FYTN03
HT10

Lecture 10, FYTN03 Computational Physics

[G&N: chap. 7, appendix F&G]
[NR: chap. 7]

1 Stochastic processes: non-uniform probability distributions and vicious walkers

In this lecture we will discuss how to generate random numbers from non-uniform distributions. As, a particular application we consider random walks with long jump lengths, and show that such a process in fact "invalidate" the central limit theorem. Finally, we consider the dynamics of interacting random walkers, through a study of reaction-diffusion processes (vicious walkers).

2 Transforming Random Numbers

On the computer, one typically has access to some random number generator that delivers (pseudo) random numbers R uniformly distributed between 0 and 1,

$$p(r) = \begin{cases} 1 & 0 < r < 1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Such random numbers are sometimes called rectangularly distributed.

In this section, we look at how to obtain random numbers with other distributions, assuming that we have uniformly distributed random numbers at our disposal.

2.1 Random integers

Suppose we are interested in generating random numbers J which takes integer values $i = 1, 2, 3, ..., N$. Then simply:

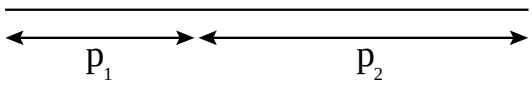
$$J = 1 + [N \cdot R] \tag{2}$$

where $[x]$ is obtained by truncation of x (for instance, $[7.9] = 7$).

2.2 The Transformation Method - discrete case

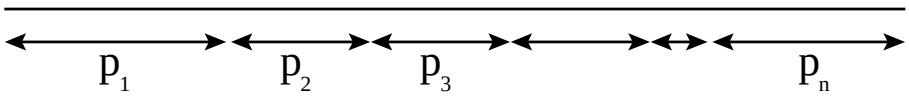
Suppose we want to generate a random number Y from a discrete probability distribution.

Consider first the case where the random number Y takes two possible values values y_1 and y_2 with probabilities p_1 and p_2 respectively. Given a uniform random number R between 0 and 1, one then simply chooses y_1 if $R < p_1$ and y_2 if $R \geq p_1$, see figure below.



For the random walk problem in the the previous lecture we use a discrete distribution with $p_1 = p_2 = 1/2$ in order to determine whether the walker took a step to the left or right.

For the case where Y is a random variable which takes values $y_1, y_2, ..., y_n$ with probabilities $p_1, p_2, ..., p_n$, the method above is straightforward to generalize. We draw a uniform random number R and assign the value y_i to Y if $\sum_{j=0}^{i-1} p_j \leq R < \sum_{j=0}^{i} p_j$, see figure below.



It is convenient to introduce the *cumulative distribution*:

$$C_i = \sum_{j=0}^{i} p_j \tag{3}$$

where we define $p_0 \equiv 0$. Note that the normalization condition becomes $C(n) = 1$. The relation above can then be written:

$$C_{i-1} \leq R < C_i. \tag{4}$$

So the procedure becomes: Calculate the cumulative distributions using Eq. (3) [this need to be done only once, if several random numbers are needed]. Draw a uniform random number R, and search the C_i:s for the i satisfying Eq. (4). Assign the value y_i to the random variable Y. This way of generating random numbers is called the (discrete) transformation method .

2.3 The Transformation Method - continuum case

Suppose that we want to generate random numbers Y from a continuous probability distribution $p(y)$. This is simply done by taking the continuum limit of Eqs. (3) and (4): we introduce the cumulative probability distribution:

$$C(y) = \int_{-\infty}^{y} p(y')dy' \tag{5}$$

and choose the Y which satisfies $R = C(Y)$, or,

$$Y = C^{-1}(R). \tag{6}$$

This way of generating random numbers with different distributions will be called the continuous transformation method. The normalization condition $\int_{-\infty}^{\infty} p(y)dy = 1$ can be written $C(\infty) = 1$. An alternative derivation of Eqs. (5) and (6) can be found in G&N appendix F.2.

Example: The exponential distribution.

