An Introduction to Monte Carlo Methods

MA213 Second Year Essay

Contents

1 Introduction

In *MA124 Maths by Computer*, we have seen an example of approximating π using computers. The method is given as follows:

1. Generate n random points (x, y) inside the square $[-1, 1] \times [-1, 1]$ uniformly.
2. Count the number of points that lie in the set $\{(x, y) : x^2 + y^2 \leq 1\}$, the unit circle.
3. Let Z be the number of points lying in the circle. A point either falls in the circle or does not falls in the circle, so $Z \sim \text{Bin}(n, p)$ where n is the number of random points generated and $p = \mathbb{P}[\text{a point lies inside circle}]$.
4. Estimate the probability that a point lies inside the circle by Z/n (this is known as the maximum likelihood estimator for data modelled by a Binomial random variable and proved in *ST220 Introduction to Mathematical Statistics*).
5. We know that
$$\mathbb{P}[\text{a point lies inside circle}] = \frac{\text{Area of the unit circle}}{\text{Area of the unit square}} = \frac{\pi}{4}$$
so we estimate π by $4 \times \frac{Z}{n}$.

The following diagram, taken from [10], illustrates this approximation:

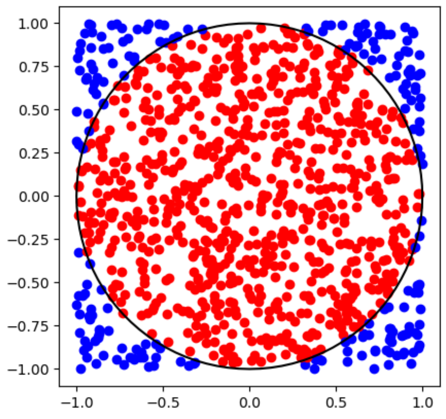

Figure 1: Monte Carlo Simulation with 1000 points

This is an example of *Monte Carlo integration* (more in Section 4) which uses randomness to approximate integrals – every time we perform a simulation we obtain different estimates of π, which is in contrast to deterministic methods such as Simpson's rule.

Monte Carlo methods is a class of simulation-based computational techniques which uses randomness to solve problems that are deterministic in nature, for example numerical integration, option pricing, optimization, etc. It is one of the most important analytical tools for modern statistical inference and has significant applications in physics and mathematical finance.

In this essay, we will explore the mathematical theory of Monte Carlo methods: what "randomness" is in the context of statistical simulations; how we can obtain random samples via simulation methods, and ultimately how we can use them to solve problems. For the majority of the essay, we follow closely from Chapters 1, 2 and 5.1 of Ritabrata Dutta's lecture notes for *ST407 Monte Carlo Methods* in [1], as well as the contents from Chapter 1-6 of Matthias Winkel's lecture notes for *Simulation* (a second-year course at the University of Oxford) in [7].

2 Pseudo-random numbers

In the example of estimating π in the introduction, we have assumed that we are able to generate random points uniformly using a computer algorithm, and for any Monte Carlo simulation we need to reproduce randomness using computers. This is indeed a philosophical paradox, since computer algorithms are deterministic in nature so the points generated in the 2×2 square cannot be truly random. It turns out that these sequences of "random numbers" generated by computers are known as **pseudo-random numbers** which are deterministic, and are usually used in Monte Carlo simulations.

As we will see in later sections, we can easily obtain samples from other distribution via deterministic algorithms by only using $U[0,1]$ as our only source of "randomness". A way to generate pseudo-random numbers, which is currently used by programming languages such as Python and R, is by using **psuedo-random number generators** (PRNGs). Their typical structure, taken from [4] Chapter 3.1, is given as follows:

There is a finite set S of states, and a function $f : S \to S$. There is also an output space U, which is usually $[0,1]$ and an output function $g : S \to U$ so that the output number always lie between 0 and 1. The generator is given an initial value for the state, S_0 which is known as the *seed*. We can then define a sequence of random numbers by

$$\begin{aligned} S_n &:= f(S_{n-1}), \;\; n \geq 1; \\ U_n &:= g(S_n) \end{aligned}$$

This is just an iteration and there is nothing random here: if we choose the same seed and run the algorithm twice we will eventually get the same output. Since S is finite, there exists some positive integer p such that $S_{n+p} = S_n$, and the smallest p that satisfies this relation is known as the *period* of the generator.

A good pseudo-random number generator should possess the following properties (taken from [3] Chapter 3.1):

- For any initial seed, the resultant sequence has the "appearance" of being a sequence of independent $U[0,1]$ random variables.
- For any initial seed, the period of the PRNG should be long. For example, the Mersenne Twister, a very popular modern PRNG, has period $2^{19937} - 1$ (see [2] Chapter 6.1.1);
- The values can be computed efficiently on a digital computer.

Here is one of the oldest and most well-known PRNGs based on simple recursions using modulo arithmetic, taken from [4] Chapter 3.2.1 and [3] Chapter 3.1, which gives a good insight on how PRNG works:

Example (Linear congruential random number generator (LCRNG))**.** The LCRNG is defined by the recurrence relation

$$X_n := (a + bX_{n-1}) \bmod m$$

where $X_1, X_2, \ldots$ is a sequence of pseudo-random values, X_0 is the seed (and should be specified by the user for reproducibility), a and b are integers in $\{0, \ldots, m-1\}$, and m is a positive integer. The state space in this case is $\{0, \ldots, m-1\}$.
We then choose our output function $g : S \to [0,1]$ as

$$g(X_n) = \frac{X_n}{m}$$

so that the output lies between 0 and 1.

We will not go into the details of this algorithm, for example, how constants a, b and m should be chosen (they have to satisfy the criteria for being a good PRNG), but it is worth nothing that this is a seriously flawed algorithm for reproducing randomness. This is illustrated by *Marsaglia's theorem* from number theory (see [5] for a detailed discussion). In fact, much better methods exist, such as the Multiply-with-carry pseudo-random number generator (developed by Marsaglia himself to address the issues of LCRNGs) which is mainly used in game development and the Mersenne Twister which is the default PRNG in many programming languages including R, Python and MATLAB.

3 Simulation methods

Now we know that computers are able to generate sequences of independent $U[0,1]$ by PRNGs, and from [6] Chapter 5.5 it turns out that this is the only "randomness" that a computer can generate naturally by itself. So the question is whether we can generate more complicated random variables and stochastic models from these uniform random variables. In this section we will present methods of obtaining samples from other distributions using psuedo-random numbers, which is known as *statistical simulation*.

3.1 The inversion method

One of the simplest methods of generating random samples from a distribution with cumulative distribution function $F(x) = \mathbb{P}(X \leq x)$ is based on the inverse of the CDF. The following theorem is taken from *ST220 Introduction to Mathematical Statistics*:

Theorem (Probability integral transform)**.** Suppose X is a continuous random variable with a continuous and strictly increasing CDF $F_X(x)$. Then the random variable defined by $Y := F_X(X) \sim U[0,1]$.

