NMSM Homework Exercises

AUTHOR Guglielmo Bordin

LAST EDITED 15th November 2023

{·1} Sampling random points within *d*-dimensional domains by hit and miss

I skipped the integration on the rectangle, solving only the disk case. The source code is in `A01b_disk_hit_miss.c`; I implemented the main part of the algorithm like this:

```
for (n = 0; n < n_plot; ++n) {
    hits = 0; // reset hit counter
    throws = (1 + n) * dn;
    for (i = 0; i < throws; ++i) {
        x = RngStream_RandU01(rngs);
        y = RngStream_RandU01(rngs);
        if (x * x + y * y < 1)
            ++hits;
    }
    mc_pi = 4 * (double)hits / throws;

    // write to file: throws + \t + relative error + \n
    fprintf(file, "%d\t", throws);
    fprintf(file, "%g\n", fabs(1 - mc_pi * M_1_PI));
}
```

The error as a function of the number of throws is shown in Fig. 1. It is comfortably under 1 % with around 25 000–30 000 iterations.

{·2} Sampling random numbers from a given distribution

The idea is to sample from the probability distribution $\rho_n(x) = cx^n$ in $[0, 1]$. First, using the normalization condition we can find out what c should be:

$$1 = \int_0^1 cx^n \, dx = \frac{c}{n+1} \implies c = n + 1. \tag{2.1}$$

Then, we find the expression of the associated cumulative density function:

$$F_n(x) = (n+1) \int_0^x y^n \, dy = x^{n+1}, \tag{2.2}$$

and invert it:

$$u = x^{n+1} \implies x = u^{1/(n+1)}. \tag{2.3}$$

So, inside the code `A02a_inversion_method.c` I sample a random `double` from a uniform distribution between 0 and 1 using `drand48()`, and I raise it to the power of $1/(n+1)$ to get x:

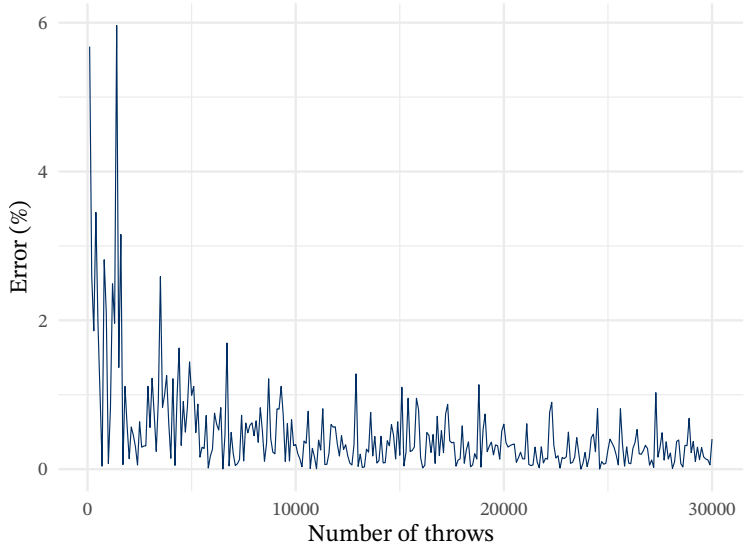


Figure 1: error in the Monte Carlo estimation of the area of a unit disk, as a function of the number of 'throws'.

```
double* x = malloc(n_smp * sizeof(*x));
for (int i = 0; i < n_smp; ++i)
    x[i] = pow(drand48(), 1.0 / (n + 1));
```

A histogram of 100 000 points sampled from ρ with $n = 3$ is displayed in Fig. 2.

Inside `A02b_inversion_method.c` I modified the code to sample from $\rho_2(x) = cx^2$ in $[0, 2]$. c is different this time, of course:

$$1 = \int_0^2 cx^2\,dx = \frac{8}{3}c \implies c = \frac{3}{8}. \tag{2.4}$$

The cumulative is then

$$F_2(x) = \frac{3}{8}\int_0^x y^2\,dy = \frac{x^3}{8} \implies x = 2u^{1/3}. \tag{2.5}$$

Once again, you can see the comparison between a 100 000-points histogram and the theoretical curve in Fig. 3.

{·3} Sampling via transformation of coordinates

If we want to sample points uniformly distributed over the unit disk, we cannot simply generate $r \sim \mathcal{U}_{[0,1]}$ and $\vartheta \sim \mathcal{U}_{[0,2\pi]}$. The result of doing that is shown in Fig. (4a), and the corresponding source code is `A03aa_disk_naive.c`. While the angular distribution poses no problem, there is an undesired higher density of points close to the centre of the disk.

The reason for this becomes clear when you consider the number of points falling inside a ring of given thickness δr. Consider a ring with inner radius r_1 and outer radius $r_1 + \delta r$, with $\delta r \ll r_1$, and another with radii r_2, $r_2 + \delta r$. Despite the ratio of their areas being r_2/r_1, the fraction of points within each ring remains the same. Thus, the number

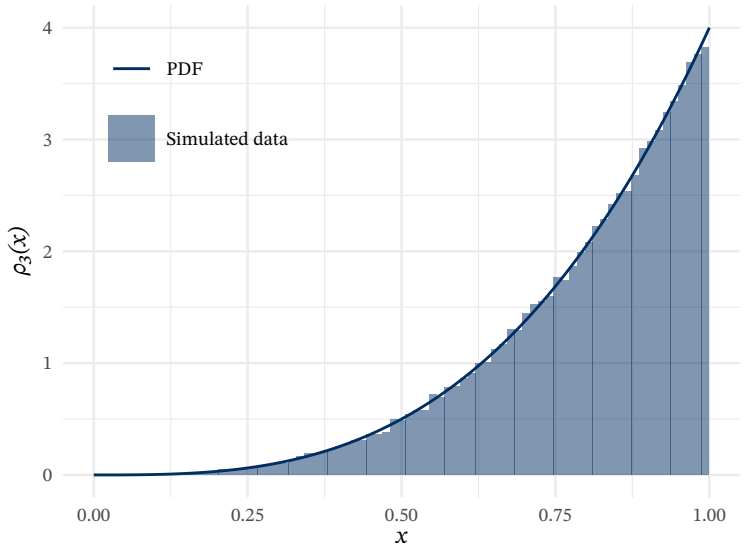


Figure 2: histogram of 100 000 points sampled from the probability distribution $4x^3$ in $[0, 1]$.

