# Scientific Computing: Monte Carlo Methods

**Aleksandar Donev**
*Courant Institute, NYU*[1]
*donev@courant.nyu.edu*

[1]Course MATH-GA.2043 or CSCI-GA.2112, Fall 2015

Nov 12th, 2015

## Outline

## What is Monte Carlo?

- Monte Carlo is any numerical algorithm that uses random numbers to compute a deterministic (non-random) answer: **stochastic** or **randomized algorithm**.
- An important example is **numerical integration in higher dimensions**:
$$J = \int_{\Omega \subseteq \mathbb{R}^n} f(\mathbf{x})\, d\mathbf{x}$$
- Recall that using a **deterministic method** is very accurate and fast for low dimensions.
- But for large dimensions we have to deal with the **curse of dimensionality**:
The number of quadrature nodes scales like at least $2^n$ (exponentially). E.g., $2^{20} = 10^6$, but $2^{40} = 10^{12}$!

## Probability Theory

- First define a set $\Omega$ of possible **outcomes** $\omega \in \Omega$ of an "experiment":
  - A coin toss can end in heads or tails, so two outcomes.
  - A sequence of four coin tosses can end in one of $4^2 = 16$ outcomes, e.g., HHTT or THTH.
- The set $\Omega$ can be finite (heads or tails), countably infinite (the number of atoms inside a box), or uncountable (the weight of a person).
- An **event** $A \subseteq \Omega$ is a **set of possible outcomes**: e.g., more tails then heads occur in a sequence of four coin tosses,

$$A = \{HHHH, THHH, HTHH, HHTH, HHHT\}.$$

- Each event has an associated **probability**

$$0 \leq P(A) \leq 1,$$

with $P(\Omega) = 1$ and $P(\emptyset) = 0$.

## Conditional Probability

- A basic axiom is that probability is **additive** for disjoint events:

$$P(A \cup B) = P(A \text{ or } B) = P(A) + P(B) \;\text{ if } A \cap B = \emptyset$$

- Bayes formula gives the **conditional probability** that an outcome belongs to set $B$ if it belongs to set $C$:

$$P(B|C) = \frac{P(B \cap C)}{P(C)} = \frac{P(B \text{ and } C)}{P(C)}$$

- Two events are said to be **independent** if their probabilities are multiplicative:

$$P(A \cap B) = P(A \text{ and } B) = P(A)\,P(B)$$

## Probability Distribution

- If $\Omega$ is uncountable, think of outcomes as **random variables**, that is, variables whose value is determined by a random outcome:

$$X = X(\omega) \in \mathbb{R}.$$

- The **probability density function** $f(x) \geq 0$ determines the probability for the outcome to be close to $x$, in one dimension

$$P\left(x \leq X \leq x + dx\right) = f(x)dx,$$

$$P(A) = P(X \in A) = \int_{x \in A} f(x)dx$$

- The concept of a **measure** and the **Lebesque integral** generalizes the traditional Riemann integral in probability theory.

## Mean and Variance

- We call the **probability density** or the **probability measure** the **law** or the **distribution** of a random variable $X$, and write:
$$X \sim f.$$
- The **cummulative distribution function** is
$$F(x) = P(X \leq x) = \int_{-\infty}^{x} f(x')dx',$$
and we will assume that this function is continuous.
- The **mean** or **expectation value** of a random variable $X$ is
$$\mu = \bar{X} = E[X] = \int_{-\infty}^{\infty} xf(x)dx.$$
- The **variance** $\sigma^2$ and the **standard deviation** $\sigma$ measure the **uncertainty** in a random variable
$$\sigma^2 = \text{var}(X) = E[(X-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 f(x)dx.$$

## Multiple Random Variables

- Consider a set of two random variables $Z = (X, Y)$ and the **joint probability distribution** $Z \sim f(x, y)$.
- The **marginal density** for $X$ is the distribution of just $X$, without regard to $Y$:

$$g(x) = \int_y f(x, y)dy, \text{ similarly } h(y) = \int_x f(x, y)dx$$

- The **conditional probability distribution** is the distribution of $X$ for a known $Y$:

$$f(x|y) = \frac{f(x, y)}{h(y)}$$

- Two random variables $X$ and $Y$ are **independent** if

$$f(x, y) = g(x)h(y) \quad \Rightarrow f(x|y) = g(x).$$

## Covariance

- The term **i.i.d.**$\equiv$**independent identically-distributed** random variables is used to describe independent **samples** $X_k \sim f,\ k = 1, \dots$.
- The generalization of variance for two variables is the **covariance**:

$$C_{XY} = \text{cov}(X, Y) = E\left[\left(X - \bar{X}\right)\left(Y - \bar{Y}\right)\right] = E\left(XY\right) - E(X)E(Y).$$

