INTRODUCTION TO THE MONTE CARLO METHOD*

A. Pelissetto
Dipartimento di Fisica, Univ. di Pisa
I.N.F.N., Sezione di Pisa

* *Lectures given at the*
2nd National Seminar of Theoretical Physics
Parma, 1-12 September 1992.

1 Uniform Random Number Generators.

An important ingredient in any Monte Carlo calculation is the random number generator. This is a small subroutine of the program which produces a sequence of numbers which are not at all random in the statistical sense but may have some properties which are similar to the properties of a truly random sequence.

A truly random number sequence can only be generated by a random physical process, for example radioactive decay, thermal noise in electronic devices, cosmic ray arrival time etc. However these methods are extremely unpractical and at present the longest sequence has been produced by Frigerio and Clark (N. A. Frigerio and N. Clark, Trans. Am. Nucl. Soc. 22 (1975) 283). They used a radioactive α-particle source and a high resolution counter turned on for periods of 20 ms. The average number of decays in this time interval is 24.315. When the count was odd they recorded a zero-bit, while when even a one-bit. To this sequence they applied subsequently a correction to eliminate the bias due to the fact that an even number of decays has a different probability with respect to an odd number, obtaining eventually 2.5×10^6 31-bit truly random numbers.

Exercise 1: By using the Poisson distribution compute the probability of a zero-bit and of a one-bit in the Frigerio and Clark experiment.

Exercise 2: Consider a random sequence of zeroes and ones, such that the probability of 1 is p and that of 0 $1-p$, with p unknown. Prove that the following method gives a new sequence where zeroes and ones have the same probability: Consider sequentially pairs of bits in the sequence; if the bits are equal reject them, otherwise accept the second one. Prove that the efficiency (percentage of accepted bits) of the method is $p(1-p)$.

This method is however not very efficient since it requires large tables which are difficult to store and long to prepare. Thus at the end of the 40s, with the introduction of computers, research began on the generation of random numbers using arithmetic operations. The idea was to produce integers $X_0, X_1, X_2, \ldots$ such that $0 \le X_n < 2^w$ where w is the word size of the computer using some rule of the form

$$X_{n+1} = f(X_n) \tag{1.1}$$

and then random numbers U_n with $0 \le U_n \le 1$ by

$$U_n = X_n/m\,. \tag{1.2}$$

There is a fairly obvious objection to this method: how can this sequence be random, since each number is completely determined by its predecessor? As J. Von Neumann said, "anyone who considers arithmetical methods to produce random numbers is in a state of sin". The answer is indeed that the sequence is not random at all, but it may appear to be random for the problem at hand if f is carefully chosen. We want to stress that the choice of f is very important and that this choice is problem-dependent. A generator which is random enough for a particular problem may be really bad for another one.

Example.

Suppose we want to generate a sequence of integers $X_0, X_1, X_2 \ldots$ in the range $0 \leq X_n < m$ using a rule $X_{n+1} = f(X_n)$ where $0 \leq f(x) < m$ for $0 \leq x < m$. Let us now choose the function f randomly, i.e. giving each of the m^m possible functions $f(x)$ equal probability. Let us now compute the probability that for whatever choice of X_0 there will be no n such that $X_{n+1} = X_n$. Indeed in this case we would have $X_k = X_n$ for every $k \geq n$ and thus the sequence would be eventually constant and thus generically the function f will not be acceptable for our purposes (we say generically because it may happen that the function f is nonetheless a good generator for a sufficiently large number of starting values). What we have to compute is the probability that $f(x) \neq x$ for all x. Now, given x, the probability that $f(x) \neq x$ is $(1 - 1/m)$. Thus the required probability is simply $(1 - 1/m)^m$ which goes to $1/e$ for $m \to \infty$. Thus 63% of the functions f produce for at least one X_0 a sequence which becomes eventually constant. Generically these functions will be poor generators of random numbers. Of course we want to exclude also functions which produce sequences with $X_{n+2} = X_n$ and so on. The lesson from this example is that only few $f(x)$ have any chance of producing decent random numbers: thus simple generators for which rigorous results are available are to be preferred to more "random" (that is more complicated) ones for which no theory exists as there is a high probability that these last methods generate series which are less random than those obtained through simpler, but better understood, algorithms.

Exercise 3: Consider a random number generator defined as in the previous example. Show that the series contains cycles, i.e. that, for a given X_0, there exist numbers λ, μ such that $X_0, X_1, \ldots, X_\mu, \ldots X_{\mu+\lambda-1}$ are distinct and $X_{n+\lambda} = X_n$ for $n \geq \mu$. λ is called the period of the sequence.

Prove that the following is a correct algorithm to determine λ and μ:

(a) Determine the first n such that $X_n = X_{2n}$;

(b) Determine the first number i such that $X_i = X_{n+i}$ and j such that $X_n = X_{n+j}$. Then $\mu = i$, $\lambda = j$.

(The main advantage of this algorithm consists in the absence of any memory requirement).

Finally show that the sequence $Y_0, Y_1 \ldots$ with $Y_{n+1} = f(Y_n)$ and $Y_0 = X_k$, $0 < k < \mu + \lambda$, has the same period as the original sequence for all k and that the lowest h for which $Y_h = Y_{h+\lambda}$ is $\max(0, \mu - k)$.

Apply these results to the middle-square method proposed by J. Von Neumann for four-digit decimal numbers. Here $m = 10^4$ and

$$f(x) = \text{mod}(\text{trunc}(x^2/10^2), 10^4) \tag{1.3}$$

where $\text{trunc}(x)$ is the largest integer such that $x \geq \text{trunc}(x)$. Compute λ and μ for every X_0.

At present the most widespread random number generators use the so-called linear congruential method, i.e. the rule

$$X_{n+1} = \text{mod}(aX_n + c, m) \tag{1.4}$$

Usually m is called the modulus, a ($0 \leq a < m$) the multiplier and c ($0 \leq c < m$) the increment. The modulus can in principle be arbitrary. However there is a very convenient choice which is $m = 2^w$ where w is the number of bits of a word of the computer. Indeed in this case the "mod m" operation does not require any division. Given m one must then choose a and c. An obvious property which can be required is that the generated sequence have the longest possible period, which is obviously m. Two necessary conditions can be determined very easily: a and c must be relatively prime to m.

Indeed if a is not relatively prime to m, then $a = pb$, $m = pq$, $c = pk + h$, with $p \neq 1$, $0 < b < q$, $0 \leq k < q$, $0 \leq h < p$. Then we can write

$$X_{n+1} = p \,\mathrm{mod}(bX_n + k\,, q) + h \tag{1.5}$$

implying that only numbers of the form $p\alpha + h$, $0 \leq \alpha < q$, are produced so that the sequence does not have maximal period.

To obtain the condition on c, let us notice that if the period is m all possible values will appear in the cycle and thus it is not restrictive to consider $X_0 = 0$. Then in this case

$$X_n = \mathrm{mod}\left[\left(\frac{a^n - 1}{a - 1}\right) c\,, m\right] \tag{1.6}$$

If c is not relatively prime to m and $p = gcd(m, c)$, X_n will be a multiple of p and the period will be at most m/p.

These two conditions are not however sufficient conditions. The maximal period congruential generators are completely known and are classified by the following theorem (Hull and Dubell, SIAM Review 4 (1962) 230):

Theorem 1: *A linear congruential generator with modulus m, multiplier $a \geq 1$ and increment c has period m if and only if:*

(i) *c is relatively prime to m;*

(ii) *$a - 1$ is a multiple of p for every prime p dividing m;*

(iii) *$a - 1$ is a multiple of 4, if m is a multiple of 4.*

In practical implementations where $m = 2^w$ this theorem requires c to be odd (and one usually considers $c = 1$ or a small odd number) and $a = 4n + 1$, for some integer n.

The previous theorem gives the conditions which ensure maximal period. Of course this property alone is not enough to guarantee the goodness of the generator. A second important quantity which must be investigated is the potency of the multiplier. It is defined as the least integer s such that

$$\mathrm{mod}((a - 1)^s\,, m) = 0 \tag{1.7}$$

Such an integer exists if $(a - 1)$ satisfies the condition (ii) of the previous theorem and m is not prime. It is easy to see that when the potency is low the sequence $\{X_n\}$ is not very random. Let us notice first of all that $s = 1$ implies $a = 1$. This is a very poor generator as

$$X_{n+1} - X_n = c + km, \tag{1.8}$$

$k = 0, -1$, meaning that successive couples (X_0, X_1), (X_1, X_2)... lie on two lines contained in the square $[0, m[\times[0, m[$.

Let us now suppose $s = 2$. Consider the sequence starting from $X_0 = 0$. Then

$$X_n \,=\, \mathrm{mod}\left[\left(\frac{a^n-1}{a-1}\right)\; c, m\right] \,=\, \mathrm{mod}\left(nc + \frac{1}{2}n(n-1)c(a-1)\,, m\right) \tag{1.9}$$

It follows

$$\mathrm{mod}(X_{n+1} - X_n\,, m) \,=\, \mathrm{mod}(c + nc(a-1)\,, m) \tag{1.10}$$

This relation implies that successive couples (X_0, X_1), (X_1, X_2), ... lie on parallel lines contained in the square $[0, m[\times[0, m[$. This is not completely unexpected. Indeed truly random numbers would obviously belong to the lines

$$X_{n+1} - X_n \,=\, q \tag{1.11}$$

with $1 - m \le q \le m - 1$. However in this case the possible values of q are much less than $2m - 1$. Indeed it is easy to see that this number is $2m/$g.c.d. $(m, a-1)\; - 1$.

Even worse is the situation for triples (X_0, X_1, X_2), (X_1, X_2, X_3), ... as

$$\mathrm{mod}(X_{n+2} - 2X_{n+1} + X_n\,, m) \,=\, \mathrm{mod}(c(a-1)\,, m) \tag{1.12}$$

implying that they will lie on the four planes

$$X_{n+2} - 2X_{n+1} + X_n \,=\, \mathrm{mod}(c(a-1)\,, m) \,+\, km \tag{1.13}$$

with $k = -2, -1, 0, 1$.

The fact that successive d-tuples from a congruential generator lie on a certain finite number of parallel hyperplanes in d-dimensional space was firstly proved by Marsaglia who also proved that the maximum number of such hyperplanes is $(d!\; m)^{1/d}$. This means that if $m = 2^{32}$, the triples lie on at most 2953 planes, the 4-tuples, the 6-tuples and the 10-tuples on at most 566, 120 and 41 hyperplanes respectively.

The potency of a generator is strictly connected with this effect: the higher the potency, the higher is the number of hyperplanes on which successive d-tuples lie. As a rule-of-thumb generators with potency less than 5 must be rejected as not sufficiently random. When $m = 2^w$ it is easy to give a condition on a ensuring high potency. Notice that because of Theorem 1 we can have either $a \bmod 8 = 5$ or $a \bmod 8 = 1$, i.e. either $(a-1) \bmod 8 = 4$ or $(a-1) \bmod 8 = 0$. In the first case the potency is $w/2$ if w is even , $(w+1)/2$ if w is odd, while in the second case we have $s \le (w+2)/3$. Thus in order to obtain maximal potency the multiplier a must have the form $a = 8n + 5$.

Exercise 4: Consider a linear congruential generator with $m = 2^w$, $a = 2^k + 1$, $2 \le k < w$, $c = 1$. Prove that it has maximal period. If $w = 32$ compute the potency for all possible values of k, showing that a potency greater than four is achieved only if $k \le 6$, i.e. for small multipliers. However small values of a must be avoided as they give rise to high serial correlations and thus this family of a's does not give rise to acceptable generators. (Notice that this choice of a is particularly appealing as X_{n+1} can be computed without any multiplication (why?) and for this reason generators of this type were of widespread use in the 50s).

Let us finally discuss some other properties of the sequences obtained from a linear congruential generator. First of all let us notice that if $m = 2^w$ the sequence cannot be used as a generator of random bits as the right-hand digits of X_n are much less random than the left-hand digits. Indeed for every divisor d of m, if $Y_n = \mathrm{mod}(X_n\, , d)$, then

$$Y_{n+1} \;=\; \mathrm{mod}\,(a_d Y_n + c_d, d) \tag{1.14}$$

where $a_d = \mathrm{mod}(a, d)$ and $c_d = \mathrm{mod}(c, d)$.

Taking $d = 2^k$ we obtain that the lowest k-bits of X_n form a sequence whose period is 2^k or less. In particular the last bit is constant or alternating.

Exercise 5: Consider the following generators:

RAN supplied by Digital on its computers with $m = 2^{32}$, $a = 69069$ and $c = 1$;

RAND the standard Unix generator with $m = 2^{31} - 1$, $a = 1103515245$, $c = 12345$;

DRAND48 provided by IBM on its RISC machines with $m = 2^{48}$, $a =$5DEECE66D$_{16}$ and $c = B_{16}$.

For each generator compute the period and the potency.

Hint: note that $2^{31} - 1$ is a prime number (Marsenne prime).

Let us now pass to discuss other possible random number generators. Another popular form is simply

$$X_{n+1} \;=\; \mathrm{mod}\,(a X_n, m) \tag{1.15}$$

These generators are linear congruential generators with $c = 0$. This makes these generators faster but reduces the period (as $c = 0$ is not relatively prime to m it cannot have maximal period according to Theorem 1). However with a proper choice of a one can still obtain sufficiently long periods. In the interesting case $m = 2^w$, $w > 3$, if $a \bmod 8 = 3, 5$ and X_0 is odd, the period is 2^{w-2} which is sufficiently large for many applications. An example of this class of generators is RANF, the standard generator on CRAY computers, which has $m = 2^{48}$, $a =$2875A2E7B175$_{16}$ and period 2^{46}.

Exercise 6: Consider the random number generator RANDU supplied by IBM on its first-generation computers; it is defined by

$$X_{n+1} \;=\; \mathrm{mod}\,(65539 X_n, 2^{31}) \tag{1.16}$$

Compute the period when X_0 is odd and prove the relation

$$9X_n \;-\; 6X_{n+1} + X_{n+2} \;=\; 0 \bmod 2^{31} \tag{1.17}$$

Show that triples (X_n, X_{n+1}, X_{n+2}) lie on at most 16 planes. Can this generator be a good source of random numbers?

Exercise 7: Prove the formula

$$X_{n+k} = a^k X_n + \frac{a^k - 1}{a - 1} c \bmod m \tag{1.18}$$

Use this result to invent a parallel random number generator.

A second important family of random number generators is based on the so-called primitive polynomials. We do not want to enter in the discussion of this subject which involves the theory of finite fields. What interests us is that it is possible to find integers k and l with $k < l$ such that the generator

$$X_n = X_{n-k} + X_{n-l} \bmod 2^w \tag{1.19}$$

where $n \geq l$ and and $X_0, \ldots, X_{l-1}$ not all even, has period at least $2^l - 1$. In general it is possible to prove that the period of the sequence $X_n \bmod 2$ is exactly $2^l - 1$ while the full sequence has period $2^f(2^l - 1)$ where $0 \leq f < w$. Because of the first property these generators are very efficient in generating random bits. However at present is not completely clear if , beside the long period, these sequences have also the other desirable properties of random numbers. A particular bad example is the Fibonacci sequence, with $k = 1$ and $l = 2$ with period $3 \times 2^{w-1}$ (see exercise 8). However generators with higher values of k and l ($l \gtrsim 100$) do not show these problems and they have proved very good under the spectral test.