Suppose we have access to random numbers R uniformly distributed between 0 and 1 and want random numbers with the distribution

$$p(y) = \begin{cases} \lambda \mathrm{e}^{-\lambda y} & y \geq 0 \\ 0 & y < 0 \end{cases} \qquad (\lambda > 0)$$

Use the transformation method. The first step is to calculate the cumulative distribution

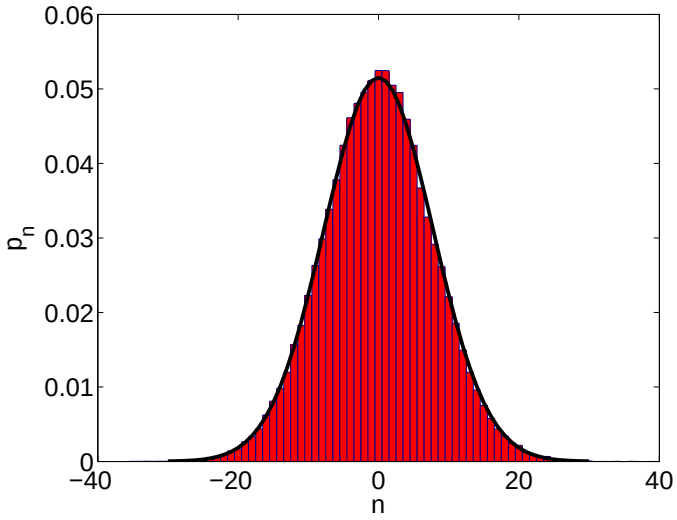
$$C(y) = \int_{-\infty}^{y} p\left(y'\right) dt = \left[-\mathrm{e}^{-\lambda y'}\right]_0^y = 1 - \mathrm{e}^{-\lambda y}$$

The transformation $R \to Y$ is then obtained by solving

$$C(Y) = R \qquad \Rightarrow \qquad 1 - \mathrm{e}^{-\lambda Y} = R \qquad \Rightarrow \qquad Y = -\frac{1}{\lambda} \ln (1 - R)$$

So, $Y = -\lambda^{-1} \ln \tilde{R}$ has the desired distribution [note that $\tilde{R} = 1 - R$ is another uniform random number between 0 and 1].

As an illustration of how to use exponentially distributed random numbers, we have below made a simulation of a random walker on a one-dimensional line, where the jump length of the walker (rather than of constant size a as in previous lecture), was taken to be exponentially distributed; steps to the left and right was taken with equal probability.



The central limit theorem tells us that the position probability distribution should be normal, and indeed we find that this is the case; the black line is the prediction of the central limit theorem. We used 10^5 simulation runs, $\Delta t = 1$, and the average jump length $\lambda = 1$, and binned the data into bins of which 80 are shown above (n labels different bins).

Example: The power-law distribution.

Suppose we have access to random numbers R uniformly distributed between 0 and 1 and want random numbers with the power-law distribution

$$p(y) = \begin{cases} \frac{\alpha}{y_c} \left(\frac{y}{y_c} \right)^{-1-\alpha} & y > y_c \\ 0 & y < y_c \end{cases}$$

for $\alpha > 0$. Use the transformation method. The cumulative distribution is:

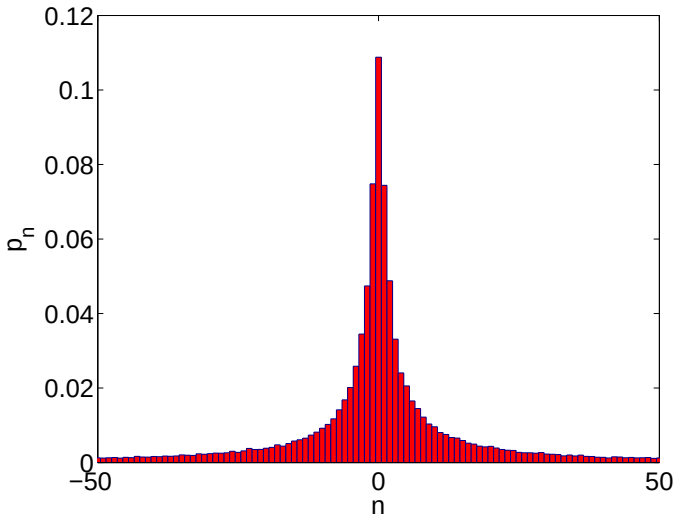
$$C(y) = \int_{-\infty}^{y} p\,(y')\,dy' = \left[-\left(\frac{y'}{y_c}\right)^{-\alpha} \right]_0^y = 1 - \left(\frac{y}{y_c}\right)^{-\alpha}$$