Proof.

$$\begin{aligned}
\mathbb{P}(Y \leq y) &= \mathbb{P}(X \leq F_X^{-1}(y)) \qquad (\text{for } y \in (0,1)) \\
&= F_X(F_X^{-1}(y)) \\
&= \begin{cases} 0 & \text{if } y < 0 \\ y & \text{if } y \in [0,1] \\ 1 & \text{if } y > 1 \end{cases} \\
f_Y(x) &= \frac{\mathrm{d}}{\mathrm{d}y} F_X(F_X^{-1}(y)) \\
&= \begin{cases} 1 & \text{if } y \in [0,1] \\ 0 & \text{otherwise .} \end{cases}
\end{aligned}$$

This implies that $Y \sim U[0,1]$. □

It turns out that we can work backwards and sample from X using $U[0,1]$:

Let $U \sim U[0,1]$ and $Z = F_X^{-1}(U)$. Then for $x \in \mathrm{Im}(F_X^{-1})$, we have

$$\begin{aligned}
\mathbb{P}(Z \leq x) &= \mathbb{P}(F_X^{-1}(U) \leq x) \\
&= \mathbb{P}(U \leq F_X(x)) \\
&= \frac{F_X(x) - 0}{1 - 0} \\
&= F_X(x)
\end{aligned}$$

so any continuous random variable X with a continuous and strictly increasing CDF can be sampled using $U[0, 1]$.

However, in cases where F_X is not strictly increasing, the inverse F_X^{-1} does not exist and if F_X is not continuous (e.g discrete random variables), then it is not differentiable (since differentiability implies continuity) and differentiating the CDF does not make sense. To deal with these potential issues, we define the *generalised inverse* (this definition is taken from [1] Chapter 2.1) as follows:

Definition (Generalised inverse function)**.** The **generalised inverse function** $F^- : [0, 1] \to \mathbb{R} \cup \{-\infty, \infty\}$ is defined as

$$F^-(u) := \inf\{x \in \mathbb{R} : F_X(x) \geq u\}$$

with the convention that $\inf\{\emptyset\} = \infty$ and for any set S that is not bounded below $\inf S = -\infty$. The following figure illustrates this definition for F which is not continuous at x_1 and not strictly increasing:

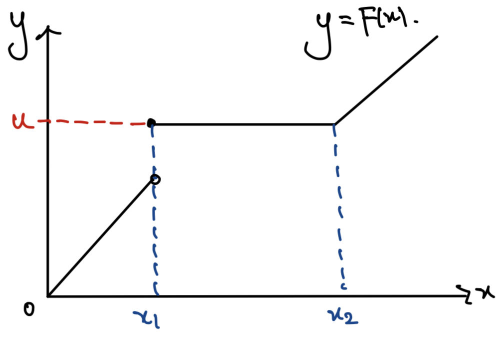

Figure 2: Illustration of the generalised inverse function

We can see that for any $a \geq x_1$ we have $F(a) \geq u$. So we pick the smallest value of a that satisfies $F(a) \geq u$, so in this case $F^-(u) = x_1$.

Lemma. $F_X^-(u) \leq x \iff u \leq F_X(x)$.

Proof. ($\Longleftarrow$) This follows directly from the definition of F^-.
($\Longrightarrow$) $F_X^-(u) \leq x \implies F_X^-(u) < \infty \implies \{x \in \mathbb{R} : F_X(x) \geq (u)\} \neq \emptyset$. So fix a decreasing sequence $(x_n) \subset \{x \in \mathbb{R} : F_X(x) \geq (u)\}$ such that $x_n \to F_X^-(u)$ as $n \to \infty$. By right-continuity of F_X,

$$\lim_{x_n \searrow F_X^-(u)} F_X(x_n) = F_X(F_X^-(u))$$

and by definition $F_X(x_n) \geq u$ for all $n \in \mathbb{N}$ so $F_X(F_X^-(u)) \geq u$.
Since CDF is increasing, $F_X^-(u) \leq x \implies F_X(F_X^-(u)) \leq F(x)$. Combining the two inequalities we have

$$F_X^-(u) \leq x \implies u \leq F(x).$$

□

With this definition and lemma we can prove the following proposition:

Proposition. Let $U \sim U[0, 1]$ and F_X be a CDF. Then $F_X^-(U)$ has CDF F_X.

Proof. From the previous lemma, $F_X^-(u) \leq x \iff u \leq F_X(x)$. Then for $U \sim U[0, 1]$,

$$\mathbb{P}(F_X^-(U) \leq x) = \mathbb{P}(U \leq F_X(x)) = F_X(x).$$

□

Now we can introduce the Inversion method algorithm, which is taken from [8] Lecture 2:

Algorithm (Inversion method)**.** The algorithm is given as follows:

1. Given CDF F, calculate F^{-};
2. Simulate independent $U_i \sim U[0,1]$ using pseudo random number generators;
3. Return $X_i = F^{-}(U_i) \sim F$.

We now provide a simple example which is taken from [7] Chapter 2.1:

Example (Weibull distribution)**.** The Weibull distribution is a model with parameters $\lambda > 0$ and $k > 0$ that is widely used in survival analysis. Its has probability density function

$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} \exp\left(-\frac{x}{\lambda}\right)^{k} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

and cumulative distribution function

$$F(x;\lambda,k) = 1 - \exp\left(-\frac{x}{\lambda}\right)^{k}.$$

F is continuous and strictly increasing , so we can just calculate F^{-1}:

$$u = 1 - \exp\left(-\frac{x}{\lambda}\right)^{k}$$
$$\implies F^{-1}(u) = \lambda \log\left(\frac{1}{1-u}\right)^{(1/k)}$$

Hence we can generate random samples from a Weibull random variable with parameters λ and k by applying the transformation $\lambda \log\left(\frac{1}{1-U}\right)^{(1/k)}$ to a uniform random variable $U \sim U[0,1]$.

The inversion method is a type of transformation method in which we generate random samples from a distribution other than the target distribution and apply transformations to them so that they come from the desired target distribution. These methods are extremely efficient but it can be difficult to find these transformations. Often, probability distributions are defined via their probability mass/density functions, and the inversion method does not work with distributions whose cumulative distribution function is not available in closed form, for example the gamma distribution and the Normal distribution. In these cases we will have to use other methods of sampling.

3.2 Rejection sampling

Rejection sampling is a method to generate random samples from a target probability distribution that is difficult or impossible (e.g. cannot be sampled via transformation methods) to sample from directly.
The basic idea of rejection sampling is as follows: we sample from a proposal distribution over a larger space than the target distribution and reject samples that are unlikely from the target distribution in a systematic way. For simplicity we only consider cases when the target distributions are continuous.

Suppose we want to sample from a target distribution X on a sample space $\Omega \subseteq \mathbb{R}$ which is continuous with pdf f_X. Rejection sampling is based on the identity

$$f_X(x) = \int_0^{f_X(x)} 1 \, \mathrm{d}u = \int_0^{\infty} \mathbf{1}_{\{[0,f_X(x)]\}}(u) \, \mathrm{d}u.$$

Note that $\int_{\mathbb{R}} f(x)\,\mathrm{d}x = 1$, and $\mathbf{1}_{\{[0,f_X(x)]\}}(u)$ is a function that depends on both x and u, so we can interpret this as a joint pdf between x and u. This implies that $f_X(x)$ can be interpreted as the marginal density of a uniform distribution over the area under $f_X(x), \{(x,u) : x \in \Omega,\ 0 \leq u \leq f_X(x)\}$. So one way of simulating X is by simulating over a uniform distribution (X,U) over a subset of $\mathbb{R}^2$ with area 1, and by the above identity X is marginally distributed from f_X. But this does not tell us how to sample uniformly from this area, which can be quite complicated, especially if we try to extend this idea to sampling from distributions of multivariate random variables (or random vectors).

