4 Feb.

Logistics: Computer project available

Paper copies @ my office 123 (TP)

Recap: Use probability to handle many (10^{23}) objects

"Outcomes" ~ "states"

Sets of outcomes → "events" w/ probabilities

Each outcome in B contains R different $X^{(r)} \in A$, one for each repetition $r = 1, \cdots, R$, and each with mean $\langle X^{(r)} \rangle = \mu$ and variance $\langle (X^{(r)} - \mu)^2 \rangle = \sigma^2$. Considering the case $R = 4$ for simplicity, any element of B can be written as $X_i^{(1)} X_j^{(2)} X_k^{(3)} X_l^{(4)} \in B$ with corresponding probability

$$P_B\left(X_i^{(1)} X_j^{(2)} X_k^{(3)} X_l^{(4)}\right) = P_A\left(X_i^{(1)}\right) P_A\left(X_j^{(2)}\right) P_A\left(X_k^{(3)}\right) P_A\left(X_l^{(4)}\right),$$

using subscripts to distinguish between the single-experiment (A) and repeated-experiment (B) probability spaces.

Averaging over all R repetitions defines the *arithmetic mean*

$$\overline{X}_R = \frac{1}{R} \sum_{r=1}^{R} X^{(r)}. \tag{5}$$

Unlike the true mean μ, the arithmetic mean $\overline{X}_R$ is a random variable—a number that may be different for each element of B. That said, $\overline{X}_R$ and μ are certainly related, and so long as the standard deviation exists—that is, so long as σ^2 is finite—this relation can be proved rigorously in the limit $R \to \infty$.[2]

1 Feb.

4 Feb.

Here we will not be fully rigorous, and take it as given that

$$\left\langle \left(X^{(i)} - \mu\right)\left(X^{(j)} - \mu\right) \right\rangle = \sigma^2 \delta_{ij} = \begin{cases} \sigma^2 & \text{for } i = j \\ 0 & \text{for } i \neq j \end{cases},$$

where the *Kronecker delta* $\delta_{ij} = 1$ for $i = j$ and vanishes for $i \neq j$. This is a consequence of the assumed independence of the different repetitions. Using this result and the relation $\left(\sum_i a_i\right)\left(\sum_j b_j\right) = \sum_{i,j} (a_i b_j)$, express the following quantity in terms of σ and R:

$\sum_{r=1}^{R} \mu = R\mu$

$$\left\langle \left(\frac{1}{R} \sum_{r=1}^{R} X^{(r)} - \mu \right)^2 \right\rangle = \frac{1}{R^2} \left\langle \left(\sum_r \left(X^{(r)} - \mu\right) \right)^2 \right\rangle$$

$$= \frac{1}{R^2} \left\langle \left(\sum_r \left(X^{(r)} - \mu\right) \right) \left(\sum_s \left(X^{(s)} - \mu\right) \right) \right\rangle$$

$$= \frac{1}{R^2} \left\langle \sum_{r,s} \left(X^{(r)} - \mu\right)\left(X^{(s)} - \mu\right) \right\rangle = \frac{\sigma^2}{R^2} \sum_{r,s} \delta_{rs}$$

$$= \frac{\sigma^2}{R^2} \sum_r 1 = \frac{\sigma^2}{R} \xrightarrow[R \to \infty]{} 0$$

You should find that your result vanishes in the limit $R \to \infty$, so long as σ^2 is finite. Since the square makes this expectation value a sum of non-negative terms, it can vanish only if every one of those terms is individually zero.

[2] In the computer project we will numerically investigate a situation where σ^2 diverges.

This establishes the **law of large numbers**:

$$\lim_{R\to\infty} \frac{1}{R} \sum_{r=1}^{R} X^{(r)} = \mu, \tag{6}$$

where we have assumed $\langle X^{(r)} \rangle = \mu$ and $\langle (X^{(r)} - \mu)^2 \rangle = \sigma^2$ are finite.

1.3 Probability distributions

It is not necessary to make the assumption (Eq. 1) that our outcome space contains only a countable number of possible outcomes. The considerations above continue to hold even if the random variable X is a continuous real number. In this case, however, the identification of probabilities with outcomes is slightly more complicated, which will be relevant when we consider the central limit theorem in the next section.