Exercise 8: Consider the random number generator

$$X_n = X_{n-1} + X_{n-2} \bmod m \tag{1.20}$$

X_0 and X_1 not both even. Prove that the relation $X_{n-1} < X_{n+1} < X_n$ never holds.

Strictly related to this class of generators are the so-called lagged Fibonacci generators which have the form

$$X_n = X_{n-k} \,.\mathrm{BIN}.\, X_{n-l} \tag{1.21}$$

where .BIN. stands for every binary operation. An example of these generators is the KirkPatrick-Stoll generator

$$X_n = X_{n-103} \,.\mathrm{XOR}.\, X_{n-250} \tag{1.22}$$

with period $2^{250} - 1$ which, however, has been recently shown to perform badly in Monte Carlo simulations of the Ising model. Empirical tests show that these generators perform poorly if $l \lesssim 100$ especially if the binary operation is the $.XOR.$.

To conclude this discussion on random numbers we want to discuss the important concept of *accuracy* of a random number generator. The accuracy is connected with what we have already said, the fact that successive couples, triples ... lie in planes and do not fill the square $[0, m[^2$, the cube $[0, m[^3$... uniformly.

Let us firstly define the accuracy: given an integer t, consider the points $P_n = (X_n, \ldots X_{n+t})$ for $n \geq 0$ and consider an arbitrary family of parallel planes passing for all P_n and call d the distance between any two successive planes (this makes sense as the planes are equidistant). Then compute the maximum d_{max} of d taken over all families of such planes. The t-dimensional accuracy ν_t is then defined by $\nu_t = m/d_{max}$. For truly random numbers $\nu_t = m$ for all t while for a generic generator $\nu_t \lesssim m^{1/t}$. A good generator has $\nu_t \approx m^{1/t}$ for all t.

Let us now discuss the meaning of ν_t. Let us consider N t-tuples $P_n = (U_{tn+1}, \ldots U_{tn+t}) = (X_{tn+1}, \ldots X_{tn+t})/m$ belonging to the hypercube $[0,1]^t$ and a small cube C of side L belonging to $[0,1]^t$. Let us compute the number of points P_n which fall in C. For truly random numbers we expect this number to be L^t/N, but this will be true only if $Lm >> 1$, that is if $L >> 1/\nu_t$ no matter how large N is. Indeed if $L < 1/m$, since the points have the form n/m, the cube can only be either empty or with one point inside. Thus the accuracy defines the "spatial" resolution of the random sequence. Analogously, if the sequence is produced by a random number generator we must keep $L >> 1/\nu_t$ otherwise we see the "granularity" of the generator. Low accuracy means deviations from random behaviour on quite large scales and thus these generators must be avoided.

Exercise 9: Suppose you want to get with uniform probability couples of integer numbers (x_n, y_n) with $0 \le x < k$ and $0 \le y < k$ and suppose you use a random number generator with modulus m and two-dimensional accuracy ν_2 in the following way: $x_n = \mathrm{trunc}(kX_{2n}/m)$ and $y_n = \mathrm{trunc}(kX_{2n+1}/m)$. What are the values of k for which deviations from equidistribution will certainly appear? And in the case of triples ? The Digital 32-bit standard random number generator (see Exercise 5) has $\nu_2^2 = 4243209856$ and $\nu_3^2 = 2072544$. Show that it can be safely used to generate couples only for $k < 10^3$ and triples for $k < 100$.

Exercise 10: Prove that for a random number generator of potency 2 we have $\nu_3^2 \le 6$ while RANDU satisfies $\nu_3^2 \le 118$. (Hint: use the relations proved in the text and in Exercise 6).

Exercise 11: A method to improve the accuracy of a generator is the shuffling method of Bays and Durham. Consider a random number generator which produces a sequence $X_0, X_1, \ldots$. Fix then a number k and initialize an auxiliary vector $V[0], \ldots, V[k-1]$ with $V[j] \leftarrow X_j$. Set $y \leftarrow k$, $j \leftarrow 0$. The algorithm is the following:

1. Compute $r = \mathrm{trunc}(kX_y/m)$ and set $Y_j \leftarrow V[r]$.

2. Set $V[r] \leftarrow X_{y+1}$, $y \leftarrow y+2$, $j \leftarrow j+1$.

The sequence $Y_0, Y_1, \ldots$ is the output of the routine.

Consider as an explicit example $X_{n+1} = \mathrm{mod}(69X_n + 1,\ 128)$, $X_0 = 0$, $k = 3$. Determine μ and λ (see Exercise 3 for the definition) for the output sequence and the number of distinct couples (x, y) such that $x = Y_j$, $y = Y_{j+1}$. Repeat the same exercise for the original sequence X_n.

2 Numerical Distributions.

We have seen in the previous Section how to generate uniform random numbers. However in many applications one needs random numbers with other types of distribution. In general we want to produce numbers $x \in [a, b]$ with probability density $f(x)$, i.e. such that

$$\mathrm{Prob}\,[x_1 < X < x_2] \;=\; \int_{x_1}^{x_2} f(x)\mathrm{d}x \tag{2.23}$$

There are various methods to obtain the correct distribution. The simplest one works as follows: define the distribution function $F(x)$ which is the probability that the random variable X does not exceed x:

$$F(x) \;=\; \mathrm{Prob}\,[X < x] \;=\; \int_{a}^{x} f(x)\mathrm{d}x \tag{2.24}$$

$F(x)$ is an increasing function with $F(a) = 0$, $F(b) = 1$. If $F(x)$ is continuous and strictly increasing there exists an inverse function $F^{-1}(x)$. Then to generate a random number X with distribution $F(x)$ one simply generates a uniform random number U between 0 and 1 and computes $X = F^{-1}(U)$. To prove that this is correct notice that

$$\mathrm{Prob}\,[X < x] \;=\; \mathrm{Prob}\,[F^{-1}(U) < x] \;=\; \mathrm{Prob}\,[U < F(x)] \;=\; F(x) \tag{2.25}$$

Exercise 1: Use this method to generate random numbers with density function:

1. $f(x) = px^{p-1}$ in the interval $[0, 1]$;
2. $f(x) = \mu e^{-\mu x}$ for $x \geq 0$;
3. $f(x) = 2xe^{-x^2}$ for $x \geq 0$.

Exercise 2: Develop an algorithm which computes two independent normally distributed variables X_1 and X_2.

Hint: notice that the couple (X_1, X_2) have density $f(x_1, x_2) = e^{-(x_1^2+x_2^2)/2}/2\pi$ which is easy to generate in polar coordinates.

This method can be applied in a limited number of cases since it requires the explicit computation of $F^{-1}(x)$. A more general technique is Von Neumann's rejection method. Suppose you want to generate a random variable with probability density $f(x)$ such that $F^{-1}(x)$ does not have a simple closed form. Choose then a second probability density $g(x)$ simple enough to allow a quick generation of random numbers distributed according to it and such that $f(x) \leq cg(x)$ for all x. Here c is a constant which in principle can be arbitrarily chosen as long as the bound is satisfied (thus in all cases $c > 1$). However, in order to obtain an efficient algorithm c must be as small as possible. Then the algorithm works as follows:

1. Generate X according to the density $g(x)$ and a uniform random number U between 0 and 1;

2. If $U \geq f(X)/cg(X)$ go back to step 1 and repeat with a new X and U; otherwise output X.

The probability of rejection is clearly

$$\int \left(1 - \frac{f(x)}{cg(x)}\right) g(x)\mathrm{d}x \; = \; 1 - \frac{1}{c} \qquad (2.26)$$

This means that step 1 will be executed c times on average (with standard deviation $\sqrt{c(c-1)}$, prove it!) and thus a good generator requires c not to be very far from 1.

The idea of the method is very simple. If X is a random variable with distribution $g(x)$ and U is uniform between 0 and 1, then the couple (x, y) with $x = X$, $y = cUg(X)$ is uniformly distributed in the plane region $R = \{(x, y) : \; a \leq x \leq b, 0 \leq y \leq cg(x)\}$. Then, in order to obtain X distributed with density $f(x)$ we accept the points such that $y < f(x)$ and reject the others. The accepted points are uniformly distributed in the region $R' = \{(x, y) : \; a \leq x \leq b, 0 \leq y \leq f(x)\}$ and thus X is distributed with probability density $f(x)$.

Exercise 3: Use the rejection method to generate :

1. $f(x) = \sqrt{2/\pi}\; e^{-x^2}$ for $x \geq 0$;
2. $f(x) = x^{a-1}e^{-x}/\Gamma(a)$, $a > 1$ for $x \geq 0$;

In the first case use a function $g(x)$ of the form $g(x) = A$ for $0 \leq x \leq p$, $g(x) = (A/p)x\exp(p^2 - x^2)$ for $x \geq p$. In the second case choose $g(x) = Ax^{a-1}$ for $0 \leq x \leq a-1$, $g(x) = Bx^{-p}$ for $x \geq a-1$.

Compute the acceptance in both cases and determine the optimal value of p.

Exercise 4: A simple method to generate random unit vectors in d-dimensional space is the following:

1. Generate $U_1, \ldots, U_d$ uniform random numbers between -1 and 1 and compute $r^2 = U_1^2 + U_2^2 + \ldots + U_d^2$;

2. If $r > 1$ go back to step 1; otherwise the required vector is $(U_1/r, \ldots, U_d/r)$.

Explain why this method is correct and compute the acceptance.

Exercise 5: Suppose you want to produce random numbers x in $(-1, 1)$ with distribution

$$\frac{1}{\pi}(1 - x^2)^{-1/2}\,\mathrm{d}x \qquad (2.27)$$

Show that both these methods are correct:

Method A: generate a uniform random number $U \in [0, 1]$, then compute $x = \sin\pi(2U - 1)$.

Method B: generate two uniform random numbers U and V in $[0,1]$. Reject them if $U^2+V^2>1$. Otherwise compute

$$x = \frac{U^2-V^2}{U^2+V^2} \tag{2.28}$$

Method B has the advantage of requiring only elementary operations.

Exercise 6: Consider the following algorithm. Pick U_1 and U_2 uniformly in $[0,1]$. If $U_1 \leq U_2$ stop. Otherwise select U_3 and stop if $U_3 \geq U_2$. Otherwise continue this process until you have $U_1 > U_2 > \ldots U_n$ and $U_n \leq U_{n+1}$. If n is odd define $X = U_1$, while if n is even reject all U_i and start again. Show that $X \in [0,1]$ is distributed with probability density proportional to e^{-X}.

Hint: prove firstly that

$$\text{Prob}\left(U_1 > U_2 \ldots > U_n\right) = \frac{1}{n!} \tag{2.29}$$

and

$$\text{Prob}\left(x < U_1 < x + \mathrm{d}x \,|U_1 > \ldots > U_n\right) = nx^{n-1}\mathrm{d}x \tag{2.30}$$

Let us finally discuss the so called binning (or stratified generation) method. This technique is a simple extension of the rejection method. The idea is very simple: suppose you want to generate random numbers X with probability density $f(x)$, $a \leq x \leq b$. Then rewrite

$$f(x) = \sum_{k=1}^{n} p_k f_k(x) \tag{2.31}$$

where

$$p_k = \int_{a_k}^{a_{k+1}} f(x)\,\mathrm{d}x \tag{2.32}$$

and

$$f_k(x) = \frac{1}{p_k} f(x)\chi([a_k, a_{k+1}]) \tag{2.33}$$

Here $\chi([\alpha,\beta])$ is the characteristic function of the interval $[\alpha,\beta]$ and $a = a_1 < a_2 \ldots < a_n < a_{n+1} = b$.

Thus the generation of X can be obtained by firstly choosing k with probability p_k and then computing $X \in [a_k, a_{k+1}]$ with probability density $f_k(x)$. If the constants a_k are properly chosen the generation of $X \in [a_k, a_{k+1}]$ can be done using the rejection method with simple trial functions, in most cases a constant $g(x)$ will be efficient enough.

In order to apply this technique a fast way of choosing k with probability p_k is needed. An ingenious trick is the so-called method of aliases. Suppose you want to generate a random number X such that $X = x_0$ with probability p_0, $X = x_1$ with probability p_1, ..., $X = x_{n-1}$ with probability p_{n-1}. Let us introduce three auxiliary arrays $P[k]$, $Q[k]$ and $Y[k]$, $k = 0, \ldots, n-1$. Initialize $Q[k] \leftarrow np_k$. Then define $P[k]$ and $Y[k]$ using the following algorithm:

1. Find k_1 such that $0 < Q[k_1] \leq 1$. Set $P[k_1] \leftarrow Q[k_1]$;

2. If $Q[k_1] = 1$ go to step 4;
3. Find m_1 such that $Q[m_1] > 1$. Set $Y[k_1] \leftarrow x_{m_1}$ and $Q[m_1] \leftarrow Q[m_1] + P[k_1] - 1$;
4. Set $Q[k_1] \leftarrow 0$. If $Q[k] = 0$ for all k exit.

Then the generation algorithm works as follows: choose two random numbers U and V uniformly distributed between 0 and 1. Let i be the integer part of nU. Then, if $V \le P[i]$ set $X \leftarrow x_i$ otherwise $X \leftarrow Y_i$. It is easy to check that each x_i is generated with the correct probability.

3 Monte Carlo Integration.

The Monte Carlo method is a very powerful technique for performing very complicated calculations. In general a Monte Carlo method is any technique that makes use of random numbers to solve the problem. The most part of Monte Carlo calculations are to all effect equivalent to an integration problem. In this Section we will discuss the use of Monte Carlo in low-dimensional integration problems. It must be clear from the beginning that Monte Carlo is generically a very bad integration method. As we shall see the error on the estimate decreases as $1/\sqrt{n}$ where n is the number of points where the function is evaluated independently on the dimensionality d of the integral (this is essentially the central limit theorem). This should be contrasted with traditional deterministic numerical methods: for instance the trapezoidal rule converges as $n^{-2/d}$, the Simpson's rule as $n^{-4/d}$ and the m-point Gauss rule as $n^{-(2m-1)/d}$. Thus it is clear that for $d = 1$ any of these algorithms will be better than Monte Carlo , while for large d Monte Carlo beats any of them. Thus Monte Carlo is the best available method when d is large enough. But there are other disadvantages in using a deterministic method. Suppose for instance you want to apply the Simpson rule in 6 dimensions. It converges as $n^{-3/4}$, faster than Monte Carlo. Let us use 50 nodes per axis (which is a very coarse mesh); then we need at least 50^6 function evaluations which is extremely time-consuming especially if the function is complicated. This brings up two new points:

(i) the feasibility limit, which is the largest number of function evaluations we can efford to make. Typically this limits the number of points to $10^{10} - 10^{15}$. This fact limits the use of higher-order rules to low dimensions and imply that the feasibility limit is reached long before the crossover point where Monte Carlo converges faster than quadrature, so that the theoretical convergence rates of higher-order rules in high dimensionalities is of pure theoretical interest.