Let $A := \{(x,u) : x \in \Omega, 0 \leq u \leq f_X(x)\}$. We can instead sample over $S \supseteq A$ in which direct sampling is "easy" – we can easily simulate using transformation methods. One way of obtaining S is by bounding f_X by $M \cdot g_X$, where $M > 1$ is a real constant and g_X is a distribution we can easily sample from. A uniform distribution on A can then be obtained by drawing from a uniform distribution on S, and rejecting samples in S that are not in A. In fact, this was the idea that we used in the introductory example of approximating π. We now have the algorithm for rejection sampling:

Algorithm (Rejection sampling)**.** Suppose we want to simulate a random variable X on a sample space $\Omega \in \mathbb{R}$ from its density f but we cannot do so directly. We then find an alternative density function g, known as the proposal distribution, such that

(i) $\exists\, M \in \mathbb{R}$ such that $\dfrac{f(x)}{g(x)} < M\, \forall\, x \in \Omega$, i.e. $\dfrac{f(x)}{g(x)}$ is bounded;

(ii) $\operatorname{supp}(g) \supset \operatorname{supp}(f)$, where $\operatorname{supp}(f) = \{x \in \Omega : f(x) \neq 0\}$ is the *support* of f. This means that the domain of f is contained in the domain of g.

If the above conditions are satisfied, then the algorithm (taken from [2] Chapter 6.3.1) is given by

1. Generate Y from g and a U from $U[0,1]$;
2. If $U \leq \dfrac{f(Y)}{Mg(Y)}$ then accept Y as a sample from f, otherwise return to step 1.

Proposition. The probability of having an acceptance is $1/M$ and the accepted values of the rejection sampling algorithm has pdf f.

The following proof is adapted from [8] Lecture 3 and [2] Chapter 6.3.2.

Proof.

$$\begin{aligned}
\mathbb{P}(Y \text{ is accepted}) &= \int_\Omega \mathbb{P}(\text{draw } Y = x \text{ and accept it})\,\mathrm{d}x \\
&= \int_\Omega g(x)\mathbb{P}\left(U \leq \frac{f(Y)}{Mg(Y)} | Y = x\right)\mathrm{d}x \\
&= \int_\Omega g(x) \cdot \frac{f(x)}{Mg(x)}\,\mathrm{d}x \\
&= \frac{1}{M}.
\end{aligned}$$

Note that to generate a X from $f_X(x)$ we need to generated many (X,U) pairs until a single value is accepted, and the probability that a single X is accepted is equal to $1/M$. This shows that we can model the number of attempts before X is accepted by $\text{Geom}(1/M)$ with expectation M. For maximal efficiency of our algorithm, we want M to be as small as possible.

Now we find the CDF of the accepted values:

$$\begin{aligned}
\mathbb{P}(Y \leq t | Y \text{ is accepted}]) &= \frac{\mathbb{P}(Y \leq t \text{ and } Y \text{ is accepted})}{1/M} \\
&= M \cdot \int_{\Omega} \mathbb{P}(\text{draw } Y = x \text{ and accept it and it is less than } t) \\
&= M \cdot \int_{\Omega} g(x) \mathbb{P}\left(Y \leq t, U \leq \frac{f(Y)}{Mg(Y)} | Y = x \right) \mathrm{d}x \\
&= M \cdot \int_{\Omega} g(x) \mathbb{P}\left(x \leq t, U \leq \frac{f(x)}{Mg(x)} \right) \mathrm{d}x \\
&= M \cdot \int_{\Omega} \mathbf{1}_{(-\infty, t]}(x) \cdot g(x) \cdot \frac{f(x)}{Mg(x)} \mathrm{d}x \\
&= \int_{\infty}^{t} f(x) \, \mathrm{d}x
\end{aligned}$$

□

The following example is taken from [8] Lecture 3:

Example (Sampling Normal distributions from a Cauchy proposal)**.** Suppose $X \sim \mathcal{N}(0,1)$ with pdf $f_X(x) = \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-x^2}{2}\right)$. We want to find a proposal distribution g which has a similar "bell" shape as the normal distribution while the domain of g contains that of f. A suitable proposal distribution is $Y \sim \text{Cauchy}(0,1)$ with pdf $g_Y(x) = \frac{1}{\pi(1+x^2)}$.

$$\begin{aligned}
\frac{f(x)}{g(x)} &= \frac{\frac{1}{\sqrt{2\pi}} \exp(-\frac{x^2}{2})}{\frac{1}{\pi(1+x^2)}} \\
&= \sqrt{\frac{\pi}{2}} (1+x^2) \exp\left(-\frac{x^2}{2}\right)
\end{aligned}$$

We can show that by differentiating this expression twice that global maximum is attained when $x = \pm 1$, so $\frac{f(x)}{g(x)} \leq \frac{f(1)}{g(1)} = 2\pi \exp\left(-\frac{1}{2}\right) = M$. Also $g(0) = 0$ and $f(0) = 0$ so both conditions hold.

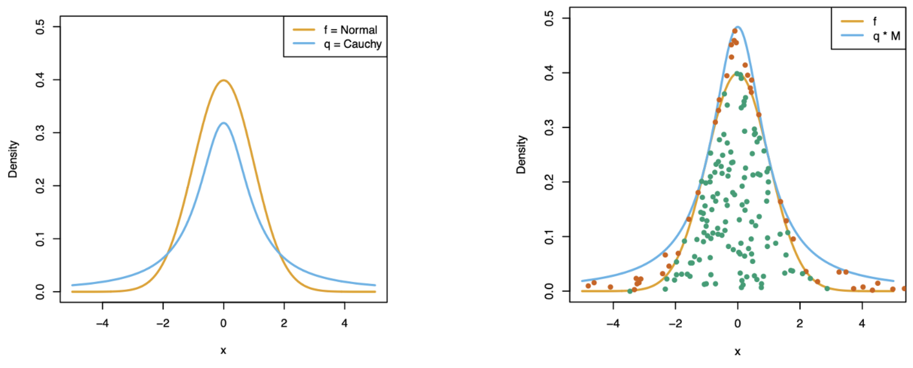

Figure 3: Sampling Normal from a Cauchy proposal

Remark. This example serves as an example to show how rejection sampling works. In fact, Normal distributions can be sampled directly using *Box-Muller transform.* This is covered in Appendix A.

4 Monte Carlo Integration and Importance Sampling

4.1 Monte Carlo Integration

We have now developed some methods to simulate from a given distribution X, i.e. we have the "randomness" needed in Monte Carlo simulations. It turns out that these are tools for us to answer more specific questions such as

(i) What is the mean of this distribution, $\mathbb{E}(X)$?

(ii) How likely is it for the distribution to produce values above a certain threshold a, i.e. what is $\mathbb{P}(X > a)$?

(iii) Given some function f, what is the expected value of the function f of X, $\mathbb{E}(f(X))$?