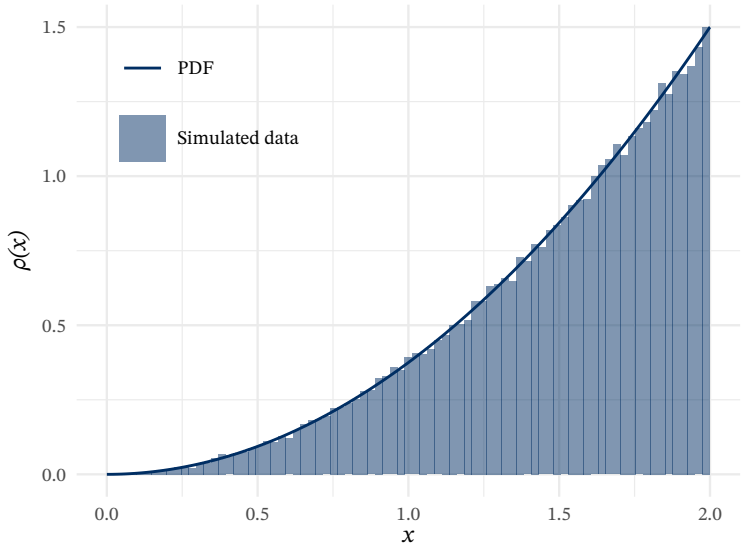


Figure 3: histogram of 100 000 points sampled from the probability distribution $3x^2/8$ in $[0, 2]$.

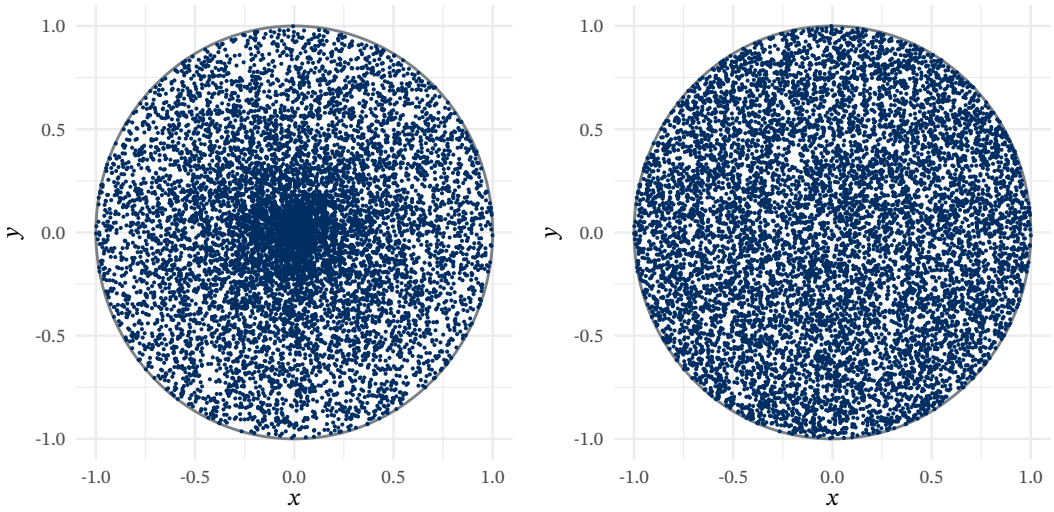


Figure 4: 10 000 points sampled on the unit disk with the wrong coordinate transformation (a) and with the correct one (b).

of points falling at a distance r from the centre should be proportional to r:

$$\rho_r(r) = 2r, \tag{3.1}$$

with the 2 in front to ensure normalization over $[0, 1]$. By computing the cumulative and inverting we get then

$$F_r(r) = 2\int_0^x x\,dx = x^2 \implies u = r^2 \implies r = \sqrt{u}. \tag{3.2}$$

So, I modified the code in `A03ab_disk_correct.c` to sample like this:

```
// allocate radius and theta arrays
double* r = malloc(n_smp * sizeof(*r));
double* t = malloc(n_smp * sizeof(*t));
for (i = 0; i < n_smp; ++i) {
    r[i] = sqrt(RngStream_RandU01(rngs));
    t[i] = 2 * M_PI * RngStream_RandU01(rngs);
}
```

The result is the correctly uniform sampling in Fig. (4b).

As regards the Box–Muller transform, I started from the factorization of $\rho_{x,y}(x, y)$:

$$\rho_{x,y}(x, y) = \frac{1}{2\pi}\exp\left(-\frac{x^2+y^2}{2}\right) = \rho_x(x)\rho_y(y), \quad \text{with } \rho_x(x) = \frac{e^{-x^2/2}}{\sqrt{2\pi}}. \tag{3.3}$$

If we switch to polar coordinates this becomes, remembering the factor r coming from the Jacobian,

$$\rho_{r,\vartheta}(r, \vartheta) = \frac{r}{2\pi}e^{-r^2/2}, \tag{3.4}$$

with $\rho_r(r) = re^{-r^2/2}$ and $\rho_\vartheta(\vartheta) = 1/2\pi$.