Thus,

$$C(Y) = R \qquad \Rightarrow \qquad 1 - \left(\frac{Y}{y_c}\right)^{-\alpha} = R \qquad \Rightarrow \qquad Y = y_c \tilde{R}^{-1/\alpha},$$

with $\tilde{R} = 1 - R$, has the desired distribution.

As an illustration of how to use power-law distributed random numbers, we have below made a simulation of symmetric random walk on a one-dimensional line, where the jump length of the walker was taken to be distributed according to a power-law.



We used 10^5 simulation runs, $\Delta t = 1$, $y_c = 1$, and $\alpha = 0.5$ and binned the data into bins of which 100 are show above. We notice that the probability distribution is *not* normal!

What went wrong above? The answer is that for a power-law distribution with $0 < \alpha < 2$ the second moment of our probability distribution for the jump lengths is infinite (for $0 < \alpha < 1$ also the first moment is infinite). In contrast, in the proof of the central limit theorem we used the assumption that $\sigma^2 < \infty$. Random processes of the type above are called Lévy flights and has received quite a lot of attention in recent years, see, for instance, the review article *R. Metzler and J. Klafter, Phys.*

Rep., vol. 339, pp. 1-77 (2000). It is possible to derive a *generalized* central limit theorem which gives an analytic expression for the probability distribution for Lévy flight type processes (but this is beyond the scope in this lecture).

Example: The Box-Müller method for normally distributed random numbers.

The transformation method cannot be directly applied to the distribution

$$p(y) = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-y^2/2} \qquad \text{(assume, for simplicity, zero mean and unit variance)}$$

because we cannot get a closed expression for $C(y)$. It turns out, however, that this problem can be circumvented by considering two independent variables Y_1 and Y_2 with the same distribution $p(y)$,

$$p\left(y_1, y_2\right) = \frac{1}{2\pi} \mathrm{e}^{-(y_1^2+y_2^2)/2} \, .$$

In polar coordinates (ρ, θ), this distribution becomes

$$p\left(\rho, \theta\right) = p\left(y_1, y_2\right) \underbrace{\left| \frac{\partial\left(y_1, y_2\right)}{\partial\left(\rho, \theta\right)} \right|}_{\rho} = \underbrace{\rho \, \mathrm{e}^{-\rho^2/2}}_{p(\rho)} \cdot \underbrace{\frac{1}{2\pi}}_{p(\theta)} \, .$$

where we defined the Jacobian

$$\left| \frac{\partial\left(y_1, y_2\right)}{\partial\left(\rho, \theta\right)} \right| = \left| \begin{array}{cc} \frac{\partial y_1}{\partial \rho} & \frac{\partial y_1}{\partial \theta} \\ \frac{\partial y_2}{\partial \rho} & \frac{\partial y_2}{\partial \theta} \end{array} \right|$$

The fact that this distribution factorizes, $p(\rho, \theta) = p(\rho)p(\theta)$, implies that ρ and θ can be generated independently.

- The θ distribution is uniform [with cumulative distribution $C(\theta) = \theta/(2\pi)$]. A random variable is then obtained as $\theta = 2\pi R_1$, where R_1 is uniformly distributed between 0 and 1.
- Use the transformation method for ρ.

  $$C(\rho) = \int_0^{\rho} t \mathrm{e}^{-t^2/2} dt = 1 - \mathrm{e}^{-\rho^2/2} = R \;\Rightarrow\; \rho = \sqrt{-2\ln(1-R)}$$

  So, ρ can be obtained as $\rho = \sqrt{-2\ln R_2}$, where R_2 is another uniformly distributed random number between 0 and 1.