Note that (iii) generalises (i) and (ii): We can set $f(x) = x$ and $f(x) = \mathbf{1}_{(a,\infty)}(x)$ to achieve (i) and (ii) respectively. In fact, many quantities of interest, such as probabilities ((ii) is an example), sums and integrals (by definition of expectations and law of the unconscious statistician), can be expressed in terms of expectations. With the simulated data, we can estimate $\mathbb{E}[f(x)]$ using the **Monte Carlo Estimator**. The definition, taken from [8] Lecture 1, is as follows:

Definition (Monte Carlo Estimator)**.** Let X be either a discrete random variable taking values in a countable or finite set Ω with probability mass function f_X, or a continuous random variable taking values in $\Omega \subseteq \mathbb{R}^n$ with probability density function f_X. Then, by the law of the unconscious statistician, the expected value of a function $\phi : \Omega \to \mathbb{R}$ of X is

$$\theta = \mathbb{E}[\phi(X)] = \sum_{x \in \Omega} \phi(x) f(x)$$

if X is discrete, and

$$\theta = \mathbb{E}[\phi(X)] = \int_{\Omega} \phi(x) f(x) \, \mathrm{d}x$$

if X is continuous.

Let $X_1, \ldots, X_n$ be independent and identically distributed random variables with pmf/pdf f_X, i.e. a sample of independent copies of X. Then we define

$$\hat{\theta} := \frac{1}{n} \sum_{i=1}^{n} \phi(X_1)$$

to be the **Monte Carlo Estimator** of the expectation θ.

Algorithm (Monte Carlo Algorithm)**.** The corresponding algorithm, known as the Monte Carlo Algorithm, is given by:

1. Simulate $X_1, \ldots, X_n$ from the given distribution;
2. Return $n^{-1}(\phi(X_1) + \ldots + \phi(X_n))$;

so the key to such algorithm is the first step – simulation.

We will focus on the case where f is continuously distributed with pdf f – in this case the estimation is known as **Monte Carlo Integration** because it is a numerical method to approximate the integral

$$\mathbb{E}(\phi(X)) = \int_{\Omega} \phi(x) f(x) \, \mathrm{d}x.$$

We return to the example in the introduction. We have

$$\mathbb{P}[\text{a point lies inside circle}] = \frac{\text{Area of the unit circle}}{\text{Area of the unit square}} = \frac{\pi}{4}.$$

How do we estimate this quantity through Monte Carlo Simulation?

Let R be the region defined by the set $\{(x,y) : x^2+y^2 \leq 1\}$, $X, Y \sim U[-1,1]$ be two independent random variables and $S = [-1,1]\times[-1,1]$. We know (from *ST112 Probability B*) that $f_{X,Y}(x,y) = \frac{1}{1+1}\cdot\frac{1}{1+1} = \frac{1}{4}$. So

$$\begin{aligned}\frac{\int_R \mathrm{d}x\,\mathrm{d}y}{\int_S \mathrm{d}x\,\mathrm{d}y} &= \int\int_S \mathbf{1}_R(X,Y)\cdot\frac{1}{4}\,\mathrm{d}x\mathrm{d}y \\ &= \mathbb{E}[\phi(X,Y)] \\ &= \theta\end{aligned}$$

where the expectation is with respect to the uniform distribution on S, and

$$\phi(X,Y) = \mathbf{1}_R(X,Y).$$

To sample uniformly on S, we use the transformation

$$X = 2U_1 - 1, \qquad Y = 2U_2 - 1$$

where $U_1, U_2 \sim U[0,1]$.

The following proposition and the subsequent proofs are adapted from [1] Chapter 1.2.1 and [8] Lecture 1:

Proposition (Properties of the Monte Carlo Estimator)**.** Assume that $\theta = \mathbb{E}[\phi(X)]$ exists. Then

(i) (Unbiasedness) The Monte Carlo estimator is an unbiased estimator for θ, i.e.

$$\mathbb{E}[\hat{\theta_n}] = \theta;$$

(ii) (Strong consistency) The Monte Carlo estimator is strongly consistent, i.e.

$$\hat{\theta_n} \to \theta \text{ almost surely as } n \to \infty.$$

(iii) If we further assume that $\mathrm{Var}(\phi(X)) = \sigma^2$ exists, then the mean-squared error is

$$\mathbb{E}[(\hat{\theta_n} - \theta)^2] = \mathrm{Var}(\hat{\theta}) = \frac{\sigma^2}{n}$$

and is equal to the variance of the Monte Carlo Estimator (since the estimator is unbiased). $\mathrm{Var}[\phi(X)]$ can then be estimated using the sample variance.

Proof. (i) $\mathbb{E}[\hat{\theta_n}] = \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n}\phi(X_i)\right)$

By linearity of expectation,

$$\mathbb{E}[\hat{\theta_n}] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[\phi(X_i)]$$

Since $X_1, \ldots, X_n$ are identically distributed and have the same pmf/pdf as X,

$$\begin{aligned}\mathbb{E}[\hat{\theta_n}] &= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[\phi(X)] \\ &= \mathbb{E}[\phi(X)] \\ &= \theta\end{aligned}$$

(ii) This follows directly from the Strong Law of Large Numbers (*ST202 Stochastic Processes*).

(iii) From (i), $\mathbb{E}[\hat{\theta_n}] = \theta$,

$$\begin{aligned}\mathbb{E}[(\hat{\theta_n} - \theta)^2] &= \mathbb{E}[\hat{\theta_n} - \mathbb{E}[\hat{\theta_n}]] \\ &= \mathrm{Var}(\hat{\theta_n}) \\ &= \mathrm{Var}\left(\frac{1}{n}\sum_{i=1}^{n}\mathrm{Var}(\phi(X_i))\right) \\ &= \frac{1}{n^2}\sum_{i=1}^{n}\mathrm{Var}(\phi(X_i)) \quad \text{(by independence of } X_1, \ldots, X_n) \\ &= \frac{\sigma^2}{n}.\end{aligned}$$

□

4.2 Importance sampling

We have seen how we can use the Monte Carlo Estimator to estimate $\theta = \mathbb{E}[\phi(X)]$, which uses samples from the pdf/pmf of X, f, to estimate θ. However, when considering problems such as the probability of a random variable exceeding a high threshold, it might not be an reliable estimator since most of the samples will be irrelevant. **Importance sampling** (IS) is another method of estimating $\mathbb{E}[\phi(X)]$, but rather than sampling directly from X, it samples from a proposal distribution (like rejection sampling). It utilises the idea of using *weights* to correct for the fact that we sample from the proposal distribution g instead of the target distribution f by adjusting the difference between f and g.

Importance sampling is based on the following identity, taken from [8] Lecture 4:

Proposition (Importance sampling identity)**.** Let X and Y be continuous (or discrete) random variables on some sample space Ω with pdf(pmf) f and g respectively. If $f(x) > 0 \implies g(x) > 0$ (so that w is well-defined) then for any (measurable) function $\phi : \Omega \to \mathbb{R}$ we have

$$\mathbb{E}_f[\phi(X)] = \mathbb{E}_g[\phi(Y)w(Y)],$$

where $w : \Omega \to \mathbb{R}^+$ is the **importance weight function**

$$w(x) := \frac{f(x)}{g(x)},$$

and the notation $\mathbb{E}_f$ and $\mathbb{E}_g$ is used to describe the expectation with respect to f and g respectively.