Then, we can marginalize over ϑ to get $\rho_r(r)$:

$$\rho_r(r) = \frac{1}{2\pi}\int_0^{2\pi} re^{-r^2/2}\,dr = re^{-r^2/2}. \tag{3.5}$$

As usual, we compute the cumulative and invert it to generate a random $r \sim \rho_r(r)$:

$$F_r(r) = \int_0^r xe^{-x^2/2}\,dx = 1 - e^{-r^2/2} \implies e^{-r^2/2} = 1 - u. \tag{3.6}$$

Since u is distributed uniformly, we can redefine it as $1 - u$ for simplicity:

$$-\frac{r^2}{2} = \log u \implies r = \sqrt{-2\log u}. \tag{3.7}$$

Then, to get the angle ϑ we recover the *conditional* distribution $\rho_{\vartheta\,|\,r}(\vartheta \mid r)$:

$$\rho_{\vartheta\,|\,r}(\vartheta \mid r) = \frac{\rho_{r,\vartheta}(r,\vartheta)}{\rho_r(r)} = \frac{1}{2\pi}. \tag{3.8}$$

Thus to get ϑ we have simply to sample from $\mathcal{U}_{[0,2\pi]}$.

The idea then is to sample two numbers u_1, u_2 from $\mathcal{U}_{[0,1]}$ at each iteration; at that point we can calculate

$$x = \sqrt{-2\log u_1}\cos(2\pi u_2), \quad y = \sqrt{-2\log u_1}\sin(2\pi u_2), \tag{3.9}$$

to get two numbers x, y distributed according to a standard Gaussian $\mathcal{N}(0,1)$. We can also get a $z \sim \mathcal{N}(\mu,\sigma)$ afterwards by multiplying x or y by σ and summing the mean: $z = \mu + \sigma x$.

The code is in `A03b_box_muller.c`; the relevant section is

```
double r, t;
double* x = malloc(n_smp * sizeof(*x));
double* y = malloc(n_smp * sizeof(*y));
for (int i = 0; i < n_smp; ++i) {
    r = sigma * sqrt(-2 * log(drand48()));
    t = 2 * M_PI * drand48();
    x[i] = mu + r * cos(t);
    y[i] = mu + r * sin(t);
}
```

You can see a histogram of 100 000 sampled points in Fig. 5.

{·4} Rejection method

In `A03ca_rejection_sampling.c` I implemented the sampling of random numbers from the probability distribution

$$f(x) = \frac{2}{\sqrt{pi}}e^{-x^2} \tag{4.1}$$

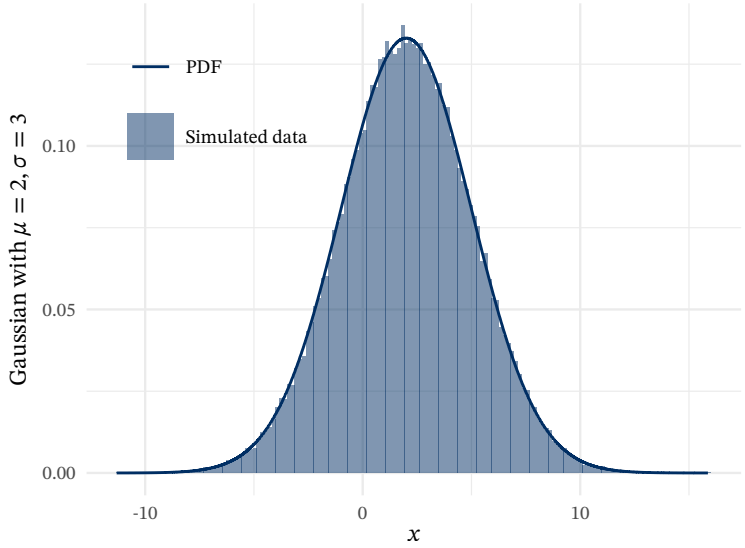


Figure 5: histogram of 100 000 points sampled with the Box–Muller algorithm from the normal distribution with $\mu = 2$ and $\sigma = 3$.

over $[0, \infty)$. I started from a function $g(x)$ that could serve as an upper limit to $f(x)$,

$$g(x) = \begin{cases} A & \text{for } 0 \leqslant x \leqslant p, \\ \dfrac{A}{p} x e^{p^2 - x^2} & \text{for } x > p. \end{cases} \tag{4.2}$$

First we normalize it to turn it into a proper probability distribution:

$$1 = \int_0^\infty g(x)\,dx = Ap + \frac{A}{p} e^{p^2} \int_p^\infty x e^{-x^2}\,dx = A\left(p + \frac{1}{2p}\right) \implies A = \frac{2p}{1 + 2p^2}. \tag{4.3}$$

Then we need a constant c such that $cg(x) \geqslant f(x)$ everywhere. The maximum of $f(x)$ is in $x = 0$, where $f(0) = 2/\sqrt{\pi}$, so we set $cA = 2/\sqrt{\pi}$. Another thing to consider is that $g(x)$ can have a maximum larger than A in $[p, \infty)$:

$$g'(x) = \begin{cases} 0 & \text{for } 0 \leqslant x \leqslant p, \\ \dfrac{A}{p}(1 - 2x^2) e^{p^2 - x^2} & \text{for } x > p. \end{cases} \tag{4.4}$$

Thus in the region $[p, \infty)$ the derivative is zero in $x = 1/\sqrt{2}$. To avoid this we have to choose $p > 1/\sqrt{2} \approx 0.707$.