When the outcome can be any number on the real line, the fundamental object is a **probability distribution** (or **density function**) $p(x)$ defined for all $x \in \mathbb{R}$. Starting from this density, a probability is determined by integrating over a given interval. Calling this interval $[a, b]$, the integration produces the probability that the outcome X lies within the interval,

$$P\,(a \le X \le b) = \int_a^b p(x)\, dx.$$

We similarly generalize the definition of an expectation value (Eq. 4) to an integral over the entire domain of the probability distribution,

$$\langle f(x) \rangle = \int f(x)\, p(x)\, dx.$$

We will omit the limits on integrals over the entire domain, so for $x \in \mathbb{R}$ we implicitly have $\int dx = \int_{-\infty}^{\infty} dx$, with $\int p(x)\, dx = 1$. An important set of expectation values is

$$\langle x^\ell \rangle = \int x^\ell\, p(x)\, dx, \tag{7}$$

which provides the mean and variance of the probability distribution $p(x)$, through generalizations of Eqs. 2–3:

$$\mu = \langle x \rangle = \int x\, p(x)\, dx \qquad\qquad \sigma^2 = \langle x^2 \rangle - \langle x \rangle^2 . \tag{8}$$

The expression for the variance should be familiar from your determination of the standard deviation in an earlier gap. Unless stated otherwise, we will assume the mean and variance are both finite for the probability distributions we consider.

1.4 Central limit theorem

The central limit theorem is a central result of probability theory (hence its name). Over the years it has been expressed in several equivalent ways, and there are also many distinct variants of the theorem accommodating different conditions and assumptions. In this module we are interested in applying rather than proving the central limit theorem; you can find proofs in many textbooks.

The version of the theorem we use in this module assumes we have N independent random variables $x_1, \cdots, x_N$, each of which has the same (finite) mean μ and variance σ^2. Such random variables are said to be *identically distributed*, and a common way to obtain them is to repeat an experiment N times, as we considered in Section 1.2. Just as in Eq. 5, the sum

$$s = \sum_{i=1}^{N} x_i \tag{9}$$

is itself a random variable.

The **central limit theorem** states that for large $N \gg 1$ the probability distribution for s is

$$p(s) \approx \frac{1}{\sqrt{2\pi N\sigma^2}} \exp\left[-\frac{(s-N\mu)^2}{2N\sigma^2}\right], \tag{10}$$

with the approximation becoming exact in the $N \to \infty$ limit.

$\int p(x)\,dx = 1$

In addition to asserting that the collective behaviour of many independent and identically distributed random variables x_i is governed by a **normal** (or **gaussian**) **distribution**, the central limit theorem further specifies the precise form of this distribution in terms of the mean and variance of *each individual* x_i.

In practice, N often doesn't need to be very large in order for the central limit theorem to provide a reasonable approximation. To illustrate this, let's again consider the roulette wheel introduced in Section 1.1. A simple game of roulette would let us place bets on whether or not the ball will end up in a red- or black-coloured pocket: If we bet correctly we get back twice the money we put in; otherwise we lose our money. We'll define our (potentially negative) *gain* to be the amount we receive minus the amount we spend on bets.

Suppose we place £5 bets on 'black' for each of N spins of the roulette wheel. What are the probabilities and gains of winning and of losing for any single one of those spins? Letting $W = 0, \cdots, N$ be the number of spins where we win, what is our total gain G_W as a function of (N, W)?

$P_{win} = \frac{18}{37}$ gain: $10 - 5 = 5$

$P_{lose} = \frac{19}{37}$ gain: $0 - 5 = -5$

N-total gain: G_W = Winnings − bets = $10W - 5N$

Recall that the number of different ways we could win W times out of N attempts is given by the binomial coefficient

$$\binom{N}{W} = \frac{N!}{W!\,(N-W)!},$$

with $0! = 1$. Setting $N = 5$, what are the six probabilities p_0 through p_5 that we win $W = 0, \cdots, 5$ times? What is the general expression for p_W for any (N, W)?

