Homework 1; due on Tuessday, September 25

PY 502, Computational Physics, Fall 2018

Department of Physics, Boston University

Instructor: Anders Sandvik

TEST OF A RANDOM NUMBER GENERATOR USING RANDOM WALKS

In this assignment, you will test the quality of a random number generator by using it to simulate a random walk, comparing the exact probability distribution with ones obtained in simulations.

Random walk

Consider a stochastic process in which at each time step $t = 1, \ldots, n$ we can move one step "up" or "down" ($x \to x \pm 1$, with probability 1/2 for + and −), starting at $t = 0$ at $x = 0$. Three realizations of such a random walk with $n = 100$ are shown in Fig. 1.

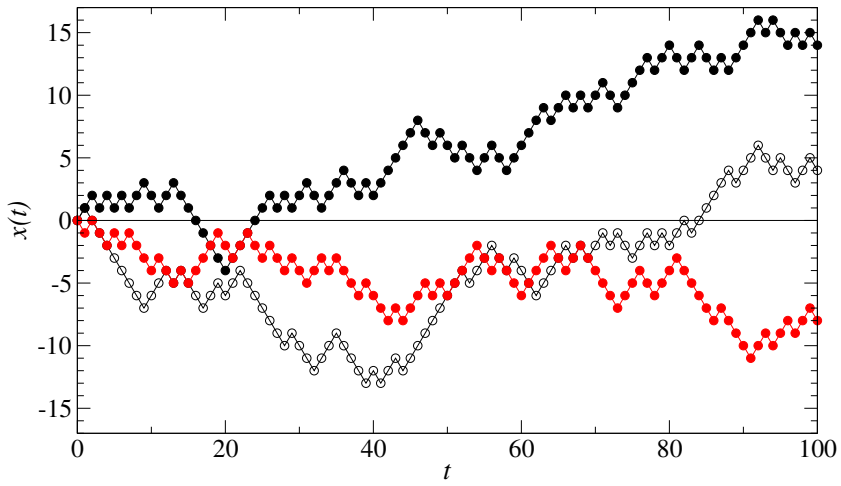

Figure 1: Three different realizations of a random walk with $n = 100$ steps.

It is easy to calculate the probability distribution of x after n steps: The total number of walks having n_+ moves up and n_- down is

$$N(n_+, n_-) = \frac{n!}{n_+! n_-!}. \tag{1}$$

The probability for each combination of up and down moves (one random walk) is the same; $1/2^n$. Since $n = n_+ + n_-$ and x after n steps is $n_+ - n_-$, we can write the distribution after n steps as;

$$P_n(x) = \frac{1}{2^n} \frac{n!}{[(n+x)/2]![(n-x)/2]!}. \tag{2}$$

Here it should be noted that if n is even, x must also be even (for a non-zero probability), whereas if n is odd x is also odd.

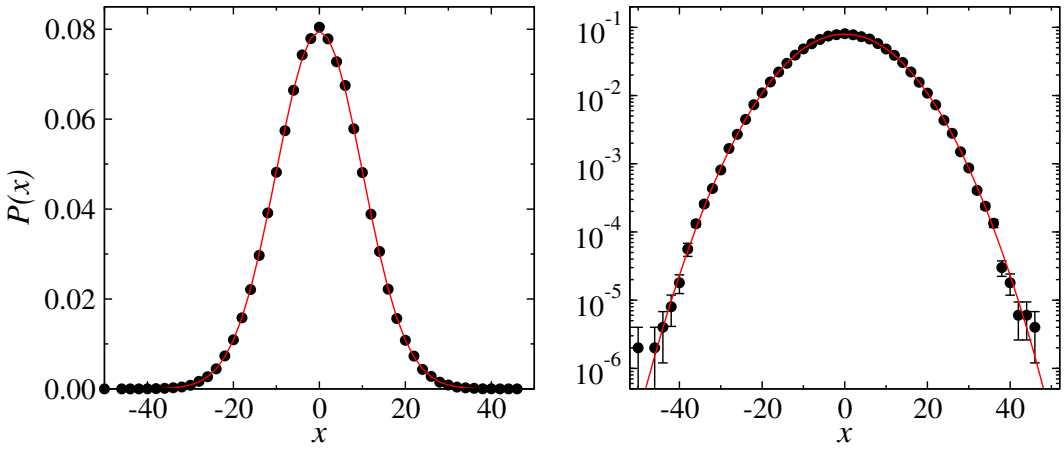

Figure 2: Probability distribution of the displacement after a random walk of $n = 100$ steps, based on 5×10^5 different walks (circles with error bars), compared with the exact distribution (solid curve) plotted on linear (left) and log (right) scales.

Simulated random walk

In a simulation, we use a random number generator to make the decisions of up or down movement at each time step (e.g., we generate a floating point number r between 0 and 1, and move up or down if $r < 1/2$ and $r \geq 1/2$, respectively). If the random numbers are not perfect, there will be deviations in the probability distribution from the exact distribution (2).

To compute the simulated distribution, we have to repeat simulations many times, using different sequences of random numbers. Fig. 2 shows the distribution for $n = 100$ steps, obtained using 5×10^5 independent simulated random walks with a good random number generator (as the examples shown in Fig. 1). Here "error bars" for the simulation results have been computed by grouping the results of the simulations into 50 "bins" and computing the standard deviation of the average result based on these bin results. The error bars, where visible, show semi-quantitatively that the deviations are consistent with the expected statistical fluctuations; the exact distribution falls mostly within the error bars (statistically, one would expect about 2/3 of the points to be within one standard deviation of the true result). For a bad random number generator, the exact distribution should fall significantly outside the error bars.