A transformation back to Cartesian coordinates gives us two independent random numbers with the desired distribution:

$$\begin{cases} Y_1 = \sqrt{-2\ln R_2}\cos 2\pi R_1 \\ Y_2 = \sqrt{-2\ln R_2}\sin 2\pi R_1 \end{cases}$$

2.4 The Accept/Reject Method

The continuous transformation method is simple and convenient provided that a closed expression for C^{-1} can be obtained, but this is far from always the case. A more general approach, which does not require knowledge of C^{-1}, is the *accept/reject method*.

To generate random numbers with a given distribution $p(x)$, the accept/reject method method makes use of an auxiliary function $f_0(x)$, which must satisfy $f_0(x) \geq p(x)$. $f_0(x)$ should be chosen so that it is easy to obtain random numbers from the distribution:

$$p_0(x) = \frac{f_0(x)}{\int_{-\infty}^{\infty} f_0(x)dx} \tag{7}$$

[note that $f_0(x)$ itself is not a normalized distribution]. For a given $f_0(x)$, the method is as follows:

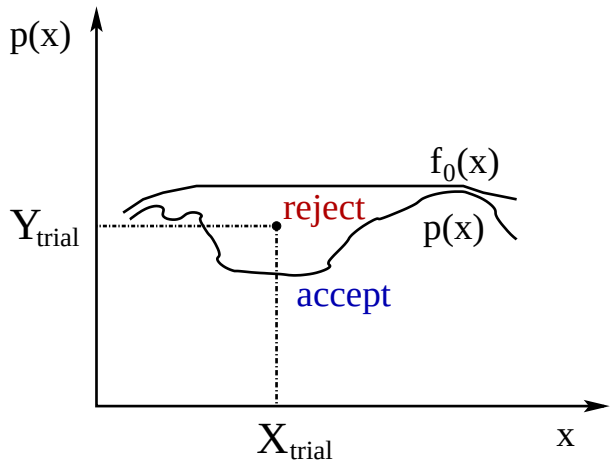
1. Draw a (trial) random number X_{trial} from the distribution $p_0(x)$.

2. Draw a uniform random number R between 0 and 1. Accept X_{trial} if

$$R < \frac{p(X_{\text{trial}})}{f_0(X_{\text{trial}})},$$

   and reject otherwise.

In the scheme above we choose a random point $(X_{\text{trial}}, Y_{\text{trial}})$ [where $Y_{\text{trial}} = f_0(X_{\text{trial}})R$] uniformly in the area under $f_0(x)$, see figure below.



We then accept only those points (X, Y) which are in the area under $p(x)$. The X component of the accepted points has the distribution $p(x)$.

Example: Kahn's method for normally distributed random numbers.

Consider the distribution

$$p(x) = \begin{cases} 2\dfrac{1}{\sqrt{2\pi}}\mathrm{e}^{-x^2/2} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

(if a random number with this distribution is given a random sign, a normally distributed random number is obtained). Use the accept/reject method with

$$f_0\,(x) = \begin{cases} \sqrt{\dfrac{2}{\pi}}\mathrm{e}^{-x+1/2} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

This choice of f_0 is OK since

$$\frac{p(x)}{f_0(x)} = \mathrm{e}^{-(x-1)^2/2} \leq 1\,,$$

and the corresponding probability distribution is

$$p_0(x) = \frac{f_0(x)}{\int_{-\infty}^{\infty} f_0(x)dx} = \begin{cases} e^{-x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

We now follow the steps above:

1. Choose $X_{\text{trial}} = -\ln R_1$ where R_1 is a uniform random number $\in [0, 1]$ (exponential distribution; see previous example).