- For independent variables

$$E\left(XY\right) = \int xy\, f(x, y) dxdy = \int xg(x)dx \int yh(y)dy = E(X)E(Y)$$

  and so $C_{XY} = 0$.
- Define the **correlation coefficient** between $X$ and $Y$ as a measure of how correlated two variables are:

$$r_{XY} = \frac{\text{cov}(X, Y)}{\sqrt{\text{var}(X)\text{var}(Y)}} = \frac{C_{XY}}{\sigma_X \sigma_Y}.$$

## Law of Large Numbers

- The average of $N$ i.i.d. samples of a random variable $X \sim f$ is itself a random variable:

$$A = \frac{1}{N} \sum_{k=1}^{N} X_k.$$

- $A$ is an **unbiased estimator** of the mean of $X$, $E(A) = \bar{X}$.
- Numerically we often use a **biased estimate** of the variance:

$$\sigma_X^2 = \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} \left(X_k - \bar{X}\right)^2 \approx \frac{1}{N} \sum_{k=1}^{N} \left(X_k - A\right)^2 = \left(\frac{1}{N} \sum_{k=1}^{N} X_k^2\right) - A^2$$

- The weak **law of large numbers** states that the estimator is also **consistent**:

$$\lim_{N\to\infty} A = \bar{X} = E(X) \text{ (almost surely).}$$

## Central Limit Theorem

- Imprecisely (and not completely accurately), the **central value theorem** says that if $\sigma_X$ is finite, in the limit $N \to \infty$ the random variable $A$ is **normally-distributed**:

$$A \sim f(a) = \left(2\pi\sigma_A^2\right)^{-1/2} \exp\left[-\frac{(a-\bar{X})^2}{2\sigma_A^2}\right]$$

- The **error of the estimator** $A$ decreases as $N^{-1}$, more specifically,

$$E\left[\left(A-\bar{X}\right)^2\right] = E\left\{\left[\frac{1}{N}\sum_{k=1}^{N}\left(X_k-\bar{X}\right)\right]^2\right\} = \frac{1}{N^2}E\left[\sum_{k=1}^{N}\left(X_k-\bar{X}\right)^2\right]$$

$$\text{var}(A) = \sigma_A^2 = \frac{\sigma_X^2}{N}.$$

- The slow convergence of the error, $\sigma \sim N^{-1/2}$, is a fundamental characteristic of Monte Carlo.

## Monte Carlo on a Computer

- In order to compute integrals using Monte Carlo on a computer, we need to be able to generate samples from a distribution, e.g., uniformly distributed inside an interval $I = [a, b]$.
- Almost all randomized software is based on having a **pseudo-random number generator** (PRNG), which is a routine that returns a pseudo-random number $0 \leq u \leq 1$ from the **standard uniform distribution:**

$$f(u) = \begin{cases} 1 & \textbf{if } 0 \leq u \leq 1 \\ 0 & \textbf{otherwise} \end{cases}$$

- Since computers (Turing machines) are deterministic, it is not possible to generate truly random samples (outcomes): Pseudo-random means **as close to random as we can get it**.
- There are well-known good PRNGs that are also efficient: One should **use other-people's PRNGs**, e.g., the **Marsenne Twister**.

## PRNGs

- The PRNG is a procedure (function) that takes a collection of $m$ integers called the **state of the generator** $\mathbf{s} = \{i_1, \ldots, i_m\}$, and updates it:

$$\mathbf{s} \leftarrow \Phi(\mathbf{s}),$$

  and produces (returns) a number $u = \Psi(\mathbf{s})$ that is a pseudo-random sample from the standard uniform distribution.
- So in pseudo-MATLAB notation, $[u, \mathbf{s}] = rng(\mathbf{s})$, often called a **random stream**.
- Simple built-in generator such as the MATLAB/C function *rand* or the Fortran function *RANDOM_NUMBER* hide the state from the user (but the state is stored somewhere in some global variable).
- All PRNGs provide a routine to **seed the generator**, that is, to set the seed **s** to some particular value.
  This way one can generate the same sequence of "random" numbers over and over again (e.g., when debugging a program).