Proof. Continuous case:

$$\begin{aligned}\mathbb{E}_f[\phi(X)] &= \int_\Omega f(x)\phi(x)\,\mathrm{d}x \\ &= \int_\Omega g(x)\cdot\frac{f(x)}{g(x)}\phi(x)\,\mathrm{d}x \\ &= \int_\Omega g(x)w(x)\phi(x)\,\mathrm{d}x \\ &= \mathbb{E}_g[\phi(Y)w(Y)].\end{aligned}$$

Replacing the integrals by sums completes the proof for the discrete case. □

This leads to the following definition of the **Importance Sampling Estimator**:

Definition (Importance Sampling Estimator)**.** Let f and g be pdf/pmfs on a sample space Ω, $\phi : \Omega \to \mathbb{R}$ and X be a continuous/ discrete random variable with pdf/pmf f. Suppose $f(x)\phi(x) \neq 0 \implies g(x) > 0$, and $Y_1, \dots, Y_n$ are independent and identically distributed samples with pdf/pmf g. The importance sampling estimator is defined as

$$\hat{\theta_n^{\text{IS}}} := \frac{1}{n} \sum_{i=1}^{n} \phi(Y_i) w(Y_i),$$

provided that $\theta = \mathbb{E}_f[\phi(X)] = \mathbb{E}_g[\phi(Y)w(Y)]$ exists.

The following properties and subsequent proofs of the Importance sampling are adapted from [1] Chapter 2.3 and [8] Lecture 4:

Proposition (Properties of the Importance Sampling Estimator)**.** The Importance Sampling Estimator satisfies the following properties:

(i) (Unbiasedness) The IS estimator is an unbiased estimator for θ, i.e.

$$\mathbb{E}[\hat{\theta_n^{\text{IS}}}] = \theta;$$

(ii) (Strong consistency) The IS estimator is strongly consistent, i.e.

$$\hat{\theta_n^{\text{IS}}} \to \theta \text{ almost surely as } n \to \infty.$$

(iii) If $\text{Var}_g(\phi(Y_i)w(Y_i))$ exists, then the mean-squared error is

$$\mathbb{E}[(\hat{\theta_n^{\text{IS}}} - \theta)^2] = \text{Var}(\hat{\theta_n^{\text{IS}}}) = \frac{1}{n}[\mathbb{E}_f[\phi(X)^2 w(X)] - \theta^2].$$

Proof. (i) $\mathbb{E}[\hat{\theta_n^{\text{IS}}}] = \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n} \phi(Y_i)w(Y_i)\right)$.

By linearity of expectation,

$$\mathbb{E}[\hat{\theta_n^{\text{IS}}}] = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_g[\phi(Y_i)w(Y_i)]$$

Since $Y_1, \dots, Y_n$ are identically distributed and have the same pmf/pdf g,

$$\begin{aligned}\mathbb{E}[\hat{\theta_n^{\text{IS}}}] &= \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_g[\phi(Y_i)w(Y_i)] \\ &= \mathbb{E}_g[\phi(Y_1)w(Y_1)] \\ &= \mathbb{E}_f(\phi(X)) \\ &= \theta.\end{aligned}$$

(ii) Let $Z_i = \phi(Y_i)w(Y_i)$. Then $Z_i, \dots, Z_n$ are i.i.d. with mean $\mathbb{E}[Z_i] = \mathbb{E}_g[\phi(Y_i)w(Y_i)] = \theta$. The result then follows directly from the strong law of large numbers.

(iii)

$$\begin{aligned}
\mathrm{Var}(\hat{\theta_n^{\mathrm{IS}}}) &= \mathrm{Var}_g\left(\frac{1}{n}\sum_{i=1}^{n}\phi(Y_i)w(Y_i)\right) \\
&= \frac{1}{n^2}\sum_{i=1}^{n}\mathrm{Var}_g(\phi(Y_i)w(Y_i)) \quad \text{(by independence of } Y_1,\dots,Y_n) \\
&= \frac{1}{n}\mathrm{Var}_g(\phi(Y_i)w(Y_i)) \quad (i \in \{1,\dots,n\}) \\
&= \frac{1}{n}\left[\mathbb{E}_g[(\phi(Y_i)w(Y_i))^2] - \mathbb{E}_g[\phi(Y_i)w(Y_i)]^2\right] \\
&= \frac{1}{n}\left[\int_\Omega g(x)[\phi(x)]^2\frac{[f(x)]^2}{[g(x)]^2}\,\mathrm{d}x - \theta^2\right] \\
&= \frac{1}{n}\left[\int_\Omega f(x)[\phi(x)]^2 w(x)\,\mathrm{d}x - \theta\right] \\
&= \frac{1}{n}[\mathbb{E}_f[\phi(X)^2 w(X)] - \theta^2].
\end{aligned}$$

□

This shows that the mean squared error of the importance sampling depends on the choice of the proposal distribution, while that of the Monte Carlo estimator only depends on the samples from the underlying distribution. We want to choose the proposal distribution g such that it minimizes the mean squared error. Before showing how to do that, we first introduce the **Importance Sampling algorithm**:

Algorithm (Importance Sampling algorithm)**.** Suppose we are given a distribution X on a sample space Ω with pdf/pmf f and a function $\phi : \Omega \to \mathbb{R}$ in which we want to estimate $\mathbb{E}[\phi(X)]$. Choose a proposal distribution g which satisfies the following conditions:

(i) $\mathrm{supp}(g) \supset \mathrm{supp}(f \cdot \phi)$, i.e. the domain of g contains the domain of $f \cdot \phi$ and g should be easy to sample from;

(ii) $\mathrm{Var}_g(\phi(Y_i)w(Y_i))$ exists, i.e. $\mathbb{E}_f\left[\phi(X)^2 \cdot w(x)\right] = \mathbb{E}_f\left[\phi(X)^2 \cdot \frac{f(X)}{g(X)}\right] < \infty$.

If $\mathrm{Var}_f[\phi(X)]$ is known to be finite, then $\mathbb{E}_f[\phi(X)^2] = \mathrm{Var}_f[\phi(X)] + \mathbb{E}_f[\phi(X)]^2 < \infty$. A sufficient condition for $\mathbb{E}_f\left[\phi(X)^2 \cdot w(X)\right] < \infty$, taken from [8] Lecture 4, would then be $w(x) \leq M$:

$$\begin{aligned}
\mathbb{E}_f\left[\phi(X)^2 \cdot w(x)\right] &= \int_\Omega f(x)w(x)\phi(x)^2\,\mathrm{d}x \\
&\leq \int_\Omega M \cdot f(x)\phi(x)^2\,\mathrm{d}x \\
&= M \cdot \mathbb{E}_f[\phi(X)] \\
&< \infty.
\end{aligned}$$

If the above conditions are satisfied, then the algorithm, taken from [1] Chapter 2.3, is given by:

1. For $i = 1,\dots,n$;

   (i) Generate Y_i from g;

(ii) Set $w(Y_i) = \dfrac{f(X_i)}{g(X_i)}$.

2. Return $\hat{\theta^{\text{IS}}} = \dfrac{\sum_{i=1}^n \phi(Y_i)w(Y_i)}{n}$.

The importance sampling algorithm is somewhat similar to the rejection sampling in terms of the conditions of the proposal distribution, but rejection sampling generates samples from f, while importance sampling provides an estimate to the expectation $\mathbb{E}[\phi(X)]$.