(ii) the growth rate is the smallest number of additional function evaluations needed to improve the current estimate. A Monte Carlo result can be improved by adding a single point, while any other deterministic rule can only be improved by going to a higher-order rule or by subdividing the space.

There is one final advantage of Monte Carlo integration. Deterministic rules usually work reasonably well for multidimensional regions of simple shape while they are of difficult application when the region of integration is irregular. Monte Carlo on the contrary can be applied to any situation independently from the shape of the boundary.

Let us now discuss how to perform integration with Monte Carlo. The least efficient method is the so-called hit-or-miss method. Suppose you want to integrate

$$I = \int_a^b f(x)\,\mathrm{d}x \tag{3.34}$$

Let $m \leq \min_{x\in[a,b]} f(x)$ and $M \geq \max_{x\in[a,b]} f(x)$. Then use the following algorithm:

1. Set $n \leftarrow 1$;

2. Choose two uniform deviates U_{2n} and U_{2n+1} between 0 and 1;

3. Set $x \leftarrow (a + (b-a)U_{2n})$ and $y \leftarrow (M-m)U_{2n+1}$;

4. If $f(x) > y + m$ set $p_n \leftarrow 1$; otherwise $p_n \leftarrow 0$;

5. If $n \geq N$, where N is the required number of iterations, stop; otherwise set $n \leftarrow (n+1)$ and go to 2.

Then

$$I = \left(\frac{1}{N}\sum_{n=1}^{N} p_n\right)(b-a)(M-m) + m(b-a) \tag{3.35}$$

(with a slight abuse of notation we have used I both for the exact integral and its estimate, since no confusion can arise). The idea behind this algorithm is very simple. Suppose for a moment $f(x) \geq 0$ for $a \leq x \leq b$ and consider the smallest rectangle R which contains the region $G = \{(x,y): \ a \leq x \leq b, 0 \leq y \leq f(x)\}$. Then pick up points in R with uniform probability. The percentage of points which belong also to G is equal to the ratio between the area of G and the area of R. Thus the required integral is equal to the percentage of hits times the area of R. If the function can be negative some care must be exercised and in this case the same method can be applied to $\hat{f}(x) = f(x) - \min f(x)$.

Exercise 1: Compute the variance of I. For the explicit case $f(x) = \cos x$, $a = 0$, $b = \pi/2$ compute N in order to get 1% accuracy.

Hint: the number of hits follows a binomial distribution.

Exercise 2: Consider the following method to compute π (this method is due to Buffon, 1777). Lay out on the floor a pattern of parallel lines separated by a distance d. Repeatedly throw randomly a needle of length d onto this striped pattern. Each time the needle lands in such a way as to cross the boundary between two stripes, count a hit, otherwise count a miss. Prove that π can be estimated as twice the number of trials divided by the number of hits.

A second method which is slightly better than the previous one is the so-called crude Monte Carlo. To compute I we use the following algorithm:

1. Initialize $n \leftarrow 1$.

2. Choose a uniform random number U_n between 0 and 1.

3. Set $x \leftarrow a + (b-a)U_n$ and $f_n \leftarrow f(x)$.

4. If $n \geq N$ where N is the required number of iterations, stop; otherwise set $n \leftarrow (n+1)$ and go to 2.

Then

$$I = \frac{b-a}{N} \sum_{n=1}^{N} f_n \tag{3.36}$$

The idea is even simpler: the numbers x are uniformly distributed between a and b, i.e. have probability measure $\mathrm{d}\mu = \mathrm{d}x/(b-a)$. Thus

$$I = (b-a) \int \mathrm{d}\mu\, f(x) = (b-a)\langle f \rangle \tag{3.37}$$

Exercise 3: Compute the variance of the previous formula. For the explicit case $f(x) = \cos x$, $a = 0$, $b = \pi/2$ compute N in order to get 1% accuracy. Compare with Exercise 1.

The crude Monte Carlo method can easily be extended to multidimensional integrals and can deal in a straightforward way with essentially any finite region. The standard technique for dealing with odd-shaped regions is to embed the domain in the smallest hyperrectangle that surrounds it. The algorithm works as follows:

1. Initialize $n \leftarrow 1$, $p \leftarrow 0$.
2. Choose a uniform random point x in the hyperrectangle.
3. If x belongs to the integration domain set $p \leftarrow (p+1)$, $f_p \leftarrow f(x)$.
4. If $n \geq N$ where N is the required number of iterations, stop; otherwise set $n \leftarrow (n+1)$ and go to 2.

Then

$$I = \frac{V}{N} \sum_{i=1}^{p} f_i \tag{3.38}$$

where V is the volume of the hyperrectangle.

Exercise 4: Prove the previous formula and compute its variance. Note that both f_i and p are random variables.

This method of integration introduces some inefficiency due to the rejected points, but its main property is its full generality. It can also be easily improved by using the stratified sampling technique. The idea here is to write the integral

$$I = \int_R f(x)\, \mathrm{d}^d x \tag{3.39}$$

as a sum of integrals over smaller subdomains R_i which have the following two properties:

1. $f(x)$ is slowly varying in each R_i;
2. R_i has a somewhat regular shape.

Then the Monte Carlo method is applied to each R_i separately, carefully choosing the number of iterations performed in each R_i. In such a way one usually obtains a sensible improvement in the variance.

Exercise 5: In order to compute the integral $\int_0^{\pi/2} \cos x \, dx$ use the stratified sampling technique with $R_1 = [0, \pi/4]$ and $R_2 = [\pi/4, \pi/2]$. If the total number of iterations is fixed equal to N, determine the optimal number of iterations n_1 and n_2 which give the best possible variance with this choice of R_i. Compute N in order to have a 1% accuracy. Compare with the results of the exercises 1 and 3.

A second method which is used to reduce the error in a Monte Carlo is the so-called importance sampling technique. The idea is very simple. Since large variations in the value of the function f lead to a large uncertainty in the final estimate, the idea is to perform a change of integration variables in such a way that in the new variables the function to be integrated is more constant. In practice this means rewriting

$$I = \int_a^b f(x)\mathrm{d}x = \int_a^b \frac{f(x)}{g(x)} g(x)\mathrm{d}x \tag{3.40}$$

with $g(x) \geq 0$, $\int_a^b g(x)\mathrm{d}x = 1$. The algorithm works as follows:

1. Initialize $n \leftarrow 1$;
2. Choose a random deviate X with probability density $g(x)$;
3. Set $h_n \leftarrow f(X)/g(X)$;
4. If $n \geq N$ where N is the required number of iterations, stop; otherwise set $n \leftarrow (n+1)$ and go to 2.

Then

$$I = \frac{1}{N}\left(\sum_{i=1}^{N} h_i\right) \tag{3.41}$$

The function $g(x)$ must be chosen such that the ratio $f(x)/g(x)$ be as nearly constant as possible in order to obtain a small variance. The importance sampling technique is a very useful one, especially when $f(x)$ has singular points. However it a has a serious drawback: it requires the generation of random numbers with probability density $g(x)$ and the class of functions for which this can be done easily is really small as we have seen in the previous Section. To overcome this difficulty a new Monte Carlo method is needed and this will be the subject of the next Section.

Exercise 6: In order to compute $\int_0^{\pi/2} \cos x \, dx$ use the important sampling technique with $g(x)$ proportional to $a + bx^2$. Determine the optimal a and b, an algorithm to generate $g(x)$ and the number of iterations needed to get a 1% accuracy.

Exercise 7: Suppose you want to perform the following integral:

$$I = \int_0^1 \mathrm{d}x \int_0^x \mathrm{d}y \, g(x, y) \tag{3.42}$$

Consider the following methods:

Method A: set $g_i \leftarrow g(U, UV)$ where U and V are uniform random numbers belonging to $[0,1]$ and $I = (1/N)\sum_{i=1}^{N} g_i$.

Method B: choose U and V uniformly in $[0,1]$; if $U < V$ reject them and try again; stop when $U > V$. Then set $g_i \leftarrow g(U,V)$ and $I = (1/N)\sum_{i=1}^{N} g_i$.

Method C: choose U and V uniformly in $[0,1]$; if $U < V$ interchange them. Then set $g_i \leftarrow g(U,V)$ and $I = (1/N)\sum_{i=1}^{N} g_i$.

Method D: choose U and V uniformly in $[0,1]$; then set $g_i \leftarrow Ug(U,UV)$ and $I = (1/N)\sum_{i=1}^{N} g_i$.

Which of these methods is correct? Which is the most efficient one when $g(x,y) = x^2 + y^2$ and when $g(x,y) = (1-x)(1-y)$?

To conclude this Section we want to stress that all the previous results can also be applied to multidimensional sums, i.e. to

$$I = \sum_{n_1,\ldots,n_d} f(n) \tag{3.43}$$

The hit-or-miss method can be applied using a function $\hat{f}(x_1,\ldots,x_d) = f(\mathrm{trunc}(x_1), \ldots, \mathrm{trunc}(x_d))$ where $\mathrm{trunc}(x)$ is the largest integer such that $x \geq \mathrm{trunc}(x)$. The crude Monte Carlo method corresponds instead in choosing successive d-tuples $(n_1,\ldots,n_d)$ uniformly, computing $f_i = f(n_1,\ldots,n_d)$ and then estimating I by $\sum_{i=1}^{N} f_i$. The stratified sampling and the importance sampling technique can also be extended to this case.

4 Dynamic Monte Carlo.

In the previous Section we have discussed the so-called static Monte Carlo that generates a sequence of statistically independent samples from the desired probability distribution. These techniques become rapidly inefficient as the dimension of the space increases and thus they are unfeasible for most applications in statistical mechanics and quantum field theory. In these cases one uses the so-called dynamic Monte Carlo.

In lattice quantum field theories and statistical mechanics one wants to compute quantities of the form

$$\langle \mathcal{O} \rangle = \frac{1}{Z} \int \Pi_i \mathrm{d}\phi_i \; \mathcal{O} \; e^{-\beta H(\phi)} \tag{4.44}$$

where ϕ_i are the basic fields of the theory, i running over the lattice sites, $\mathcal{O}$ is a generic observable and Z the partition function

$$Z = \int \Pi_i \mathrm{d}\phi_i \; e^{-\beta H(\phi)} \tag{4.45}$$

The fields ϕ_i can be either continuous or discrete. In the latter case the integrals are replaced by sums. For instance for the Ising model

$$\langle \mathcal{O} \rangle \; = \; \frac{1}{Z} \sum_{\{\sigma\}} \mathcal{O}(\sigma) \exp\left(\beta \sum_{\langle ij \rangle} \sigma_i \sigma_j\right) \tag{4.46}$$

where $\sum_{\{\sigma\}}$ indicates the sum over all possible configurations of spins, each spin σ_i assuming the values ± 1.

To compute $\langle \mathcal{O} \rangle$ one could try to use the methods presented in the previous Section. For instance, in the Ising case, we could generate successively random configurations $\{\sigma\}_{i=1\ldots N}$ where each σ_i is obtained by choosing randomly the spin at each site. Then

$$\langle \mathcal{O} \rangle \; = \; \frac{\sum_{i=1}^{N} \mathcal{O}(\{\sigma\}_i) e^{-\beta H(\{\sigma\}_i)}}{\sum_{i=1}^{N} e^{-\beta H(\{\sigma\}_i)}} \tag{4.47}$$

This method is certainly correct, but is totally inefficient. Indeed from statistical mechanics we know that the configurations which give the relevant contribution to (4.46) have energy E such that $E_m - \Delta E \le E \le E_m + \Delta E$ where E_m is the average value of the energy at the given β; ΔE can be taken as some multiple of the standard deviation of E_m and has the property of going to zero as the volume of the system goes to infinity (self-averaging property of the energy). Now in the sum (4.47) we are trying to estimate $\langle \mathcal{O} \rangle$ at inverse temperature β using configurations distributed according to the Gibbs measure at $\beta = 0$. Thus if β is not very small the configurations we are using are not the configurations which dominate the sum and thus the estimate is completely unreliable .

The way out is the importance sampling technique. We produce configurations distributed with probability $\pi(\{\sigma\}) \; = \; \exp(-\beta H)/Z$ and then we use $\sum_{i=1}^{N} \mathcal{O}(\{\sigma\}_i)/N$ to estimate $\langle \mathcal{O} \rangle$. The problem here is that we do not know how to generate independent configurations from the given probability π. However as we shall see there are fairly simple methods which generate correlated configurations from the given probability. These methods are called dynamic Monte Carlo.

Let us introduce some notation: we will call the state space where the fields live S and the probability measure π. S can be either continuous or discrete: in the following we will suppose S to be discrete (as in the Ising model) but all formulas extend easily to the continuous case substituting sums with integrals, matrices with kernels .....π is a generic probability measure: in statistical mechanics and lattice quantum field theory applications π will be the Gibbs measure $\exp(-\beta H)/Z$.

Let us now discuss the dynamic Monte Carlo methods. In this case we give up the idea of producing statistically independent points in S. Instead the choice of the new point depends on the previous one. Of course this has to be done in such a way that the final points are distributed with probability measure π. To accomplish this task the idea is to invent a stochastic process (a Markov chain usually) having π as equilibrium distribution. Let us firstly define a Markov chain:

Definition: *A Markov chain with state space S is a sequence of S-valued random variables $X_0, X_1, X_2, \ldots$ such that the successive transitions $X_t \rightarrow X_{t+1}$ are statistically independent.*

This means that in a Markov chain the probability that $X_t \to X_{t+1}$ depends explicitly only on X_t and not on X_τ with $\tau < t$. Thus the whole process is completely defined by the one-step transition probability matrix $P = \{p_{ij}\}_{ij\in S}$. As p_{ij} is the probability of the transition $i \to j$ we must have $p_{ij} \geq 0$ for all i, j and $\sum_{j\in S} p_{ij} = 1$. Matrices satisfying these two conditions are called stochastic matrices.

A Markov chain is said to be irreducible if from each state it is possible to get to each other state: that is, if, for each pair $i, j \in S$, there exists an $n \geq 0$ such that the probability $\mathrm{Prob}(X_{t+n} = j | X_t = i) = (P^n)_{ij}$ is strictly positive. An irreducible chain is said to have period T if T is the g.c.d. of $\{n : P^n(i,i) > 0\}$ for every state i. A chain with period 1 is said to be aperiodic.

Exercise 1: Consider Markov chains with transition probability matrices:

$$P_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} . \tag{4.48}$$

$$P_2 = \begin{pmatrix} 1/2 & 1/2 \\ 1 & 0 \end{pmatrix} . \tag{4.49}$$

$$P_3 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1/4 & 0 & 3/4 \end{pmatrix} . \tag{4.50}$$

Are they irreducible? If they are, compute the period.

