pi n x) = \sqrt{\frac{\pi}{v}} \sum_n \exp\left(-\frac{\pi^2}{v}(n+x)^2 \right) \tag{87}$$

is an example of a Jacobi theta function, hence our name for the diaphony. The two alternative forms of $\phi(v,x)$, related by Poisson summation, are alternatively appropriate for large and for small values of v, respectively; the sums typically

converge well with only a few terms (note that e^{-16} is already as small as 10^{-7}). The normalization factor is

$$K(v) = -1 + \phi(v;0)^D \quad . \tag{88}$$

The evaluation of this diaphony is, therefore, actually not much more complicated than that of the previous one.



The function $\phi(v,x)$ as a function of x for $v = 0.1, 0.3, 1$. For decreasing values of v, the function (that always integrates to unity), becomes more and more peaked at $\{x\} \approx 0$: in the limit $v \to 0$ we actually have $\phi(0,x) = (\delta(x) + \delta(1-x))/2$.

Other choices are of course also possible. For instance, if for some reason modes with $|\vec{n}| \sim m$ are especially relevant, one might choose $\sigma_{\vec{n}}^2$ to be proportional to $|\vec{n}|^{m-1}\exp(-v|\vec{n}|^2)$, so that the function ϕ can then be written (at least for even m) as a derivative of a Jacobi theta function.

4.7 Assessing point sets

The discrepancy or diaphony can be used as a test of whether a given point set 'looks random'. For diaphonies as the ones discussed above, with expectation value 1 for truly random point sets, the observed diaphony should therefore be 'close to' 1 – but how close? In order to answer this question, we may consider not only the expectation value of T but also its second moment:

$$\left\langle T(X)^2 \right\rangle = \frac{1}{N^2}\left\langle \sum_{j_{1,2,3,4}=1}^{N} \beta(\vec{x}_{j_1} - \vec{x}_{j_2})\beta(\vec{x}_{j_3} - \vec{x}_{j_4}) \right\rangle \quad , \tag{89}$$

from which we see that the indices $j_{1,2,3,4}$ only contribute if they are pairwise equal. The result is

$$\begin{aligned} \left\langle T(X)^2 \right\rangle &= \frac{1}{N^2}\left(\sum_j \int d\vec{x}\; \beta(\vec{0})^2 + 2\sum_{j_1 \neq j_2} \int d\vec{x}_1\; d\vec{x}_2\; \beta(\vec{x}_1 - \vec{x}_2)^2 \right) \\ &= \frac{1}{N}\beta(\vec{0})^2 + 2\frac{N-1}{N^2}\int \beta(\vec{x})^2 \quad , \end{aligned} \tag{90}$$

so that the variance of the T distribution is given by

$$\sigma\left(T(X)\right)^2 = 2\int d\vec{x}\;\beta(\vec{x})^2 + \mathcal{O}\left(\frac{1}{N}\right) \approx 2\sum_{\vec{n}\neq\vec{0}} \sigma_{\vec{n}}^4 \quad , \tag{91}$$

where the last result holds for large N. In fact, we can prove more: in the limit of large N, the moment-generating function of the $T(X)$ distribution for truly random points is given by

$$\langle \exp(zT(X)) \rangle = \exp(\psi(z)) \quad , \quad \psi(z) = \sum_{m\geq 1} \frac{(2z)^m}{2m} \sum_{\vec{n}\neq\vec{0}} \sigma_{\vec{n}}^{2m} \quad . \tag{92}$$

From this we may construct the actual $T(X)$ distribution $H(t)$, the probability density for $T(X)$ to have the value $t > 0$, by hook, crook or saddle point method, using the inverse Laplace transform:

$$H(t) = \frac{1}{2i\pi} \int_{-i\infty}^{+i\infty} dz \exp(-zt + \psi(z)) \quad , \tag{93}$$

where the integral runs to the left of any singularity. The saddle-point approximation can be formulated as follows: choosing any real value z^* between $-\infty$ and the first singularity of $\psi(z)$ on the positive real axis, compute

$$t^* = \psi'(z^*) \quad . \tag{94}$$

The saddle-point approximation is then given by

$$H(t^*) \approx \frac{1}{\sqrt{2\pi\psi''(z^*)}} \exp\left(\psi(z^*) - z^* t^*\right) \quad . \tag{95}$$

Whether the saddle-point approximation is justified depends on the situation. For z^* close to the singularity, we obtain the high-t tail of $H(t)$; for $z^* \to -\infty$ we obtain the behaviour close to $t = 0$; finally, choosing $z^* = 0$ gives the approximate height of the peak at $t = \langle T \rangle$. Also the $1/N$ correction terms can be systematically computed to any desired order, using the methods of quantum field theory. In practice, it may be sufficient to just use the variance and obtain Chebyshev confidence levels to judge whether a point set is acceptably random. In addition, it can be proven that for large dimensionality the $T(X)$ distribution for random point sets approaches a Gaussian, so we may even employ the stricter Gaussian confidence limits[10].

As an application, we discuss an evaluation of the Jacobi diaphony for three different generators: **(a)** the `RCARRY` algorithm in the form recommended by

[10]The Gaussian limit in this case does *not* arrive from the limit where the number of points becomes large, but rather from the fact that there are usually many $\sigma_{\vec{n}}^2$ with the same magnitude: this is the limit of *large number of degrees of freedom*.

Luescher; **(b)** an admittedly poor linear congruential method with $m = 2^{10} + 9$, $a = 2^7 + 3$, and $c = 1$ (which has maximal period), and **(c)** the three-dimensional `CORPUT` generator (to be discussed later). In all three cases we constructed 3-dimensional vectors $\vec{x}$ and computed the Jacobi diaphony using $v = 0.1$. For truly random point the expectation value of the diaphony is unity, with variance equal to 0.0034. The results are given below.

generator	T	# of st.dev.
`RCARRY`	1.086	1.36
lin.congr.	1.89	14.0
`CORPUT`	0.245	11.9

From the point of view of this test, the `RCARRY` appears satisfactory. The linear congruential generator is much too non-uniform, which can be ascribed to the abovementioned lattice structure and its small period of only 1033. The `CORPUT` generator, on the other hand, produces points that are spread out much more evenly than expected. The last two generators would therefore be rejected as models of random number streams. Experience with the Jacobi diaphony is still limited. It appears to become more discriminatory for larger values of N. If a generator shows a lattice structure with a moderate number k of hyperplanes, $k = 20$, say, the Jacobi diaphony will become sensitive to it only if v is of the order of $1/k^2$, and then there will be roughly k^D modes with about the same or smaller wavevector, so that the 'dangerous' mode has to compete with many others to be picked out.

5 Quasi-random number generators

5.1 Low-discrepancy point sets

A message we may receive from the Koksma-Hlawka inequalities is that the lower the discrepancy of the point set is, the more accurate the integral. If the only aim is the accurate computation of an integral[1], we may forego randomness completely, and simply strive for point sets with very low discrepancy. This field goes under the name of Quasi-Monte Carlo, a bit inappropriately since we have now dropped the mask and carry nonrandomness as a banner; and the corresponding point sets are called quasi-random point sets.

In finding quasi-random point sets there are two issues to be confronted. In the first place, the number of points used. Suppose that we have a magical way of finding for given N, in some dimension, the set of points with the very lowest discrepancy/diaphony possible. In one dimension, this set is known: it consists of the points $x_j = (2j-1)/2N$, $j = 1, 2, \ldots, N$. In higher dimensions, it is not known how to find the optimal point set in any simple manner (for more, see below). Now suppose that, having computed the integral using N points, we are still not satisfied with the answer, and want to use $N + p$ points. It is clear that the optimal point set with $N + p$ points will, in general, *not* contain the previously used points unless $p = N$, and our effort has consequently been wasted. There is therefore a great difference between fixed-size point sets and potentially infinite streams of quasirandom numbers in which the discrepancy is low in some average sense, or at least is low as often as we can make it. The second point to keep in mind is that quasi-random points are *not* independent of one another, and we have to consider again the way in which we derived the Monte Carlo error estimate discussed in chapter 2. This we shall do at the end of this section.

5.2 Finite point sets: Korobov sets

We first consider finite point sets. A common way of obtaining low discrepancy is the so-called *method of good lattice points*, which is as follows. Take a D-dimensional vector $\vec{g}$ with integer components. The set of N points is then defined by

$$x_j^\mu = \left(\frac{j}{N} g^\mu \right) \mod 1 \ , \ \mu = 1, 2, \ldots, N \ . \tag{96}$$

The discrepancy depends of course crucially on $\vec{g}$. An obvious requirement is that each g^μ be relatively prime to N. The resulting sets are called Korobov sets. An example applicable to $D = 2$ is based on the Fibonacci numbers:

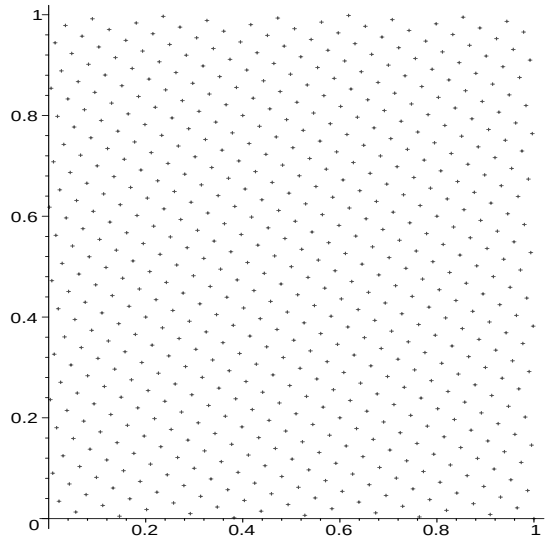
$$F_1 = F_2 = 1 \quad , \quad F_n = F_{n-1} + F_{n-2} \ , \ n \geq 3 \ . \tag{97}$$

[1] This is not obviously the case, *e.g.* in the case of Monte Carlo simulations in particle phenomenology, where fluctuations are often as important as the result itself: for instance, in the development of hadronic jets.

Each pair of F_n and F_{n-1} is relatively prime, so that we may take

$$\vec{g} = (1, F_{n-1}) \quad , \quad N = F_n \quad . \tag{98}$$

The result for $n = 15$ is depicted below.



The Fibonacci point set for $n = 15$, which contains 610 points. The low value of L^* can be ascribed to the tilting of the regular trapezoidal grid, so that no two points are precisely aligned horizontally or vertically. The regularity of the grid, however, makes it dangerous for integrands with a similar pattern, and can be picked out by other measures of nonuniformity such as T.

5.3 Infinite streams: Richtmeyer sequences

A drawback of Korobov sets is, as mentioned, the finiteness of the number of different points. In one dimension, the sequence $x_j = jm/N \mod 1$ will start to repeat after N points, and less if m and N contain common prime factors. If we replace the factor m/N by an *irrational* number, an infinite stream is possible. In fact, the situation is even better, since it can be proven that for every irrational number θ, the sequence defined by

$$x_j = (j\theta) \mod 1 \quad , \quad j = 1, 2, \ldots, N \quad , \tag{99}$$

is equidistributed, *i.e.* the discrepancy goes to zero as N approaches infinity. The speed with which this happens depends, of course, on the number-theoretical properties of the irrational number θ. Some insight can be gleaned from the *continued fraction representation*. In this representation, we write, for $0 < \theta < 1$,

$$\theta = \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{a_4 + \cfrac{1}{\ddots}}}}} \quad , \tag{100}$$

where the a's are positive integers, and are called the continued fraction coefficients.

The infinite set $a_1, a_2, a_3, \ldots$ is completely equivalent to a unique θ. Now, if a_k, say, happens to be infinite, the continued fraction actually ends at a_{k-1},

and θ is then rational rather than irrational. The sequence (99) will then start to repeat after a given number of points, and the discrepancy does not decrease any further. We may therefore conjecture (and it can be proven) that the discrepancy is lower if the coefficients are smaller, and θ 'more irrational'. The 'most irrational' number is the one with $a_j = 1$ for all j, and this is the golden mean $(-1 + \sqrt{5})/2$. The following results are easily proven: if all a_j are equal:

$$a_j = a \ , \ \ j = 1, 2, 3, \ldots \quad \Rightarrow \quad \theta = \frac{1}{2}\left(-a + \sqrt{4 + a^2}\right) \ , \tag{101}$$

and if the a_j form a periodic pattern with period p after some j_0:

$$a_{j+p} = a_j \ , \ \ j \geq j_0 \quad \Rightarrow \quad \theta = \frac{1}{2}\left(-A + \sqrt{B}\right) \tag{102}$$

for some integers A en B[2]. Below we give the continued fraction coefficients for some simple numbers.