Expected statistical fluctuations

We can only perform a finite number of these simulated walks, and therefore we will not obtain the exact distribution (2). We then have to analyze the statistics of the expected *deviations* from the exact distrubution, to determine whether the observed deviations from (2) are just due to natural statistical fluctuations based on a finite sample, or whether they are larger than would be expected (in which case they must be due to flaws in the random number generator used). Instead of just computing error bars, let us analyze the statistical fluctuations theoretically.

Let us estimate what deviations from the exact distribution (2) that should be expected based solely on a finite number of walks used to estimate the distribution. If we approximate the statistics of

the final point of the random walk (after n steps) as "counting statistics" (which would results from independent Poisson distribution for all x—this is not strictly true here because we do a fixed number N_w if walks but for large N_w this will not matter), then if we end up with N_x walks at final position x the expected standard deviation would be $\sqrt{N_x}$. Thus, the expected number of "counts" and its standard deviation for x based on a total number N_w of random walks is

$$C_n(x) = N_w P_n(x) \pm \sqrt{N_w P_n(x)}. \tag{3}$$

Let us now divide this by N_w, to get a properly normalized probability, and compute the expected deviation from the actual distribution;

$$\Delta_n(x) = C_n(x)/N_w - P_n(x). \tag{4}$$

The expected value of this quantity (for a perfect random number generator) is 0, and from (3) we see that the expected (standard) deviation is;

$$\sigma_n(x) = \sqrt{\frac{P_n(x)}{N_w}}. \tag{5}$$

We can now compute the total *squared* deviation (in order to have a sum over positive numbers characterizing the average deviation) over all final outcomes x, and call it Δ^2.

$$\Delta^2 = \sum_{x=-n}^{n} \Delta_n^2(x). \tag{6}$$

According to Eq. (5) its expected value should be

$$\langle \Delta^2 \rangle = \sum_x \sigma_n^2(x) = \frac{1}{N_w} \sum_x P_n(x) = \frac{1}{N_w}. \tag{7}$$

It is appropriate to take the square root, to get an RMS (root-mean-square) deviation;

$$\Delta = \sqrt{\sum_x \Delta_n^2(x)} = \frac{1}{\sqrt{N_w}}. \tag{8}$$

We can test the random number generator by computing Δ according to (6) with (4), where $C_n(x)$ is the actual count based on N_w simulations. According to Eq. (8), for increasing number N_w of simulated random walks, this number should decay as $1/\sqrt{N_w}$ if the random number generator is good. If, on the other hand, the generator is bad, we expect Δ to approach some non-zero value as $N_w \to \infty$, reflecting a distribution for the "random" walk different from (2). We can also look at $\sqrt{N_w}\Delta$, which should approach 1 for a good generator and diverge as $\propto \sqrt{N_w}$ for a bad generator.

Programming task

The random number generator you should test here is the 64-bit linear congruential generator discussed in class, which uses the algorithm

$$r_{n+1} = a \cdot r_n + c, \tag{9}$$

with $a = 2862933555777941757$ and $c = 1013904243$ (remember to include `_8` at the end of these constants in the program, to make them "long" integers).

Normally, when using this kind of random number generator, the integers would be converted into double precision floating-point numbers between 0 and 1. Here we will instead test the individual bits of the sequence of integers. The Fortran 90 function `btest(r,b)` can be used to test (true or false) the value of bit $b = 0, \ldots 31$. We can use the value of a given bit b for the $x = \pm 1$ decision in the random walk. We want to check for differences in the randomness among the bits, through the quality of random walks based on them.

Write a program that tests all the bits in the same run, i.e., perform 64 different random walks simultaneously, using the 64 different bits (i.e., you need a 64-element array to keep the positions x of all the "walkers"). The program should read in the number of random walks to be performed, N_w, and the number of steps to be taken in each walk, n, from a file `read.in` (both integers on the same line). Read the initial integer r_0 to be used as a seed for the random number generator (9) from this same file (the third integer on the single line).

After each set of walks, accumulate the counts $C_n(x)$ of x values for for each bit b in an array (i.e., it should be a two-dimensional array, with indices x, b). After all the N_w walks have been performed, compute the deviations Δ based on Eqs. (6) and (4). Write $\sqrt{N_w}\Delta$ for all b to a file `d.dat` (containing 64 lines, with b and $\sqrt{N_w}\Delta(b)$ on each line). Also, the program should write the full distributions for each b to files with names `pnn.dat`, where `nn` stands for the bit numbers; `nn=00,01,...,63` [use `achar(48+i)` for the character `i`, $i = 0, \ldots, 9$]. Write a separate line for each x and the corresponding simulated probability (number of counts divided by N_w), but include only x for which the count is not zero. Write the exact probability $P_n(x)$ in the third column.

Hint: To compute $P_n(x)$, write down an expression for its logarithm first and evaluate it, in order to avoid numerical problems with very large and very small numbers.

Simulation tasks and report

Do calculations for $n = 100$ and vary the number of walks. Use $N_w = 10^m$, with $m = 2, 3, \ldots$, up to m as high as you can reasonably do ($m = 6$ or 7 will be sufficient but go further if you wish).

Write a report showing the ability (or lack thereof) of the individual bits to reproduce the correct statistics of the random walk. Show some illustrative graphs of the simulated probability distributions compared with the correct $P_n(x)$ (it is useful to plot both on linear and log scales, as in Fig. 2). Show the behavor of $\sqrt{N_w}\Delta$ versus the bit number $b = 0, \ldots, 63$ for the different N_w you used. Explain qualitatively why you think the results look as they do.

Assuming that there are no correlations between the bits b (which may not be true and could also be tested), how do you judge this generator's ability to produce random floating-point numbers when the integers between -2^{63} and $2^{63} - 1$ are linearly converted to the range $[0, 1)$?