To completely define a Markov chain we must also define the probability distribution of X_0. Usually we consider two cases:

1. the Markov chain starts from a state i, i.e. $\mathrm{Prob}(X_0 = j) = \delta_{ij}$;

2. the Markov chain is in "equilibrium". In this case $\mathrm{Prob}(X_0 = j) = \pi(j)$ where $\pi(j)$ is the equilibrium distribution we will discuss later in this Section.

The transition probability matrix P and the probability distribution of X_0 (let us call it $\alpha(j)$) completely define the probability distribution of the random variables X_t and of the functions thereof. Moreover they induce a probability measure on the set of n-step chains through

$$\mathrm{Prob}(X_0 = i_0, X_1 = i_1, \ldots, X_n = i_n) = \alpha(i_0)\, p_{i_0 i_1} p_{i_1 i_2} \cdots p_{i_{n-1} i_n} \tag{4.51}$$

Using this probability measure one can compute mean values of functions of the $\{X_t\}$. We will indicate with $\langle \ldots \rangle_\alpha$ the mean value when the Markov chain is started with initial distribution α, omitting the subscript when the chain is in equilibrium ($\alpha = \pi$).

Let us now study the asymptotic properties of Markov chains. The standard theory tells us the following about the long-term behaviour of an aperiodic irreducible chain. Firstly the limit

$$\lim_{n\to\infty} (P^n)_{ij} \tag{4.52}$$

exists for every i and j in S and is independent of i; call it $\pi(j)$. Next, if S is finite, then

$$\sum_{j \in S} \pi(j) = 1 \tag{4.53}$$

and

$$\sum_{i \in S} \pi(i) p_{ij} = \pi(j) \tag{4.54}$$

for all $j \in S$. Moreover π is the only non-negative solution to (4.53) and (4.54). In the general case there are two possibilities: $\pi(j) = 0$ for all j and in this case no solution to (4.53) and to (4.54) exists, or $\pi(j)$ is the unique nonnegative solution to the previous equations. If π exists the chain is called positive recurrent or ergodic and π is called the equilibrium or stationary distribution.

All these results extend to periodic irreducible chains. In this case the limit (4.52) does not exist, however one can prove that the same results hold true taking the mean limit of P^n.

Exercise 2: For each of the matrices defined in Exercise 1, compute P^n, its limit (or mean-limit) for $n \to \infty$ and all the nonnegative solutions to the equations (4.53) and (4.54).

The probability π represents the fraction of time that the chain spends in each state in the long run irrespective of the initial state (law of large numbers). To state it precisely, let $u_j^{(k)}$ be a function with value 1 if the k-th step of the chain is j and zero otherwise. If

$$f_j^{(n)} = \frac{1}{n} \sum_{k=1}^{n} u_j^{(k)} \tag{4.55}$$

we have

Theorem 1: *(Law of Large Numbers) If P is ergodic*

$$\lim_{n \to \infty} \langle f_j^{(n)} \rangle_\alpha = \pi(j) \tag{4.56}$$

and

$$\lim_{n \to \infty} \mathrm{Prob}_\alpha[|f_j^{(n)} - \pi(j)| > \epsilon] = 0 \tag{4.57}$$

for every $\epsilon > 0$, independently of the starting distribution α.

Exercise 3: For an aperiodic positive recurrent chain define $\Pi_{ij} = \pi(j)$ and assume that the matrix $Z = (I - P + \Pi)^{-1}$ exists. Prove

$$\lim_{n \to \infty} \langle n \left(f_j^{(n)} - \pi(j) \right) \rangle_\alpha = \sum_{i \in S} \alpha(i)(Z - \Pi)_{ij} \tag{4.58}$$

If $\alpha = \pi$ show that the limit is zero. This result shows that if $\alpha \neq \pi$, $f_j^{(n)}$ is a biased estimate of $\pi(j)$, the bias being of order $1/n$.

Hint: Notice that $Z = I + \sum_{k=1}^{\infty}(P^k - \Pi)$ and $\langle u_j^{(k)} \rangle_\alpha = \sum_i \alpha(i)(P^k)_{ij}$.

Thus, if the chain is positive recurrent and we observe it for a sufficiently long time then the data will be well distributed according to π and thus to estimate π-averages of an observable f we can consider Markov time averages. The rigorous result (ergodic theorem) tells us that for a real-valued function f defined on the state space S and a positive recurrent chain with stationary probability π we have

$$\lim_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} f(X_t) = \sum_{i\in S} f(i)\pi(i) \tag{4.59}$$

with probability one if the r.h.s converges absolutely. Moreover, because of the central limit theorem, the fluctuations are of order $1/\sqrt{n}$.

Exercise 4: Consider a maximal period random number generator with recursion $X_{n+1} = f(X_n)$. Show that it defines a Markov chain. Is it irreducible? And what is the period? Prove that $\pi(n) = 1/m$ satisfies (4.53) and (4.54). This implies that for every function g we have

$$\lim_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} g(X_t) = \frac{1}{m} \sum_{i=0}^{m-1} g(i) \tag{4.60}$$

Is this property of interest for the random number generation problem? (Answer: No)

As a final comment let us notice that static Monte Carlo can also be thought as a Markov chain. Consider indeed the transition matrix $p_{ij} = \pi(j)$. It is obvious from this definition that X_{t+1} does not depend on X_t and that it has the correct distribution.

After this introduction to the theory of Markov chains it is clear how to generate samples from the desired probability distribution π. It is enough to invent a transition probability matrix $P = \{p_{ij}\}$ which satisfies the following two conditions:

(A) Irreducibility: For each pair $i, j \in S$ there exists $n \geq 0$ such that $(P^n)_{ij} > 0$.

(B) Stationarity of π. For each $i \in S$

$$\sum_{j\in S} \pi(j) p_{ji} \;=\; \pi(i) \tag{4.61}$$

In many cases, instead of (B) one requires a stronger condition:

(C) Detailed balance: For every pair $i, j \in S$

$$\pi(j) p_{ji} \;=\; \pi(i) p_{ij} \tag{4.62}$$

It is easy to check that (C) implies (B). A Markov chain that satisfies (C) is called reversible as the probabilities Prob$(X_t = i, X_{t+1} = j)$ and the reversed one Prob$(X_t = j, X_{t+1} = i)$ are equal in equilibrium.

Conditions (A) and (B) are necessary for a correct Monte Carlo. However there are many choices of P which satisfy both (A) and (B) and their efficiency will be in general very different. The key difficulty in dynamic Monte Carlo is that successive states $X_0, X_1, \dots$ of the Markov chain are correlated so that the variance of the estimates produced from the simulation may be much higher than in static Monte Carlo. To make this precise, let $f(x)$ be a real valued function defined on the state space S. We have already seen that time averages of $f(X_t)$ converge to $\langle f \rangle$. We want now to study the variance of this estimate , i.e.

$$\lim_{n\to\infty} n \; \mathrm{Var}_\alpha \left(\frac{1}{n} \sum_{k=1}^{n} f(X_t) \right) \tag{4.63}$$

where α is the starting distribution of the Markov chain. It can be shown that such a limit does not depend on the initial distribution and thus one can compute it in equilibrium. Let us notice that when $\alpha(j) = \pi(j)$ all the random variables $\{X_t\}$ have distribution π because of property (B). This implies that all mean values are time translation invariant, i.e.

$$\langle f_1(X_{t_1}) \dots f_n(X_{t_n}) \rangle \;=\; \langle f_1(X_{t_1+t}) \dots f_n(X_{t_n+t}) \rangle \tag{4.64}$$

for all t. Then

$$\begin{aligned} \mathrm{Var}_\pi \left(\frac{1}{n} \sum_{k=1}^{n} f(X_t) \right) &= \frac{1}{n^2} \sum_{ts=1}^{n} \langle f(X_t) f(X_s) \rangle - \langle f \rangle^2 && (4.65) \\ &= \frac{1}{n^2} \sum_{ts=1}^{n} C_{ff}(t-s) && (4.66) \\ &= \frac{1}{n} \sum_{t=-(n-1)}^{n-1} C_{ff}(t) \left(1 - \frac{|t|}{n} \right) && (4.67) \end{aligned}$$

Here $C_{ff}(t)$ is the autocorrelation function defined by

$$\begin{aligned} C_{ff}(t) &= \langle f(X_s) f(X_{s+t}) \rangle - \langle f \rangle^2 && (4.68) \\ &= \sum_{ij} f(i) [\pi(i) (P^t)_{ij} - \pi(i)\pi(j)] \; f(j) && (4.69) \end{aligned}$$

$C_{ff}(t)$ is a (usually exponentially) decreasing function and thus for large n we can approximate the variance by

$$\frac{1}{n} \sum_{t=-(n-1)}^{n-1} C_{ff}(t) \tag{4.70}$$

Let us then define the integrated autocorrelation time

$$\tau_{int,f} \;=\; \frac{1}{2} \sum_{t=-\infty}^{\infty} \frac{C_{ff}(t)}{C_{ff}(0)} \;=\; \frac{1}{2} \;+\; \sum_{t=1}^{\infty} \frac{C_{ff}(t)}{C_{ff}(0)} \tag{4.71}$$

Then we obtain for the variance

$$\frac{1}{n} \; (2\tau_{int,f}) C_{ff}(0) \tag{4.72}$$

Thus the variance is a factor $2\tau_{int,f}$ larger than it would be if the $\{f(X_t)\}$ were statistically independent. Stated differently, the number of effectively independent samples in a run of length n is roughly $n/2\tau_{int,f}$

Exercise 5: Let S be finite, P an aperiodic chain and f, g functions defined on S. Consider

$$\mathrm{Cov}^{(n)}(f;g) \;= \mathrm{Cov}_\pi\left(\frac{1}{n}\sum_{k=1}^{n} f(X_k)\;;\frac{1}{n}\sum_{k=1}^{n} g(X_k)\right) \tag{4.73}$$

Prove that

$$\lim_{n\to\infty} n\;\mathrm{Cov}^{(n)}(f;g) = \sum_{ij} f(i)C_{ij}g(j) \tag{4.74}$$

where the limiting covariance operator C is given by

$$C_{ij} \;=\; Z_{ij}\pi(i) + \; Z_{ji}\pi(j) - \pi(i)\delta_{ij} - \pi(i)\pi(j) \tag{4.75}$$

and Z is defined in Exercise 3 (assume that Z exists; this is always true if S is finite). Hint: prove firstly that

$$C_{ij} = \lim_{n\to\infty}\left[\sum_{k=1}^{n-1}\pi(i)(P^k)_{ij}\left(1-\frac{k}{n}\right) + (i\leftrightarrow j) + \pi(i)\delta_{ij} - n\pi(i)\pi(j)\right] \tag{4.76}$$

and then use the Cesaro theorem to prove

$$\sum_{k=1}^{n-1}(P^k)_{ij}\left(1-\frac{k}{n}\right) \;=\; Z_{ij} - \delta_{ij} + \frac{1}{2}(n-1)\pi(j) + O(1/n) \tag{4.77}$$

A second important autocorrelation time is the exponential autocorrelation time. It is defined by the decay for large times of $C_{ff}(t)$. The proper definition is

$$\tau_{exp,f} \;=\; \lim_{t\to\infty}\sup\frac{t}{-\log|\rho_{ff}(t)|} \tag{4.78}$$

where $\rho_{ff}(t) \;=\; C_{ff}(t)/C_{ff}(0)$ and

$$\tau_{exp} \;=\; \sup_{f\in l^2(\pi)} \tau_{exp,f} \tag{4.79}$$

Thus τ_{exp} is the relaxation time of the slowest mode in the system.

An equivalent definition involves the spectrum of the transition probability matrix P considered as an operator in $l^2(\pi)$. One can prove the following facts:

(a) the spectrum of P lies in the closed unit disk (this follows immediately from the fact that $\sum_j p_{ij} = 1$);

(b) 1 is a simple eigenvalue of P and $(1,\ldots,1)$ is the corresponding eigenvalue;

(c) if the chain is aperiodic, then 1 is the only eigenvalue of P on the unit disk.

Within this formalism τ_{exp} is defined through the spectral radius R of P acting on the orthogonal complement of the constant functions in $l^2(\pi)$. Then

$$R = \;\exp(-1/\tau_{exp}) \tag{4.80}$$

If S is finite, R is simply the second largest (in absolute value) eigenvalue of P.

The interest in τ_{exp} lies in the fact that it governs the convergence to equilibrium of the Markov chain. Let α be an arbitrary probability distribution defined on S and let us consider the Markov chain with initial probability α. Then at time t the probability distribution of $\{X_t\}$ is

$$(\alpha P^t)(j) = \sum_{i\in S} \alpha(i)(P^t)_{ij} \tag{4.81}$$

Now, if $f \in l^2(\pi)$, let us consider

$$\begin{aligned} |\langle f(X_t)\rangle_\alpha - \langle f\rangle| &= \left|\sum_{ij} \alpha(i) P^t_{ij} f(j) - \sum_i \pi(i) f(i)\right| && (4.82)\\ &= \left|\sum_{ij} (\alpha(i) - \pi(i))(P^t_{ij} - \pi(j)) f(j)\right| && (4.83)\\ &= \left|\sum_{ij} (\alpha(i) - \pi(i))(P - \Pi)^t_{ij} f(j)\right| && (4.84)\end{aligned}$$

where $\Pi_{ij} = \pi(j)$ and we have repeatedly used the stationarity property of P. Now the spectral radius of $P - \Pi$ is R by definition as Π is the projector over the constant functions and thus the spectral radius of $(P - \Pi)^t$ is of the order of R^t with exact equality if P is self-adjoint. Then

$$|\langle f(X_t)\rangle_\alpha - \langle f\rangle| \le \exp(-t/\tau_{exp}) \ |\langle f(X_0)\rangle_\alpha - \langle f\rangle| \tag{4.85}$$

To conclude this Section we want to prove an important relation satisfied by the integrated autocorrelation time when the chain is reversible. In this case, using (C), we immediately see that P is self-adjoint in $l^2(\pi)$ and thus its spectrum is real. Then, by the spectral theorem, we can write

$$C_{ff}(t) = \int_{-1}^{1} \lambda^{|t|} \, \mathrm{d}\mu_{ff}(\lambda) \tag{4.86}$$

where $\mathrm{d}\mu_{ff}(\lambda)$ is a nonnegative measure with support in $[-\exp(-1/\tau_{\mathrm{exp},f}), \exp(-1/\tau_{\mathrm{exp},f})]$. Then

$$\tau_{int,f} = \frac{1}{2}\frac{\int_{-1}^{1} \frac{1+\lambda}{1-\lambda} \mathrm{d}\mu_{ff}(\lambda)}{\int_{-1}^{1} \mathrm{d}\mu_{ff}(\lambda)} \ge \frac{1}{2}\frac{1+\rho_{ff}(1)}{1-\rho_{ff}(1)} \tag{4.87}$$

by Jensen's inequality. This is a very useful relation in proving lower bounds on the autocorrelation times.