Now, to generate numbers distributed according to $g(x)$ we compute, as usual, the cumulative and invert it:

$$G(x) = \begin{cases} Ax & \text{for } 0 \leqslant x \leqslant p, \\ 1 - \dfrac{A}{2p} e^{p^2 - x^2} & \text{for } x > p. \end{cases} \tag{4.5}$$

Since it is piecewise defined, we have to pay attention to the limits too:

$$\begin{cases} x = u/A & \text{for } u \leqslant Ap, \\ u = 1 - \dfrac{A}{2p} e^{p^2 - x^2} & \text{for } u > Ap, \end{cases} \tag{4.6}$$

which implies

$$x = \begin{cases} u/A & \text{for } u \leqslant Ap, \\ \sqrt{p^2 - \log\left[\frac{2p}{A}(1-u)\right]} & \text{for } u > Ap. \end{cases} \tag{4.7}$$

With the choice $cA = 2/\sqrt{\pi}$ the test $cg(x)\xi < f(x)$, with $\xi \sim \mathcal{U}_{[0,1]}$, becomes

$$\frac{2}{\sqrt{\pi}}\xi < \frac{2}{\sqrt{\pi}}e^{-x^2} \implies \xi < e^{-x^2}, \tag{4.8}$$

if we generated x with the uniform part of $g(x)$, or

$$\xi\frac{2}{\sqrt{\pi}}\frac{x}{p}e^{p^2-x^2} < \frac{2}{\sqrt{\pi}}e^{-x^2} \implies \xi x < pe^{-p^2} \tag{4.9}$$

otherwise. Let's take a look at the main loop in the code to make it clear:

```
int acc = 0;
while (acc < n_smp) {
    u = drand48();
    if (u < A * P) {
        // sample from unif g(x) = A
        x = u / A;
        if (drand48() < exp(-x * x))
            smp[acc++] = x;
    } else {
        // sample from exp g(x) = (A/p) x e^(p^2 - x^2)
        x = sqrt(p2 - log_2pA - log(1 - u));
        if (drand48() * x < P * exp(-p2))
            smp[acc++] = x;
    }
}
```

As you can see from the example histogram in Fig. 6, the sampling is correct.

In `A03cb_rejection_analysis.c` I modified slightly the code in order to analyse the acceptance rate of the algorithm as a function of p – subject to the condition I was mentioning earlier, $p > 1/\sqrt{2}$. You can see the results in Fig. 7: the best performance, unsurprisingly, is obtained with $p = 1/\sqrt{2}$, since for larger values of p $g(x)$ tends to move further away from $f(x)$.

{·5} Importance sampling

The goal was to integrate the function $f(x) = g(x)e^{-x^2}$ on $[0, \infty)$, with $g(x)$ being a slowly varying function. I chose $g(x) = x\cos x^2$, so the exact value of the integral is

$$\int_0^\infty f(x)\,dx = \int_0^\infty x\cos(x^2)e^{-x^2}\,dx = \cdots = \frac{1}{4}. \tag{5.1}$$

Following the suggestion, we multiply and divide by a weighting function defined as

$$w(x) = \frac{2}{\sqrt{\pi}}e^{-x^2}, \tag{5.2}$$

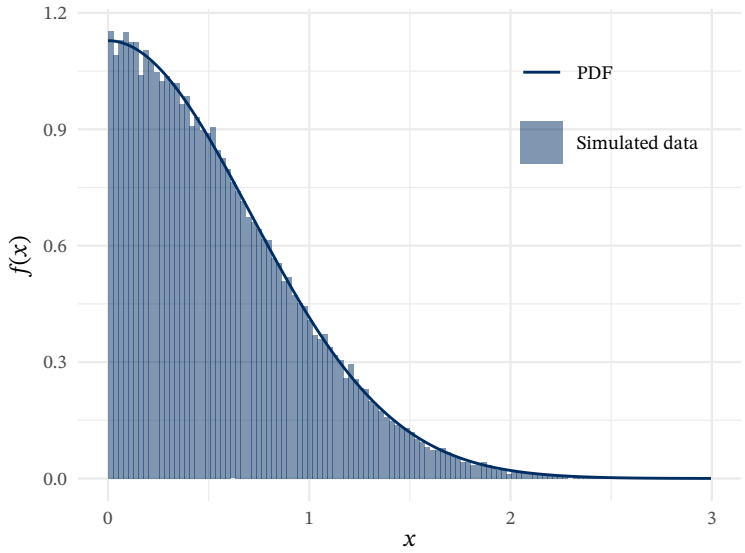


Figure 6: histogram of 100 000 points sampled from $f(x) = 2e^{-x^2}/\sqrt{\pi}$ with the rejection method implemented in `A03ca_rejection_sampling.c`.

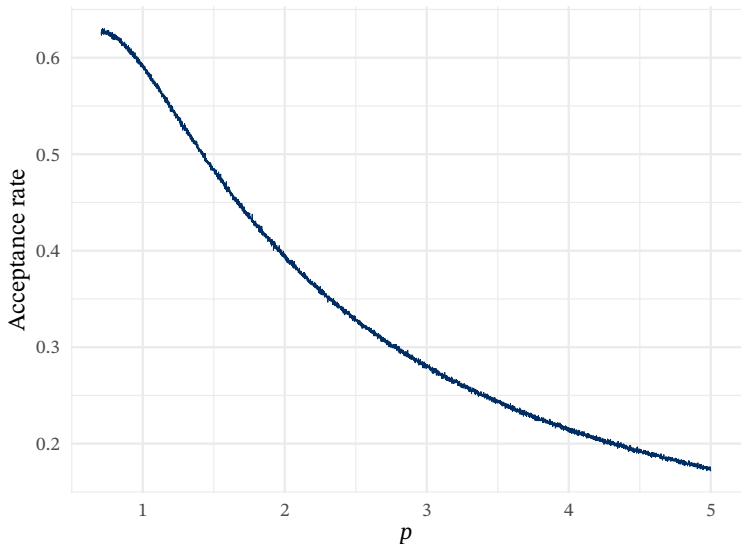


Figure 7: acceptance rate of 100 000 samples generated with the rejection method algorithm in `A03cb_rejection_analysis.c`., as a function of the breakpoint of $g(x)$.

and so we can write

$$\int_0^\infty g(x)e^{-x^2}\,dx = \frac{\sqrt{\pi}}{2}\int_0^\infty g(x)w(x)\,dx = \frac{\sqrt{\pi}}{2}\int_0^\infty g(x)\,dw(x). \tag{5.3}$$