2. Accept X_{trial} if

$$R_2 < \frac{p(X_{\text{trial}})}{f_0(X_{\text{trial}})} = \mathrm{e}^{-(X_{\text{trial}}-1)^2/2}$$

How efficient is this method? This depends crucially on the acceptance rate, which is the area under $p(x)$ (which is 1) divided by the area under $f_0(x)$. In this particular example, we find an acceptance probability of

$$\frac{1}{\sqrt{\frac{2e}{\pi}}\int_0^\infty \mathrm{e}^{-x}dx} \approx 0.76\,.$$

3 Random Number Generators*

Random numbers are generally pseudo random numbers on the computer, obtained by a deterministic algorithm. Whether or not these random numbers are "random enough" depends on both the algorithm used and the problem at hand. The use of deterministic algorithms may seem strange, but has two major advantages, speed and reproducibility. Reproducibility can be a valuable property when debugging a program.

Most programming languages provide some built-in random number generator, but there are sometimes good reasons not to use these. One possible reason is portability. Another possible reason is that built-in random number generators are occasionally of quite poor quality.

3.1 The Linear Congruential Method

A historically much used deterministic algorithm for generating random numbers is the linear congruential method. Many random number generators rely on this algorithm or combines this method with other algorithms. In its simplest version, this method uses a recursion formula of the form

$$i_{j+1} = ai_j + c \qquad \mod m$$

to generate a sequence $i_1, i_2, \ldots$ of integers [$a \mod b$ gives the remainder when a is divided by b]. Th random number generator has to initiated with a "seed" number i_0. Putting $x_j = i_j/m$, we obtain a normalized sequence $x_1, x_2, \ldots$ of numbers between 0

and 1. The hope is that these numbers, for a suitable choice of the integer parameters a, c and m (see table in NR), will behave as approximately independent and uniformly distributed random numbers. However, clearly, these x_j's are not independent, and therefore it is important to test how strong the correlations between the x_j's are. One way to do this is as follows:

- x_1, x_2, ... should be uniformly distributed on the unit interval from 0 to 1.
- (x_1, x_2), (x_3, x_4), ... should be uniformly distributed on the unit square.

$\vdots$

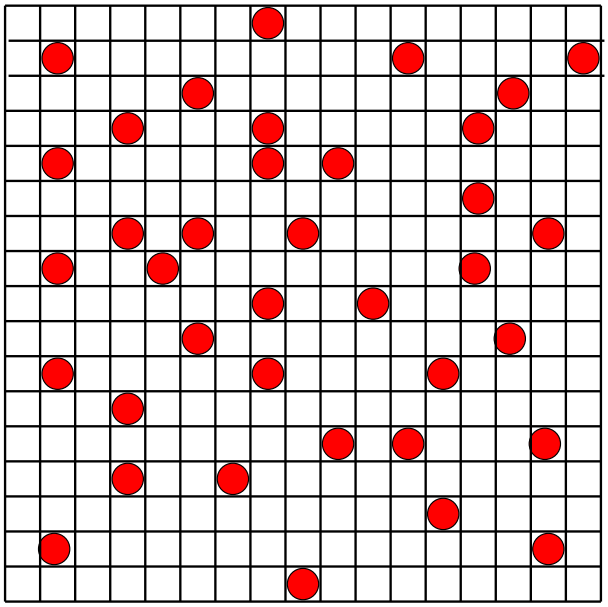
- $(x_1, \ldots, x_n)$, $(x_{n+1}, \ldots, x_{2n})$, ... should be uniformly distributed on the unit cube in n dimensions.

For the linear congruential method with $c = 0$, it can be shown that all possible n-tuples $(x_{j+1}, \ldots, x_{j+n})$ fall onto one of at most $(n!\, m)^{1/n}$ different hyperplanes. This number is not very large; if we, for example, take $m = 2^{32}$ and $n = 10$, then $(n!\, m)^{1/n} \approx 42$. This shows that this method exhibits correlations that definitely may cause problems in applications where many random numbers are needed. It is also worth noting that the recursion formula above is periodic, with a period of m or smaller, so m should be very large.

There are methods that are better than this one; for a discussion, see NR. Nevertheless, it should be kept in mind that random number generators are not perfect, and they should be chosen with care in applications where lots of random numbers are needed.