In summary, the autocorrelation times τ_{exp} and $\tau_{int,f}$ play different roles in Monte Carlo simulations. The first one places an upper bound on the number of iterations n_{disc} which should be discarded at the beginning of the run, before the system has attained equilibrium; for example, $n_{disc} \approx 20\tau_{exp}$ is usually more than adequate. On the other hand $\tau_{int,f}$ determines the statistical errors in the Monte Carlo measurement of $\langle f\rangle$ once equilibrium has been attained.

Most commonly it is assumed that τ_{exp} and $\tau_{int,f}$ are of the same order of magnitude, at least for reasonable observables f. But this is not true in general. In fact, in statistical mechanical problems near a critical point, one usually expects the autocorrelation function $\rho_{ff}(t)$ to obey a dynamic scaling law of the form

$$\rho_{ff}(t;\beta) \sim |t|^{-a} \, F\left((\beta - \beta_c)|t|^b\right) \tag{4.88}$$

valid in the region

$$|t| >> 1, \; |\beta - \beta_c| << 1, \; |\beta - \beta_c||t|^b \text{ bounded} \tag{4.89}$$

Here a, b are dynamic critical exponents and F is suitable scaling function; β is some "temperature-like" parameter and β_c is the critical point. Now suppose that F is continuous and strictly positive, with $F(x)$ decaying exponentially for large x. Then it is not hard to see that

$$\tau_{exp,f} \sim |\beta - \beta_c|^{-1/b} \tag{4.90}$$

$$\tau_{int,f} \sim |\beta - \beta_c|^{-(1-a)/b} \tag{4.91}$$

$$\rho_{ff}(t; \beta = \beta_c) \sim |t|^{-a} \tag{4.92}$$

so that $\tau_{exp,f}$ and $\tau_{int,f}$ have different critical exponents unless $a = 0$. Actually this should not be surprising: replacing "time" by "space" we see that $\tau_{exp,f}$ is the analogue of a correlation length while $\tau_{int,f}$ is the analogue of a susceptibility; and in general this two quantities have different critical exponents. Thus it is crucial to distinguish between these two types of autocorrelation time.

5 Statistical Analysis of Dynamic Monte Carlo Data.

In the previous Section we have discussed the theory behind Monte Carlo simulations and in particular we have defined the autocorrelation times associated with a Markov chain. In this Section we want to discuss how to use them in the analysis of the data coming from a simulation.

There are two fundamental and quite distinct issues in dynamic Monte Carlo simulations:

- Initialization bias. If the Markov chain is started with distribution $\alpha \neq \pi$ then there is an "initial transient" in which the data do not reflect the desired equilibrium distribution π. This results in a systematic error which however goes to zero as the sample size goes to infinity.
- Autocorrelation in equilibrium. The Markov chain, once it reaches equilibrium, provides correlated samples from π. This correlation causes the statistical error (variance) to be a factor $2\tau_{int,f}$ larger than in independent sampling.

Let us firstly discuss the problem of initialization. At the beginning of the run we usually choose an easy-to-prepare configuration. There are various possibilities. For instance in the Ising model we can either start from a cold (or ordered) configuration or from a hot (or random) configuration. In the first case all the spins are aligned, while in the second case the spins are initialized randomly and independently, with equal probability of up and down. Another possibility, which reduces the thermalization time, consists in using a configuration thermalized at a nearby value of β (of course if we have it). The main feature of all these methods is that the initial configuration is out of equilibrium. We know from the results of the previous Section that the system approaches equilibrium as $t \to \infty$ as $(P)^n_{ij} \to \pi(j)$ but what we are really interested in is the rate of convergence, that is the number of iteration necessary for the system to thermalize.

Using the exponential autocorrelation time τ_{exp} we can set an upper bound on this amount of time. For example $20\tau_{exp}$ iterations will be enough for all practical purposes. There are however two difficulties in applying this method: first of all we do not know how to evaluate τ_{exp} and in very few (non trivial) cases we have a theoretical knowledge of it. Secondly this method may be overly conservative. For instance there are perfectly good algorithms for which $\tau_{exp} = \infty$ (the correlation function decays here as a power law). Does this mean that equilibrium can never be reached? Of course not: $\tau_{exp} = \infty$ means that there many "bad" starting configurations which require an enormous time to equilibrate, but this does not exclude that we can find "good" configurations which equilibrate quite fast.

In practice to determine empirically when equilibrium has been achieved one plots selected observables as a function of time and notes when the initial transient appears to end. The danger in all these methods is the possibility of metastability. That is, it could appear that equilibrium has been achieved, when in fact the system has only settled down to a long-lived metastable region of configuration space that may be far from equilibrium. A good method to check if metastability is present is to try different initial configurations and see if the results of the different runs are consistent.

Once equilibrium has been attained one can try to make a rough empirical estimate of τ_{exp} by measuring the autocorrelation function $C_{ff}(t)$ for a suitably large set of observables f; but there is always the danger that our chosen set of observables has failed to include one that has strong enough overlap with the slowest mode, again leading to a gross underestimate of τ_{exp}.

As a final comment, let us notice that discarding the initial transient is asymptotically not necessary. As we have seen in the previous Section the sample mean and its variance converge, when the sample size goes to infinity, to a limit independent from the initial distribution. Indeed the bias due to the initial transient (see e.g. exercise 3 in the previous Section) scales as $1/n$ while the statistical fluctuations are of order $1/\sqrt{n}$. In practice, however, the coefficient of $1/n$ may be fairly large if the starting configuration is very far from equilibrium so that throwing away tha data from the initial transient remains necessary. Remark that the shorter the run the more careful one has to be about equilibration.

Let us now discuss the second issue. As we have seen in the previous Section the sample mean of f, i.e.

$$\bar{f} = \frac{1}{n}\sum_{i=1}^{n} f_i \tag{5.93}$$

gives an estimate of $\langle f \rangle$ which is unbiased if the chain is in equilibrium (that is $\langle \bar{f} \rangle = \langle f \rangle$). The next problem is to set an error on this estimate. One of the simplest procedures is the method of the "batched means". If the run consists of n measurements, divide it into some relatively small number T of equal length subsequences, or "batches". Let $b = n/T$ be the number of measurements in each batch and let Y_i be the average of the i-th batch

$$Y_i = \frac{1}{b}\sum_{j=(i-1)b+1}^{ib} f_j \tag{5.94}$$

If b is much larger than τ_{exp} then the Y_i's are approximately independent and Gaussian with mean $\langle f\rangle$ and variance $C_{ff}(0)(2\tau_{int,f})/b$. Then, for the overall average is the average of the Y_i's we can estimate its variance using the sample variance of the Y_i's. This is a very quick method. However it has a serious drawback: the assumption $b >> \tau_{exp}$. In particular it cannot be applied to those algorithms where $\tau_{exp} = \infty$ or where $\tau_{exp} >> \tau_{int,f}$ since in this case the total number of iterations can be much less than τ_{exp}. Moreover the results of the procedure cannot be used as a check on the assumption.

A much better method uses the results of the previous Section. Indeed we know that the variance of the sample mean $\bar{f}$ is $C_{ff}(0)(2\tau_{int,f})$. $C_{ff}(0)$ is simply the static variance and is estimated by

$$C_{ff}(0) = \frac{1}{n}\sum_{i=1}^{n}\left(f_i - \bar{f}\right)^2 \tag{5.95}$$

Notice that this quantity depends uniquely on the model and not on the algorithm used to simulate it. What instead depends on the algorithm is $\tau_{int,f}$. To evaluate it we must first estimate $C_{ff}(t)$. The natural estimator is

$$\hat{C}_{ff}(t) \;=\; \frac{1}{n-|t|}\sum_{i=1}^{n-|t|}(f_i-\bar{f})(f_{i+|t|}-\bar{f}) \tag{5.96}$$

This is a biased estimator of $C(t)$ the bias being of order $1/n$. The natural estimator of $\rho_{ff}(t) = C_{ff}(t)/C_{ff}(0)$ is thus

$$\hat{\rho}_{ff}(t) \;=\; \frac{\hat{C}_{ff}(t)}{\hat{C}_{ff}(0)} \tag{5.97}$$

Then the natural estimator of τ_{int} would seem to be

$$\hat{\tau}_{int,f} = \frac{1}{2}\sum_{t=(1-n)}^{n-1}\hat{\rho}_{ff}(t) \tag{5.98}$$

but this is wrong. Indeed this estimator has a variance that does not go to zero as $n \longrightarrow \infty$. Roughly speaking this is because the sample autocorrelation $\hat{\rho}_{ff}(t)$ for $|t| >> \tau_{int,f}$ contains much noise but little signal (see the following exercise).

Exercise 1: Let f_i be independent Gaussian variables of mean μ/n and variance σ^2/n. Compute the mean and the variance of $\sum_{i=1}^{n} f_i$.

The solution is to cut off the sum defining

$$\hat{\tau}_{int,f} \;=\; \frac{1}{2}\sum_{t=-M}^{M}\hat{\rho}_{ff}(t) \tag{5.99}$$

Of course this cut off introduces a bias and in general this formula will underestimate the correct value. However we can choose M in such a way that the bias is small. This can be achieved using the authomatic windowing procedure by Madras and Sokal: choose M self-consistently as the smallest integer such that $M \geq c\hat{\tau}_{int,f}(M)$. If $\rho_{ff}(t)$ were a pure exponential one could take for instance $c = 4$, in this case making an error of the $e^{-4} = \;2\%$. However in many cases $\rho_{ff}(t)$ has a slower preasymptotic decay and thus in these cases c

must be larger, for instance $c = 4 - 6$ in order to have an error of the same magnitude. In general the determination of c requires the study of the behaviour of the autocorrelation function $C_{ff}(t)$. Indeed c must not be too small, otherwise the bias would be too large, but neither too large otherwise we would include some noise. What small and large mean, depends on the number of iterations and in general, increasing the number of iterations one must at the same time increase the value of c in order to keep the systematic error smaller than the statistical one. Let us finally quote the variance of $\hat{\tau}_{int,f}$:

$$\mathrm{Var}(\hat{\tau}_{int,f}) \;=\; \frac{2(2M+1)}{n}\tau^2_{int,f} \tag{5.100}$$

valid in the regime $\tau_{int,f} << n$.

As a final remark, notice that this procedure works well only if $n >> \tau_{int,f}$. Empirically one usually finds that the results are reliable if $n \gtrsim 1000\tau_{int,f}$.

From what we have said it is clear that the performance of the algorithm is completely characterized by $\tau_{int,f}$ measured in CPU-time units. The higher $\tau_{int,f}$ the less efficient is the algorithm. Now what is the behaviour of $\tau_{int,f}$? Generically, away from phase transitions (i.e. in the regions where the correlation length is small) $\tau_{int,f}$ remains reasonably small. However near a critical point the autocorrelation time diverges as

$$\tau \;\sim\; \xi^z(\beta) \tag{5.101}$$

where $\xi(\beta)$ is the correlation length of the infinite volume system at temperature $1/\beta$ and z is a dynamical critical exponent. This phenomenon is called dynamic critical slowing down and it is the most severe limitation to Monte Carlo studies. We will see in the next Section that for local Monte Carlo one usually have $z \gtrsim 2$ (overrelaxation is an exception with $z \gtrsim 1$) while using non-local methods (cluster algorithms, Fourier acceleration, multigrid ...) one can achieve sensible improvements (in many cases $z \approx 0$).

Now, how can we determine z? First of all, as we have already remarked, z depends on the type of autocorrelation time we are considering and thus we must distinguish between z_{exp} and $z_{int,f}$ and in many algorithms different quantities f have different exponents (for instance in Wolff's algorithms for RP^n σ-models the exponent of the susceptibility is ~ 1 while the exponent of the energy is ~ 0). Secondly, notice that (5.101) is true only when $L \to \infty$. In practice experience with two-dimensional models shows that corrections to scaling are usually very strong. For instance, using Wolff's algorithm for the $O(4)$ σ-model one finds for $L = 32$, $\beta = 1.70$ $\xi = 3.54$ and $\tau_{int,\chi} = 4.53(16)$ while for $L = 32$, $\beta = 2.20$ $\xi = 11.59$ and $\tau_{int,\chi} = 10.61(70)$. Using (5.101) and these two values we would get $z_{int,\chi} \sim 0.8$ but this is wrong. Indeed for this model one has $z_{int,\chi} \lesssim 0.1$. The wrong result is due to the fact that we have neglected the finite-size corrections. The correct way of performing the analysis uses a finite-size scaling Ansatz of the form

$$\tau_{int,f}(\beta, L) \;\sim\; \xi(\beta, L)^{z_{int,f}}\; g_f(\xi(\beta, L)/L) \tag{5.102}$$

where g_f is an unknown scaling function with $g_f(0)$ supposed to be finite and non-zero. To determine $z_{int,f}$ one plots $\tau_{int,f}(\beta, L)/\xi(\beta, L)^{z_{int,f}}$ as a function of $\xi(\beta, L)/L$ fixing $z_{int,f}$ so that all the points lie on a unique curve within error bars. In order to have a reliable estimate many different values of L must be used and sufficiently accurate estimates of $\tau_{int,f}$ are required.

Let us now pass to a second subject: so far we have discussed how to compute from the Monte Carlo data the mean values of the observables we are interested in and their error bars. Now let us discuss how to use them in some practical problem.

In many cases we want to compute invariant ratios, i.e. quantities of the form

$$R = \frac{\langle A\rangle^p}{\langle B\rangle^q} \tag{5.103}$$

where A and B are two different observables and p and q some exponents. Of course we estimate this quantity from the sample means $\bar{A}$ and $\bar{B}$:

$$R_{est} = \frac{\bar{A}^p}{\bar{B}^q} \tag{5.104}$$

But then, what is the error bar on R_{est}? The main problem in its estimation is that $\bar{A}$ and $\bar{B}$ come from the same run; consequently they are correlated and this must be kept into account in setting the error bars. Let us compute the variance of R_{est}. By definition

$$\text{Var}\,(R_{est}) = \langle \frac{\bar{A}^{2p}}{\bar{B}^{2q}}\rangle - \langle R_{est}\rangle^2 \tag{5.105}$$

Then introduce

$$\Delta A = \frac{\bar{A} - \langle A\rangle}{\langle A\rangle} \tag{5.106}$$

$$\Delta B = \frac{\bar{B} - \langle B\rangle}{\langle B\rangle} \tag{5.107}$$

Expanding in ΔA and ΔB (valid in the large sample limit) we get

$$\text{Var}\,(R_{est}) = \frac{\langle A\rangle^{2p}}{\langle B\rangle^{2q}}\left(p^2\langle\Delta A^2\rangle + q^2\langle\Delta B^2\rangle - 2pq\langle\Delta A\Delta B\rangle\right) + o(1/n) \tag{5.108}$$

Notice that $\langle\Delta A^2\rangle = \sigma_A^2/\langle A\rangle^2$ and $\langle\Delta B^2\rangle = \sigma_B^2/\langle B\rangle^2$ while the last term keeps into account the correlation between the estimates of A and B. Using the fact that (Schwartz inequality)

$$|\langle\Delta A\Delta B\rangle| \le \langle\Delta A^2\rangle^{1/2}\langle\Delta B^2\rangle^{1/2} \tag{5.109}$$

we obtain an upper bound in terms of the variances of A and B alone

$$\text{Var}\,(R_{est}) \le \frac{\langle A\rangle^{2p}}{\langle B\rangle^{2q}}\left(p\frac{\sigma_A}{\langle A\rangle} + q\frac{\sigma_B}{\langle B\rangle}\right)^2 \tag{5.110}$$

This upper bound is usually very far from being sharp since A and B are strongly correlated (in some cases (5.110) is ten times larger than the correct formula (5.108)). For this reason it is always convenient to use (5.108). However in this case we must compute the covariance $\langle\Delta A\Delta B\rangle$. This can be avoided using a little trick, i.e. rewriting (5.108) as

$$\text{Var}\,(R_{est}) = \frac{\langle A\rangle^{2p}}{\langle B\rangle^{2q}}\,\text{Var}\,(p\Delta A - q\Delta B) \tag{5.111}$$

and this last variance can be computed applying the time-series methods we have presented before to the time series

$$p\,\frac{A_i - \bar{A}}{\bar{A}} \;-\; q\,\frac{B_i - \bar{B}}{\bar{B}} \tag{5.112}$$

In this way one usually gets smaller (but correct!) error bars. We want to emphasize that these results are not connected with dynamic Monte Carlo, in the sense that the important thing here is the static correlation of A and B (usually $p\Delta A - q\Delta B$ has an autocorrelation time which is of the order of the largest of $\tau_{int,A}$ and $\tau_{int,B}$) and thus it is important to keep it into account also in the case of independent sampling.