## Generating Non-Uniform Variates

- Using a uniform (pseudo-)random number generator (**URNG**), it is easy to generate an outcome drawn uniformly in $I = [a, b]$:

$$X = a + (b - a)U,$$

  where $U = rng()$ is a standard uniform variate.
- We often need to generate **(pseudo)random samples** or **variates** drawn from a distribution $f(x)$ other than a uniform distribution, where $f(x) \geq 0$ and $f(x)$ is normalized, $\int f(x)dx = 1$.
- Almost all **non-uniform samplers are based on a URNG**.
- Sometimes it may be more efficient to replace the URNG with a **random bitstream**, that is, a sequence of random bits, if only a few random bits are needed (e.g., for discrete variables).
- We need a method to convert a uniform variate into a non-uniform variate.

## Generating Non-Uniform Variates

- Task: We want to sample a random number with **probability distribution** $f(x)$. For now assume $f(x)$ is a **probability density**:

$$P(x \leq X \leq x + dx) = f(x)dx,$$

- Tool: We can generate samples from some special distributions, e.g., a sample $U$ from the standard uniform distribution.
- Consider applying a non-linear **differentiable one-to-one** function $g(x)$ to $U$:

$$X \equiv X(U) = g(U) \quad \Rightarrow \quad dx = g'(U)du$$

- We can find the probability density of $X$ by using the informal differential notation

$$P(u \leq U \leq u + du) = du = \frac{dx}{g'(u)} = P(x \leq X \leq x + dx) = f(x)dx$$

$$f[x(u)] = \left[g'(u)\right]^{-1}$$

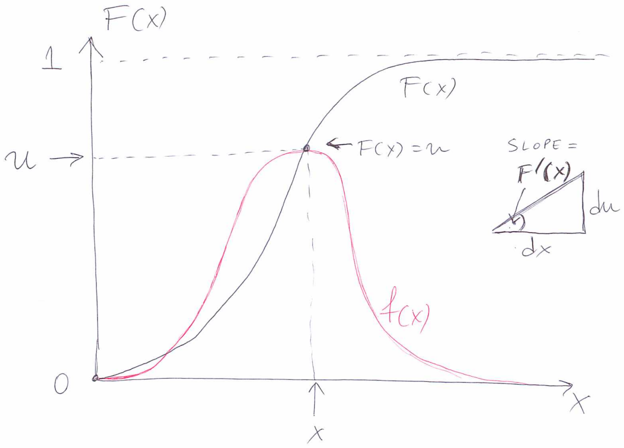
## Inverting the CDF

$$f\left[x(u)\right] = \left[g'(u)\right]^{-1}$$

- Can we find $g(u)$ given the target $f(x)$? It is simpler to see this if we invert $x(u)$:
$$u = g^{-1}(x) = F(x).$$
- Repeating the same calculation
$$P\left(u \leq U \leq u + dx\right) = du = F'(x)dx = f(x)dx$$
$$F'(x) = f(x)$$
- This shows that $F(x) = g^{-1}(x)$ is the **cummulative probability distribution**:
$$F(x) = P(X \leq x) = \int_{-\infty}^{x} f(x')dx'.$$
- Note that $F(x)$ is monotonically non-decreasing because $f(x) \geq 0$.

## Sampling by Inversion

**Inversion algorithm**: Generate a standard uniform variate $u$ and then solve the **non-linear equation** $F(x) = u$ to get $x$.



## Exponentially-Distributed Number

- As an example, consider generating a sample from the **exponential distribution with rate** $\lambda$:

$$f_\lambda(t) = \begin{cases} \lambda e^{-\lambda t} & \text{if } t \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

- Related to the **Poisson process** of events whose rate of occurence is $\lambda$ and whose occurence does not depend on the past (history):

$$P(t \leq T \leq t + dt \mid T \geq t) = P(T < dt) = \lambda dt.$$

- Using the **inversion technique** we get

$$F(t) = P(T \leq t) = \int_{t'=0}^{t} \lambda e^{-\lambda t} dt = 1 - e^{-\lambda t} = u' \equiv 1 - u$$

$$T = -\lambda^{-1} \ln(U),$$

where numerical care must be taken to ensure the log does not overflow or underflow.