$P_W = \binom{N}{W}\left(\frac{18}{37}\right)^W \left(\frac{19}{37}\right)^{N-W}$

$P_0 = \left(\frac{19}{37}\right)^5 = 0.0357$

$P_1 = 5\left(\frac{18}{37}\right)\left(\frac{19}{37}\right)^4 = 0.169$

$P_2 = 10\left(\frac{18}{37}\right)^2\left(\frac{19}{37}\right)^3 = 0.320$

$P_3 = 10\left(\frac{18}{37}\right)^3\left(\frac{19}{37}\right)^2 = 0.304$

$P_4 = 5\left(\frac{18}{37}\right)^4\left(\frac{19}{37}\right) = 0.144$

$P_5 = 1\left(\frac{18}{37}\right)^5 = 0.0273$

Now let's apply the central limit theorem to this setup. What are the mean gain and its variance for a single spin of the wheel? What is the resulting probability distribution $p(G)$ given by the central limit theorem for the gain after N spins?

$$p(G) = \frac{1}{\sqrt{2\pi N\sigma^2}} \exp\left[-\frac{(G - N\mu)^2}{2N\sigma^2}\right] \approx \frac{1}{\sqrt{49.96\,\pi N}} \exp\left[-\frac{(G+0.135N)^2}{49.96\,N}\right]$$

$$\mu = \langle x \rangle = \sum_{x \in A} x\, p(x) = 5\left(\frac{18}{37}\right) - 5\left(\frac{19}{37}\right) = \frac{-5}{37} \simeq -0.135$$

$$\langle x^2 \rangle = 25\left(\frac{18}{37}\right) + 25\left(\frac{19}{37}\right) = 25 \rightarrow \sigma^2 = 25 - \left(\frac{5}{37}\right)^2 \approx 24.982$$

Since the central limit theorem gives us an approximate *distribution*, in order to compare its prediction against the exact p_i computed above, we need to extract

probabilities by integrating over appropriate intervals as discussed in Section 1.3. Natural intervals to consider are

$$P_{\text{integ}}(G_W) \equiv \int_{G_W-\Delta G/2}^{G_W+\Delta G/2} p(g)\, dg,$$

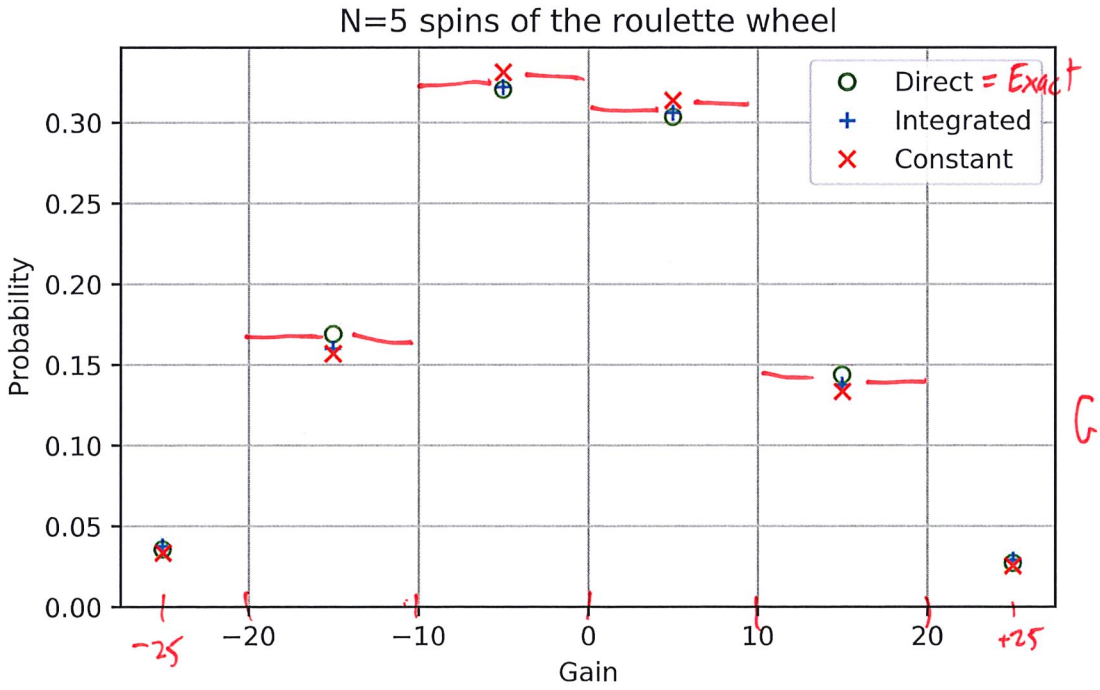
2.10 where $\Delta G = G_{W+1} - G_W$ is a constant you can read off from your work above. These numerical integrations are not convenient to do by hand, but can easily be performed by Maple, Python, MATLAB, Mathematica, etc. Alternately, we can simplify further by approximating $p(G)$ as a constant within each interval, which would give us