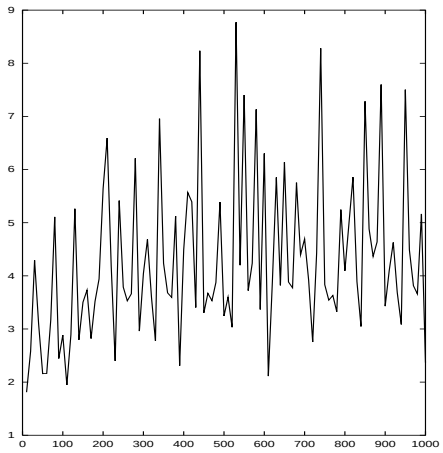
θ	$a_1, a_2, a_3, a_4, \ldots$
$\sqrt{2} - 1$	$2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, \ldots$
$\sqrt{3} - 1$	$1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, \ldots$
$\sqrt{5} - 2$	$4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, \ldots$
$\sqrt{6} - 2$	$2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, \ldots$
$\sqrt{7} - 2$	$1, 1, 1, 4, 1, 1, 1, 4, 1, 1, 1, 4, \ldots$
$\sqrt{8} - 2$	$1, 4, 1, 4, 1, 4, 1, 4, 1, 4, 1, 4, \ldots$
$\sqrt{10} - 3$	$6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, \ldots$
$\sqrt{11} - 3$	$3, 6, 3, 6, 3, 6, 3, 6, 3, 6, 3, 6, \ldots$
$\sqrt{12} - 3$	$2, 6, 2, 6, 2, 6, 2, 6, 2, 6, 2, 6, \ldots$
$\sqrt{2} - 1/2$	$1, 10, 1, 1, 1, 10, 1, 1, 1, 10, 1, 1, \ldots$
$\sqrt{3} - 1/2$	$4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, \ldots$
$\pi - 3$	$7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2, 1, 1, 2, \ldots$
$\pi - 5/2$	$1, 1, 1, 3, 1, 3, 4, 73, 6, 3, 3, 2, 1, 3, \ldots$
$e - 2$	$1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, 1, 1, 10, 1, \ldots$

We see that the irrational (and even transcedental) number π is actually quite close to the rational number 355/113, and a sequence based on $\pi \mod 1$ does not have a very low discrepancy. Note also the interesting pattern for $e \mod 1$.

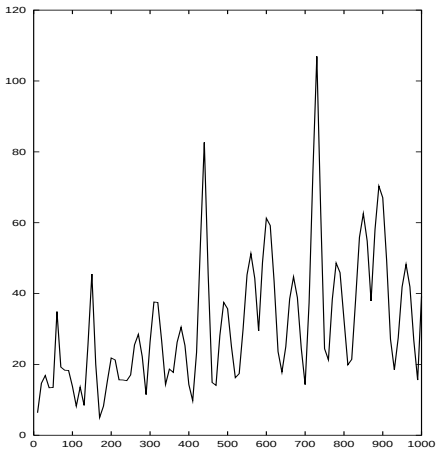
In the plots below we display the quadratic *discrepancy of some typical sequences. We have generated sets of 1000 points each, and computed L_2^* for $N = 10, 20, 30, \ldots, 1000$. The minimal L_2^* for a set of N points is that of the set defined by $x_j = (2j - 1)/N$, and is equal to $1/(12N^2)$; therefore we have multiplied the result by $12N^2$. The discrepancy for the golden ratio behaves

[2]This has an interesting corollary. Suppose θ is an irrational number that does *not* satisfy a quadratic equation with integer coefficients. In that case, its partial fraction coefficients form *an aperiodic sequence of natural numbers*. This might be considered a marvellous random number source, if it were not for the fact that (a) we usually know nothing about the range of the coefficients, and (b) we know even less about their distribution. Finally, computing a_j for very large j involves computing θ to *very* many digits.

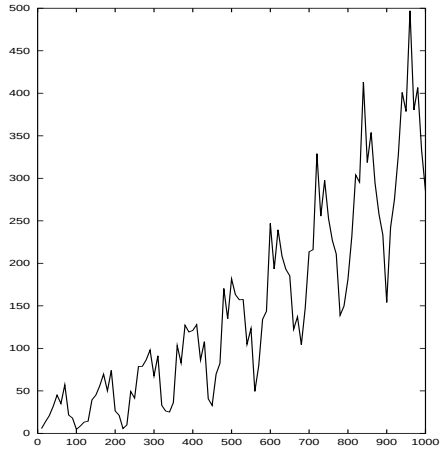
acceptably, being of the order of a few times the minimum possible. The irrational number $\sqrt{37}$ mod 1 has all its continued fraction coefficients equal to 12, with a discrepancy that is indeed some 12 times that for the golden ratio. If we choose $\theta = \pi$, the almost-rationality of π shows up in the discrepancy, which shows almost no decrease after the first 113 points: this shows up as a rising curve with a period-113 structure imposed. Finally, for $\theta = \exp(1)$, the discrepancy shows an acceptable behaviour, which we may ascribe to the fact that its continued fraction coefficients a_j remain modest up to quite large j: for N not too large, it is therefore quite acceptable.



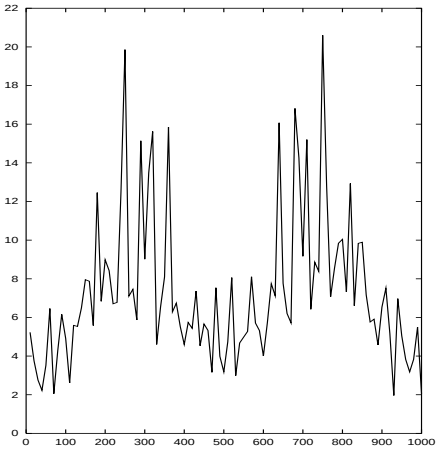
$\theta = (\sqrt{5} - 1)/2$



$\theta = \sqrt{37}$ mod 1



$\theta = \pi$ mod 1

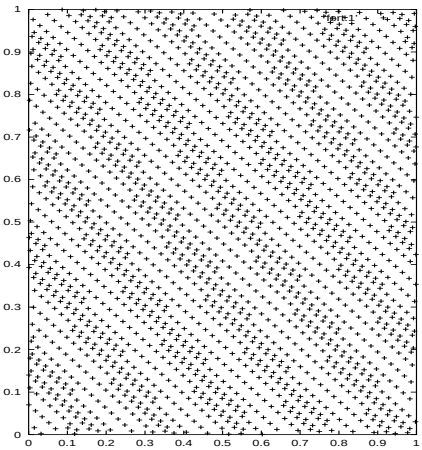


$\theta = e$ mod 1

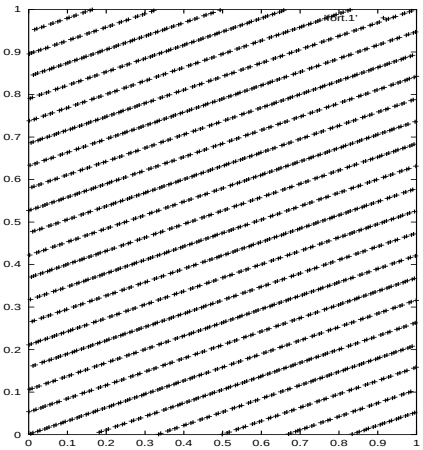
Multidimensional sequences, the so-called Richtmeyer sequences[3], may easily be constructed by choosing

$$x_j^\mu = (j\theta^\mu) \bmod 1 \ , \tag{103}$$

where θ^μ, $(\mu = 1, 2, \ldots, D)$ are irrational numbers that are all relatively prime to one another. A simple choice is to take $\theta^\mu = \sqrt{p_\mu} \bmod 1$, where p_μ is the μ^{th} prime number. Below we give some results for sets of 2000 points in two dimensions.



$\vec{\theta} = (\sqrt{2} \bmod 1, \sqrt{3} \bmod 1)$



$\vec{\theta} = (\sqrt{2} \bmod 1, \sqrt{5} \bmod 1)$

Although we are guaranteed that the eventual distribution of points will be uniform, the plots show that this may happen only for really large N. It is therefore important to choose the irrationals with great care.

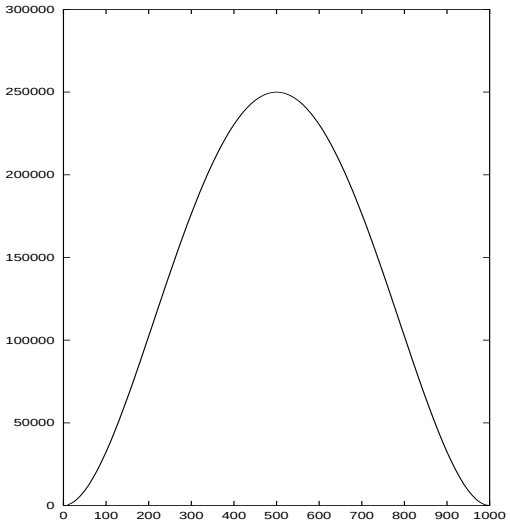
5.4 van der Corput sequences

In the previous section we discussed Richtmeyer sequences, and presented results for $N = 1000$, for one dimension, for various θ's. The 1000-point point set with the minimal discrepancy is given by

$$x_j = \frac{2j-1}{2000} \ , \quad j = 1, 2, \ldots, 1000 \ . \tag{104}$$

In the following figure we plot $12N^2 L_2^*$ for this set, where N denotes the first N points.

[3] In the mathematical literature these are also called Kronecker sequences.



The curve reads a maximum of about 250,000 in the middle. This is due to the fact that, for $N = 500$, the interval $(0, 1/2)$ contains precisely 500 points, and the interval $(1/2, 1)$ precisely none. The value for the 500-point point set with the minimal discrepancy would be 1, and this value would be obtained for $N = 500$ if we had constructed the point set by defining $x_j = ((2j - 1)/1000) \bmod 1$.

It is clear that the *order* in which the points are generated is extremely important if we want to maintain a low, or low-ish, discrepancy during the stream. Optimally, we want a sequence that returns to the optimal value, $L_2^* = 1/(12N^2)$, or thereabouts, for as many N values as possible. This can be achieved by the following algorithm. Consider a numbering system in base b: that is, given an integer $b \geq 2$, we can write any natural number n as

$$n = n_0 + n_1 b + n_2 b^2 + n_3 b^3 + \cdots + n_k b^k \quad , \quad n < b^{k+1} \quad . \tag{105}$$

The *van der Corput transform* of n in base b is then

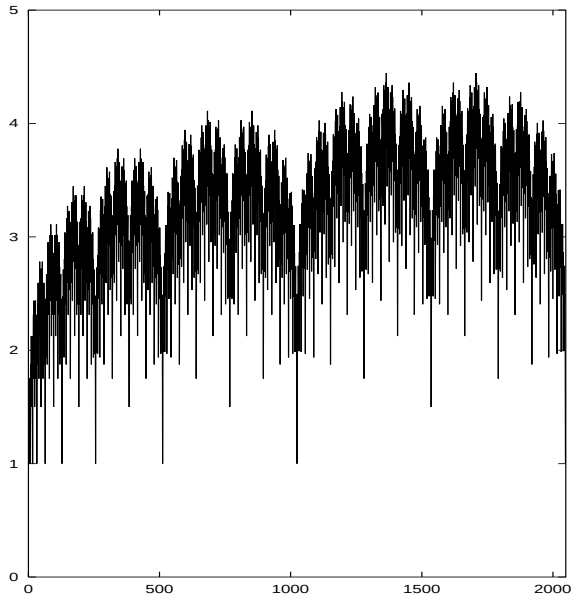
$$\phi_b(n) = n_0 b^{-1} + n_1 b^{-2} + n_2 b^{-3} + \cdots + n_k b^{-k-1} \quad . \tag{106}$$

As n runs up through the integers, the least leading bit in n, that is, the most leading bit in $\phi_b(n)$, will change most rapidly, followed by the next-to-least(most) leading bit, and so on. For $b = 2$, the first few van der Corput transforms are given here:

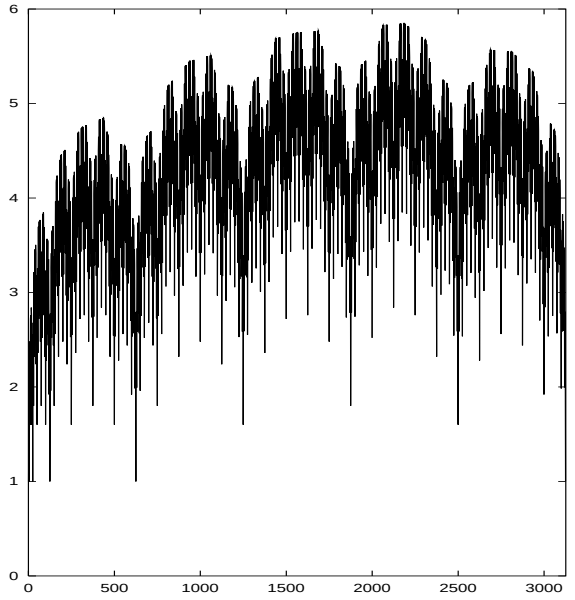
n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\phi_2(n)$	0	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{3}{4}$	$\frac{1}{8}$	$\frac{5}{8}$	$\frac{3}{8}$	$\frac{7}{8}$	$\frac{1}{16}$	$\frac{9}{16}$	$\frac{5}{16}$	$\frac{13}{16}$	$\frac{3}{16}$	$\frac{11}{16}$	$\frac{7}{16}$	$\frac{15}{16}$

These numbers have been used in the `CORPUT` generator mentioned before. This sequence attempts to fill the unit interval in as uniform a way as possible; after 2^k points the interval is filled almost optimally, with $x_j = (j - 1)/2^k$, $j = 0, 1, \ldots, 2^k - 1$. The L_2^* in that case is $1/(3N^2)$, almost optimal[4].

[4]For point sets defined by $x = (2j - \alpha)/(2N)$, $j = 1, 2, \ldots, N$ and $-1 \leq \alpha \leq 1$, the discrepancy is given by $L_2^* = (1 + 3(1 - \alpha)^2)/(12N^2)$,



The 'normalized' extreme discrepancy NL^*_∞ for the first N van der Corput numbers with base $b = 2$,for N from 1 to 2^{11}. Note the fractal structure. In fact, if we plot only the values for N multiples of 2^c, we obtain exactly the same plot as that for $N = 2^{11-c}$.