## Normally-Distributed Numbers

- The **standard normal distribution** is a Gaussian “bell-curve”:

$$f(x) = \left(2\pi\sigma^2\right)^{-1/2} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right),$$

  where $\mu$ is the **mean** and $\sigma$ is the **standard deviation**.
- The **standard normal distribution** has $\sigma = 1$ and $\mu = 0$.
- If we have a sample $X_s$ from the standard distribution we can generate a sample $X$ from $f(x)$ using:

$$X = \mu + \sigma X_s$$

- There are many specialized and optimized samplers for normally-distributed numbers, *randn* in MATLAB.

## Histogram Validation

- We need some way to test that a sampler is correct, that is, that the generated sequence of random numbers really comes from the specified distribution $f(x)$. One easy way to do that is by computing the **histogram** of the samples.
- Count how many $N_x$ samples of the $N$ samples are inside a **bin of width** $h$ centered at $x$:

$$f(x) \approx P_x = \frac{1}{h} P(x - h/2 \leq X \leq x + h/2) \approx \frac{N_x}{hN}.$$

- If we make the bins smaller, the **truncation error** will be reduced:

$$P_x - f(x) = \frac{1}{h} \int_{x-h/2}^{x+h/2} f(x')dx' - f(x) = \alpha h^2 + O(h^4)$$

- But, this means there will be fewer points per bin, i.e., **statistical errors** will grow. As usual, we want to find the optimal tradeoff between the the two types of error.

## Statistical Error in Histogramming

- For every sample point $X$, define the **indicator** random variable $Y$:

$$Y = \mathbb{I}_x(X) = \begin{cases} 1 & \text{if } x - h/2 \le X \le x + h/2 \\ 0 & \text{otherwise} \end{cases}$$

- The mean and variance of this **Bernoulli random variable** are:

$$E(Y) = \bar{Y} = hP_x \approx hf(x)$$

$$\sigma_Y^2 = \int (y - \bar{Y})^2 f(y) dy = \bar{Y} \cdot (1 - \bar{Y}) \approx \bar{Y} \approx hf(x)$$

- The number $N_x$ out of $N$ trials inside the bin is a sum of $N$ random Bernoulli variables $Y_i$:

$$f(x) \approx \frac{1}{h} \frac{N_x}{N} = h^{-1} \left( \frac{1}{N} \sum_{i=1}^{N} Y_i \right) = \hat{P}_x$$

## Optimal Bin Width

- The central limit theorem gives us the uncertainty in our estimate of $f(x)$

$$\sigma\left(\hat{P}_x\right) \approx h^{-1}\frac{\sigma_Y}{\sqrt{N}} = \sqrt{\frac{f(x)}{hN}} = \frac{\sqrt{N_x}}{hN}.$$

- This means that the **empirical distribution** $f(x)$ should be reported with a 95% **confidence interval**,

$$P\left\{f(x) \in \left[\frac{N_x - 2\sqrt{N_x}}{hN}, \frac{N_x + 2\sqrt{N_x}}{hN}\right]\right\} \approx 95\%.$$

- The optimal bin width is when the truncation and statistical errors are equal:

$$\alpha h^2 \approx \sqrt{\frac{f(x)}{hN}} \quad \Rightarrow \quad h \sim N^{-1/5},$$

  with total error $\varepsilon \sim (hN)^{-1/2} \sim N^{-2/5}$.

- Typically we choose $h$ based on how well we want to resolve $f(x)$, and accept the fact that **statistical errors dominate**.