We have now come up with an algorithm for importance sampling, and we know that the mean-squared error of the IS estimator depends on the proposal distribution g. A natural question then is whether there exists a proposal distribution g that minimizes the mean-squared error? The following theorem and its subsequent proof, which is taken from [1] Chapter 2.3, answers this question:

Theorem (Optimal proposal)**.** The proposal distribution g that minimizes the mean-squared error (equivalently, the variance due to unbiasedness) of the IS estimator is

$$g^*(x) = \frac{f(x)|\phi(x)|}{\mathbb{E}_f[|\phi(X)|]}$$

with the mean-squared error being

$$\mathbb{E}[(\hat{\theta_n^{\text{IS}}} - \theta)^2] = \text{Var}(\hat{\theta_n^{\text{IS}}}) = \frac{1}{n}[\mathbb{E}_f[|\phi(X)|]^2 - \theta^2].$$

Proof. We want to minimize

$$\text{Var}(\hat{\theta_n^{\text{IS}}}) = \frac{1}{n}[\mathbb{E}_f[\phi(X)^2 w(X)] - \theta^2]]$$

and since $\theta = \mathbb{E}_f(\phi(X))$ is independent of g, the problem now is to minimize $\mathbb{E}_f[\phi(X)^2 w(X)]$.

$$\begin{aligned}
\mathbb{E}_f[\phi(X)^2 w(X)] &= \mathbb{E}_g\left[\phi(Y_i)^2\left(\frac{f(Y_i)}{g(Y_i)}\right)^2\right] \qquad (i \in \{1,\dots,n\}) \\
&= \mathbb{E}_g\left[|\phi(Y_i)|^2\left(\frac{f(Y_i)}{g(Y_i)}\right)^2\right] \\
&\geq \left(\mathbb{E}_g\left[|\phi(Y_i)|\frac{f(Y_i)}{g(Y_i)}\right]\right)^2 \qquad \text{(by Jensen's inequality : } x \to x^2 \text{ is convex)} \\
&= \mathbb{E}_f[|\phi(X)|]^2.
\end{aligned}$$

We have achieved a lower bound for $\mathbb{E}_f[\phi(X)^2 w(X)]$, and this lower bound is attained when $g = g^*$:

$$\mathbb{E}_f\left[\phi(X)^2\frac{f(X)}{g^*(X)}\right] = \mathbb{E}_f[|\phi(X)|\mathbb{E}[|\phi(X)|]] = \mathbb{E}_f[|\phi(X)|]^2.$$

□

In the proof we have used the trick $\phi(X)^2 = |\phi(X)|^2$, because g^* is a probability density function so it must be non-negative. g^*, however, cannot be used in practice because it requires us to know $\mathbb{E}[|\phi(X)|]$ which is exactly what we want to estimate. It serves as a guideline to choose the proposal distribution g: choose $g(x)$ such that it has a similar shape to $f(x) \cdot |\phi(x)|$.

The following example is taken from [7] Chapter 5.3 as an illustration of Importance Sampling in rare event simulation:

Example (Tails of Normal distribution)**.** Consider the problem of estimating $\theta = \mathbb{P}[X > 3]$ where $X \sim \mathcal{N}(0,1)$. The Monte Carlo estimator will not be a good choice, because the majority of the samples will not lie in $(3,\infty)$ so to obtain a good estimate we need n to be really huge, which will make our estimation inefficient. So we use importance sampling with a proposal distribution that has a similar shape to $f_X(x) \cdot |\phi(x)|$, where $f_X(x)$ is the pdf of X and $\phi(x) = \mathbf{1}_{(3,\infty)}(x)$. This is basically identical to the shape of $\mathcal{N}(0,1)$ for $x > 3$, so we choose another Normal distribution with arbitrary mean μ and a same variance 1 (to preserve the shape) as our proposal distribution: $Y \sim \mathcal{N}(\mu,1)$ with pdf

$$g_Y(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2}\right)$$

Now our importance weight function is

$$w(x) = \frac{f_X(x)}{g_Y(x)} = \exp\left(-\frac{x^2-(x-\mu)^2}{2}\right) = \exp\left(\frac{\mu(\mu-2x)}{2}\right)$$

so if we generate $Y_1,\dots,Y_n$ from g_Y, the importance sampling estimator is given by

$$\hat{\theta_n^{\text{IS}}} = \frac{1}{n}\sum_{i=1}^{n} \phi(Y_i)w(Y_i) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{(3,\infty)}(Y_i)\exp\left(\frac{\mu(\mu-2Y_i)}{2}\right).$$

It was shown that in [7] Chapter 5.3 that by trying different values of μ ranging from 0 to 4, the value of μ that minimizes the variance is approximately 3.15 with 95% confidence interval $[1.3478, 1.3509]$.

One can generate $Y_1,\dots,Y_n$ by means of rejection sampling, or by applying Box-Muller transform to generate i.i.d. $\mathcal{N}(0,1)$ and adding 3.15 to all the samples, since if $X \sim \mathcal{N}(0,1)$ then $Y := \sigma X + \mu \sim \mathcal{N}(\mu,\sigma^2)$.

5 Markov Chain Monte Carlo

In this section we aim to quickly introduce, but not going into full detail, the idea of **Markov Chain Monte Carlo** (MCMC), which has significant applications in Bayesian statistics to compute the marginal likelihood, as well as statistical physics and computer science. The Metropolis-Hastings algorithm, which we will briefly cover, is one of the most widely used MCMC algorithms and is widely regarded as one of the most important algorithms of the 20th century.

We have seen how we can estimate $\theta = \mathbb{E}[\phi(X)]$, where X is a a random variable with pdf/pmf f and $\phi : \Omega \to \mathbb{R}$ using the Monte Carlo estimator and the Importance sampling estimator. What we have done is by simulating independent and identically distributed samples from f itself via transformation methods or a proposal distribution with pdf/pmf g by the means of rejection sampling. It turns out that these methods works well in low-dimensions but are generally inefficient as the number of dimensions of our target distribution increases – this is known as the *curse of dimensionality.* The following example illustrates how rejection sampling becomes inefficient in high dimensions:

Consider sampling from a standard multivariate normal distribution $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, I)$, where $\mathbf{0}$ is the n-dimensional zero vector and I is the $n \times n$ identity matrix with a proposal distribution $\mathbf{Y} \sim \mathcal{N}(\mathbf{0}, 1.01^2 I)$. We know (from *ST220 Introduction to Mathematical Statistics*) that for an n-dimensional multivariate normal distribution $\mathcal{N}(\mu,\Sigma)$ where Σ is positive-definite, it has pdf

$$f(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det(\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)\right),$$

so by letting $f_{\mathbf{X}}$ and $g_{\mathbf{Y}}$ be the pdf of $\mathbf{X}$ and $\mathbf{Y}$ respectively, we have

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n}} \exp\left(-\frac{1}{2}(\mathbf{x})^T I(\mathbf{x})\right) = \frac{1}{\sqrt{(2\pi)^n}} \exp\left(-\frac{1}{2}(|\mathbf{x}|^2)\right)$$

and

$$g_{\mathbf{Y}}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n (1.01)^{2n}}} \exp\left(-\frac{1}{2}(\mathbf{x})^T \frac{1}{1.01^2} I(\mathbf{x})\right) = \frac{1}{1.01^n \sqrt{(2\pi)^n}} \exp\left(-\frac{1}{2 \cdot 1.01^2}(|\mathbf{x}|^2)\right)$$

so

$$\frac{f_{\mathbf{X}}(\mathbf{x})}{g_{\mathbf{Y}}(\mathbf{x})} \leq (1.01)^{2n} = M.$$

This implies that the number of attempts before obtaining a sample is equal to 1.01^{2n}. For low dimensions our algorithms is pretty efficient, but as n get larger the expected number increases exponentially. Take $n = 1000$, the expected number of attempts before obtaining a sample $\approx 4.39 \times 10^8$, so it makes no sense to use rejection sampling in high dimensions.