Therefore, we can sample numbers x_i from $w(x)$ and estimate the integral with

$$\int_0^\infty g(x)e^{-x^2}\,dx \approx \frac{\sqrt{\pi}}{2N}\sum_{i=1}^{N} g(x_i). \tag{5.4}$$

To sample from $w(x)$, which is a half-normal distribution with variance 1/2, we can use a modified Box-Muller. The idea is to restrict the number generation to the $x > 0$ half-plane. First, since in this case the variance is not unitary we can recompute the cumulative. The marginal distribution over r needs a factor 2 in front for normalization, and we obtain

$$\rho_r(r) = 2re^{-r^2} \implies F_r(r) = 1 - e^{-r^2} \implies r = \sqrt{-\log u_1}, \quad \text{with } u_1 \sim \mathcal{U}_{[0,1]}. \tag{5.5}$$

For ϑ we sample uniformly over $[-\pi/2, +\pi/2]$ in order to stay in the $x > 0$ half-plane:

$$\vartheta = -\frac{\pi}{2} + \pi u_2, \quad \text{with } u_2 \sim \mathcal{U}_{[0,1]}. \tag{5.6}$$

Once we have r and ϑ we generate x, y distributed according to $w(x)$ in this way:

$$x = r\cos\vartheta, \quad y = r|\sin\vartheta|. \tag{5.7}$$

Here is the function that implements this algorithm; the full source code is in the file `A04a_gauss_crude_vs_import.c`.

```
double integ_impor(int N)
{
    double r, t;
    double g_sum = 0.0;
    int n = 0;
    while (n < N) {
        // sample x, y with modified Box-Muller
        r = sqrt(-log(drand48()));
        t = -M_PI_2 + M_PI * drand48();
        g_sum += g(r * cos(t)) + g(r * fabs(sin(t)));
        n += 2;
    }

    return g_sum / (M_2_SQRTPI * N);
}
```

In contrast to the importance sampling technique, inside the same C file I also implemented the 'crude' Monte Carlo integration, that is a simple average of a set of $f(x_i)$ with x_i sampled uniformly over an interval from 0 to a given $x_{\max}$:

$$\int_0^\infty g(x)e^{-x^2}\,dx \approx= \frac{1}{N}\sum_{i=1}^{N} g(x_i)e^{-x_i^2}, \quad \text{with } x_i \sim \mathcal{U}_{[0,x_{\max}]}. \tag{5.8}$$

The implementation is in this simple function:

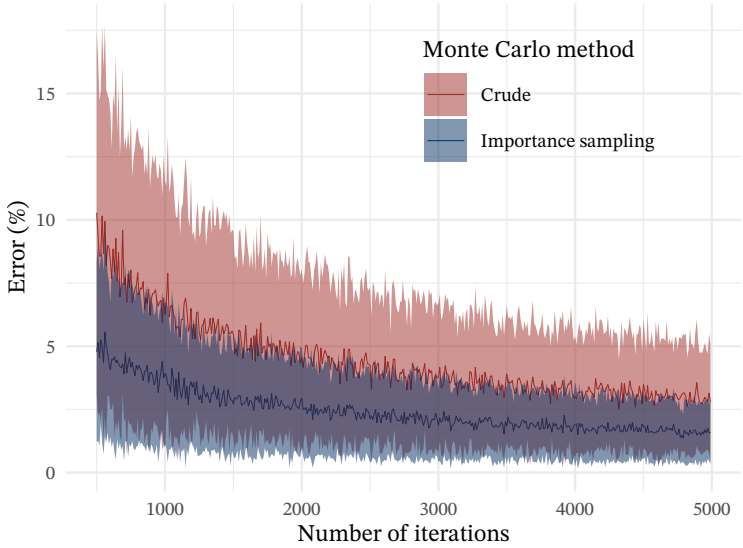


Figure 8: error of the Monte Carlo integration of $x\cos(x^2)e^{-x^2}$ on $[0,\infty)$ as a function of the number of iterations, both with the 'crude' method and the importance sampling. The error bars are obtained from the standard deviation of 100 different runs.

```
double integ_crude(int N)
{
    double f_sum = 0.0;
    for (int i = 0; i < N; ++i)
        f_sum += f(MAX_X * drand48());

    return MAX_X * f_sum / N;
}
```

I have compared the error of the two methods as a function of the number of iterations in Fig. 8. The importance sampling is definitely more efficient, although the crude method has a higher variance which can lead to a lower error in lucky runs. Also, as N grows the difference between the two methods shrinks, but this is rather unsurprising.

The second exercise asked to compute the integral

$$\int_0^{\pi/2} \cos x \, dx \tag{5.9}$$

using once again the importance sampling technique, with a weighting function $g(x) = a + bx^2$, a and b to tune. To have the best performance, the idea is to have a $g(x)$ that is large where the integrand is large and small when the integrand is small. First, $g(x)$ should be normalized, which brings down the number of parameters to one:

$$1 = \int_0^{\pi/2} (a + bx^2)\, dx = \frac{\pi}{2}a + \frac{\pi^3}{24}b \implies a = \frac{2}{\pi} - \frac{\pi^2}{12}b. \tag{5.10}$$

Next, since we want $g(x)$ to be as similar as possible to $\cos x$, we can restrict ourselves to $b < 0$. If b is too negative, though, $g(x)$ becomes negative in part of the domain. The best

solution is to impose $g(\pi/2) = 0$:

$$0 = \frac{2}{\pi} - \frac{\pi^2}{12}b + \frac{\pi^2}{4}b \implies b = -\frac{12}{\pi^3} \implies g(x) = \frac{3}{\pi}\left(1 - \frac{4}{\pi^2}x^2\right). \tag{5.11}$$

To sample from g, we compute the cumulative as usual:

$$G(x) = \frac{3}{\pi}\int_0^x \left(1 - \frac{4}{\pi^2}y^2\right) dy = \frac{3}{\pi}x - \frac{4}{\pi^3}x^3. \tag{5.12}$$