The 'normalized' extreme discrepancy NL^*_∞ for the first N van der Corput numbers with base $b = 5$,for N from 1 to 5^5. The fractal structure is in this case of a 5-fold nature.

In the van der Corput sequence with base b, the 'normalized' extreme discrepancy, $D(N) \equiv NL^*_\infty$, enjoys the property that $D(bN) = D(N)$, so it is forced to remain small. Since the powers b^k are further apart with increasing k, the value of $D(N)$ can move further and further out before having to return to its minimum 1. The envelope of the D curve therefore grows logarithmically: $L^*_\infty \propto (\log N)/N$, with a proportionality constant that grows with b. For $b = 2$,

computer experimentation shows the following:

$$\begin{aligned} D(2n) &= D(n) \ , \\ D(2n+1) &= \frac{1}{2}\left(D(n) + D(n+1) + 1\right) \ , \end{aligned} \tag{107}$$

and the envelope is described by the points $N = K(m)$ with[5]

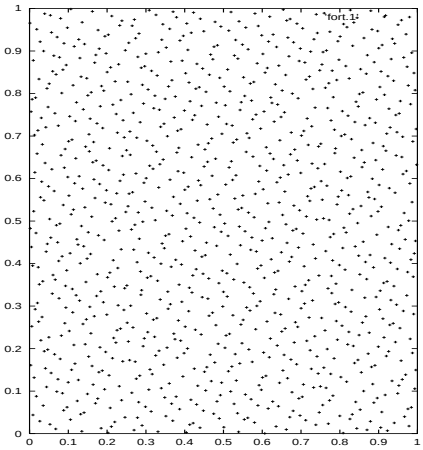
$$\begin{aligned} K(m) &= \frac{1}{3}\left(2^{m+1} + (-)^m\right) \ , \\ D(K(m)) &= \frac{2^{1-m}}{18}\left((3m+7)2^m + 2(-)^m\right) \ . \end{aligned} \tag{108}$$

The form of $D(n)$ is therefore indeed bounded by a constant times $\log(n)$.

In more dimensions, we may define the vector sequence $\vec{x}_j$ by choosing a vector $\vec{b}$ with natural components, and defining

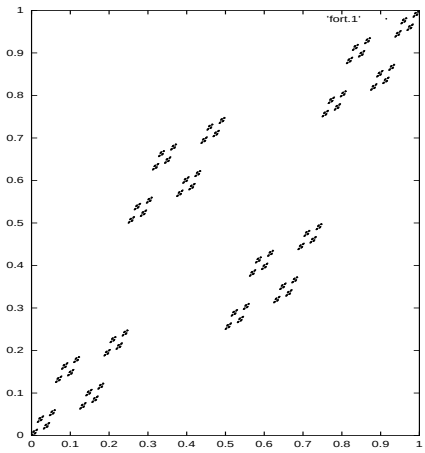
$$x_j^\mu = \phi_{b^\mu}(j) \ , \tag{109}$$

where the bases b^μ, $\mu = 1, 2, \ldots, D$ are all relatively prime.

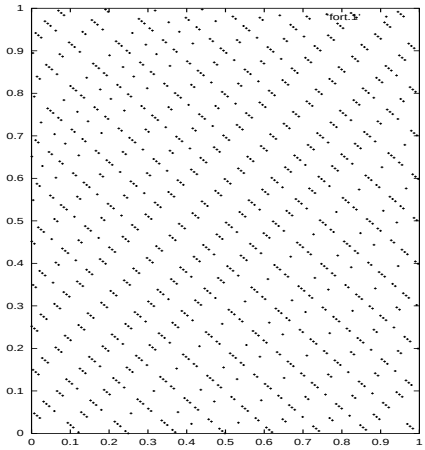


The van der Corput points $(\phi_2(n), \phi_3(n))$ for $1 \leq n \leq 1000$.

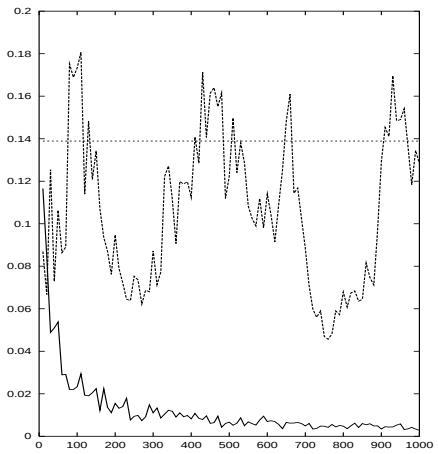
[5] These forms are the solutions of the easily observed recursion relations $K(m) = 2K(m-1) + (-)^m$, $m \geq 2$ and $D(K(m)) = d(m)/2^{m-1}$ with $d(m) = 2d(m-1) + K(m-2)$, with initial values $K(10 = 1$, $d(1) = 1$, and $d(2) = 3$.



The van der Corput points $(\phi_2(n), \phi_4(n))$ for $1 \leq n \leq 1000$. This plot shows why the bases of the axes should be relatively prime to one another.



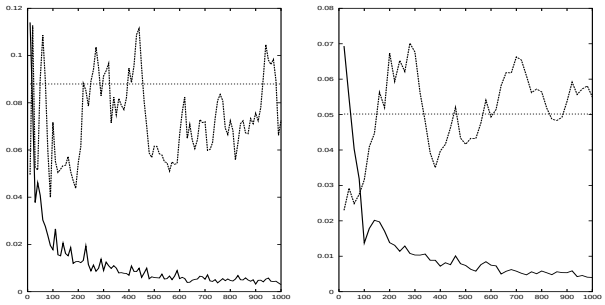
The van der Corput points $(\phi_{17}(n), \phi_{19}(n)$ for $1 \leq n \leq 1000$. The onset of uniformity is seen to be quite slow in this case, due to the fact that **(a)** the primes 17 and 19 are 'large' for $N = 1000$ ($1000 \sim 2 \times 19^2$), and **(b)** they are close to one another.



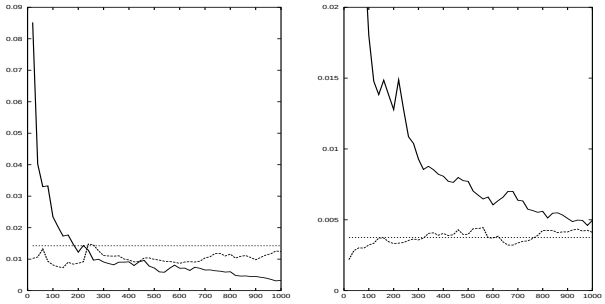
Here we give the normalized discrepancy NL_2^* as a function of N, for the two-dimensional van der Corput sequence with $\vec{b} = (2, 3)$ (lower curve), and for the `RCARRY` algorithm (upper curve). The straight line at $2^{-2} - 3^{-2} = 5/36$ is the expected value of NL_2^* for truly random points. The NL_2^* for the pseudorandom set from `RCARRY` shows appreciable fluctuations around the expected value, while that for the quasirandom set from `CORPUT` is steadily decreasing.

Since we must require that the components of the basis vector $\vec{b}$ be all relatively prime, we can draw a few inferences. In the first place, since no integer

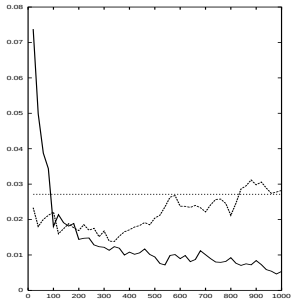
can be equal to both a power of b and a power of b' if b and b' are relatively prime, the 'return' phenomenon so conspicuous in one dimension is absent, thus allowing for faster growth of the normalized discrepancy. In the second place, we would like the b^μ to be quite small, and it is natural to choose them to be the smallest prime numbers. For $D = 8$, we must therefore already use 17 and 19 in the base. In addition, the known upper bounds on the discrepancy contain an overall factor $\prod_{\mu=1}^{D} b^\mu$, which therefore grows faster than $(D!)$, and this superexponential growth makes the exact upper bound useless as a guarantor of low discrepancy for given N in high dimensions.



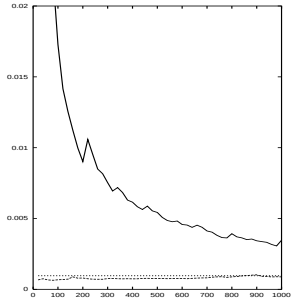
$D = 3$ $D = 4$



$D = 6$ $D = 8$



$D = 5$



$D = 10$

The above plots show the evolution of the N-dependence of NL_2^* with increasing dimension D. In all cases, the van der Corput results[6] show an initial steep decrease with N, while the `RCARRY` values are close to the expected one. For D 8 or larger, the van der Corput set is actually *less* uniform than the pseudorandom one for N not extremely large, thus bearing out the conjecture made above.

5.5 Nierreiter sequences

The problem mentioned at the end of the last section is circumvented by the so-called Niederreiter sequences. In these sequences one may use $b = 2$ for all coordinates: we have

$$\vec{x}_j = \left(\phi_2(p_1(j)) \, , \, \phi_2(p_2(j)) \, , \ldots , \, \phi_2(p_D(j)) \right) \quad , \quad j = 1, 2, 3, \ldots \quad , \tag{110}$$

where the functions $p_\mu(j)$ are cleverly chosen permutations. They have the property that if j runs from 1 to 2^k, the values of $p_\mu(j)$ are also restricted to the same range, for all k. The 'return' phenomenon will therefore be much more effective, and it can be shown that the discrepancy bound *decreases* super-exponentially with increasing D.

[6] The bases used are the choices recommended in the text: coordinate x_j^μ is generated with base $b = p_\mu$, the μ^{th} prime number.

5.6 Discrepancy minimization

Another possible strategy to arrive at low-discrepancy point sets is by explicit minimization of the discrepancy/diaphony. Since the discrepancy/diaphony depends in a complicated way on the point set, we expect that its shape (viewed as an ND-dimensional landscape) has many local minima. The most practical way to tackle this problem is by using the Metropolis algorithm. That is, consider a point set X consisting of N points. For this point set, compute the diaphony $T(X)$, say. Then, make a small random change in the point set, for instance by changing the position of a single point. This changed point set, X', is a candidate for an updated point set. Compute also $T(X')$, and determine

$$P(X \to X') = \exp\left(-\frac{T(X') - T(X)}{t}\right) \quad . \tag{111}$$

If $T(X') < T(X)$, P will be larger than 1; the candidate point set has the lower diaphony, and we adopt it. If $T(X') > T(X)$, the candidate point set has higher diaphony, but we may still accept it, with probability[7] given by P. This seeming deterioration allows the point set to move away from a local minimum in the diaphony, hopefully to settle later on in a deeper minimum somewhere else. The number t plays the rôle of temperature: if $t \approx 0$, X' is very unlikely to be accepted if it leads to a diaphony increase, whereas when $t \to \infty$ any change will be accepted regardless of any increase in diaphony. The method of *simulated annealing* consists of starting at high temperature and gradually lowering it, allowing the point set to settle gently in a low-diaphony situation.

Simulated annealing has proven its worth for problems like the travelling-salesman problem, in which the shortest route connecting a given set of fixed points has to be found. If there are m points, there are of the order of $\Gamma(m)$ possible routes, so that an explicit search through all possibilities quickly becomes prohibitively time-consuming. A similar situation arises in the search for optimal wiring of microprocessors. Discrepancy minimization is again much harder since the possible changes are continuous rather than discrete. For small N, of the order of a few tens, one can observe a lowering of the discrepancy, but for large N, of the order of tens of thousands, the computation of quadratic measures of nonuniformity, taking time of $\mathcal{O}\left(N^2\right)$ becomes itself time-consuming[8]. At the end of the day, we are anyway left with a fixed-size point set, with its own drawbacks as discussed above.

5.7 Error estimates revisited

In standard Monte Carlo, the expectation value of the squared error decreases as $1/N$ under the assumption that the points are iid. For Quasi-Monte Carlo, this is explicitly *not* true: indeed, the various points 'know' about each other's

[7] That is, generate a uniform random number r in $(0, 1)$, and accept X' if $r < P$, otherwise stick to X.