One possible way to address this problem is instead of simulating i.i.d. samples we construct a Markov Chain $X_0, \dots, X_N \sim \text{Markov}(\lambda, P)$ (so our samples are not independent) whose invariant/stationary distribution is the target distribution f using the detailed-balance equations (i.e. reversibility, see *ST202 Stochastic Processes*). We will need the following theorem on Markov Chains, taken from [6] Chapter 1.10 and also covered in *ST202 Stochastic Processes*, which is very similar to the Strong Law of Large Numbers:

Theorem (Ergodic Theorem for Markov Chains)**.** Let $(X_n)_{n\geq 0}$ be an irreducible Markov Chain on a state space I. Assume that the Markov Chain admits an invariant distribution π (i.e. $\pi P = \pi$). Suppose $\phi : I \to \mathbb{R}$ is a bounded function, X is a discrete random variable on I with pmf π and $\theta = \mathbb{E}_\pi[\phi(X)]$. If $(X_n)_{n\geq 0}$ is positive recurrent, i.e. for every state $i \in I$ the expected amount of time for the chain to return to i given that it started at i is finite, then for any initial distribution λ,

$$\frac{1}{n}\sum_{k=0}^{n-1} \phi(X_k) \xrightarrow{a.s.} \theta.$$

Now we can define the **Markov Chain Monte Carlo estimator**, taken from [8] Lecture 6:

Definition (Markov Chain Monte Carlo estimator)**.** Let X be a discrete random variable defined on I with pmf π, $\phi : \Omega \to \mathbb{R}$ be a bounded function and $\theta = \mathbb{E}_f[\phi(X)]$. Suppose $(X_n)_{n\geq 0} \sim \text{Markov}(\lambda, P)$ with initial distribution λ and an irreducible transition matrix P which has an invariant distribution π. Then for any initial distribution λ we define the **MCMC estimator**

$$\hat{\theta_n}^{\text{MCMC}} = \frac{1}{n}\sum_{k=1}^{n-1} \phi(X_k).$$

The MCMC estimator is strongly consistent as a consequence of Ergodic theorem.

There are many different ways to construct reversible Markov Chains with a given invariant distribution. One way is by utilising the idea of rejection sampling, but the proposed value now depends on the last accepted value rather than being independent. This is the key idea of the famous **Metropolis-Hastings** algorithm. One simple version of it, taken from [8] Lecture 7 and only applies to finite state spaces, is given as follows:

Algorithm (Metropolis-Hastings)**.** Let I be a finite state space with $|I| = n$ and $\pi = (\pi_i : i \in I)$ be a distribution on I such that $\pi_i > 0 \,\forall\, i \in I$. Choose a proposal transition density $q(y|x)$ (which has to be easy to sample from), and denote $\mathbb{P}[Y = y|X = x]$ by $Y \sim q(\cdot|x)$. Set a starting value $x_0 \in I$ and set $X_0 = x_0$. Then the Metropolis-Hastings algorithm is given by: For $i = 1, 2, \dots, n-1$,

1. Assume $X_{i-1} = x_{i-1}$.
2. Simulate $Y_i \sim q(\cdot|x_{i-1})$ and $U_i \sim U[0, 1]$.

3. If $U_t \leq \alpha(Y_i|x_{i-1})$, where $\alpha(Y_i|x_{i-1})$ is the acceptance probability and is defined by

$$\alpha(y|x) = \min\left\{1, \frac{\pi_y q(x|y)}{\pi_x q(y|x)}\right\},$$

   then set $X_i = Y_i$ (accept), otherwise set $X_i = x_{i-1}$ (reject).

This generates a reversible Markov Chain whose invariant distribution is π.

Here we have only considered the case where I is finite. In fact, MCMC methods are applicable to countably infinite or continuous state spaces and is one of the most versatile and widespread classes of Monte Carlo algorithms. It can also be generalised to high-dimensional state spaces $I \subseteq \mathbb{R}^n$ where we replace X and Y be random vectors $\mathbf{X}$ and $\mathbf{Y}$ respectively (see [1] Chapter 5.1).

It is worth noting that the samples generated by MCMC algorithms are not independent (since they form a Markov Chain with a non-trivial dependency structure), so they should only be used as a last resort when Monte Carlo methods that generate independent samples fail.

A Box-Muller Transform

We have seen in section 3.2 that we can generate samples from $\mathcal{N}(0,1)$ using rejection sampling. However, this involves choosing a proposal distribution and generating samples that has a probability of being rejected $1-1/M$, so it is not an efficient way of sampling compared to direct transformations. It turns out that there exists such method to sample from $\mathcal{N}(0,1)$ directly – this is known as the *Box-Muller transformation method*, named after British statistician George E. P. Box and American mathematician Mervin E. Muller. We will need the following theorem from *ST220 Introduction to Mathematical Statistics*:

Theorem (Multivariate transformation of random variables)**.** Let $\mathbf{X}$ be an absolutely continuous random vector on $\mathbb{R}^n$ with density $f_{\mathbf{X}}(\mathbf{x})$, $g : \mathbb{R}^n \to \mathbb{R}^n$ be bijective, continuously differentiable with a continuously differentiable inverse $h : \mathbb{R}^n \to \mathbb{R}^n$. Then $\mathbf{Y} := g(\mathbf{X})$ is also absolutely continuous with density

$$f_{\mathbf{Y}}(\mathbf{y}) = f_{\mathbf{X}}(h(\mathbf{Y})) \cdot |J_h(\mathbf{y})|,$$

where $J_h(\mathbf{y})$ is the Jacobian given by

$$J_h(\mathbf{y}) = \det\left(\frac{\partial h}{\partial \mathbf{y}}\right) = \det\begin{pmatrix} \frac{\partial h_1}{\partial y_1} & \cdots & \frac{\partial h_1}{\partial y_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_n}{\partial y_1} & \cdots & \frac{\partial h_n}{\partial y_n} \end{pmatrix}.$$

The theorem also holds if $g : U \to V$ where U, V are open subsets of $\mathbb{R}^n$ as long as the other conditions are satisfied, i.e. g is a diffeomorphism (*MA259 Multivariable Calculus).*

Before moving on to the method we first prove a handy lemma (the statement of this lemma is taken from [1] Chapter 2.1):

Lemma. $R^2 \sim$ Exponential(1/2) and $\Theta \sim U(0, 2\pi)$ are two independent random variables if and only if $X := R\cos\Theta$ and $Y := R\sin\Theta$ are independent and identically distributed standard Normal random variables $\mathcal{N}(0,1)$.

Proof. R^2 and Θ are independent if and only if

$$f_{R^2,\Theta}(r^2,\theta) = f_{R^2}(r^2) \cdot f_\Theta(\theta) = \frac{1}{2}\exp\left(-\frac{1}{2}r^2\right) \cdot \frac{1}{2\pi}.$$

Consider the function $g : (0,\infty) \times (0, 2\pi) \to \mathbb{R}^2$ defined by

$$g(R^2, \Theta) = (R\cos\Theta, R\sin\Theta) = (X, Y).$$

The Jacobian of g is:

$$J_g(r^2,\theta) = \det\begin{pmatrix} \cos\theta \cdot \frac{1}{2r} & \sin\theta\frac{1}{2r} \\ -r\sin\theta & r\cos\theta \end{pmatrix} = r\cos^2\theta \cdot \frac{1}{2r} + r\sin^2\theta \cdot \frac{1}{2r} = \frac{1}{2} \neq 0$$

so by Inverse function theorem (*MA259 Multivariable Calculus) g* is bijective, continuously differentiable with continuously differentiable inverse h. To compute h, we solve

$$R\cos\Theta = X, \qquad R\sin\Theta = Y$$

which gives

$$h(X,Y) = \left(X^2 + Y^2, \arctan\left(\frac{Y}{X}\right)\right)$$

where $h : \mathbb{R}^2 \to (0,\infty) \times \left(0, \frac{\pi}{2}\right)$.