The inversion requires solving a cubic equation,

$$u = \frac{3}{\pi}x - \frac{4}{\pi^3}x^3 \implies x^3 - \frac{3}{4}\pi^2 x + \frac{\pi^3}{4}u = 0. \tag{5.13}$$

Let's define $\alpha = \pi^2/4$:

$$x^3 - 3\alpha x + \pi\alpha u = 0. \tag{5.14}$$

This is a depressed cubic, so thankfully it has a trigonometric solution given by[1]

$$x_k = 2\sqrt{\alpha}\cos\left[\frac{1}{3}\arccos\left(-\frac{3\pi\cancel{\alpha}u}{6\cancel{\alpha}}\sqrt{\frac{\cancel{3}}{\cancel{3}\alpha}}\right) - \frac{2\pi}{3}k\right], \quad \text{with } k = 0, 1, 2. \tag{5.15}$$

With some algebra, and selecting the right k, we can arrive at the much simpler form

$$x = \pi \sin\left(\frac{1}{3}\arcsin u\right), \quad \text{with } u \sim \mathcal{U}_{[0,1]}. \tag{5.16}$$

To wrap up, what I ended up doing is

$$\int_0^{\pi/2} \cos x\, dx \approx \frac{1}{N}\sum_{i=1}^{N} \frac{\cos x_i}{g(x_i)} = \frac{\pi}{3N}\sum_{i=1}^{N} \frac{\cos x_i}{1 - 4x_i^2/\pi^2}, \tag{5.17}$$

with each x_i generated according to Eq. (5.16). The code is in `A04b_cosx_importance.c`, with the main function being

```
double integral_cos(int N)
{
    double x;
    double f_sum = 0.0;
    for (int i = 0; i < N; ++i) {
        // sample x ~ g(x) = (3/pi)(1 - (4/pi^2)x^2)
        x = M_PI * sin(asin(drand48()) / 3);
        f_sum += cos(x) / (1 - M_2_PI * M_2_PI * x * x);
    }

    return M_PI * f_sum / (3 * N);
}
```

The integration error as a function of the number of iterations is reported in Fig. 9. Remarkably, as low as 100 iterations are sufficient to get on average an error that is comfortably under 1 %.

1. https://en.wikipedia.org/wiki/Cubic_equation#Trigonometric_and_hyperbolic_solutions.

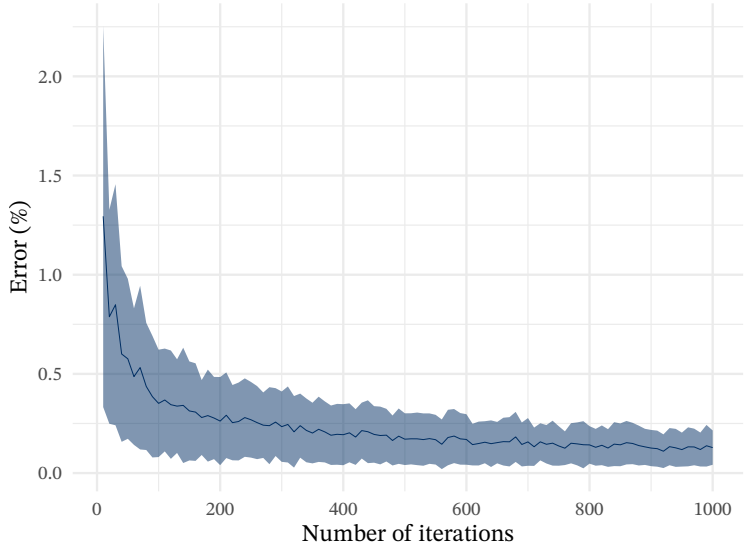


Figure 9: error of the Monte Carlo integration with importance sampling of $\cos x$ on $[0, \pi/2]$ as a function of the number of iterations. The error bars are obtained from the standard deviation of 100 different runs.

The third exercise asked to investigate the improvement one can get with importance sampling applied to a simple integral, the average on the distribution $\rho(x) = e^{-x}$, $x \geqslant 0$, of the function

$$f(x) = \begin{cases} 0 & \text{for } x < T, \\ 1 & \text{for } x \geqslant T. \end{cases} \tag{5.18}$$

This average can be readily calculated as

$$\langle f \rangle_\rho = \int_0^\infty f(x) e^{-x}\, dx = \int_T^\infty e^{-x}\, dx = e^{-T}. \tag{5.19}$$

The problem of this integral is that $f(x)$ is different from 0 in the region where $\rho(x)$ is quickly vanishing. This means that if we were to sample numbers from $\rho(x)$ we would waste a lot of iterations.

Consider instead the function $g(x; a) = ae^{-ax}$ defined for $x \geqslant 0$ and $a \in (0, 1]$. We can use it for an importance sampling and optimize a to minimize the variance. In particular, defining $F(x) \equiv f(x)\rho(x)/g(x)$ we can write

$$\langle f \rangle_\rho = \int_0^\infty f(x)\rho(x)\, dx = \int_0^\infty \frac{f(x)\rho(x)}{g(x)} g(x)\, dx = \langle F \rangle_g. \tag{5.20}$$

Let's calculate the second moment now:

$$\begin{aligned} \langle F^2 \rangle_g = \int_0^\infty \frac{f^2(x)\rho^2(x)}{g^2(x)} g(x)\, dx &= \int_T^\infty \frac{\rho^2(x)}{g(x)}\, dx \\ &= \frac{1}{a} \int_T^\infty e^{-(2-a)x}\, dx = \frac{e^{-(2-a)T}}{a(2-a)}. \end{aligned} \tag{5.21}$$

By subtracting the first moment squared we get the variance,

$$\sigma_g^2(F) = \langle F^2 \rangle_g - \langle F \rangle_g^2 = \frac{e^{-(2-a)T}}{a(2-a)} - e^{-2T}. \tag{5.22}$$

Table 1: variance ratios for some values of T, all evaluated at a^*.