4 Reaction-diffusion

Consider a number of particles positioned on a lattice in one, two or three dimensions.



We then let the particles hop randomly on the lattice (diffusion), and assume that whenever two particles happen to find themselves on the same lattice site they annihilate each other (reaction). For the blood-thirsty readers, it may be more helpful to think of a number of drunkards walking around and whenever they happen to meet they kill each other - vicious walkers.

The example above is the simplest example of a reaction-diffusion process - all of the particles are identical (of type A), and have the same diffusion constants (hop rate multiplied by the lattice spacing squared). The type of process described above is usually written as:

$$A + A \to 0. \tag{8}$$

More complicated reaction-diffusion process may involve more types of particles (A,B,C etc), other geometries, reactions may require more complicated criterion than double occupancy etc. Typically one is interested in investigating how the ensemble averaged density $\langle \rho(t) \rangle$ of particles decay with time. The mathematical machinery for analytically solving reaction-diffusion problems is rather involved, and for most kinds of processes no analytical solutions exists. For the type of process in Eq. (8) one can obtain the ensemble averaged density decay for *long* times, which satisfy:

$$\begin{aligned} \langle \rho(t) \rangle \quad \sim \quad & t^{-1/2}, \qquad \text{in 1d} \\ & t^{-1} \log(t) \qquad \text{in 2d} \\ & t^{-1} \qquad \text{in 3d} \end{aligned} \tag{9}$$

in the thermodynamic limit. A simple mean-field equation for the type of process above would be to write $d\langle \rho(t) \rangle / dt = -\lambda \langle \rho(t) \rangle^2$, this equation has solution

const $\times\, t^{-1}$ for long times. We thus see that it is only in 3d that such a mean-field treatment is correct. In lower dimension (particularly in 1d) the system is instead said to be fluctuation-dominated [see the review article: A.R. Cadilhe, M.L. Glasser and V. Privman, Int. J. Mod. Phys. B, vol. 11, pp. 109-114 (1997), http://arxiv.org/abs/cond-mat/9606224].

Let us make a sketch how to simulate reaction-diffusion systems. A simple pseudo-code is as follows:

1. Randomly position n particles on the (1d, 2d or 3d) lattice. Avoid double occupancy. Set the time t equal to zero.

2. Pick one particle at random [see Eq. (2)]. Also, pick, with equal probability, any nearest neighbor site to the chosen particle.

3. Check if the chosen nearest neighbor site is occupied by another particle. If so, remove the corresponding two particles from the system (and decrease the number of particles n accordingly), otherwise update the position of the chosen particle. Make sure that an attempt to move out of the system is not successful. Update the time: $t \to t + \Delta t/n$; we here divide by n so that *each* particle's hop rate $= 1/\Delta t$ is kept fixed during the simulation.

Steps 2 and 3 above are then repeated until some specified stop time is reached; during this scheme the number of particles at pre-determined sampling times is stored. Once the stop-time is reached one goes back to step 1 for a simulation run etc. If this process is repeated many times one gets the ensemble averaged density $\langle \rho(t_s) \rangle$ at different sampling times t_s. It is usually a good idea to plot $\langle \rho(t) \rangle$ vs t on a log-log plot: since often $\langle \rho(t) \rangle \sim t^{-\gamma}$, a log-log plot give a straight line with slope $-\gamma$.

In step 1 above, one must randomly position n particles on M lattice sites avoiding double occupancy. A simple algorithm for this is due to Bebbington [see A.C. Bebbington, Appl. Stat. vol. 24, p. 136 (1975).]:

- Start with the first lattice site. Choose to occupy that lattice site with probability n/M. If the lattice site was chosen to be occupied decrease n by 1 ($n \to n-1$). Reduce M by one ($M \to M-1$), and proceed to the next lattice site and occupy with probability n/M etc. Note that n and M changes during the loop over lattice sites.

Going through all lattice sites, $1, ..., M$, using the scheme above results in exactly n lattice sites being chosen (with equal probability).