Monte Carlo Methods Lecture Notes

Apurva Nakade

1 Welcome

Preface

Note: These notes are very much under construction and subject to significant changes.

These are the lecture notes for *Monte Carlo Methods* (EN.553.433). The notes are loosely based on course materials by Dr. Jim Spall and the textbook by Rubinstein & Kroese (Rubinstein and Kroese 2017).

These notes represent only half of the course content. The remaining material consists of Jupyter notebooks assigned as homework, which are not published online.

Thanks to Kyle Beaty for help with cleaning these notes.

2 Introduction to Monte Carlo Methods

Monte Carlo methods are a powerful class of computational algorithms that harness the power of random sampling to solve complex numerical problems. Named after the famous Monte Carlo Casino in Monaco, these methods transform deterministic problems into probabilistic ones, allowing us to approximate solutions through statistical simulation.

2.1 What are Monte Carlo Methods?

At their core, Monte Carlo methods rely on **repeated random sampling** to obtain numerical results for problems that might be difficult or impossible to solve analytically. By generating large numbers of random samples and analyzing their statistical properties, we can approximate solutions with quantifiable uncertainty.

2.2 Key Application Areas

Monte Carlo methods excel in three primary problem domains:

> **i** Core Applications
>
> - **Numerical integration and estimation** — Computing integrals, expected values, and other mathematical quantities
> - **Physical and mathematical system simulation** — Modeling complex systems with inherent randomness
> - **Optimization** — Finding optimal solutions in high-dimensional or complex parameter spaces

2.3 The Monte Carlo Workflow

A typical Monte Carlo analysis follows this systematic approach:

1. **Model formulation**: Develop a mathematical representation of the real-world system, explicitly incorporating relevant random variables

2. **Distribution specification**: Determine the probability distributions governing the random variables, using available data, theoretical knowledge, or expert judgment
3. **Simulation**: Generate representative samples from the specified distributions and execute computational experiments
4. **Validation and refinement**: Compare simulation results with empirical observations to assess model adequacy and guide iterative improvements

2.4 Why Monte Carlo Methods Matter

These methods are particularly valuable when dealing with:

- High-dimensional problems where traditional numerical methods become computationally prohibitive
- Systems with complex, nonlinear relationships
- Problems involving uncertainty quantification
- Situations where analytical solutions are intractable

Throughout these notes, we'll explore how Monte Carlo methods provide both theoretical insights and practical solutions across diverse fields, from finance and physics to machine learning and engineering.

Part I

Monte Carlo Estimation

One of the most fundamental applications of Monte Carlo methods is **estimation** — using random sampling to approximate unknown quantities. This approach is remarkably versatile, applying equally well to inherently random phenomena and deterministic mathematical problems.

Types of Estimation Problems

Monte Carlo estimation addresses two distinct categories of problems:

Stochastic quantities: Values that are naturally random or uncertain

- Future stock prices
- Weather predictions

- System reliability measures
- Risk assessments

Deterministic quantities: Fixed mathematical values that are difficult to compute directly

- The value of π
- Complex integrals
- Solutions to differential equations
- High-dimensional optimization problems

The Monte Carlo Estimation Strategy

The key insight behind Monte Carlo estimation is transforming any estimation problem into a probabilistic framework, even when the original problem contains no randomness.

! Core Monte Carlo Principle

To estimate any quantity , we:

1. **Construct** a random variable X such that $\mathbb{E}[X] = \theta$
2. **Generate** independent samples $x_1, x_2, \ldots, x_n$ from the distribution of X

3. **Estimate** using the sample average:

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

In fact, we do not even need $\mathbb{E}[X] = \theta$. It suffices that $\lim_{n\to\infty} \hat{\theta} = \theta$ with high probability. For details, see **?@sec-estimation-theory**.

Why This Works

This approach leverages the **Law of Large Numbers**: as the sample size n increases, our estimate $\hat{\theta}$ converges to the true expected value θ. The beauty lies in converting complex analytical problems into straightforward sampling and averaging procedures.

What's Next

In the following sections, we'll see:

- Concrete examples demonstrating this estimation framework
- How to construct appropriate random variables X for different problems

- Methods for assessing and improving estimation accuracy
- Advanced sampling techniques for enhanced efficiency

The examples will illustrate how this simple yet powerful principle enables us to tackle problems that would otherwise require sophisticated analytical techniques or be computationally intractable.