T	$\sigma_\rho(f)/\langle f\rangle_\rho$	$\sigma_g(F)/\langle F\rangle_g$	$\sigma_\rho(f)/\sigma_g(F)$
3	4.37	1.95	2.24
5	12.14	2.55	4.76
10	148.41	3.65	40.71
20	22026.47	5.18	4249.17

The variance of f with respect to ρ is given instead by

$$\begin{aligned}\sigma_\rho^2(f) = \int_0^\infty (f(x) - \langle f\rangle_\rho)^2 \rho(x)\,dx &= \int_T^\infty \rho(x)\,dx + \langle f\rangle_\rho^2 - 2\langle f\rangle_\rho \int_T^\infty \rho(x)\,dx \\ &= \langle f\rangle_\rho + \langle f\rangle_\rho^2 - \langle f\rangle_\rho^2 \\ &= e^{-T}(1 - e^{-T}).\end{aligned} \tag{5.23}$$

Now, let's minimize $\sigma_g^2(F)$ with respect to a:

$$\frac{\partial}{\partial a}\sigma_g^2(F) = e^{-(2-a)T}\left[\frac{Ta(2-a) - 2(1-a)}{a^2(2-a)^2}\right], \tag{5.24}$$

which gets us to

$$Ta^2 - 2(1+T)a + 2 = 0 \iff a = a_\pm^* = \frac{1 + T \pm \sqrt{1+T^2}}{T}. \tag{5.25}$$

Only a_-^* is compatible with the condition $a \in (0, 1]$, so the optimal a is given by

$$a^* = \frac{1 + T - \sqrt{1+T^2}}{T}. \tag{5.26}$$

In Tab. 1 I calculated the ratios between each of the two variances and $\langle f\rangle_\rho$ and the ratio between the two variances, for some values of T. While $\sigma_g(F)/\langle f\rangle_\rho$ grows slowly, $\sigma_\rho(f)/\langle f\rangle_\rho$ has an exponential increase; indeed

$$\frac{\sigma_\rho(f)}{\langle f\rangle_\rho} = \frac{\sqrt{e^{-T}(1-e^{-T})}}{e^{-T}} = \sqrt{e^{-T} - 1}. \tag{5.27}$$

In Fig. 10 you can see the plot of the variance ratio as a function of T.

{·6} Markov chains

If we define the row vector of state probabilities at time n as $\boldsymbol{\mu}_n$ and the stochastic matrix as $\mathcal{P}$, we can write

$$\boldsymbol{\mu}_n = \boldsymbol{\mu}_{n-1}\mathcal{P}. \tag{6.1}$$

The (i, j)-th entry of $\mathcal{P}$, p_{ij}, is the probability to go from state i to state j. If we work out the vector-matrix product in the previous formula, we get, for state i,

$$\mu_n(i) = \sum_{j\in S} \mu_{n-1}(j)p_{ji}, \tag{6.2}$$

where S denotes the set of possible states. Then, we can split the sum and obtain

$$\mu_n(i) = \mu_{n-1}(i)p_{ii} + \sum_{j\neq i} \mu_{n-1}(j)p_{ji}. \tag{6.3}$$

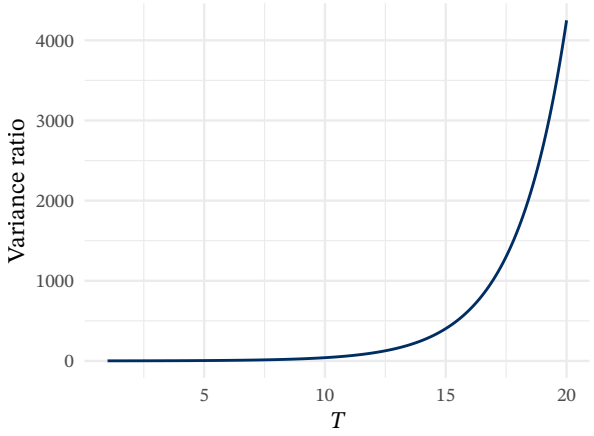


Figure 10: ratio between the two variances, $\sigma_\rho(f)/\sigma_g(F)$, computed with a^* as a function of T.

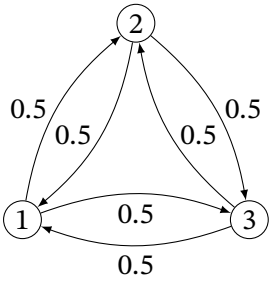
The first term is representative of the situation in which the system was in the same state i at time $n-1$, while the second represents all the possible transitions from another state to i. We can also rewrite the first term using the fact that the rows of the stochastic matrix are normalized:

$$\sum_{j\in S} p_{ij} = 1 \implies p_{ii} = 1 - \sum_{j\neq i} p_{ij}. \tag{6.4}$$

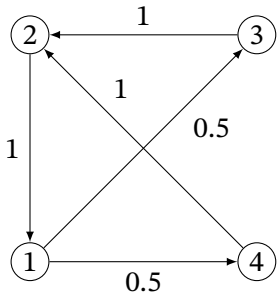
In this way we obtain the formula we needed to prove,

$$\mu_n(i) = \left(1 - \sum_{j\neq i} p_{ij}\right)\mu_{n-1}(i) + \sum_{j\neq i} \mu_{n-1}(j)p_{ji}. \tag{6.5}$$