The Jacobian of h is then given by

$$J_h(x,y) = \det \begin{pmatrix} 2x & 2y \\ -\frac{y}{x^2} \cdot \frac{x^2}{x^2+y^2} & \frac{1}{x} \cdot \frac{x^2}{x^2+y^2} \end{pmatrix} = \frac{2x^2}{x^2+y^2} + \frac{2y^2}{x^2+y^2} = 2$$

Now we compute the joint density of X and Y:

$$\begin{aligned} f_{X,Y}(x,y) &= f_{R^2,\Theta}\left(x^2+y^2, \arctan\left(\frac{y}{x}\right)\right) \cdot |2| \\ &= 2 \cdot \frac{1}{2\pi} \cdot \frac{1}{2} \exp\left(-\frac{1}{2}(x^2+y^2)\right) \\ &= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right) \cdot \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}y^2\right) \\ &= f_X(x) \cdot f_Y(y) \end{aligned}$$

Realising that this is a product of two densities of the standard Normal distribution and using $f_{X,Y}(x,y) = f_X(x)f_Y(y) \iff X$ and Y are independent completes the proof. □

$Y := R^2 \sim$ Exponential(1/2) has CDF

$$F_Y(y) = \begin{cases} 1 - \exp(-\frac{1}{2}y) & \text{if } y \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

which is available in closed form so we can sample it directly using the inversion method. The inverse of CDF of Y is $F_Y^{-1}(U) = -2\log(1-U)$ and if $U \sim U[0,1]$ then the random variable $F_Y^{-1}(U)$ has the same distribution as Y, so we can use $\sqrt{-2\log(1-U)}$ to simulate R.

Now we present the Box-Muller transformation method which is taken from [10]:

Algorithm (Box-Muller transform)**.** The Box-Muller transformation method is given as follows:

1. Simulate independent $U_1, U_2 \sim U[0,1]$;
2. Generate $R = \sqrt{-2\log(1-U_1)}$ and $\Theta = 2\pi U_2$'
3. Return $X = \sqrt{-2\log(1-U_1)} \cdot \cos(2\pi U_2), Y = \sqrt{-2\log(1-U_1)} \cdot \sin(2\pi U_2)$

It follows directly from the above lemma that X, Y are identically and independently distributed $\mathcal{N}(0,1)$ random variables. A code illustrating the Box-Muller transformation method is included in Appendix B3.

B Code for Monte Carlo Simulations

The following code are written using the programming language R. The last line of each chunk shows the output of the code.

B.1 Rejection Sampling: Normal via Cauchy Proposal

In this code we generate 10000 samples from $\mathcal{N}(0,1)$ via rejection sampling:

```r
f_X <- function(x) { 1 / sqrt(2 * pi) * exp(-0.5 * x ** 2)} #target distribution
f_Y <- function(x) { 1 / pi / (1 + x ** 2)} #proposal distribution
M <- sqrt(2 * pi) * exp(-1 / 2)
n <- 10000 #number of samples
x <- array(NA, n)
i <- 1
while(i <= n) {
  U1 <- runif(1) #psuedo random number between 0 and 1
  Xp <- tan(pi * (U1 - 0.5)) #inverse transform method
  U2 <- runif(1)
  if (U2 <= (f_X(Xp) / f_Y(Xp) / M)) {
    x[i] <- Xp
    i <- i + 1 }
}
(c(mean(x), var(x))) #returns the mean and variance of the resulting sample

## [1] -0.004277369  0.980464614
```

We can see that the mean and variance of the resulting sample are approximately 0 and 1 respectively.

B.2 Introductory example: Approximating π

```r
n <- 1000 #number of random points
x <- array(0, c(2,n)) #array to store coordinates of points generated
t <- array(0, c(1,n)) #array to store whether the point is inside circle or not
for (i in 1:n) {
  x[1,i] <- 2*runif(1)-1
  x[2,i] <- 2*runif(1)-1
  if (x[1,i]*x[1,i] + x[2,i]*x[2,i] <= 1) {
    t[i] <- 1
  } else {
    t[i] <- 0 }
}
print(sum(t)/n*4) #count the number of points in the circle and hence pi

## [1] 3.176
```

The Monte Carlo estimation of π is 3.152 in this case.

B.3 Box-Muller Transform

```
n <- 100000 #100000 realisations
u1 <- runif(n) #n psuedo-random numbers between 0 and 1
u2 <- runif(n)
lambda <- 1 / 2 #parameter of the exponential distribution
r2 <- -2 * log(1 - u1) #step 2 of the algorithm
theta <- 2 * pi * u2
r <- sqrt(r2)
x <- r * cos(theta)
y <- r * sin(theta)
round(c(mean(x), var(x)), 3) #mean and variance of X to 3dp

## [1] 0.003 0.997

round(c(mean(y), var(y)), 3) #mean and variance of Y to 3dp

## [1] 0.005 1.003

round(cor(x, y),5) #correlation between X and Y

## [1] 7e-05
```

We can see that $\text{corr}(X, Y) \approx 0$ since X and Y are independent.

References

[1] Dutta, R. (2020). *ST407 Monte Carlo Methods*, Lecture Notes, University of Warwick, Autumn Term 2020.

[2] Peng R. (2022). *Advanced Statistical Computing*, Self-published. URL: `https://bookdown.org/rdpeng/advstatcomp/`

[3] Ross, S. (2013). *Simulation* (5th ed.). Academic Press. DOI: `https://doi.org/10.1016/C2011-0-04574-X`

[4] Kennedy, T. (2016). *Monte Carlo Methods – a special topics course*, Lecture Notes, University of Arizona. URL:`https://www.math.arizona.edu/~tgk/mc/book.pdf`

[5] Marsaglia, G. (1968). *Random numbers fall mainly in the planes.* DOI: `https://www.pnas.org/doi/pdf/10.1073/pnas.61.1.25`

[6] Norris, J. (1997). Markov Chains (Cambridge Series in Statistical and Probabilistic Mathematics). Cambridge: Cambridge University Press. DOI:`https://doi.org/10.1017/CBO9780511810633`. URL: `https://www.statslab.cam.ac.uk/~james/Markov/`

[7] Winkel M. (2011). *Part A Simulation*, Lecture Notes, University of Oxford, Trinity Term 2011. URL: `https://www.stats.ox.ac.uk/~winkel/ASim11.pdf`

[8] Davies, R. (2020). *Part A Simulation and Statistical Programming*, Slides for Lectures 1,2,3,4,6,7, University of Oxford, Hilary Term 2020. URL: `https://www.stats.ox.ac.uk/~rdavies/teaching/PartASSP/2020/index.htm`

[9] Barkley, D. (2022). *MA124 Maths by Computer*, Jupiter Notebook for Week 6: Introduction to Monte Carlo, University of Warwick, Spring Term 2022.

[10] Box, G., & Muller, M. (1958). *A Note on the Generation of Random Normal Deviates.* Annals of Mathematical Statistics, 29, 610-611.