These formulas can easily be extended to generic functions of $\langle A\rangle$ and $\langle B\rangle$ and to functions of more than two observables.

Exercise 2: In lattice gauge theories one usually defines the Creutz ratio

$$R(I,J) \;=\; \frac{W(I,J)W(I-1,J-1)}{W(I,J-1)W(I-1,J)} \tag{5.113}$$

where $W(I,J)$ denotes the expectation of a rectangular Wilson loop of lattice dimensions I and J. Compute the variance of $R(I,J)$ in the large sample limit expressing the result in terms of variances of suitably defined observables for which the usual time-series analysis can be applied.

Let us now present two applications of the previous results. The first one is concerned with the so-called hystogram method introduced by Falcioni et al. and Ferrenberg and Swendsen. Suppose you have made a simulation of the system at a given value of β. Can this run be used to get some information on the mean values of the various observables we have measured at a different temperature β_0? The answer is yes, and it is based on the following observation: if $\mathcal{O}$ is any observable we have

$$\langle\mathcal{O}\rangle_{\beta_0} \;=\; \frac{1}{Z(\beta_0)}\sum_{\{\sigma\}}\mathcal{O}(\sigma)\exp(-\beta_0 H(\sigma)) \tag{5.114}$$

$$= \frac{\sum_{\{\sigma\}}\mathcal{O}(\sigma)\exp((\beta-\beta_0)H(\sigma))\exp(-\beta H(\sigma))}{\sum_{\{\sigma\}}\exp((\beta-\beta_0)H(\sigma))\exp(-\beta H(\sigma))} \tag{5.115}$$

$$= \frac{\langle\exp((\beta-\beta_0)H)\mathcal{O}\rangle_\beta}{\langle\exp((\beta-\beta_0)H)\rangle_\beta} \tag{5.116}$$

where $\langle\ldots\rangle_\beta$ is the estimate at inverse temperature β. Thus an estimate of $\langle\mathcal{O}\rangle_{\beta_0}$ is obtained as a ratio of the sample means of $\exp((\beta-\beta_0)H)\mathcal{O}$ and $\exp((\beta-\beta_0)H)$ and the error bar is computed using (5.108).

One can also compute $\langle\mathcal{O}\rangle_{\beta_0}$ using runs at different, but nearby, values β_i. In this case the estimates are independent and thus it is very simple to write down the final estimate and its error bar. If $\mathcal{O}_i$ is the estimate of $\langle\mathcal{O}\rangle$ obtained using the runs at temperature $1/\beta_i$ and σ_i^2 its variance we get the final estimate

$$\mathcal{O}_{est} \;=\; \sum_i \frac{\mathcal{O}_i}{\sigma_i^2}\left(\sum_i\frac{1}{\sigma_i^2}\right)^{-1} \tag{5.117}$$

with variance

$$\sigma^2 = \left(\sum_i \frac{1}{\sigma_i^2} \right)^{-1} \tag{5.118}$$

The hystogram method is very useful in many cases. However we must stress two important points. First of all $\langle O \rangle_{\beta_0}$ can be computed from a run at inverse temperature β only if β_0 lies in a small interval around β and the width of this interval is of the order of the fluctuations of the energy and as such decreases as $1/\sqrt{V}$ (except at phase transition points where fluctuations go as $\sqrt{C_E/V}$ where $C_E \sim V^{\alpha/d\nu}$ is the specific heat). Outside this interval the estimates become completely unreliable and formula (5.108) cannot be applied since higher order corrections are still important. Secondly the points obtained in this way are correlated and this must be kept in mind in any subsequent analysis (see exercise 4).

Exercise 3: Let E_m be the mean value of the energy at inverse temperature β of an Ising model and $n(E)$ the number of spin configurations with energy E. Show how one can compute the ratio $n(E)/n(E_m)$ from a run at inverse temperature β. (The results will be reliable only for E not very different from E_m, explain why!).

A second important case concerns the evaluation of conditional expectations. Let $\mathcal{O}^1$ and $\mathcal{O}^2$ be two observables and suppose you want to compute the mean value of $\mathcal{O}^1$ subject to the condition that $\mathcal{O}^2$ assumes a certain value (or satisfies a certain relation). For instance this what you need when preparing a hystogram, a plot of $\mathcal{O}^1$ as a function of $\mathcal{O}^2$.

Let us introduce the function χ defined on the state space S which assumes the value 1 if the condition on $\mathcal{O}^2$ is satisfied and 0 otherwise. Then the conditional expectation can be computed using the formula

$$\langle \mathcal{O}^1 \rangle_{\text{Conditional on } \mathcal{O}^2} = \frac{\frac{1}{n}\sum_{i=1}^{n} \mathcal{O}_i^1 \chi_i}{\frac{1}{n}\sum_{i=1}^{n} \chi_i} \tag{5.119}$$

and its variance from (5.108).

It is easy to see that this formula corresponds to the very simple idea of estimating $\langle \mathcal{O}^1 \rangle_{\text{Conditional on } \mathcal{O}^2}$ using the censored time-series $\tilde{X}_1, \ldots \tilde{X}_{n'}$ defined by

$$\tilde{X}_j = X_{T_j} \tag{5.120}$$

where $T_1 < T_2 < \ldots < T_{n'}$ are the times for which the condition on $\mathcal{O}^2$ is satisfied. Indeed it is easy to see that the previous formula corresponds to

$$\langle \mathcal{O}^1 \rangle_{\text{Conditional on } \mathcal{O}^2} = \frac{1}{n'} \sum_{i=1}^{n'} \mathcal{O}^1_{T_i} \tag{5.121}$$

Notice that in this formula both the T_j and the censored sample size n' are random variables and thus we cannot apply the usual time-series analysis to this sequence.

Let us conclude this Section by discussing another problem of practical importance, the determination of the exponential correlation length. Let us firstly review its definition. Given a model, let $\mathcal{O}$ be a generic observable. Then compute the two-point function

$$\langle \mathcal{O}(0)\ \mathcal{O}(x) \rangle - \langle \mathcal{O} \rangle^2 \tag{5.122}$$

Near a critical point, for large $|x|$, this quantity has a behaviour of the form

$$\langle \mathcal{O}(0)\ \mathcal{O}(x)\rangle - \langle \mathcal{O}\rangle^2 \ \sim\ \exp(-|x|/\xi_{\mathcal{O}}) \tag{5.123}$$

where $\xi_{\mathcal{O}}$ is the exponential correlation length associated to $\mathcal{O}$. In general different operators have different correlation lengths but at the critical point all these quantities scale with the same exponent usually called ν. In Monte Carlo computations, in order to have a better signal and thus a better estimate of $\xi_{\mathcal{O}}$ one usually uses space-averages of $\mathcal{O}$, i.e. computes

$$G_{\mathcal{O}}(t) = \sum_{\vec{x},\vec{y}} \Big(\langle \mathcal{O}(\vec{x},0)\ \mathcal{O}(\vec{y},t)\rangle\ - \langle \mathcal{O}\rangle^2 \Big) \tag{5.124}$$

which is expected to behave for large t as

$$G_{\mathcal{O}}(t)\ \sim\ \exp(-|t|/\xi_{\mathcal{O}}) \tag{5.125}$$

On a finite lattice with extension L in the time direction and periodic boundary conditions one must keep into account the periodicity and the correct formula is $(0 \le t \le L)$:

$$\begin{aligned} G_{\mathcal{O}}(t) &\sim\ (\exp(-t/\xi_{\mathcal{O}})\ +\ \exp((L-t)/\xi_{\mathcal{O}})) && (5.126)\\ &\sim\ \mathrm{ch}\,((L/2-t)/\xi_{\mathcal{O}}) && (5.127) \end{aligned}$$

Let us now pass to describe how to compute $\xi_{\mathcal{O}}$ from Monte Carlo data. For simplicity we will compute $\xi_{\mathcal{O}}$ from the Ansatz

$$G_{\mathcal{O}}(t)\ =\ \alpha \exp(-t/\xi_{\mathcal{O}}) \tag{5.128}$$

which we assume valid for $0 << t_{min} < t < t_{max} << L/2$ but the whole discussion can be easily extended to the general formula (5.127). Taking logarithms we get

$$\log G_{\mathcal{O}}(t)\ =\ \log\alpha - t/\xi_{\mathcal{O}}\ =\ A - t/\xi_{\mathcal{O}} \tag{5.129}$$

Now, how do we evaluate A and $\xi_{\mathcal{O}}$? The first idea is to use the standard χ^2 method. Let $\hat{G}(t)$ be the sample two-point function, i.e. [1]

$$\hat{G}(t)\ =\ \frac{1}{n}\sum_{i=1}^{n} G_{\mathcal{O}}^{(i)}(t) \tag{5.130}$$

and $V(t)$ be its variance computed for each t by analyzing the time series of $G_{\mathcal{O}}^{(i)}(t)$; then define

$$\chi^2\ =\ \sum_{t=t_{min}}^{t_{max}} \left(\log\hat{G}(t) - A - \frac{t}{\xi_{\mathcal{O}}}\right)^2 \left(\frac{\hat{G}(t)^2}{V(t)}\right) \tag{5.131}$$

A and $\xi_{\mathcal{O}}$ are computed as the values which minimize χ^2 and the errors on A and $\xi_{\mathcal{O}}$ are obtained from the standards results on the linear regressions. But, is this method correct? Unfortunately it is not, since we have overlooked a very important fact: the values of $\hat{G}(t)$ for different t are (strongly) correlated due to the fact that we have made the measurements

[1] Notice that in this discussion we may indicate with time two different quantities: there is the simulation time which we indicate with i and goes from 1 to n where n is the total number of iterations and the lattice time t which goes from 1 to L.

on the same configurations. This means that the error bars that we have computed using the standard formulas on linear regressions grossly underestimate the true error bars.

The correct procedure consists in computing firstly the covariance matrix

$$V(t_1, t_2) \;=\; \mathrm{Cov}\left(\frac{1}{n}\sum_{i=1}^{n} G_{\mathcal{O}}^{(i)}(t_1);\; \frac{1}{n}\sum_{i=1}^{n} G_{\mathcal{O}}^{(i)}(t_2)\right) \tag{5.132}$$

This computation can be done in many different ways. The simplest one consists in noting that (A and B generic quantities)

$$\begin{aligned}\mathrm{Cov}(A;B) &= \langle AB\rangle - \langle A\rangle\langle B\rangle & (5.133)\\ &= \frac{1}{2}\left(\langle A^2\rangle + \langle B^2\rangle - \langle (A-B)^2\rangle - 2\langle A\rangle\langle B\rangle\right) & (5.134)\\ &= \frac{1}{2}\left(\mathrm{Var}(A) + \mathrm{Var}(B) - \mathrm{Var}(A-B)\right) & (5.135)\end{aligned}$$

From this formula one sees that the problem of computing $\mathrm{Cov}(A;B)$ has been reduced to the (by now very well known) problem of computing three variances, i.e. to the computation of $C_{A-B,A-B}(0)$, $C_{AA}(0)$, $C_{BB}(0)$, $\tau_{int,A-B}$, $\tau_{int,A}$ and $\tau_{int,B}$ as we have discussed at length. Then, once you have V, compute its inverse, i.e. the matrix $V^{-1}(t_1,t_2)$ such that

$$\sum_{t=t_{min}}^{t_{max}} V^{-1}(t_1,t)V(t,t_2) \;=\; \delta_{t_1,t_2} \tag{5.136}$$

and define

$$\Sigma(t_1,t_2) \;=\; V^{-1}(t_1,t_2)\hat{G}_{\mathcal{O}}(t_1)\hat{G}_{\mathcal{O}}(t_2) \tag{5.137}$$

Then the χ^2 is defined by

$$\chi^2 = \sum_{t_1,t_2=t_{min}}^{t_{max}} \left(\log\hat{G}(t_1) - A - \frac{t_1}{\xi_{\mathcal{O}}}\right)\left(\log\hat{G}(t_2) - A - \frac{t_2}{\xi_{\mathcal{O}}}\right)\Sigma(t_1,t_2) \tag{5.138}$$

and the mean values A and $\xi_{\mathcal{O}}$ and their error bars are computed using the standards methods for linear regressions.

This discussion applies to many other similar cases which can be dealt with along the same lines. Another example is presented in the following exercise.

Exercise 4: Consider the Ising model on a two-dimensional square lattice. Suppose you have three runs at $\beta = 0.39,\ 0.41,\ 0.43$ and that you have measured the susceptibility χ and the energy E. Then imagine that, using the hystogram method, you have computed the susceptibility also for $\beta = 0.38,\ 0.40,\ 0.42$. How do you perform a fit of the form $\chi \sim (\beta_c - \beta)^\gamma$ in order to estimate γ and β_c?