[8] If we change just one point at a time, the *difference* $T(X') - T(X)$ may be computed in time $\mathcal{O}(N)$.

position. For simplicity, let us restrict ourselves to the hypercube. The combined probability density for truly random points is, by definition,

$$P(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N) = 1 \ . \tag{112}$$

Since quasirandom points are not independent, their combined probability density[9] is different; we therefore write

$$P(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N) = 1 - \frac{1}{N} F_N(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N) \ , \tag{113}$$

where the factor $-1/N$ is a convention. Clearly, F_N must be symmetric in all its arguments. Moreover, let us define

$$F_k(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_k) = \int_0^1 d\vec{x}_{k+1} d\vec{x}_{k+2} \cdots d\vec{x}_N F_N(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N) \tag{114}$$

for $k < N$. If the point set is to be of any use at all, we must have $F_1(\vec{x}) = 0$, otherwise the integral estimate is biased. Furthermore, we shall assume (not unreasonably) that the combined probability density of the points in the point set is translation-invariant modulo 1:

$$F_N(\vec{x}_1 + \vec{y}, \ldots, \vec{x}_N + \vec{y}) = F(\vec{x}_1, \ldots, \vec{x}_N) \ \ \forall \vec{y} \ , \tag{115}$$

where the mod 1 for each component is understood. Therefore, the combined probability density of a pair of points depends only on their difference:

$$P(\vec{x}_1, \vec{x}_2) = 1 - \frac{1}{N} F_2(\vec{x}_1 - \vec{x}_2) \ . \tag{116}$$

We now redo the calculations of section 2. The integral estimator is,again,

$$E_1 = \frac{1}{N} \sum_{j=1}^{N} f(\vec{x}_j) \ , \tag{117}$$

and we still find $\langle E_1 \rangle = J_1$ as required. For the expectation value of its square, we now find

$$\begin{aligned} \left\langle E_1^2 \right\rangle &= \frac{1}{N^2} \left(N J_2 + N^{\underline{2}} J_1^2 - \frac{N^{\underline{2}}}{N} \int f f F_2 \right) \ , \\ \int f f F_2 &\equiv \int d\vec{x}_1 \ d\vec{x}_2 \ f(\vec{x}_1) \ f(\vec{x}_2) \ F_2(\vec{x}_1 - \vec{x}_2) \ . \end{aligned} \tag{118}$$

The squared error therefore has the expectation value

$$\sigma \left(E_1 \right)^2 = \frac{1}{N} \left(J_1 - J_2^2 - \frac{N-1}{N} \int f f F_2 \right) \ . \tag{119}$$

[9]Strictly speaking, for a given set of quasirandom points, there is no probability density: but for large N we may assume that we approach a continuum situation.

Here, we see the error-reducing virtue of quasirandom point sets at work: if $F_2(\vec{x}) > 0$ for $\vec{x}$ 'small' in some sense, the expected squared error is smaller than for truly random points. Note, also, that a deviation from the iid property by only $\mathcal{O}(1/N)$ is already sufficient!

To actually estimate the squared error, we also have to modify E_2. It turns out that the definition that gives $\langle E_2 \rangle = \sigma(E_1)^2$ is given by

$$E_2 = \sum_{j,k=1}^{N} f(\vec{x}_j)\, f(\vec{x}_k) \left(\left(\frac{1}{N^{\underline{2}}} + \frac{F_2(\vec{0})}{N^3} \right) \delta_{j,k} - \frac{1}{NN^{\underline{2}}} - \frac{1}{N^3} F_2(\vec{x}_j - \vec{x}_k) \right) \quad . \tag{120}$$

Unfortunately, in this case we do have to perform $\mathcal{O}\left(N^2\right)$ sums. It is therefore customary, even if not justified, to employ the standard form of E_2. An alternative method is the following. Suppose that a set of 100,000 quasirandom points, say, is used in a calculation. We may then collect the weights in 100 groups of 1,000 each, and study the distribution of the 100 averages in addition to *their* average, in order to arrive at an error estimate.

Quite another issue is the question of what $F_2(\vec{x}_1 - \vec{x}_2)$ actually is. In standard Monte Carlo, we use $F_2 = 0$ by the implicit assumption that *the point set used is a 'typical' member of the set of all N-point point sets with iid uniformly distributed points.* Note that we make this assumption with blithe disregard for the fact that a pseudorandom point set is not really typical because it comes from a well-defined algorithm. In the same spirit, it seems appropriate to embody the most important property of quasirandom point sets, namely their low diaphony/discrepancy, by assuming that *the quasirandom point set with diaphony equal to t is a 'typical' member of the class of all N-point point sets with uniformly distributed points, under the constraint that their diaphony equals t.* In that case, techniques similar to that of sect.4.7 lead to

$$\begin{aligned} F_2(\vec{x}) &= -\frac{1}{2i\pi} \int_{-i\infty}^{+i\infty} dz\; R(z;x) \exp(-zt + \psi(z)) \quad , \\ R(z;x) &= \sum_{\vec{n} \neq \vec{0}} \frac{2z\sigma_{\vec{n}}^2}{1 - 2z\sigma_{\vec{n}}^2} \exp(2i\pi \vec{n} \cdot \vec{x}) \quad . \end{aligned} \tag{121}$$

Computing $F_2(\vec{x})$ in a readily evaluable form is in general *not* easy! Experience with this kind of 'improved Quasi-Monte Carlo error estimate' is nonexistent; and other assumptions on the class of point sets of which X is a 'typical' member are also possible.

6 Variance reduction

Another, and more common, approach to improving the Monte Carlo error estimate does not aim at improving the uniformity of the point set, but rather at modifying the effective integrand such that its variance $J_2 - J_1^2$ is smaller. A large number of techniques and tricks exist, and here we only describe a few.

6.1 Stratified sampling

For simplicity we shall discuss integration over the D-dimensional hypercube $\Omega \equiv (0,1)^D$. Let us dissect the hypercube in K non-overlapping parts Ω_k, $k = 1, 2, \ldots, K$. Each Ω_k has volume ω_k, so that

$$\sum_{k=1}^{K} \omega_k = 1 \quad . \tag{122}$$

The strategy of stratified sampling now consists of integrating $f(\vec{x})$ separately over the regions Ω_k, using a *predetermined* number of iid uniform points n_k inside each region. To compare with standard Monte Carlo, we therefore must have

$$\sum_{k=1}^{K} n_k = N \quad . \tag{123}$$

The probability density of the points is therefore

$$P_k(\vec{x}) = \frac{1}{\omega_k} \theta\left(\vec{x} \in \Omega_k\right) \quad . \tag{124}$$

The integral estimator is given by

$$E_s = \sum_{k=1}^{K} E(k) \quad , \quad E(k) = \frac{\omega_k}{n_k} \sum_{j=1}^{n_k} f(\vec{x}_j) \quad , \tag{125}$$

where for each k the points $\vec{x}_j$ accord to $P_k(\vec{x}_j)$. We have

$$\langle E(k) \rangle = \frac{\omega_k}{n_k} \sum_{j=1}^{n_k} \langle f(\vec{x}_j) \rangle = \omega_k \int\limits_{\Omega_k} d\vec{x}\; P_k(\vec{x})\; f(\vec{x}) = \int\limits_{\Omega_k} d\vec{x}\; f(\vec{x}) \quad , \tag{126}$$

so that, indeed, $\langle E_s \rangle = J_1$ as it should. This is just the additive property of Riemann integration. Since the points in each region are generated independently, we have

$$\sigma\left(E_s\right)^2 = \sum_{k=1}^{N} \sigma\left(E(k)\right)^2 \quad , \tag{127}$$

and

$$\sigma\left(E(k)\right)^2 = \frac{\omega_k}{n_k} J_2(k) - \frac{1}{n_k} J_1(k)^2 \quad , \quad J_m(k) = \int\limits_{\Omega_k} d\vec{x}\; f(\vec{x})^m \quad . \tag{128}$$

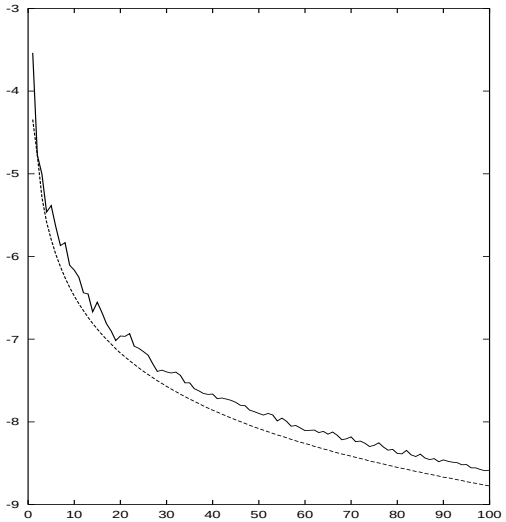
The improvement (if any) in the estimated squared error is therefore

$$\begin{aligned}\delta\eta^2 &= \sigma\left(E_1\right)^2 - \sigma\left(E_s\right)^2 \\ &= \frac{1}{N}\left(\sum_{k=1}^{K}\left(1-\frac{N\omega_k}{n_k}\right)J_2(k) + \sum_{j=1}^{K}\frac{N}{n_k}J_1(k)^2 - \left(\sum_{k=1}^{N}J_1(k)\right)^2\right) \quad (129)\end{aligned}$$

In *uniform stratified sampling*, we simply take the number of points n_k proportional to the volume ω_k: $n_k = N\omega_k$ (of course this assumes that the ω_k all have rational ratios to one another; in practice this poses no problem). In that case, the J_2 terms drop out and we have

$$\delta\eta^2 = \frac{1}{N}\sum_{k,m=1}^{K}\left(\sqrt{\frac{\omega_m}{\omega_k}}J_1(k) - \sqrt{\frac{\omega_k}{\omega_m}}J_1(m)\right)^2 \quad , \qquad (130)$$

which is nonnegative: uniform stratified sampling usually improves the error, although the improvement may be very small. The underlying reason for the error improvement is easily recongized: by imposing $n_k = N\omega_k$, we force some amount of uniformity on the point set. If we consider the regions Ω_k as 'bins' in the sense of the χ^2 test, uniform stratified sampling leads to $\chi^2 = 0$, whereas without stratification we would expect $\chi^2 \approx K - 1$.



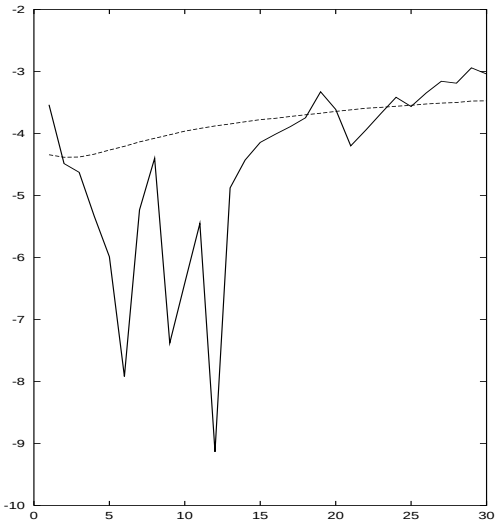
The effect of uniform stratified sampling. The integrand is $969969x^{10}(2-x)^{10}/262144$, which integrates to one on the unit interval. We plot the actual absolute error (upper curve), and its estimate (lower curve) as a function of the number K of bins, taken equal-sized. For each K, exactly the same stream from `RCARRY` has been used. The total number of points used is $K\lfloor(10^4/K)\rfloor$, *i.e.* as close to 10,000 as the binning allows.

Another approach may be called *rigid stratified sampling*: here we simply force equal numbers of points in every region, $n_k = N/K$. Then,

$$\delta\eta^2 = \frac{1}{N}\left(\sum_{k,m=1}^{K}\left(J_1(k) - J_1(m)\right)^2 + \sum_{k=1}^{K}J_2(k)(1-K\omega_k)\right) \quad . \qquad (131)$$

In this case, the error may actually increase, especially if some of the ω_k are

much larger than the other ones. An example is given below.



The results for rigid stratification, for the same integrand as above. The volumes have been chosen as $\omega_k = 2^(k - 1)/(2^K - 1)$ for each K. The number of points in each bin is still N/K. Almost all of the integral is contained in $\Omega_K \approx (1/2, 1)$, which gets fewer and fewer points as K increases; the other Ω's contribute almost nothing, and the resulting error becomes larger and larger. Needless to say, this particularly stratification is totally unsuited to this integrand.

6.2 Importance sampling

The idea of importance sampling is radically different from that of stratified sampling. It works best when the integrand, $f(x)$, is nonnegative (for instance, when it is a cross section in particle scattering). We may write

$$J_1 = \int f(x)\, dx = \int \frac{f(x)}{g(x)}\, g(x)\, dx \tag{132}$$

with a nonnegative function $g(x)$ which is nonzero where $f(x)$ is nonzero, and $\int g(x)\ dx = 1$: in other words, $g(x)$ is some probability density. We may then interpret the above truism as follows: if the points x are not distributed uniformly but rather according to the density $g(x)$, then the desired integral J_1 is the expectation value of the weight $w(x) = f(x)/g(x)$ under this new probability distribution. If we can arrange that these new weights have a smaller variation than the original weights $f(x)$, the expected squared Monte Carlo error will be reduced to

$$\frac{1}{N}\left(\int w(x)^2\, g(x)\, dx - (J_1)^2\right) = \frac{1}{N}\left(\int \frac{f(x)^2}{g(x)}\, dx - (J_1)^2\right) \tag{133}$$

Obviously, the error will vanish if $g(x)$ is strictly proportional to $f(x)$, since then the weights will be constant: but since this supposes that we can integrate $f(x)$ analytically, this ideal case is moot. Importance sampling is, in essence, the method of choice in most particle phenomenology Monte Carlo calculations. To make it work one has to

1. find an 'approximant' $g(x)$ that captures the main fluctiations of $f(x)$, in particular its peaks;

2. find an algorithm to generate random points x with the prescribed probability density $g(x)$.

The basic idea here is to generate more random points where the integrand is 'more important', hence the name. The simplicity of the method is concomitant with a great deal of 'art' in its application. In particular the generation of non-uniform sets of random numbers is a whole field in itself [8]. This will be discussed later. Note that importance sampling is not *guaranteed* to reduce the error. In particular, larger errors may arise if $g(x) \ll f(x)$ somewhere, and this should be avoided. The opposite case, $g(x) \gg f(x)$, is not so dangerous since it will simply lead to a small weight $w(x)$.