$$P_{\text{const}}(G_W) \equiv p(G_W)\, \Delta G.$$

Keeping $N = 5$, what are the six P_{integ} and P_{const}?

Tutorial ...

You can check your results against those plotted below, which come from this Python code that is available for you to peruse, run for different N or modify. As promised, the approximate P_{integ} and P_{const} are not far off the exact "Direct" probabilities, even though $N = 5$ doesn't strictly satisfy the condition $N \gg 1$.



G_W = 10W - 5N

MATH327: Statistical Physics, Spring 2022

Computer Project — Part 1

Instructions

In this first part of the computer project you will numerically analyze ordinary diffusive behaviour in a one-dimensional random walk. This will allow you to verify your numerical results by comparing them with exact analytic predictions based on the law of large numbers and central limit theorem. The verified numerical methods can then be generalized to consider anomalous diffusion in the second part of the project, where exact analytic predictions will not be available.

There are three exercises below, the first two of which include some background information on pseudo-random numbers and inverse transform sampling. While the exercises mention some syntax specific to Python, you may use a different programming option if you prefer. This demo illustrates all the Python programming tools needed for the project. Even running slowly in the cloud via replit.com, the computing for each exercise should complete in a minute or less.

This part of the project is **due by 23:59 on Thursday, 17 February**. Submit it by file upload on Canvas.[1] **Both** your answers to the questions below and the code that produces your results must be submitted. These can be uploaded as separate files or in a combined file, as you prefer. With the exception of Mathematica `.nb` files, it will be quicker for me to check code submitted in its native format (for example, a `.py` file for Python code or a `.m` file for MATLAB code). Anonymous marking is turned on, and I will aim to return feedback promptly in case this may be helpful when working on the second part of the project.

Exercise 1: Pseudo-random numbers

Background

We have discussed how statistical physics is based on considering systems that involve some element of randomness. Because computer programs are deterministic, it is not possible to use them to generate a truly random sequence of numbers.[2] Instead, computer algorithms generate *pseudo*-random numbers, which are entirely sufficient for our purposes.

A sequence of pseudo-random numbers is defined to be a sequence that *looks* random, in the sense that knowing the first $N-1$ elements in the sequence

[1] By submitting solutions to this assessment you affirm that you have read and understood the Academic Integrity Policy detailed in Appendix L of the Code of Practice on Assessment and have successfully passed the Academic Integrity Tutorial and Quiz. The marks achieved on this assessment remain provisional until they are ratified by the Board of Examiners in June 2022.

[2] New quantum technologies are being developed as a way to produce truly random numbers. This is part of the motivation for large investments in quantum technologies around the world.

does not suffice to predict the Nth element with a high probability of correctness. Equivalently, it takes a very long time for the sequence to start repeating itself—such repetition *will* eventually happen, because computers encode numbers in a finite set of bits, which can represent only a finite set of numbers. For example, 32 bits can represent all integers from 0 through $2^{32} - 1 \sim 10^9$ while 64 bits increase the upper bound to $2^{64} - 1 \sim 10^{19}$. Python uses the Mersenne Twister algorithm as its default pseudo-random number generator (PRNG). This algorithm can provide $2^{19937} - 1 \sim 10^{6000}$ numbers before its sequence repeats.

We can view the absence of true randomness as an advantage rather than a limitation. Deterministic pseudo-random numbers allow our computer programs to be reproducible up to the (very high) precision of the computer. Each exercise below starts by initializing the PRNG with a "seed". Given the same seed, the PRNG will subsequently generate the same sequence of pseudo-random numbers. In Python, as shown in the demo, this initialization is done by calling the function `random.seed(s)`, where s is the seed we specify.