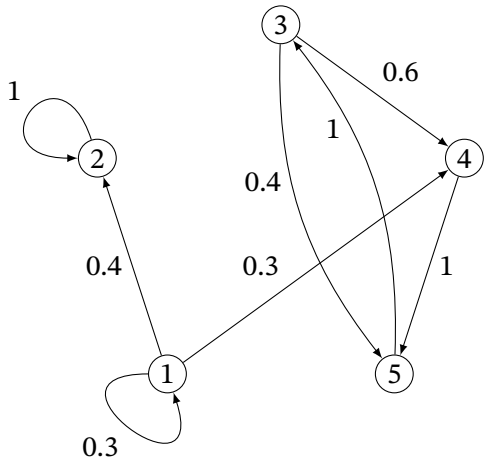
Now, for the second exercise, I will display the directed graphs that correspond to the given stochastic matrices.



$$\mathcal{P} = \begin{pmatrix} 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \\ 0.5 & 0.5 & 0 \end{pmatrix} \tag{6.6}$$



$$\mathcal{P} = \begin{pmatrix} 0 & 0 & 0.5 & 0.5 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{6.7}$$



$$\mathcal{P} = \begin{pmatrix} 0.3 & 0.4 & 0 & 0 & 0.3 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.6 & 0.4 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{6.8}$$

For the third exercise, let's consider the Markov chain defined by the matrix

$$\mathcal{P}_1 = \begin{pmatrix} 1/2 & 1/2 \\ 1 & 0 \end{pmatrix}. \tag{6.9}$$

Is it irreducible? For every pair of states but one, $(2,2)$, the corresponding entry in $\mathcal{P}_1$ is different from zero. So, we can directly reach 1 from 1 and 2, and 2 from 1. We have to find a path of any length n that starts from 2 and comes back to 2, i.e. a n such that $p^n_{ij} > 0$. Let's see:

$$\mathcal{P}_1^2 = \begin{pmatrix} 3/4 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}. \tag{6.10}$$

So, there is a path of length 2 that starts from 2 and ends back there – indeed, we can go from 2 to 1 with probability 1 and then from 1 to 2 with probability 1/2. Therefore, the Markov chain described by $\mathcal{P}_1$ is irreducible.

To find $\mathcal{P}_1^n$, it is best to diagonalize the matrix first:

$$\mathcal{P}_1^n = T\mathcal{D}^n T^{-1}. \tag{6.11}$$

The transformation matrix is given by

$$T = \begin{pmatrix} 1 & 1 \\ 1 & -2 \end{pmatrix}, \tag{6.12}$$

while the diagonal matrix to the n-th power is

$$\mathcal{D}^n = \begin{pmatrix} 1 & 0 \\ 0 & -1/2 \end{pmatrix}^n = \begin{pmatrix} 1 & 0 \\ 0 & (-2)^{-n} \end{pmatrix}. \tag{6.13}$$

Putting all together we obtain

$$\mathcal{P}_1^n = \begin{pmatrix} 1 & 1 \\ 1 & -2 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 2^{-n} \end{pmatrix}\begin{pmatrix} 2/3 & 1/3 \\ 1/3 & -1/3 \end{pmatrix} = \frac{1}{3}\begin{pmatrix} 2+(-1)^n 2^{-n} & 1-(-1)^n 2^{-n} \\ 2-(-1)^n 2^{1-n} & 1+(-1)^n 2^{1-n} \end{pmatrix}. \tag{6.14}$$

In the limit $n \to \infty$ this becomes

$$\lim_{n\to\infty} \mathcal{P}_1^n = \begin{pmatrix} 2/3 & 1/3 \\ 2/3 & 1/3 \end{pmatrix} \tag{6.15}$$

Now, for the second matrix:

$$\mathcal{P}_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1/4 & 0 & 3/4 \end{pmatrix}, \tag{6.16}$$

let's find directly $\mathcal{P}_2^n$. The transformation matrix is

$$T = \begin{pmatrix} -4 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}, \tag{6.17}$$

while the diagonalized one to the n-th power is

$$\mathcal{D}^n = \begin{pmatrix} -1/4 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}^n = \begin{pmatrix} (-1)^n 4^{-n} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{6.18}$$

Putting everything together,

$$\begin{aligned} \mathcal{P}_2^n &= \begin{pmatrix} -4 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} (-1)^n 4^{-n} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1/5 & 0 & 1/5 \\ 1/5 & 0 & 4/5 \\ 0 & 1 & 0 \end{pmatrix} \\ &= \frac{1}{5} \begin{pmatrix} 1 + (-1)^n 4^{1-n} & 0 & 4 - (-1)^n 4^{1-n} \\ 0 & 5 & 0 \\ 1 - (-1)^n 4^{-n} & 0 & 4 + (-1)^n 4^{-n} \end{pmatrix}. \end{aligned} \tag{6.19}$$

Therefore, since $\mathcal{P}_2^n$ has entries which are zero, the corresponding chain is *not* irreducible. The limit $n \to \infty$ is

$$\lim_{n \to \infty} \mathcal{P}_2^n = \begin{pmatrix} 1/5 & 0 & 4/5 \\ 0 & 1 & 0 \\ 1/5 & 0 & 4/5 \end{pmatrix}. \tag{6.20}$$

Moving to the next exercise, the goal is to prove whether a Markov chain defined by a given matrix is regular. To do it, it suffices to find a n such that every entry of $\mathcal{P}^n$ is positive. This implies that every entry of $\mathcal{P}^m$ is positive whenever $m \geqslant n$. In the first case it is simple, $n = 2$ suffices to end the proof:

$$\begin{pmatrix} 1/2 & 1/4 & 1/4 \\ 1/2 & 0 & 1/2 \\ 1/4 & 1/4 & 1/2 \end{pmatrix}^2 = \frac{1}{16} \begin{pmatrix} 7 & 3 & 6 \\ 6 & 4 & 6 \\ 6 & 3 & 7 \end{pmatrix}. \tag{6.21}$$

The second matrix instead is problematic:

$$\begin{pmatrix} 1 & 0 \\ 1/2 & 1/2 \end{pmatrix}. \tag{6.22}$$

If we look at it, we can realize that we cannot reach state 2 from state 1 in any way. So it is not regular either.