6.3 Multichanneling

A refinement of importance sampling that has become popular in the last years is that of multichanneling[9]. It is useful if the approximant $g(x)$ can be written as

$$\begin{aligned} g(x) &= \sum_{n=1}^{m} \alpha_n \, g_n(x) \quad , \quad \alpha_n > 0 \ , \ \sum_{n=1}^{m} \alpha_n = 1 \quad , \\ g_n(x) &\geq 0 \quad , \ \int g_n(x) = 1 \ . \end{aligned} \tag{134}$$

It frequently happens that there is no simple algorithm to generate x values according to $g(x)$ but there exist separate and different algorithms to generate the m densities $g_n(x)$, called the 'channels'. In that case, $g(x)$ can be generated by generating a uniform random number ρ in $[0, 1]$, finding the index k for which

$$\sum_{n=1}^{k-1} \ \leq \ \rho \ < \ \sum_{n=1}^{k} \ ,$$

and then generating x according to $g_n(x)$. As we shall see later, the end result is that the x values are then distributed according to $g(x)$. For *any* choice of the coefficients α_n, the integral estimate is unbiased, which means that we may change the α_n during the Monte Carlo run. Therefore we may employ the coefficients α_n to minimize the Monte Carlo error estimate, in other words, minimizing

$$W(\vec{\alpha}) = \int dx \, \frac{f(x)^2}{g(x)} \ . \tag{135}$$

Formally, $W(\vec{\alpha})$ is homogeneous of degree -1 in the α's. Using the Lagrange multiplier method, the optimal choice then satisfies

$$\frac{\partial}{\partial \alpha_n} \left(W(\vec{\alpha}) - \lambda \sum_{j=1}^{m} \alpha_j \right) = \frac{\partial}{\partial \alpha_n} W(\vec{\alpha}) - \lambda = 0 \ , \tag{136}$$

which means that all these derivatives are equal, and

$$-\frac{\partial}{\partial\alpha_n}W(\vec{\alpha}) = \int dx\; g_n(x)\; w(x)^2 = \sum_{j=1}^{m}\alpha_j\frac{\partial}{\partial\alpha_j}W(\vec{\alpha}) = -W(\vec{\alpha})\;\;. \qquad (137)$$

In this case, therefore, each channel contributes the same amount to the total error. Note that the partial contributions are expressed in terms of the weights and the channel densities, which are computed anyway during the Monte Carlo run, so that they can be computed rapidly. It is a simple matter to devise updating strategies for the α_n so as to approximate the optimal error. This usually consists of setting some initial values for the α's, then generating enough events so that each channel has been picked an appreciable number of times, analyzing the error contributions, and determining an 'optimized' setting of the α's. This can be iterated a number of times. The best setting can then be found among the inspected ones, and this setting is then kept for the rest of the run. Some Monte Carlo simulations have used as many as 2,000 channels. The advantage of multichanneling is that it can be performed essentially for free, while the channel densities can be optimized to describe just a part of the peaking structure of the whole integrand.

6.4 Non-uniform random number generation

The considerations about the very important method of importance sampling require us to be able to generate (one- or more-dimensional) points x according to some prescribed probability density $g(x)$. This field consists of a handful of general strategies and a host of particular tricks. Many of these can be found in [8]: here we simply describe the main strategies. In practice, it is of course easy to write an algorithm that combines, transforms or otherwise juggles random numbers so as to come up with some non-uniform random variate, and often it is much harder to ascertain what is the precise distribution of this variate. It turns out to be useful to develop a formalism to discuss the analysis of such algorithms.

6.5 Unitary-algorithm formalism

In what follows we shall assume that we have available a stream of iid uniform truly random numbers ρ_j, $j = 1, 2, 3, \ldots$. We shall assume that these are given with infinite precision in the interval $(0, 1)$. This is of course an idealization: but any algorithm sensitive to the infinite precision-assumption is *not* reliable and should not be used anyway[1]. The 'unitary algorithm formalism' allows us to translate mathematical operations such as variable transforms directly into

[1] As an example, one might try to be 'prudent' in the use of the random number stream by, say, cutting a number in half so as to obtain 'two for the price of one': $\rho = 0.n_1n_2n_3n_4n_5\ldots \rightarrow \rho_1 = 0.n_1n_3n_5\ldots$ and $\rho_2 = 0.n_2n_4n_6\ldots$. This is *not* a good idea.

computer (pseudo)code. In the first place, the equality

$$1 = \int dx\ p(x) \quad , \tag{138}$$

with $p(x)$ a probability density, is interpreted to read 'we have at hand an algorithm to generate x values according to $p(x)$. For instance,

$$1 = \int d\rho\ B(\rho) \tag{139}$$

simply specifies that we have available a good source of iid uniform random numbers in the unit interval[2], as assumed. Next, the occurrence of

$$dy\ \delta(y - H(x)) \quad ,$$

with an implied integral over y over the real axis, is interpreted as the (pseudo-)code statement

$$y \leftarrow H(x)$$

where x is a given number and H a known function. As a simple example, the equality

$$\begin{aligned} 1 \quad = \quad & \int d\rho_1\ B(\rho_1)\ d\rho_2\ B(\rho_2)\ dx \\ & \times \left\{\theta(\rho_2 < 1/2)\delta(x - \rho_1^2) + \theta(\rho_2)\delta(x - \sqrt{\rho_1})\right\} \end{aligned} \tag{140}$$

can be formulated in pseudo-code as

generate ρ_1 and ρ_2;
if $0 < \rho_2 < 1/2$, put $x \leftarrow \rho_1^2$;
if $1/2 < \rho_2 < 1$, put $x \leftarrow \sqrt{\rho_1}$.

By standard methods, we may eliminate ρ_1 and ρ_2 from Eq.(140), and arrive at

$$1 = \int_0^1 dx\ \left(x + \frac{1}{4\sqrt{x}}\right) \quad , \tag{141}$$

which we can interpret as the statement that the above algorithm generates x values with probability density $x + 1/4\sqrt{x}$ in the unit interval. The above simple rules allow one to work out the probability distribution implied by a given algorithm. As a variation, in Eq.(140) we might have left out the integral over dx: in that case, the left-hand side is of course not unity but simply x's probability density itself.

Finally, a fundamental but usually invisible property of the above algorithms is that they *finish* with certainty or, at least, with probability one. That is, once you enter the algorithm, a value is certainly returned. This is especially important for rejection-type algorithms, described below.

[2]Whether $[0,1]$, $[0,1)$, $(0,1]$ or $(0,1)$ depends on the implementation. Some care has to be taken here since there are algorithms that behave unacceptably if the random number is precisely zero or one.

6.6 Inversion methods

The most elegant method for generating a desired one-dimensional density $g(x)$ in an interval (x_0, x_1) is that of inversion. This method requires that we can compute the cumulative distribution

$$G(x) = \int_{x_0}^{x} dt\ g(t) \quad , \tag{142}$$

and, moreover, that we can compute its inverse. Now, $G(x)$ is non-decreasing from $G(x_0) = 0$ to $G(x_1) = 1$, and hence the equation

$$G(x) = \rho \ \in \ (0, 1) \tag{143}$$

has a unique solution. The infinite-precision assumption assures that where $g(x) = 0$ so that $G(x)$ is constant in some x interval, the probability of picking precisely the corresponding value of ρ is zero. The unitary-algorithm formalism now shows that this x has the correct distribution:

$$\int d\rho\ B(\rho)\ \delta(x - G^{-1}(\rho)) = \int_{0}^{1} d\rho\ G'(x)\ \delta(\rho - G(x)) = g(x)\theta(x_0 < x < x_1) \quad . \tag{144}$$

For many simple densities, inversion works perfectly, and then it is the method of choice. For instance, the density $P(x) = \exp(-x)\theta(x \geq 0)$ can be generated almost trivially, using $x \leftarrow -\log(\rho)$.

Sometimes, of course, the inverse of G can only be found by purely numerical methods, calling for repeated evaluation of G. This can make inversion time-consuming. Also, the method is strictly one-dimensional. For a more-dimensional distribution, therefore, we must compute two successive cumulative densities, and generate variables on after the other. Especially in cases where the inversion has to be performed numerically this can become prohibitive.

6.7 Rejection methods

Rejection methods (also called von Neumann methods) use the following strategy for generating a density proprtional to some nonnegative function $g(x)$ in an interval X (which may be more-dimensional). Suppose that we can generate x in X according to another density, $p(x)$, and that there is a constant c such that

$$g(x) \leq c\ p(x) \quad , \quad x \in X \quad . \tag{145}$$

The algorithm now reads

generate x according to $p(x)$;
generate a random number ρ;
if $c\rho < g(x)$, accept this value of x;
otherwise, go back and try a new value of x.

The unitary-algorithm formulation for the resulting probability density $P(x)$ is

$$P(x) = \int dy\, d\rho\, p(y)\ \{\theta(c\rho \le g(y))\, \delta(x-y) + \theta(c\rho > g(y))P(x)\}\ \ ; \qquad (146)$$

here, the last term in the two alternatives represents the step where the new value is generated[3]. By simple algebra we arrive at

$$P(x) = \frac{1}{N} g(x)\ \ , \quad N = \int\limits_X g(t)\, dt\ \ , \qquad (147)$$

which proves the algorithm. It is important to realize that this proof depends crucially on the fact that $g(x)$ is indeed bounded by $cp(x)$, otherwise the piece of $g(x)$ that 'sticks out' will simply be omitted. The efficiency of the algorithm (that is, the average number of times a trial value y has to be generated before it is accepted as x), depends of course on our ability to find a good and generable approximant $p(x)$.

6.8 Clever tricks and combination methods

The field abounds with simple and not-so-simple tricks to generate particular densities. A veritable bible is [8]: here we just give few examples. Perhaps the most impressive and useful is the Box-Müller algorithm for obtaining the normal density, $P(x) = \exp(-x^2)/\sqrt{2\pi}$. Note that this density is not easily generated by rejection because of its infinite tails, nor by inversion since that involves computing error functions. There exists the well-known 'doubling trick' by which the integral over $\exp(-x^2/2)$ is computed:

$$\left(\int dx \exp\left(-x^2/2\right)\right)^2 = \int dx\, dy\ \exp\left(-(x^2+y^2)/2\right)$$
$$= \int\limits_0^{2\pi} d\phi \int\limits_0^{\infty} ds\ s\ exp(-s^2/2) = 2\pi \int\limits_0^{\infty} dt\ \exp(-t) \ = \ 2\pi\ , \qquad (148)$$

where the substitutions $x = s\cos(\phi)$, $y = s\sin(\phi)$, and $s = \sqrt{t}$ have been applied. This immediately implies the following algorithm for generating *pairs* of independent, normally distributed random numbers:

generate ρ_1 and ρ_2 uniformly in $(0, 1]$;
put $t \leftarrow -\log(\rho_1)$ and $\phi \leftarrow 2\pi\rho_2$;
put $x \leftarrow \cos(\phi)\sqrt{t}$ and $y \leftarrow \sin(\phi)\sqrt{t}$.

Although this is not the very fastest algorithm for generating normal variates, it is certainly among the most elegant, and easy to remember as well.

[3] This can easily be checked by writing the algorithm as a formally infinite series of branches.

The above example displays what is probably the defining characteristic of 'clever tricks': the judicious combination of several random variables into one. As another example, the density $P(x) = x^{N-1}e^{-x}/\Gamma(N)$ does not lend itself easily to either rejection or inversion; however, from the discussion of the Central Limit theorem in section 2.4 we see immediately that (for integer N) we may use the simple algorithm $x \leftarrow -log(\rho_1\rho_2\cdots\rho_N)$.

As a further example of a clever trick we mention the following. Suppose that the probability density $P(x)$ is zero for $x < 0$. From the unitary-algorithm description

$$\int_0^\infty d\xi P(\xi) \int_0^1 d\rho\ \delta(x - \rho\xi) = \int_x^\infty d\xi \frac{1}{\xi} P(\xi) \tag{149}$$

we derive that $x \leftarrow -\rho_1 \log(\rho_2)$ leads to the density $E_1(x)$, the first exponential integral. In a similar vein, it is easily proved that $x \leftarrow 2\sqrt{\log(\rho_1)\log(\rho_2)}$ generates the density $xK_0(x)$, where K_0 is the modified Bessel function of the second kind, of order 0.

It may be fair to say that, if a particular difficult-looking nonnegative function can be integrated by an aesthetically pleasing trick, this usually implies a 'clever' algorithm to generate it. A drawback is, of course, that each density requires its own particular trick.

6.9 Kinderman-Monahan algorithms: ratio-of-uniforms

A particularly elegant-looking class of algorithms can be based on the following strategy. Let $f(x)$ be a nonnegative function. Consider the two-dimensional region R in the (u, v) plane determined by the following conditions:

$$-\infty < u < +\infty \quad , \quad 0 \le v < +\infty \quad , \quad v^2 \le f(u/v)\ . \tag{150}$$

The area A_R of this region can be computed as follows:

$$\begin{aligned} A_R \equiv \int_R du\ dv &= \int dx \int_R du\ dv\ \delta(x - u/v) \\ &= \int dx \int_{-\infty}^{\infty} du \int_0^\infty dv\ v\ \delta(vx - u)\ \theta(v^2 \le f(x)) \\ &= \int dx \int_0^\infty d(v^2)\ \theta(v^2 \le f(x)) = \frac{1}{2}\int f(x)\ dx\ . \end{aligned} \tag{151}$$

We then arrive at the following algorithm for generating x according to the density $P(x) \propto f(x)$:

generate (u, v) uniformly in the region R;
put $x \leftarrow (u/v)$.