## Integration via Monte Carlo

- Define the random variable $Y = f(\mathbf{X})$, and generate a sequence of $N$ **independent uniform samples** $\mathbf{X}_k \in \Omega$, i.e., $N$ random variables distributed uniformly inside $\Omega$:

$$\mathbf{X} \sim g(\mathbf{x}) = \begin{cases} |\Omega|^{-1} & \text{for } \mathbf{x} \in \Omega \\ 0 & \text{otherwise} \end{cases}$$

  and calculate the mean

$$\hat{Y} = \frac{1}{N} \sum_{k=1}^{N} Y_k = \frac{1}{N} \sum_{k=1}^{N} f(\mathbf{X}_k)$$

- According to the weak law of large numbers,

$$\lim_{N \to \infty} \hat{Y} = E(Y) = \bar{Y} = \int f(\mathbf{x}) g(\mathbf{x}) dx = |\Omega|^{-1} \int_{\Omega} f(\mathbf{x})\, d\mathbf{x}$$

## Accuracy of Monte Carlo Integration

- This gives a Monte Carlo approximation to the integral:

$$J = \int_{\Omega \in \mathbb{R}^n} f(\mathbf{x})\, d\mathbf{x} = |\Omega|\, \bar{Y} \approx |\Omega|\, \hat{Y} = |\Omega|\, \frac{1}{N} \sum_{k=1}^{N} f\left(\mathbf{X}_k\right).$$

- Recalling the central limit theorem, for large $N$ we get an **error estimate** by evaluating the standard deviation of the estimate $\hat{Y}$:

$$\sigma^2\left(\hat{Y}\right) \approx \frac{\sigma_Y^2}{N} = N^{-1} \int_{\Omega} \left[f(\mathbf{x}) - |\Omega|^{-1} J\right]^2 d\mathbf{x}$$

$$\sigma\left(\hat{Y}\right) \approx \frac{1}{\sqrt{N}} \left[\int_{\Omega} \left[f(\mathbf{x}) - \overline{f(\mathbf{x})}\right]^2 d\mathbf{x}\right]^{1/2}$$

- Note that this error goes like $N^{-1/2}$, which is order of convergence $1/2$: **Worse than any deterministic quadrature**.
- But, the same number of points are needed to get a certain accuracy **independent of the dimension**.

## Monte Carlo Error Bars

- **Monte Carlo (MC) answers should always be reported with error bars**, or equivalently, with confidence intervals!
- Since the answer is approximately normally-distributed, we have the well-known **confidence intervals**:

$$P\left(\frac{J}{|\Omega|}\in\left[\hat{Y}-\sigma,\hat{Y}+\sigma\right]\right)\approx 66\%$$

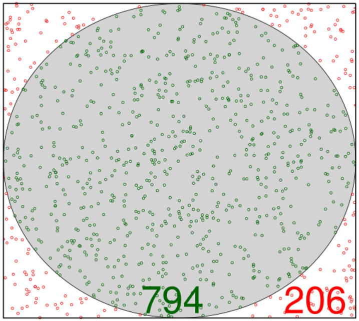
$$P\left(\frac{J}{|\Omega|}\in\left[\hat{Y}-2\sigma,\hat{Y}+2\sigma\right]\right)\approx 95\%$$

- In practice we estimate the uncertainty **empirically** as

$$\sigma^2\left(\hat{Y}\right)\approx\frac{1}{N^2}\sum\left(Y_i-\overline{Y}\right)^2=\frac{1}{N}\left[\left(\frac{1}{N}\sum Y_i^2\right)-\left(\frac{1}{N}\sum Y_i\right)^2\right].$$

- This is done in a **single MC loop**: Average the $Y$'s to get the answer but also average the squares $Y^2$ to get the uncertainty in the answer.

## Employing Rejection



Note how this becomes **less efficient as dimension grows** (most points are outside the sphere).

- Integration requires $|\Omega|$, which is hard to compute for complicated domains,

$$\int_{\Omega \in \mathbb{R}^n} f(\mathbf{x})\, d\mathbf{x} \approx |\Omega| \frac{1}{N} \sum_{k=1}^{N} f(\mathbf{X}_k)$$

- Consider $\Omega$ being the unit circle of radius 1.
- Rejection: Integrate by sampling points inside an **enclosing region**, e.g, a square of area $|\Omega_{encl}| = 4$, and rejecting any points outside of $\Omega$:

$$\int_{\Omega \in \mathbb{R}^n} f(\mathbf{x})\, d\mathbf{x} \approx |\Omega_{encl}| \frac{1}{N} \sum_{\mathbf{X}_k \in \Omega} f(\mathbf{X}_k)$$