6 Metropolis and Heat-Bath Algorithms

In Section 4 we discussed the requirements which must be satisfied by a Markov chain in order to produce a correct Monte Carlo. Let us now start discussing some explicit ways of building transition matrices satisfying the conditions (A) and (B) (formulas (4.53) and (4.54)). A very general method was introduced by Metropolis et al. and is now called Metropolis algorithm. Let $P^{(0)}$ be an arbitrary irreducible transition matrix on S. We call $P^{(0)}$ the proposal matrix and we use it to generate proposed transitions $i \longrightarrow j$ which are then accepted with probabilities a_{ij} and rejected with probability $1 - a_{ij}$. If a proposed move is rejected, then we make a "null transition", $i \longrightarrow i$. Therefore the full transition matrix P is given by

$$p_{ij} = p_{ij}^{(0)} a_{ij} \text{ for } i \neq j \tag{6.139}$$

$$p_{ii} = p_{ii}^{(0)} + \sum_{j \neq i} p_{ij}^{(0)} (1 - a_{ij}) \tag{6.140}$$

where of course $a_{ij} \leq 1$ for all i, j. It is easy to see that P satisfies detailed balance for π if and only if

$$\frac{a_{ij}}{a_{ji}} = \frac{\pi_j p_{ji}^{(0)}}{\pi_i p_{ij}^{(0)}} \tag{6.141}$$

for all pairs $i \neq j$. But this is easily arranged: just set

$$a_{ij} = F\left(\frac{\pi_j p_{ji}^{(0)}}{\pi_i p_{ij}^{(0)}}\right), \tag{6.142}$$

where $F : [0, \infty] \longrightarrow [0, 1]$ is any function satisfying

$$\frac{F(z)}{F(1/z)} = z \tag{6.143}$$

for all z. The choice of Metropolis et al. is

$$F(z) = \min(z, 1) \tag{6.144}$$

which is the maximal function satisfying (6.143). Another choice is

$$F(z) = \frac{z}{1+z} \tag{6.145}$$

Of course it is still necessary to check that P is irreducible; but this is usually done on a case-by-case basis.

In the most part of the applications, the proposal matrix $P^{(0)}$ is symmetric, i.e. $p_{ij}^{(0)} = p_{ji}^{(0)}$ and in this case the transition probabilities have the simpler expression:

$$a_{ij} = F\left(\frac{\pi(j)}{\pi(i)}\right) \tag{6.146}$$

In statistical mechanics and lattice quantum theories, π is the Gibbs measure and thus the transition probabilities become

$$a_{ij} = F\left(e^{-\beta(E_j - E_i)}\right) \tag{6.147}$$

where E_i is the energy of the state i. Then, using the F given in (6.144), we obtain the following algorithm:

1. If $i \in S$ is the current configuration choose j with probability $p^{(0)}_{ij}$;
2. Compute E_i and E_j, the energies of the old and of the proposed configuration;
3. If $E_j - E_i \le 0$ accept the proposal, i.e. j is the new current configuration;
4. If $E_j - E_i > 0$ accept the proposal with probability $e^{-\beta(E_j - E_i)}$; that is, choose a random number U uniformly distributed between 0 and 1; then if $U \le e^{-\beta(E_j-E_i)}$ accept the proposal so that j is the new current configuration, otherwise make a null transition, that is keep i as current configuration.

Exercise 1 : Write the algorithm in the case $F(z) = z/(1+z)$.

Let us see some simple applications of this procedure. Let us firstly consider the Ising model: this is the simplest spin model, the fields σ_i assuming the values ± 1. The Hamiltonian is

$$H(\{\sigma\}) \;=\; -\sum_{\langle ij\rangle} \sigma_i\sigma_j \tag{6.148}$$

where the sum runs over all nearest-neighbour pairs. Let us now define the algorithm. Fix some site i. The proposal is to flip σ_i, hence

$$p^{(0)}_i(\{\sigma\} \to \{\sigma'\}) \;=\; 1 \text{ if } {\sigma'}_i = -\sigma_i \text{ and } {\sigma'}_j = \sigma_j \text{ for all } j \neq i \tag{6.149}$$

$$p^{(0)}_i(\{\sigma\} \to \{\sigma'\}) \;=\; 0 \text{ otherwise} \tag{6.150}$$

Here $P^{(0)}_i$ is symmetric so that we can immediately apply the algorithm with

$$E(\{\sigma'\}) - E(\{\sigma\}) \;=\; 2\sigma_i \sum_j \sigma_j \tag{6.151}$$

where the sum extends to all nearest neighbours j of i.

This defines a transition matrix P_i in which only the spin at site i is touched. The full algorithm involves sweeping through the entire lattice. There are various ways of doing this. One can for instance visit the lattice sites randomly, i.e. choose subsequent random points i. In this case the complete transition matrix is

$$P \;=\; \frac{1}{V}\sum_i P_i \tag{6.152}$$

Another possibility is sequential updating. In this case one orders the lattice sites and sequentially update the spin at each site. The full transition matrix is in this case

$$P \;=\; P_{i_1} P_{i_2} \ldots P_{i_V} \tag{6.153}$$

Notice that in this case P does not satisfy detailed balance for π but it satisfies stationarity for π which is what really matters.

An important type of sequential updating is the so-called checkerboard update used on vector and parallel machines. Suppose you are working on a d-dimensional hypercubic lattice and define the parity function

$$p(n_1, n_2, \ldots, n_d) \;=\; (-1)^{\sum_i n_i} \tag{6.154}$$

where $(n_1, n_2, \ldots, n_d)$ are the lattice coordinates. Then notice that, if i and j have the same parity, P_i and P_j commute so that the order in which we apply P_i and P_j is irrelevant. In particular, if our computer is a parallel machine we can perform the updates at points i and j simultaneously. In this way a single sweep over the whole lattice is performed in two steps (in the first step we update all points i with $p(i) = 1$, in the second one those with $p(i) = -1$).

Exercise 2: Consider an Ising model on a triangular lattice. Show that on a parallel machine a single sweep can be performed in three steps.

Exercise 3: On a square lattice consider an Ising model which has a nearest-neighbour and diagonal interactions. Show that a single sweep can be performed in at most 4 steps on a parallel machine.

Exercise 4: On a square lattice consider an Ising model which has a hamiltonian

$$\sum_i \sum_{\hat\mu} \sigma_i \left(J_1 \sigma_{i+\hat\mu} + J_2 \sigma_{i+2\hat\mu}\right) \tag{6.155}$$

Show that a single sweep can be performed in at most 3 steps on a parallel machine.

Let us now pass to discuss a second model: the $O(N)$ σ-model. In this case the fields are N-dimensional unit vectors $\vec\sigma_i$ and the Hamiltonian is given by

$$H(\{\vec\sigma\}) \;=\; -\sum_{\langle ij\rangle} \vec\sigma_i \cdot \vec\sigma_j \tag{6.156}$$

where the summation extends to all nearest neighbour pairs $\langle ij\rangle$. The algorithm we have in mind here is a simple generalization of what we have presented for the Ising model. Firstly choose a site i. Then propose a new configuration $\{\vec\sigma'\}$ in the following way:

$$\vec\sigma'_j \;=\; \vec\sigma_j \quad \text{for all } j \neq i \tag{6.157}$$

$$\vec\sigma'_i \;=\; \vec\sigma_i \cos\theta + \vec r \sin\theta \tag{6.158}$$

where $\vec r$ is a uniformly distributed unit vector perpendicular to σ_i and θ is a random angle with probability density $f(\cos\theta)$, $0 \leq \theta \leq \pi$. Let us now show that for every probability density $f(\cos\theta)$ the proposal matrix is symmetric and thus we can apply the standard Metropolis procedure. To prove this, notice that the probability distribution of $\vec r$ and θ is proportional to

$$\mathrm{d}\theta\ \mathrm{d}^N\vec r\ \delta(\vec r\cdot\vec\sigma_i)\ \delta(r^2-1) f(\cos\theta) \tag{6.159}$$

From this we can easily obtain the probability distribution of $\vec\sigma'$. Eliminating $\vec r$ and θ in favour of $\vec\sigma'_i$ we get

$$\mathrm{d}^N\vec\sigma'_i\ \delta(\vec\sigma'^2_i - 1)\left(1-(\vec\sigma_i\cdot\vec\sigma'_i)^2\right)^{1-N/2}\ f(\vec\sigma_i\cdot\vec\sigma'_i) \tag{6.160}$$

which proves that $P^{(0)}$ is symmetric.

Exercise 5: Let S be the space of $U(N)$ matrices with probability measure the Haar measure. Consider the transitions $U \to U'$ where $U' = UV$ with V is chosen with probability measure $f(V)\mathrm{d}V$. Show that the associated transition matrix is symmetric if and only if $f(V) = f(V^+)$.

At this point we must specify $f(\cos\theta)$. All possible choices are correct; however not all of them will be equally efficient. It is clear that the probability of acceptance will decrease with increasing θ and thus if θ is very large the move will be rejected with high probability and this makes the algorithm inefficient. On the other hand, if θ is very small the new configuration will be accepted with high probability. However in this case the new spin will be only slightly different from the previous one and thus the evolution of the system will be slow and the algorithm will again be inefficient. Thus $f(\cos\theta)$ must be a compromise between high acceptances and large proposed modifications of the configuration. In practice this is achieved by choosing θ uniformly between 0 and some cutoff ϵ and then tuning ϵ in order to have a mean acceptance ratio of the order 50%. Let us notice that the value of ϵ will be temperature dependent and will decrease as β goes to infinity.

An important technique which is used with the Metropolis update is the so-called multihit technique. The idea is to perform many subsequent updates (hits) at point i before moving to another point. In this case the total transition probability is simply

$$P = \frac{1}{V}\sum_i P_i^n \tag{6.161}$$

for a random update and

$$P = P_{i_1}^n P_{i_2}^n \dots P_{i_V}^n \tag{6.162}$$

for a sequential update.

To understand the convenience of this technique let us notice that in the σ-model we have

$$E(\{\vec{\sigma}'\}) - E(\{\vec{\sigma}\}) = (\vec{\sigma}_i - \vec{\sigma}_i') \cdot \sum_j \vec{\sigma}_j \tag{6.163}$$

where the sum extends over all nearest neighbours j of i. Thus the difference of energy is simply given by the difference of the spins times a vector which needs to be computed only before the first hit. In general the technique is convenient in all those cases where the energy difference has the form

$$E(\{\phi'\}) - E(\{\phi\}) = ({\phi'}_i - \phi_i)A \tag{6.164}$$

where the two configurations differ only at site i and the quantity A does not depend on the fields at point i. In these cases A needs to be computed only before the first hit, and this is convenient if A requires a lot of computation (this is the case for instance of $SU(3)$ gauge theories).

Let us now discuss a second important class of algorithms which use the so-called heat-bath method. This method is based on the following idea: given a model an a lattice, let us indicate the fields with ϕ_i, i running over the lattice sites. Fix a point i and consider the conditional probability of ϕ_i, keeping all the other fields fixed. Then choose ${\phi'}_i$ independently of the old value ϕ_i from the conditional distribution while all the other fields remain unchanged. Then sweep over the lattice points i as in the Metropolis case, either randomly or sequentially.

Let us see how it works in a specific example, the Ising model. The first thing we have to do is computing the conditional distribution. We have in this case

$$\mathrm{const}\left(\{\sigma\}_{j\neq i}\right)\exp\left(\beta\sigma_i\sum_j \sigma_j\right) \tag{6.165}$$

where the sum extends to all nearest neighbours j of i. The new spin σ' is then chosen with this probability. Thus the update at site i works as follows:

1. Set $A \leftarrow \beta\sum_j \sigma_j$;
2. Set $p_+ \leftarrow 1/(1+e^{-2A})$;
3. Choose a uniform random number U between 0 and 1;
4. If $p_+ > U$ set $\sigma' \leftarrow 1$, else $\sigma' \leftarrow -1$.

Exercise 6: Show that for the Ising model the heat-bath update is equivalent to a Metropolis update with $F(z) = z/(1+z)$.

In a similar way one can deal with the σ-model. In this case the conditional probability measure is proportional to

$$\exp\left(\beta\vec{\sigma}_i\cdot\sum_j \vec{\sigma}_j\right)\mathrm{d}^N\vec{\sigma}_i\ \delta(\vec{\sigma}_i^2-1) \tag{6.166}$$

Thus in order to apply the heat-bath method to the σ-model we must generate random variable with probability given by (6.166). This can be done although it is not completely trivial, except when $N = 3$.

Exercise 7: Develop the heat-bath algorithm for the $O(3)$ σ-model.

Hint: introduce polar coordinates choosing the z-axis parallel to the vector $\sum_j \vec{\sigma}_j$.

Let us finally notice that the multihit Metropolis update is equivalent to a heat-bath update in the limit in which the number of hits goes to infinity. Indeed, by construction, P_i restricted to the site i leaves invariant the conditional probability distribution of σ_i, given all the other spins $\{\sigma_j\}_{j\neq i}$. Let us indicate this distribution with $P^\pi(\sigma_i)$. Then by the standard results of the theory of Markov chains we have presented in Section 4 we have

$$\lim_{n\to\infty} P_i^n = P^\pi(\sigma_i) \tag{6.167}$$

and the r.h.s. is by definition the heat-bath transition probability.

Let us now discuss the performance of Metropolis and heat-bath algorithms. Their main feature is that each update is local. In a single step of the algorithm, "information" is transmitted from a given site only to its nearest neighbours. Crudely one might guess that this information executes a random walk around the lattice. In order for the system to evolve to an "essentially new" configuration, the information has to travel a distance of the order ξ, the static correlation length. One would guess, therefore, that $\tau \sim \xi^2$ near criticality. These results can be proven for the free Gaussian model (see next Section) and in general extensive simulations have shown that it is approximately true for every model and any local update (with the exception of overrelaxation).

Exercise 8: Consider a one-dimensional random walk, i.e. a Markov chain defined on the integers with $p_{ii} = 0$, $p_{i,i\pm1} = 1/2$, $p_{ij} = 0$ if $|i-j| > 1$ and $\mathrm{Prob}(X_0 = j) = \delta_{j0}$. Show that the mean square distance from the origin of the walker at time t is t, i.e. $\langle X_t^2\rangle_0 = t$.

Hint: write $X_t = \sum_{i=0}^{t-1}(X_{i+1} - X_i) + X_0$ and then prove $\langle (X_{i+1} - X_i)(X_{j+1} - X_j)\rangle_0 = \delta_{ij}$.

To conclude this Section let us discuss the embedding technique which was introduced by Cabibbo and Marinari who developed a pseudo-heat-bath algorithm for $SU(3)$ gauge theories and which has recently been extended to more general algorithms.

The idea is to consider a group of transformations $\{g_x\}$ which acts on the configurations $\{\phi_x\}$, i.e.

$$\phi_x \rightarrow \{g_x \phi_x\} \tag{6.168}$$

and leaves the integration measure invariant.

Then we consider $\{g_x\}$ as fields and update them using the induced hamiltonian

$$H'(\{g_x\}) = H(\{g_x\phi_x\}) \tag{6.169}$$

A valid algorithm is the following:

1. Keeping the fields $\{\phi_x\}$ fixed, compute the induced hamiltonian.
2. Initialize $g_x = Id_x$.
3. Update g_x with a transition probability $p(\phi; g_1 \rightarrow g_2)$ satisfying $p(\phi; g_1 g \rightarrow g_2 g) = p(g\phi; g_1 \rightarrow g_2)$ for all g and which is stationary with respect to the Boltzmann weight $\exp(-\beta H'(g))$.
4. Set $\phi'_x = g_x \phi_x$.