That this is indeed a correct algorithm is easily seen by the same method:

$$P(x) = \frac{1}{A_R} \int\limits_R du\, dv\, \delta(x - u/v) = \frac{1}{2A_R} f(x) = f(x) \left(\int dx'\, f(x') \right)^{-1} \quad . \tag{152}$$

So, provided that we can generate points (u, v) in R efficiently, the function $f(x)$ is easily generated in a self-normalizing way, even avoiding the need to compute the normalization. As an example, let R be the rectangle with vertices $(-1, 0)$, $(-1, 1)$, $(1, 1)$ and $(1, 0)$. It is easily seen that in that case

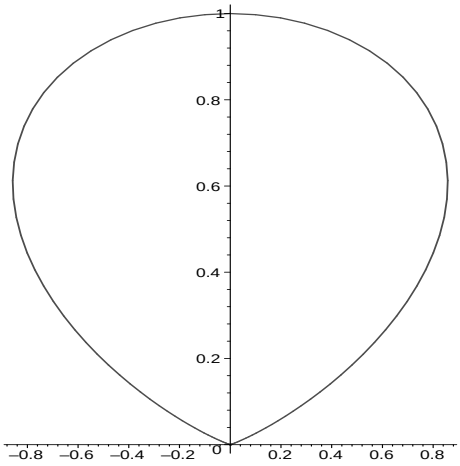
$$P(x) = \frac{1}{4} \left(\theta(|x| \leq 1) + \frac{1}{x^2} \theta(|x| > 1) \right) \quad . \tag{153}$$

Any function that can be fitted under this curve can be generated by generating inside this rectangle, followed by rejection of points outside the region R. For instance, let R be the half-circle given by $u^2 + v^2 \leq 1$. This will lead to the Cauchy distribution $P(x) = (1 + x^2)^{-1}/\pi$. Although this density can of course be generated by inversion as well, the above algorithm is computationally cheaper since it avoids taking arc-tangent functions (on the other hand, it uses two-dimensional pseudorandom points rather than one-dimensional ones).

The boundary of the region R can be parametrically obtained: if τ is a parameter running from $-\infty$ to $+\infty$, we simply have

$$v = \sqrt{f(\tau)} \quad , \quad u = \tau \sqrt{f(\tau)} \quad . \tag{154}$$

As an example, we show the boundary of R for the normal distribution $f(x) = \exp(-x^2/2)$;



The region R for the normal distribution. It is bounded by the curve $u = 2v\sqrt{\log(v)}$. Its area is $\sqrt{\pi/2}$, or about 1.253, and it fits in a rectangle of area $\sqrt{8/e}$, or about 1.716. The efficiency of generation is therefore 73%.

An advantage of this method is that it is often easy to determine whether a point (u, v) is inside R, and many shortcuts can be invented to simplify and/or speed up this check.

Conversely, the density arising from a given region R can be determined simply: if the boundary is given by the equation $F(u, v) = 0$, then $P(x)$ can be obtained by solving $F(P, xP) = 0$ for P. This is especially convenient if $P(x)$ is actually defined as the root of some implicit equation.

The method may of course be extended. We may generate a value x according to the normal distribution $P(x) = \exp(-x^2/2)/\sqrt{2\pi}$, and then generate, for this value x, a value y between 0 and $P(x)$. By construction, we then fill the shape under the curve $y = P(x)$ uniformly. The variate $z = x/y$ will then be distributed according to the density $W(z^2/(2\pi))/z^2$, where W is the Lambert function defined implicitly by $W(t)\exp(W(t)) = t$. Quite unbelieveable-looking densities can be obtained in this way, using simple manipulations.

6.10 Generation under constraints

6.11 The Metropolis algorithm

This algorithm is somewhat special in the sense that it delivers not single numbers but a stream of numbers, of which the distribution approaches the desired one asymptotically. Let the desired probability density of the number x be given by $F(x)$, so that $F(x) \geq 0$ and $F(x)$ integrates to 1. The algorithm by which the next point, x_{k+1}, is determined using the current one, x_k, can be formulated as follows.

1. Generate a number y under some density $g(x_k; y)$ that depends on x_k. Here, the requirement of *detailed balance* is important: $g(x; y) = g(y; x)$, that is, it is as probable to pick y if the current number in the stream is x as it would be to pick x if the current number in the stream were y;
2. Compute the ratio $R = F(y)/F(x_k)$;
3. If $R \geq 1$, assign $x_{k+1} \leftarrow y$;
4. If $0 \leq R < 1$, assign $x_{k+1} \leftarrow y$ with probability R, and $x_{k+1} \leftarrow x_k$ with probability $1 - R$. This is of course done by generating a uniform random number in $[0, 1]$ and comparing with R.

The unitary-algorithm description of this algorithm is as follows. Let the numbers x_k be distributed according to $F_k(x_k)$. After one step of the Metropolis algorithm, the distribution $F_{k+1}(x_{k+1})$ of the new numbers x_{k+1} is given by

$$\begin{aligned} F_{k+1}(x_{k+1}) \quad = \quad & \int dx_k \; F_k(x_k) \; dy \; g(x_k; y) \\ & \times \Bigg[\theta(F(y) \geq F(x_k)) \delta(x_{k+1} - y) \\ & \quad + \theta(F(y) < F(x_k)) \left\{ \frac{F(y)}{F(x_k)} \delta(x_{k+1} - y) \right. \\ & \qquad \left. + \left(1 - \frac{F(y)}{F(x_k)} \right) \delta(x_{k+1} - x_k) \right\} \Bigg] \end{aligned}$$

$$= \int dx\ g(x_{k+1};x) \times \left[\theta(F(x_{k+1} \geq F(x)) \left\{ F_k(x) + F_k(x_{k+1}) - \frac{F_k(x_{k+1})F(x)}{F(x_{k+1})} \right\} + \theta(F(x_{k+1} < F(x)) \frac{F_k(x)F(x_{k+1})}{F(x)} \right] \quad (155)$$

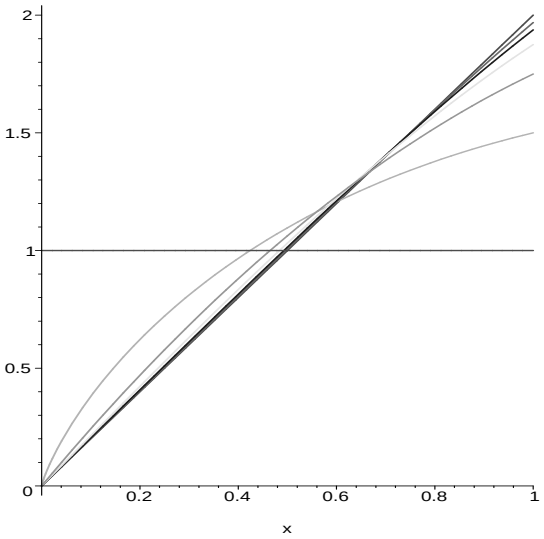
Here, we have used the detailed balance requirement. It is seen that $F(x)$ is a fixed point of the mapping $F_k \to F_{k+1}$ since if we insert $F_k(x) = F(x)$ then also $F_{k+1}(x) = F(x)$. Note that the dimensionality of x again plays no rôle here so that the algorithm is dimension-independent; moreover, since only *ratio's* of the desired density F enter, the algortithm does not require knowledge of the normalization of F but only its nonnegativity.

To investigate this algorithm by a simple example we shall put $F(x) = 2x$ and restrict ourselves to the interval $[0,1]$. Also, we shall simply take y independent of x, that is, $g(x;y)$ is the uniform density in $[0,1]$, which certainly satisfies detailed balance. Simple algebra then tells us that the successive F_k obey

$$F_{k+1}(x) = \int_0^x F_k(z)dz + \int_x^1 F_k(z)\frac{x}{z}dz + \frac{x}{2}F_k(x) \ . \quad (156)$$

Starting with points distributed according to the uniform density, we have

$$F_1(x) = 1 \quad , \quad F_2(x) = \frac{3}{2}x - x\log(x) \quad ,$$
$$F_k(x) = \left(2 + \frac{1}{2^{k-2}(k-2)}\right)x - \frac{k}{2^{k-1}(k-2)}x^{k-1} \ , \ k \geq 3 \ . \quad (157)$$



The plot shows the distributions $F_n(x)$ for $1 \leq n \leq 6$, and the desired distribution $2x$. As n increases, there is a quite rapid approach to the asymptotic case.

To gauge the approach to the asymptotically desired distribution, we may define the difference norm

$$D_n = \int (F_n(x) - 2x)^2 \, dx = \frac{n - 1/3}{4^{n-1}(n+1)(2n-1)} \quad . \tag{158}$$

The approach to the desired distribution is roughly exponential.

The Metropolis algorithm is general and flexible, but it suffers from drawbacks. In the first place, the distribution actually generated after a given finite number of steps is usually unknown, which leads to an unknown bias in, *e.g.* importance sampling. For this reason it is not commonly used in high-precision Monte Carlo integration in particle phenomenology. In the second place, the nonnegligible fraction of cases where $x_{k+1} = x_k$ is also considered unattractive in many cases. Thirdly, since the normalization of $F(x)$ is irrelevant, the algorithm cannot serve to compute the integral of $F(x)$ itself. The main application of the Metropolis algorithm is in the simulation of complicated distribution with extremely high dimension, where the goal is the computation of certain moments of $F(x)$: the natural type of problem is, therefore, in statistical physics and lattice field theories.

7 Phase space algorithms

One of the fundamental integrations to be performed in particle phenomenology is that of transition rates over a multiparticle phase space. Under the usual convention that the speed of light has numerical value unity, the phase space integration element for a final state of n particles with masses $m_1, m_2, \ldots, m_n$ can be written as

$$\prod_{j=1}^{n} \left(d^4 p_j \; \delta\left(p_j^2 - m_j^2\right) \; \theta(p_j^0) \right) \; \delta^3 \left(\sum_{j=1}^{n} \vec{p}_j \right) \; \delta \left(\sqrt{s} - \sum_{j=1}^{n} p_j^0 \right) \quad ,$$

where the Minkowski square of the four-vector $p^\mu = (p^0, \vec{p})$ is defined by $p^2 = (p^0)^2 - \vec{p}^2$. Here the total four-momentum of the system is at rest, with a total invariant mass of $\sqrt{s}$. Because of the essentially nonlinear character of the energy conservation constraint, this is a very nontrivial integration problem. In the absence of additional information about the transition rate[1], or for transition rates that fluctuate mildly, one may consider performing the phase space integral without attempting variance reduction. In that case, it becomes important to search for algorithms that generate phase space points as uniformly as possible. This is the subject of the next few sections.

7.1 Hierarchical methods

7.2 `RAMBO` algorithm

The `RAMBO` algorithm[2] aims at generating points in an n-particle phase space in the ultra-relativistic limit, where the particle masses can be neglected. The phase space element is therefore

$$\prod_{j=1}^{n} \left(d^4 p_j \delta\left(p_j^2\right) \theta(p_j^0) \right) \quad \delta^3 \left(\vec{P} \right) \delta \left(P^0 - \sqrt{s} \right) \quad , \tag{159}$$

where

$$P^\mu = \sum_{j=1}^{n} p_j^\mu \quad , \quad s = P^2 \quad . \tag{160}$$

The task is not only to obtain four-momenta that satisfy the constraints of energy and momentum conservation, but also we have to ensure that the whole phase space is covered, with as uniform a density as possible.

From the integral

$$\int d^4 q \; exp(-q^0/w) \; \delta(q^2) \; \theta(q^0) = 2\pi \int_0^\infty dq^0 \; q^0 \; \exp(-q^0/w) = 2\pi w^2 \quad , \tag{161}$$

[1] In practice, such information *is* available if the transition rate is computed from Feynman diagrams or the like.