## (Importance) Sampling Function

- In the basic MC algorithm described above, the samples $\mathbf{X}$ have a uniform distribution over the integration domain.
- Instead, we can sample our points from some probability distribution function $g(\mathbf{X}) \geq 0$, $\int g(\mathbf{x})d\mathbf{x} = 1$, and rewrite:

$$\int f(\mathbf{x})\, d\mathbf{x} = \int \frac{f(\mathbf{x})}{g(\mathbf{x})} g(\mathbf{x})d\mathbf{x} = E\left[\frac{f(\mathbf{X})}{g(\mathbf{X})}\right] \text{ where } \mathbf{X} \sim g.$$

- This now corresponds to taking **samples from the sampling function** $g(\mathbf{x})$:

$$\int f(\mathbf{x})\, d\mathbf{x} \approx \frac{1}{N}\sum_{k=1}^{N} \frac{f(\mathbf{X}_k)}{g(\mathbf{X}_k)} \text{ where } \mathbf{X} \sim g$$

- Note that $|\Omega|$ does not appear since it is implicitly included in the normalization of $g(\mathbf{x})$. The previous uniform sampling algorithm corresponds to $g(\mathbf{x}) = |\Omega|^{-1}$ for $\mathbf{x} \in \Omega$.

## Variance Reduction

- The order of convergence (accuracy) of Monte Carlo is always $1/2$ and cannot be improved. Instead, all of the focus is on improving the error constant, i.e., the **variance** for a constant number of samples $N$.
- The most important thing in Monte Carlo is **variance reduction**, i.e., finding methods that give the same answers in the limit $N \to \infty$ but have a much smaller $\sigma$.
- There are several methods for variance reduction, the most general and powerful of which is **importance sampling**.
- Importance sampling simply means choosing the sampling function $g(x)$ to give more importance to those points that dominate the value of the integral. We call $g(x)$ an **importance sampling function**.

## Importance Sampling

- Repeating the variance calculation for

$$Y(\mathbf{X}) = \frac{f(\mathbf{X})}{g(\mathbf{X})}, \text{ with mean } \overline{Y} = \int f(\mathbf{x}) d\mathbf{x}$$

- The variance of the emprical mean $\hat{Y} = N^{-1} \sum Y_i$ is

$$\sigma^2\left(\hat{Y}\right) \approx \frac{\sigma_Y^2}{N} = N^{-1} \int \left[Y(\mathbf{x}) - \overline{Y}\right]^2 g(\mathbf{x}) d\mathbf{x}$$

$$\sigma\left(\hat{Y}\right) \approx \frac{1}{\sqrt{N}} \left[\int \left[\frac{f(\mathbf{x})}{g(\mathbf{x})} - \overline{Y}\right]^2 g(\mathbf{x}) d\mathbf{x}\right]^{1/2}.$$

## The Importance Function

- We therefore want $f(\mathbf{x})/g(\mathbf{x}) = \overline{Y}$ to be as close as possible to a constant, **ideally**

$$g_{ideal}(\mathbf{x}) = \frac{f(\mathbf{x})}{\int f(\mathbf{x})dx}$$

  but this requires being able to create independent samples from $f(\mathbf{x})$, which is harder than the problem of integrating $f(\mathbf{x})$!

- The importance sampling function $g(\mathbf{x})$ must be a probability distribution function that we know how to sample from, such that

$$h(\mathbf{x}) = \frac{f(\mathbf{x})}{g(\mathbf{x})}$$

  is **as close to constant as possible**, and in particular, it must be **bounded** from above (i.e., finite for all $\mathbf{x}$ in the relevant domain).

- Choosing the right $g$ given $f$ is an art form; an example is in the homework.

## Conclusions/Summary

- Monte Carlo is an umbrella term for **stochastic computation** of deterministic answers.
- Monte Carlo answers are random, and their accuracy is measured by the **variance** or uncertaintly of the estimate, which typically scales like $\sigma \sim N^{-1/2}$, where $N$ is the number of **samples**.
- Implementing Monte Carlo algorithms on a computer requires a PRNG, almost always a **uniform pseudo-random number generator** (URNG).
- One often needs to convert a sample from a URNG to a sample from an arbitrary distribution $f(x)$, including inverting the cummulative distribution and rejection sampling.
- Monte Carlo can be used to perform **integration in high dimensions** by simply evaluating the function at random points.