It is easy to check that this algorithm has the correct equilibrium distribution. Indeed the total transition probability is given by

$$P(\phi_1 \rightarrow \phi_2) = \int \mathrm{d}g\; p(\phi_1; Id \rightarrow g)\delta(\phi_2 - g\phi_1) \tag{6.170}$$

Then

$$\int \mathrm{d}\phi_1 e^{-\beta H(\phi_1)}\; P(\phi_1 \rightarrow \phi_2) = \tag{6.171}$$

$$= \int \mathrm{d}\phi_1 \mathrm{d}g_1 \mathrm{d}g_2 e^{-\beta H(g_1\phi_1)}\; p(\phi_1; g_1 \rightarrow g_2)\delta(\phi_2 - g_2\phi_1) \tag{6.172}$$

$$= \int \mathrm{d}\phi_1 \mathrm{d}g_2 e^{-\beta H(g_2\phi_1)}\; \delta(\phi_2 - g_2\phi_1) = e^{-\beta H(\phi_2)} \tag{6.173}$$

Of course one has to check also for ergodicity. If the algorithm turns out non ergodic the simplest remedy consists in mixing it with conventional update steps.

To clarify how the method works let us consider a simple example. Let $\vec{\sigma}$ be an N-dimensional unit spin and suppose we want to generate $\vec{\sigma}$ according to the probability density

$$\exp\left(\vec{A}\cdot\vec{\sigma}\right)\, d\vec{\sigma}\,\delta(\vec{\sigma}^2-1) \tag{6.174}$$

We want to use the embedding method and we consider the transformations

$$\sigma_i' = \sum_j V_{ij}^{(\alpha\beta)}(\theta)\sigma_j \tag{6.175}$$

where $\alpha\neq\beta$ run from 1 to N,

$$V^{(\alpha\beta)}(\theta) = \exp\left(\theta T^{(\alpha\beta)}\right)\,, \tag{6.176}$$

with $T_{ij}^{(\alpha\beta)} = (\delta_i^\alpha\delta_j^\beta - \delta_j^\alpha\delta_i^\beta)$ and θ is a parameter. The transformations $V^{(\alpha\beta)}(\theta)$ are a one-parameter group of rotations which leave invariant the measure $d\vec{\sigma}\,\delta(\vec{\sigma}^2-1)$. Let us now compute the induced Hamiltonian. As

$$V_{ij}^{(\alpha\beta)}(\theta) = \cos\theta\,\delta_{ij} + \sin\theta\, T_{ij}^{(\alpha\beta)} \tag{6.177}$$

we have

$$\vec{A}\cdot\vec{\sigma}' = \cos\theta\vec{A}\cdot\vec{\sigma} + \sin\theta\sum_{ij} A_i T_{ij}^{(\alpha\beta)}\sigma_j \tag{6.178}$$

Thus a valid algorithm in this case is given by:

1. Choose randomly $\alpha\neq\beta$ between 1 and N.

2. Compute the induced hamiltonian coefficients $h_1 = \vec{A}\cdot\vec{\sigma}$ and $h_2 = \sum_{ij} A_i T_{ij}^{(\alpha\beta)}\sigma_j$.

3. Initialize $\theta = 0$.

4. Update θ with an algorithm which is stationary with respect to the probability measure

$$d\theta\,\exp\left(h_1\cos\theta + h_2\sin\theta\right) \tag{6.179}$$

and whose transition probability satisfies

$$p(\vec{\sigma};\theta_1+\theta\rightarrow\theta_2+\theta) = p(V^{(\alpha\beta)}(\theta)\vec{\sigma};\theta_1\rightarrow\theta_2) \tag{6.180}$$

for all θ. (Prove that a heat-bath update satisfies the last condition).

5. Set $\vec{\sigma}' = V^{(\alpha\beta)}\vec{\sigma}$.

Using the embedding method the original problem of generating unit spins with the given probability distribution is reduced to the problem of generating an angle θ with distribution (6.179). In this specific example this reduction does not help much. However this method has been used with success in unigrid simulations of $O(N)$ σ-models.

Exercise 9: Let $\vec{\sigma}$ be an N-dimensional unit spin and suppose we want to generate $\vec{\sigma}$ according to the probability density (6.174). Now, given a random unit vector $\vec{r}$, consider the transformations

$$\vec{\sigma}' = \vec{\sigma} + (\epsilon - 1)(\vec{\sigma} \cdot \vec{r})\vec{r} \tag{6.181}$$

where $\epsilon = \pm 1$ is an Ising spin. Write down a valid algorithm to update σ based on this class of transformations.

Exercise 10: (Cabibbo-Marinari algorithm) Consider matrices $U \in SU(N)$ and the probability measure

$$\exp\left(-\beta \mathrm{Re}\ \mathrm{Tr}\ (AU)\right) dU \tag{6.182}$$

where dU is the Haar measure and A is a given matrix. Consider the transformations $U \to VU$, where V belongs to some $SU(2)$ subgroup of $SU(N)$. Write down a valid algorithm based on this group of transformations, paying attention to the ergodicity of the algorithm.

7 Overrelaxation

In this Section we want to discuss an important local algorithm which has $z = 1$, the overrelaxation algorithm. Let us firstly describe it for the Gaussian model. Here the fields ϕ_i are real variables and the action on a hypercubic lattice has the form

$$H = \frac{1}{2}\sum_i \sum_{\hat{\mu}} (\phi_{i+\hat{\mu}} - \phi_i)^2 + \frac{1}{2} m^2 \sum_i \phi_i^2 \tag{7.183}$$

As we have done for the Metropolis and the heat-bath algorithm we firstly define a transition probability matrix P_i which updates the field at site i leaving all the other ϕ_j unchanged by requiring that P_i satisfies detailed balance with respect to the conditional probability of ϕ_i, given $\{\phi_j\}_{j\neq i}$. It is easy to see that this probability, let us indicate it with $P^\pi(\phi_i)$, has the form

$$P^\pi(\phi_i) \;=\; \text{const } \times \exp\left(-\frac{1}{2\sigma^2}(\phi_i - \mu)^2\right) \mathrm{d}\phi_i \tag{7.184}$$

with

$$\sigma^2 = \frac{1}{2D + m^2} \tag{7.185}$$

and

$$\mu = \sigma^2 \sum_j \phi_j \tag{7.186}$$

where the sum extends to all nearest neighbours j of i. Let us now define the proposal. We set

$${\phi'}_i = \alpha\phi_i + \beta + \gamma X \tag{7.187}$$

where X is a Gaussian random variable with mean zero and variance one and α, β and γ constants to be determined. (7.187) corresponds to the transition probability matrix

$$P_i(\phi_i \to {\phi'}_i) = \frac{1}{\sqrt{2\pi}\gamma}\ \exp\left(-\frac{1}{2\gamma^2}({\phi'}_i - \alpha\phi_i - \beta)^2\right) \tag{7.188}$$

Let us compute α, β and γ by requiring

$$P^\pi(\phi_i)\ P_i(\phi_i \to \phi'_i) \ = \ P^\pi(\phi'_i)\ P_i(\phi'_i \to \phi_i) \tag{7.189}$$

We get two equations

$$\frac{1}{\sigma^2} \ = \ \frac{1}{\gamma^2}(1-\alpha^2) \tag{7.190}$$

$$\frac{\mu}{\sigma^2} \ = \ \frac{\beta}{\gamma^2}(1+\alpha) \tag{7.191}$$

Since $\sigma^2 \geq 0$ we see that we must have $1-\alpha^2 \geq 0$. It is conventional to use instead of α the overrelaxation parameter $\omega = 1-\alpha$, which must then satisfy $0 \leq \omega \leq 2$. The constants β and γ are then computed in terms of ω. In this way we find that the proposal

$$\phi'_i \ = \ (1-\omega)\phi_i + \mu\omega + \sigma\sqrt{\omega(2-\omega)}X \tag{7.192}$$

gives rise for every ω to a Monte Carlo algorithm with the correct probability distribution. Let us notice a few limiting cases. For $\omega = 1$ we have

$$\phi'_i \ = \ \mu + \sigma X \tag{7.193}$$

and thus it corresponds to heat-bath, while for $\omega = 2$ we have

$$\phi'_i = \ -\phi_i + 2\mu \tag{7.194}$$

which is a microcanonical non-ergodic update.

(7.192) defines P_i. In order to obtain the complete transition matrix we must then sweep over the lattice. We will study the case of a sequential update with checkerboard ordering, but the results are true for any sequential update (but not for random updates as we shall see!). In this case a single sweep corresponds to two steps, in the first one we update the even sites, while in the second we update the odd ones.

Let us firstly perform a Fourier expansion. Let $\tilde{\phi}_\pm$ be the Fourier transform of the fields, i.e.

$$\tilde{\phi}_\pm(p) \ = \ \frac{1}{2}\sum_x e^{-ipx}\ (1 \pm (-1)^{\sum_\mu x_\mu})\phi_x \tag{7.195}$$

and $\tilde{X}_\pm(p)$ the Fourier transform of the random numbers

$$\tilde{X}_\pm(p) \ = \ \frac{1}{2}\sum_x e^{-ipx}\ (1 \pm (-1)^{\sum_\mu x_\mu})X_x \tag{7.196}$$

Then the update of the even sites corresponds to

$$\tilde{\phi}'_+(p) \ = \ (1-\omega)\tilde{\phi}_+(p) + \omega\lambda(p)\tilde{\phi}_-(p) + \tilde{X}_+(p) \tag{7.197}$$

$$\tilde{\phi}'_-(p) \ = \ \tilde{\phi}_-(p) \tag{7.198}$$

while the update of the odd sites corresponds to

$$\tilde{\phi}'_+(p) \ = \ \tilde{\phi}_+(p) \tag{7.199}$$

$$\tilde{\phi}'_-(p) \ = \ (1-\omega)\tilde{\phi}_-(p) + \omega\lambda(p)\tilde{\phi}_+(p) + \tilde{X}_-(p) \tag{7.200}$$

where

$$\lambda(p) \;=\; \frac{2D - \hat{p}^2}{2D + m^2} \tag{7.201}$$

and

$$\hat{p}^2 \;=\; 4 \sum_{\mu=1}^{D} \sin^2 \frac{p_\mu}{2} \tag{7.202}$$

Then introducing

$$\chi(p) \;=\; \begin{pmatrix} \tilde{\phi}_+(p) \\ \tilde{\phi}_-(p) \end{pmatrix} \tag{7.203}$$

and an analogous vector for the random numbers, a full sweep reads

$$\chi' \;=\; M(\lambda,\omega)\chi + \; A(\lambda,\omega)X \tag{7.204}$$

where

$$M(\lambda,\omega) \;=\; \begin{pmatrix} 1-\omega & \omega\lambda \\ (1-\omega)\omega\lambda & 1-\omega+\omega^2\lambda^2 \end{pmatrix} \tag{7.205}$$

and

$$A(\lambda,\omega) \;=\; \begin{pmatrix} 1 & 0 \\ (1-\omega) & 1 \end{pmatrix} \tag{7.206}$$

The convergence of the algorithm is determined by the largest (in absolute value) eigenvalue of M. In the heat-bath case $\omega = 1$, the eigenvalues are 0 and $\lambda(p)$. Then

$$\tau_{exp,p} \;=\; -\frac{1}{2\log|\lambda(p)|} \tag{7.207}$$

Since $|\lambda(p)| \le \lambda(0) < 1$ the critical behaviour is dominated by the modes with low momentum p. Then

$$\tau_{exp} \approx \; \frac{D}{m^2} \sim \xi^2 \tag{7.208}$$

which shows that in this model $z_{exp} = 2$. The same is then expected to be true for $z_{int,A}$ for all observables A which are strongly coupled with the low-momentum modes. The typical one is the magnetization

$$M = \sum_i \phi_i = \; \tilde{\phi}_+(0) + \tilde{\phi}_-(0) \tag{7.209}$$

Let us now compute the eigenvalues for general ω. We find

$$1 - \omega + \omega^2\lambda^2/2 \,\pm \sqrt{(1-\omega+\omega^2\lambda^2/2)^2 - (1-\omega)^2} \tag{7.210}$$

The term under the square root can be rewritten as

$$\omega^2\lambda^2(\omega^2\lambda^2 - 4\omega + 4)/4 \tag{7.211}$$

Then let us choose ω such that

$$\omega^2\lambda(0)^2 - 4\omega + 4 = 0 \tag{7.212}$$

We get

$$\omega = \frac{2}{\lambda(0)^2}\left(1-\sqrt{1-\lambda(0)^2}\right) = 2\left(1+\frac{m^2}{2D}\right)\left(1-\frac{m}{\sqrt{2D}}\right) \tag{7.213}$$

which satisfies $0 < \omega < 2$ if $0 < m < \sqrt{2D}$. In this case, for $p \neq 0$ the square root is always imaginary and then it is easy to check that the absolute value of the eigenvalues is always $\omega - 1$. This means that

$$\tau_{exp,p} = \frac{\sqrt{D}}{2m} \sim \xi \tag{7.214}$$

for all modes p. It follows that, if we tune ω as

$$2 - \omega \sim 1/\xi \tag{7.215}$$

we obtain an algorithm which has a critical exponent $z \approx 1$.

We want to stress that this result is strictly connected with the fact that we have performed a sequential update (we used the checkerboard update but this is not essential). Had we used a random update, we would have got $z_{exp} = 2$ for all ω's. Indeed overrelaxation works because the eigenvalues of the transition matrix can be complex and this is possible only if the full transition matrix P does not satisfy detailed balance, since in the latter case P is self-adjoint and thus with real spectrum.

This form of overrelaxation with a tunable parameter ω cannot be easily extended to non Gaussian models. There is however a variant, called hybrid overrelaxation, which can be easily extended to more general models, and in particular to gauge theories. The idea here is to mix microcanonical and heat-bath sweeps. More precisely an iteration of the algorithm consists in one heat-bath (sequential) sweep and in N microcanonical (sequential) sweeps. If N scales like ξ it is likely that one iteration of this algorithm is equivalent to $N+1$ sweeps of the tunable algorithm with ω satisfying (7.215). Thus one expects the autocorrelation time (expressed in sweeps) to scale again like ξ, i.e. $z \approx 1$. This is supported by an analytic study of the Gaussian model and by extensive simulations of two-dimensional σ-models and lattice gauge theories.

Exercise 1: Consider the following algorithm: choose a lattice site i; then perform N microcanonical updates followed by 1 heat-bath update; then sweep over the lattice. What is the expected critical exponent z?

8 Bibliography

Much of the material presented in these lectures is taken from A.D.Sokal's Lausanne lectures, *Monte Carlo Methods in Statistical Mechanics: Foundations and New Algorithms.* There one can also find a very readable introduction to cluster algorithms, multigrid methods and algorithms for self-avoiding walks. For the material presented in the first two Sections we refer to D. E. Knuth, *The Art of Computing Programming*, Vol.2, Chapt. 3, Addison-Wesley. An extensive discussion of static Monte Carlo methods can be found in Y. Sobal, Le Méthode de Monte Carlo, Editions Mir. An easy introduction to the theory of (finite) Markov chains can be found in J. Kemeny and L. Snell, *Finite Markov Chains*, Springer-Verlag. For the latest developments on algorithms we refer to the Proceedings of the various Lattice conferences.