[2] `RAMBO` is an acronym of RAndom Momenta BOoster.

where w is an arbitrary positive scale, we see that the following unitary-algorithm formulation holds:

$$1 = (2\pi w^2)^{-n} \int \prod_{j=1}^{n} \left(d^4 q_j \; \delta\left(q_j^2\right) \; \theta(q_j^0) \; \exp(-q_j^0/w) \right) \quad . \tag{162}$$

The algorithm now proceeds as follows: first, we compute $k^\mu = \sum_j q_j^\mu$. In general, this vector is not at rest, nor does its square equal s. Therefore, we perform on every q_j the Lorentz boost Λ which brings k^μ to its rest frame, and a scaling transform that ensures the correct overal invariant mass. The resulting vectors are denoted by p_j. Equation (162) then becomes

$$\begin{aligned} 1 \;\; = \;\; & (2\pi w^2)^{-n} \int \prod_{j=1}^{n} \left(d^4 q_j \; \delta\left(q_j^2\right) \; \theta(q_j^0) \; d^4 p_j \; \delta^4 \left(p_j - \frac{1}{x} \Lambda(q_j) \right) \right) \\ & d^4 k \; \delta^4 \left(k - \sum_j q_j \right) \; dx \; \delta \left(x - \frac{\sqrt{k^2}}{\sqrt{s}} \right) \quad . \end{aligned} \tag{163}$$

The scaling factor x runs from 0 to ∞. Owing to the Lorentz invariance of the four-dimensional Dirac delta, we may write

$$\delta^4 \left(p_j - \frac{1}{x} \Lambda(q_j) \right) \; \delta\left(q_j^2\right) = x^2 \; \delta^4 \left(q_j - x \Lambda^{-1}(p_j) \right) \; \delta\left(p_j^2\right) \quad . \tag{164}$$

Furthermore,

$$\begin{aligned} \delta^4 \left(k - \sum_j q_j \right) \; \delta \left(x - \frac{\sqrt{k^2}}{\sqrt{s}} \right) \;\; &= \;\; \frac{2s}{x^3} \; \delta^4 \left(P - \frac{1}{x} \Lambda(k) \right) \; \delta\left(k^2 - x^2 s\right) \\ &= \frac{2s}{x^3} \; \delta^3 \left(\vec{P} \right) \; \delta\left(P^0 - \sqrt{s}\right) \; \delta\left(k^2 - x^2 s\right) \quad . \end{aligned} \tag{165}$$

The unitary formula (163) can therefore be written as

$$1 = N \int \prod_{j=1}^{n} \left(d^4 p_j \; \delta\left(p_j^2\right) \; \theta(p_j^0) \right) \; \delta^3 \left(\vec{P} \right) \; \delta\left(P^0 - \sqrt{s}\right) \quad , \tag{166}$$

where

$$N = 2s(2\pi w^2)^n \int_0^\infty dx \int d^4 k \; x^{2n-3} \; \exp(-k^0/w) \; \sqrt{(k^0)^2 - x^2 s} \quad . \tag{167}$$

A fairly straightforward computation of the double integral[3] then leads to the volume of phase space in the ultra-relativistic limit:

$$V_{\rm UR} = N^{-1} = \left(\frac{\pi}{2} \right)^{n-1} \frac{s^{n-2}}{\Gamma(n)\Gamma(n-1)} \quad . \tag{168}$$

[3]Helped by the substitution $k^0 = tx\sqrt{s}$, with a new integration variable t.

We see that the RAMBO algorithm is essentially the best possible: the whole of phase space is covered uniformly. The energy scale w drops out as it should, and therefore in practice we use $w = 1$.

The algorithm for the generation of the phase space events is straightforward. Let us denote by ρ_i random numbers from the stream. The energy q_j^0 and its polar and azimuthal angles, θ_j and ϕ_j, are generated by

$$q_j^0 \leftarrow -\log(\rho_1\rho_2) \ , \ \theta_j \leftarrow 2\rho_3 - 1 \ , \ \phi_j \leftarrow 2\pi\rho_4 \ . \tag{169}$$

Using $u = \sqrt{k^2}$, the combined boost and scaling transform from the q's into the p's is given by

$$\begin{aligned} p^0 &\leftarrow \frac{\sqrt{s}}{u^2}(k\cdot q) \ , \\ \vec{p} &\leftarrow \frac{\sqrt{s}}{u^2}\left(u\vec{q} - \vec{k}\frac{(k\cdot q) + uq^0}{u + k^0}\right) \ . \end{aligned} \tag{170}$$

The total number of (pseudo-)random numbers used per event is $4n$, quite reasonable since the minimum possible number necessary is $3n - 4$. By finding a clever algorithm to generate the distribution $x\exp(-x)$ using only one ρ, the number of random variates used may be reduced to $3n$; but since the algorithm analysis contains integration over k^μ and x, information is always lost, and the algorithm becomes irreversible: in other words, from the resulting momenta p_j^μ one cannot work backwards to find out what the original q_j^μ (or the random variates that went into them) were.

7.3 Inclusion of particle masses

Quite often, the ultra-relativistic approximation is not appropriate, and particles may be relatively light but not massless. The following discussion shows how we can accomodate this. Let us assume that a 'massless' phase space point has been generated according to RAMBO:

$$1 = \frac{1}{V_{\rm UR}} \int \prod_{j=1}^{n} \left(d^4q_j\delta\left(q_j^2\right)\right) \ \delta^3\left(\sum_j \vec{q}_j\right) \ \delta\left(\sum_j q_j^0 - \sqrt{s}\right) \ . \tag{171}$$

We may now scale the three-momenta down by a factor ξ between 0 and 1, and adjust the energies so as to build in the masses, while keeping energy conservation intact. That is, we determine the unique root ξ_0 of the function

$$F(\xi) = -\sqrt{s} + \sum_{j=1}^{n} \sqrt{\xi^2(q_j^0)^2 + m_j^2} \ . \tag{172}$$

Since $F(\xi)$ is monotonic for $\xi > 0$ with positive derivative, and the root ξ_0 is known to be inside $(0, 1)$, this is a simple task suited for Newton-Raphson iteration starting at $\xi = 1$. Having found ξ_0, we then replace q_j^μ by p_j^μ:

$$\vec{p}_j \leftarrow \xi\vec{q}_j \ \ , \ \ p_j^0 \leftarrow \sqrt{\vec{p}_j^{\,2} + m_j^2} \ . \tag{173}$$

Let us now analyse this algorithm. We may rewrite equation (171) as

$$1 = \frac{1}{V_{\rm UR}} \int \left(d^4q_j \; \delta\left(q_j^2\right) \; d^4p_j \; \delta^3\left(\vec{p}_j - \xi\vec{q}_j\right) \; \delta\left(p_j^0 - \sqrt{|\vec{p}_j|^2 + m_j^2}\right)\right) \delta^3\left(\sum_j \vec{q}_j\right) \; \delta\left(\sum_j q_j^0 - \sqrt{s}\right) \; d\xi \; \delta\left(\xi - \xi_0\right) \quad . \tag{174}$$

We now perform a few manipulations. In the first place,

$$\delta\left(\xi - \xi_0\right) = F'(\xi)\delta\left(F(\xi)\right) = \delta\left(\sum_j p_j^0 - \sqrt{s}\right) \; \sum_{j=1}^n \frac{|\vec{p}_j|^2}{\xi p_j^0} \quad . \tag{175}$$

Secondly,

$$d^4q_j \; \delta\left(q_j^2\right) \; d^4p_j \; \delta^3\left(\vec{p}_j - \xi\vec{q}_j\right) \; \delta\left(p_j^0 - \sqrt{|\vec{p}_j|^2 + m_j^2}\right) = \frac{p_j^0}{\xi|\vec{p}_j|} \; d^4p_j \; \delta\left(p_j^2 - m_j^2\right) \quad . \tag{176}$$

Finally,

$$\delta\left(\sum_j q_j^0 - \sqrt{s}\right) = \frac{\xi}{\sqrt{s}} \; \delta\left(\xi - \frac{1}{\sqrt{s}} \sum_j |\vec{p}_j|\right) = \frac{\xi}{\sqrt{s}} \; \delta\left(\xi - \xi_0\right) \tag{177}$$

and

$$\delta^3\left(\sum_j \vec{q}_j\right) = \xi^3 \; \delta^3\left(\sum_j \vec{p}_j\right) \quad . \tag{178}$$

It is seen that, once the vectors p_j^μ have been obtained, ξ is simply[4] given by $\sum_j |\vec{p}_j|/\sqrt{s}$. Putting everything together, we see that Eq.(174) becomes

$$1 = \int \prod_{j=1}^n \left(d^4p_j \; \delta\left(p_j^2 - m_j^2\right)\right) \; \delta^3\left(\sum_j \vec{p}_j\right) \; \delta\left(\sum_j p_j^0 - \sqrt{s}\right) \times \frac{\xi_0^{3-2n}}{V_{\rm UR}} \left(\sum_{j=1}^n \frac{|\vec{p}_j|^2}{p_j^0\sqrt{s}}\right) \prod_{j=1}^n \left(\frac{p_j^0}{|\vec{p}_j|}\right) \quad . \tag{179}$$

We see that, in constrast to the ultra-relativistic case, the phase space is not filled uniformly, but rather with a fluctuating density given by the second line of Eq.(179). This implies that we have to assign to each generated phase space point a weight

$$w(p_1, p_2, \ldots, p_n) = V_{\rm UR} \; \left(\sum_{j=1}^n \frac{|\vec{p}_j|}{\sqrt{s}}\right)^{2n-3} \left(\prod_{j=1}^n \frac{|\vec{p}_j|}{p_j^0}\right) \left(\sum_{j=1}^n \frac{|\vec{p}_j|^2}{p_j^0\sqrt{s}}\right)^{-1} \quad . \tag{180}$$

[4]Computationally, this is of course useless, since the p's are only determined once ξ has been found.

Note that this 'rescaling' is actually reversible since ξ_0 is given in terms of the finally resulting p's.

The physics behind the weight is clear: for massive particles, the phase space is smaller than for massless particles, so the rescaling transform is actually a contraction. This contraction is most noticeable at the edges of phase space,and least so in the 'center'. Since the boundaries of massless phase space have sharp corners, whereas for massive particles the corners are rounded (as can be seen from, *e.g.*, the Dalitz plot for $n = 3$), there exists no uniform transformation between the two, and the weights fluctuate. An exception is $n = 2$, where

$$w(p_1, p_2) = V_{\rm UR}\ \xi_0 = \frac{\pi}{2s}\lambda(s, m_1^2, m_2^2)^{1/2} \quad , \tag{181}$$

where λ is the Källén function,

$$\lambda(x, y, z) = x^2 + y^2 + z^2 - 2xy - 2xz - 2yz \quad , \tag{182}$$

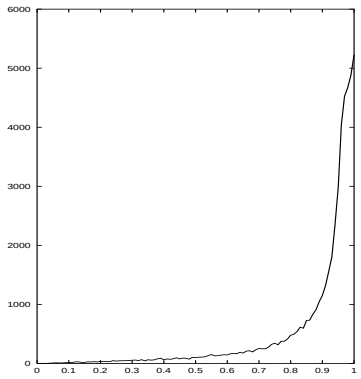
and phase space is actually covered in a uniform manner[5]. For larger n, rejection may be used to arrive at a uniformly distributed phase space point set, and then it becomes relevant what the maximum possible value of ξ is. In general, this is not known and has to be determined empirically. For all masses equal, $m_j = m$, the maximum weight is expected to reside where all particles are as relativistic as possible, since the weight vanishes at the boundaries of phase space where one or more three-momenta vanish. This point is given by

$$p_j^0 = \frac{1}{n}\sqrt{s} \quad , \quad |\vec{p}_j| = \frac{1}{n}\sqrt{s - n^2m^2} \quad , \quad j = 1, 2, \ldots, n \quad ,$$

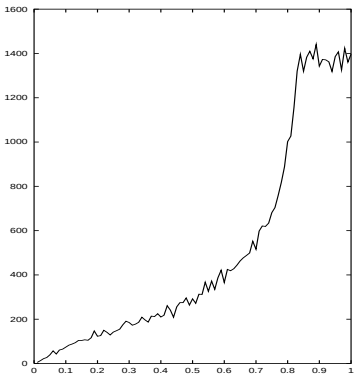
and the weight at that point is given by

$$w(p_1, p_2, \ldots, p_n) = \left(1 - \frac{n^2m^2}{s}\right)^{(3n-5)/2} \quad . \tag{183}$$

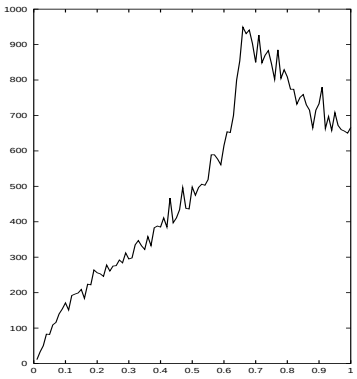
[5]Of course, generating a two-body phase space by `RAMBO` and rescaling is using *very* heavy artillery; still, some programs do it...



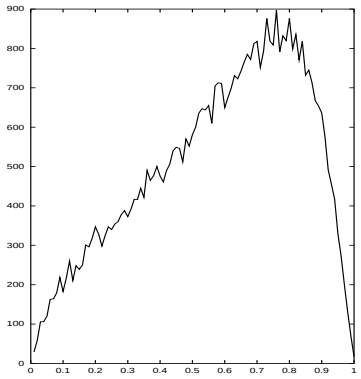
$n = 3\ ,\ r = 0.3$



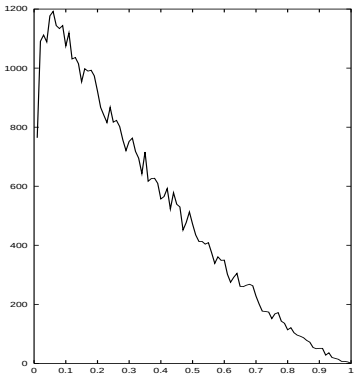
$n = 3\ ,\ r = 0.6$



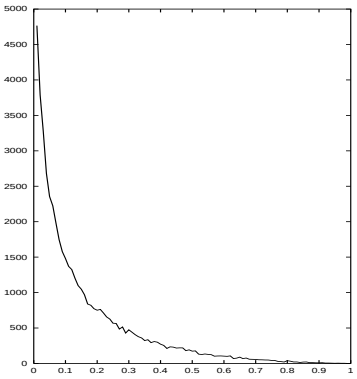
$n = 3\ ,\ r = 0.9$



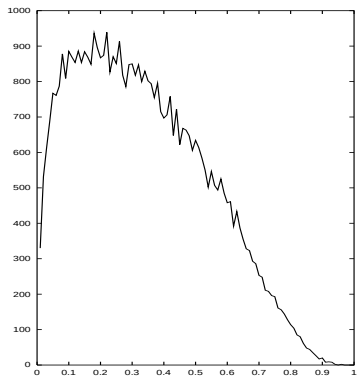
$n = 6\ ,\ r = 0.3$



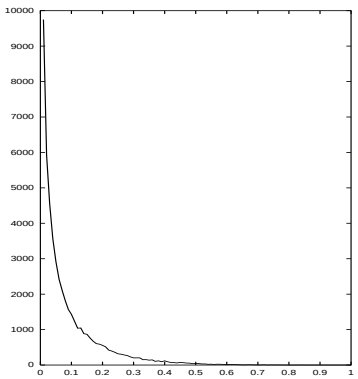
$n = 6\ ,\ r = 0.6$



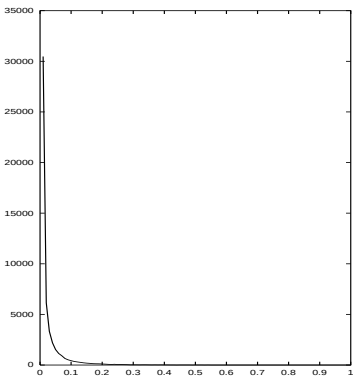
$n = 6\ ,\ r = 0.9$



$n = 10\ ,\ r = 0.3$



$n = 10\ ,\ r = 0.6$



$n = 10\ ,\ r = 0.9$

Depicted are some examples of the distribution of the weights, normalized to the conjectured maximum (183). We have described the 'massiveness' of the final state by taking all masses equal to

$$m_j = r\frac{\sqrt{s}}{n} \quad , \quad 0 < r < 1 \quad , \quad j = 1, 2, \ldots, n \ . \tag{184}$$

The conjectured maximum weight is indeed never exceeded. Not surprisingly, the weight distribution deteriorates as n or r increases. For moderately large number of particles (meaning even up to a 10-particle final state!) and fairly small masses, the weight distribution is perfectly acceptable for rejection purposes.