Task

The Python function `random.random()` generates a pseudo-random number u with the uniform probability distribution

$$p(u) = \begin{cases} 1 & \text{for } 0 \le u < 1 \\ 0 & \text{otherwise} \end{cases} .$$

Clearly $\int p(x)\,dx = \int_0^1 dx = 1$, as required. What are the exact mean μ and standard deviation σ of this probability distribution?

[2 marks]

Initialize the PRNG with seed $s = 327$. For each of the five $R = 10$, 100, 1000 10,000 and 100,000, generate a sequence of R pseudo-random numbers u_r distributed according to $p(u)$. (Don't re-initialize the PRNG when changing R, or else these sequences will partially duplicate each other.) Use each sequence to estimate the mean and standard deviation via the law of large numbers,

$$\overline{u}_R = \frac{1}{R}\sum_{r=1}^{R} u_r \qquad\qquad \overline{\sigma}_R \equiv \sqrt{\left(\frac{1}{R}\sum_{r=1}^{R} u_r^2\right) - \overline{u}_R^2}. \tag{1}$$

How do your numerical results compare to your exact analytic predictions above? Rounding to four decimal places should suffice for these comparisons.

[5 marks]

In class we saw $\left\langle (\overline{u}_R - \mu)^2 \right\rangle \propto 1/R$ (page 14 of the lecture notes). Let's test this numerically by repeating the above computation of $\overline{u}_R$ another 99 times, ignoring $\overline{\sigma}_R$ for simplicity. Together with the result you reported above, this gives a total of 100 estimates of the random variable $(\overline{u}_R - \mu)^2$, which we can use to approximate the expectation value as

$$\overline{(\overline{u}_R - \mu)^2} \equiv \frac{1}{100} \sum_{i=1}^{100} (\overline{u}_R - \mu)_i^2 \,. \tag{2}$$

Rather than reporting your results as numerical values, plot $R \times \overline{(\overline{u}_R - \mu)^2}$ vs. R and see whether the five points appear approximately constant. If so, is the size of this constant roughly what you would expect?

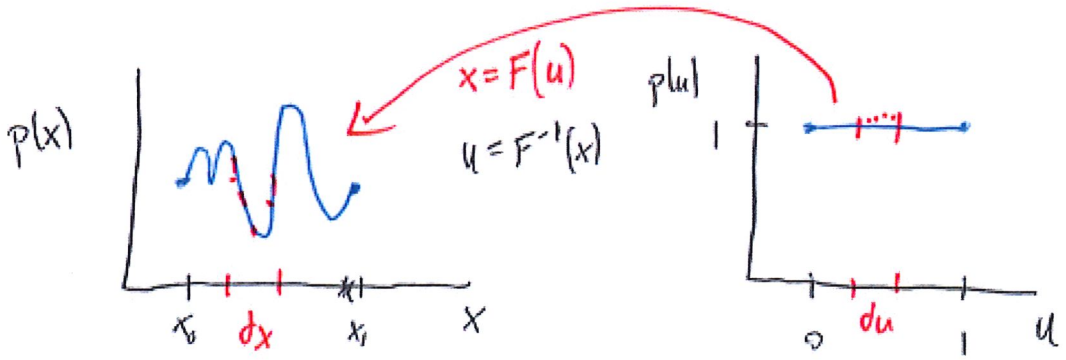
Hints: Include 0 on the y-axis of your plot to maintain a sense of scale. The Matplotlib Python plotting library provides (via its `pyplot` module) the option `xscale('log')` that sets a logarithmic scale for the x-axis, to produce even spacing between these five R.

[8 marks]

Exercise 2: Inverse transform sampling

Background

The uniform distribution is a bit boring. Inverse transform sampling is a technique that allows us to consider more interesting probability distributions, while still generating pseudo-random numbers using the `random.random()` function. The idea is illustrated by the sketch below.



In words, we take our uniformly distributed u_r and act on them with some invertible transformation F to define $x_r = F(u_r)$ that follow the distribution of interest, $p(x)$. We require $p(u)du = p(x)dx$, which allows us to relate $p(x)$ and the transformation $F(u)$:

$$p(x) = p(u)\frac{du}{dx} = p(u)\frac{d}{dx}F^{-1}(x),$$