7.4 BOLTZ algorithm

Although it is not usually relevant for particle phenomenology, we may consider the opposite limit, that of nonrelativistic particles. The phase space integration element is then given by that of the *microcanonical ensemble*:

$$\int \prod_{j=1}^{n} d^3\vec{p}_j \; \delta^3\left(\sum_j \vec{p}_j\right) \; \delta\left(\sum_j \frac{|\vec{p}_j|^2}{2m_j} - U\right) \quad ,$$

where U is the total available kinetic energy. We start with the integral

$$\int d^3\vec{p} \; \exp(-|\vec{p}|^2/2m) = (2\pi m)^{3/2} \quad , \tag{185}$$

so that the first step in the unitary-algorithm treatment is

$$1 = \prod_{j=1}^{n} (2\pi m_j)^{-3/2} \int \prod_{j=1}^{n} \left(d^3\vec{q}_j \; \exp(-|\vec{q}_j|^2/2m_j)\right) \quad . \tag{186}$$

Instead of a Lorentz boost, we now perform a Galilei transform to bring the total velocity of the system to zero. Eq.(186) is then modified into

$$\begin{aligned} 1 \;\; = \;\; & \prod_{j=1}^{n} (2\pi m_j)^{-3/2} \\ & \int \prod_{j=1}^{n} \left(d^3\vec{q}_j \; \exp(-|\vec{q}_j|^2/2m_j) \; d^3\vec{r}_j \; \delta^3\left(\vec{r}_j - (\vec{q}_j - M\vec{v})\right)\right) \\ & d^3\vec{r} \; \delta^3\left(\vec{r} - \frac{1}{M}\sum_j \vec{q}_j\right) \quad , \end{aligned} \tag{187}$$

where $M = \sum_j m_j$ is the total mass. After the usual manipulations, we can perform the $\vec{v}$ integral and arrive at

$$\begin{aligned} 1 \;\; = \;\; & \left(\frac{2\pi}{M}\right)^{3/2} \prod_{j=1}^{n} (2\pi m_j)^{-3/2} \\ & \int \prod_{j=1}^{n} d^3\vec{r}_j \; \delta^3\left(\sum_j \vec{r}_j\right) \; \exp\left(-\sum_j \frac{|\vec{r}_j|^2}{2m_j}\right) \quad . \end{aligned} \tag{188}$$

Next, we perform the scaling transform that brings the total kinetic energy to the value U. This transforms Eq.(188) into

$$\begin{aligned}
1 &= \left(\frac{2\pi}{M}\right)^{3/2} \prod_{j=1}^{n} (2\pi m_j)^{-3/2} \\
&\quad \int \prod_{j=1}^{n} \left(d^3\vec{r}_j \; d^3\vec{p}_j \; \delta^3\left(\vec{p}_j - \frac{1}{x}\vec{r}_j\right)\right) \delta^3\left(\sum_j \vec{r}_j\right) \\
&\qquad dx \; 2x \; \delta\left(x^2 - \frac{1}{U}\sum_j \frac{|\vec{r}_j|^2}{2m_j}\right) \exp\left(-\sum_j \frac{|\vec{r}_j|^2}{2m_j}\right) \\
&= 2U \left(\frac{2\pi}{M}\right)^{3/2} \prod_{j=1}^{n} (2\pi m_j)^{-3/2} \\
&\quad \int \prod_{j=1}^{n} d^3\vec{p}_j \;\; \delta^3\left(\sum_j \vec{p}_j\right) \;\; \delta\left(\sum_j \frac{|\vec{p}_j|^2}{2m_j} - U\right) \\
&\quad \int_0^\infty dx \; x^{3n-4} \; \exp(-Ux^2) \\
&= \frac{1}{V_{\rm NR}} \int \prod_{j=1}^{n} d^3\vec{p}_j \;\; \delta^3\left(\sum_j \vec{p}_j\right) \;\; \delta\left(\sum_j \frac{|\vec{p}_j|^2}{2m_j} - U\right) \quad , \qquad (189)
\end{aligned}$$

where $V_{\rm NR}$ is the nonrelativistic phase space volume, given by

$$V_{\rm NR} = \frac{(2\pi)^{(3n-3)/2}}{\Gamma((3n-3)/2)} \frac{U^{(3n-5)/2}}{M^{3/2}} \left(\prod_{j=1}^{n} m_j\right)^{3/2} \quad . \qquad (190)$$

The – apparently – naive change from the ultrarelativistic expression for the energy to the nonrelativistic one, and from Lorentz to Galilei transform, turns out to give again the optimal algorithm. Since this describes the microcanonical ensemble, we may dub it the `BOLTZ`(mann) algorithm.

7.5 `MAMBO` algorithm

7.6 `SARGE` algorithm

8 Appendices

8.1 Falling powers

The falling powers $n^{\underline{k}}$, where $k \geq 0$ is integer, are defined by

$$n^{\underline{k}} = \frac{n!}{(n-k)!} = n(n-1)(n-2)\cdots(n-k+1) \ . \tag{191}$$

They can be combined into the generating function

$$\sum_{k\geq 0} n^{\underline{k}} \frac{z^k}{k!} = (1+z)^n \ . \tag{192}$$

In terms of regular powers, we have for the first few values of k:

$$\begin{aligned} n^{\underline{1}} &= n \ , \\ n^{\underline{2}} &= n^2 - n \ , \\ n^{\underline{3}} &= n^3 - 3n^2 + 2n \ , \\ n^{\underline{4}} &= n^4 - 6n^3 + 11n^2 - 6n \ , \ldots \end{aligned} \tag{193}$$

This can be inverted to express regular powers in terms of falling powers:

$$\begin{aligned} n^2 &= n^{\underline{2}} + n^{\underline{1}} \ , \\ n^3 &= n^{\underline{3}} + 3n^{\underline{2}} + n^{\underline{1}} \ , \\ n^4 &= n^{\underline{4}} + 6n^{\underline{3}} + 7n^{\underline{2}} + n^{\underline{1}} \ , \ldots \end{aligned} \tag{194}$$

The coefficients in these two series of expressions are the so-called Stirling numbers.

8.2 Unequal-index sums

Averages of quantities like S_1^2 cannot be taken without more ado. To tackle this problem, we introduce an extension of the objects S_m:

$$S_{m_1,m_2,\ldots,m_k} = \sum_{j_1,j_2,\ldots,j_k} (w_{j_1})^{m_1} (w_{j_2})^{m_2} \cdots (w_{j_k})^{m_k} \ , \tag{195}$$

with the explicit constraint that the labels $j_1, j_2, \ldots, j_k$ are all different. In that case, the various w's all refer to different random numbers, and since these are iid we immediately have

$$\langle S_{m_1,m_2,\ldots,m_k} \rangle = N^{\underline{k}} J_{m_1} J_{m_2} \cdots J_{m_k} \ . \tag{196}$$

It is easily seen that the following rule holds:

$$\begin{aligned} &S_{m_1,m_2,\ldots,m_k} S_n = \\ &\quad S_{m_1+n,m_2,\ldots,m_k} + S_{m_1,m_2+n,\ldots,m_k} + \cdots \\ &\quad \cdots + S_{m_1,m_2,\ldots,m_k+n} + S_{m_1,m_2,\ldots,m_k,n} \ , \end{aligned} \tag{197}$$

according to whether the label of the random number in the sum S_n is equal to j_1, or to j_2, and so on, or different from all of them. In this way, we can reduce various products of S's to sums of single terms:

$$\begin{aligned} S_1^2 &= S_2 + S_{1,1} \ , \\ S_2 S_1 &= S_3 + S_{2,1} \ , \\ S_1^3 &= S_3 + 3S_{2,1} + S_{1,1,1} \ , \\ S_3 S_1 &= S_4 + S_{3,1} \ , \\ S_2^2 &= S_4 + S_{2,2} \ , \\ S_2 S_1^2 &= S_4 + 2S_{3,1} + S_{2,2} + S_{2,1,1} \ , \\ S_1^4 &= S_4 + 4S_{3,1} + 3S_{2,2} + 6S_{2,1,1} + S_{1,1,1,1} \ . \end{aligned} \tag{198}$$

These results suffice to compute the first- and second-order error estimates. Although the nicely linear unequal-index sums makes for easier evaluation of their expectation values, they should not be used in the actual application: if it takes time of order N to evaluate a single sum S_m, then it takes time of order N^k to evaluate $S_{m_1,\ldots,m_k}$. This would make the error estimation very cumbersome.

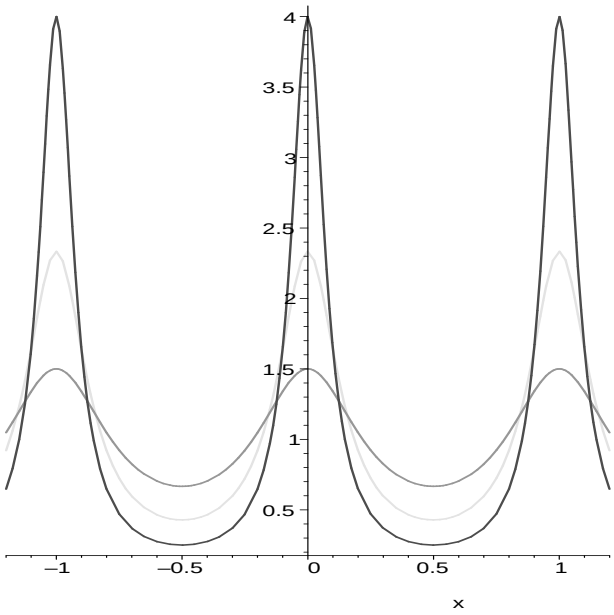
8.3 The Poisson summation formula

Here we give a derivation of the Poisson summation formula used in the description of the Jacobi diaphony, sect.4.6. Define the convergent sum

$$\Delta_\alpha(x) = \sum_n \alpha^{|n|} \exp(2i\pi n x) \quad , \quad -1 < \alpha < 1 \ . \tag{199}$$

By explicit summation we find

$$\Delta_\alpha(x) = \frac{1-\alpha^2}{1 - 2\alpha\cos(2\pi x) + \alpha^2} \ . \tag{200}$$



The function $\Delta_\alpha(x)$ for $\alpha = 0.2, 0.4$ and 0.6. As α increases, the function becomes more and more singular, and approaches zero inbetween the peaks.

We find that

$$\lim_{\alpha \to 1} \Delta_\alpha(x) = 0 \quad , \quad x \text{ not integer} \ . \tag{201}$$

Moreover,

$$\int_{-1/2}^{1/2} dx \ \Delta_\alpha(x) = 1 \ , \tag{202}$$

since only the term with $n = 0$ contributes to this integral. Therefore, we have

$$\sum_n \exp(2i\pi nx) = \sum_k \delta(x-k) \ . \tag{203}$$

Now, consider any function $f(x)$ that has a Fourier transform. Assuming that the sum exists, we may write

$$\begin{aligned} \sum_k f(k) &= \int dx \ f(x) \ \sum_k \delta(x-k) \\ &= \int dx \ f(x) \ \sum_n \exp(2i\pi nx) = \sum_n g(n) \ , \end{aligned} \tag{204}$$

where

$$g(y) = \int dx \ f(x) \ \exp(2i\pi xy) \ . \tag{205}$$

References

[1] J.M. Hammersley and D.C. Handscomb, *Monte Carlo methods*, London, Methuen, 1964

[2] Fred James

[3] D.E. Knuth, *The art of computer programming, Vol.2: seminumerical algorithms*. Reading, Mass: Addison-Wesley, 1981.

[4] Achilleas & me.

[5] G. Marsaglia and A. Zaman, *Ann. Appl. Prob.* **1** (1991) 462.

[6] Luescher

[7] wozniakowski

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

[9] Kleiss-Pittau