3 Estimating π

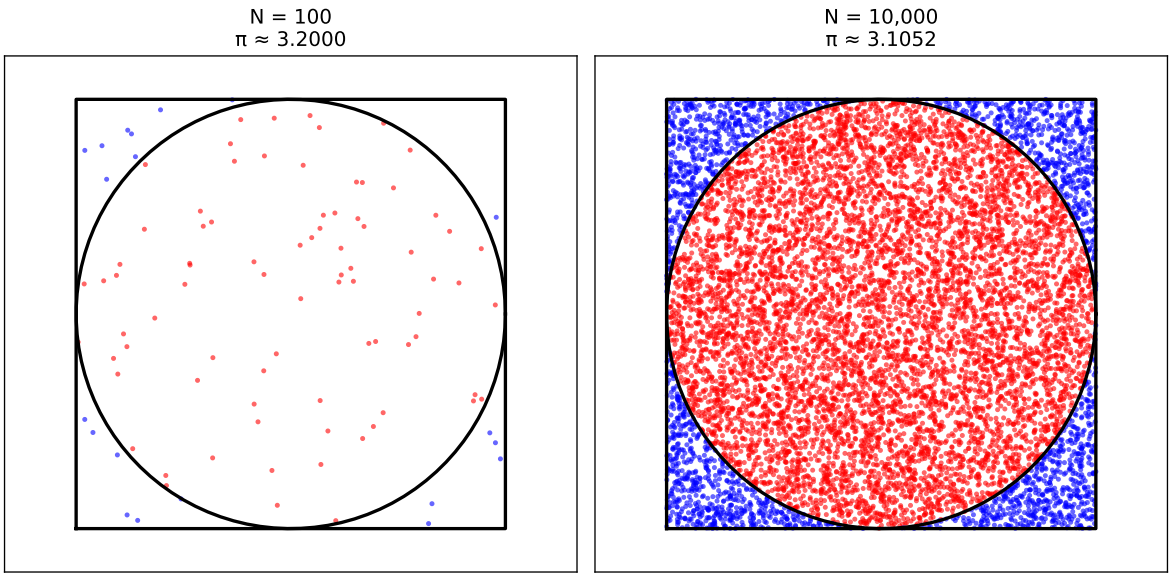


Figure 3.1: Estimating using Monte Carlo method with different sample sizes

Monte Carlo Estimation Results

Sample Size	Points Inside	Estimate	Absolute Error	Standard Error
100	80	3.2000	0.0584	0.1600
10,000	7,763	3.1052	0.0364	0.0167

We begin with a classic Monte Carlo application: estimating the value of π using geometric probability. This example illustrates the core principles of Monte Carlo simulation while providing an intuitive geometric interpretation.

Consider a unit circle inscribed in a square with side length 2, both centered at the origin. The circle has radius $r = 1$ and area $A_{\text{circle}} = \pi$, while the square has area $A_{\text{square}} = 4$. The ratio of these areas is $\pi/4$.

Geometric Relationships

- **Circle**: radius $r = 1$, area $A_{\text{circle}} = \pi r^2 = \pi$
- **Square**: side length $2r = 2$, area $A_{\text{square}} = (2r)^2 = 4$
- **Area ratio**: $\frac{A_{\text{circle}}}{A_{\text{square}}} = \frac{\pi}{4}$

If we randomly sample points uniformly within the square, the probability that any point falls inside the inscribed circle equals this area ratio. This geometric probability provides our pathway to estimating π.

To formulate this as a Monte Carlo problem, let (X, Y) be uniformly distributed on $[-1, 1] \times [-1, 1]$ and define the indicator random variable:

$$I(X, Y) = \mathbf{1}_{\{X^2+Y^2 \leq 1\}} = \begin{cases} 1 & \text{if } X^2 + Y^2 \leq 1 \text{ (inside circle)} \\ 0 & \text{if } X^2 + Y^2 > 1 \text{ (outside circle)} \end{cases}$$

The expected value of this indicator function gives us the desired probability:

$$\mathbb{E}[I(X, Y)] = P(X^2 + Y^2 \leq 1) = \frac{\text{Area of unit circle}}{\text{Area of square}} = \frac{\pi}{4}$$

Therefore $\pi = 4\mathbb{E}[I(X, Y)]$, and given N independent samples $(X_1, Y_1), \ldots, (X_N, Y_N)$, our Monte Carlo estimator is:

$$\hat{\pi}_N = 4 \cdot \frac{1}{N} \sum_{i=1}^{N} I(X_i, Y_i) = \frac{4 \cdot (\text{number of points inside circle})}{N}$$

3.1 Statistical Properties

This estimator possesses several important statistical properties. First, it is unbiased:

$$\mathbb{E}[\hat{\pi}_N] = 4\mathbb{E}\left[\frac{1}{N} \sum_{i=1}^{N} I_i\right] = 4\mathbb{E}[I] = 4 \cdot \frac{\pi}{4} = \pi$$

Since $I \sim \text{Bernoulli}(\pi/4)$, we can compute the variance. The indicator has variance $\text{Var}(I) = \frac{\pi}{4}(1 - \frac{\pi}{4}) = \frac{\pi(4-\pi)}{16}$, which gives our estimator variance:

$$\text{Var}(\hat{\pi}_N) = 16 \cdot \frac{\text{Var}(I)}{N} = \frac{\pi(4 - \pi)}{N}$$

The standard error is therefore $\text{SE}(\hat{\pi}_N) = \sqrt{\frac{\pi(4-\pi)}{N}} \approx \frac{1.64}{\sqrt{N}}$.

Convergence Rate

The standard error decreases as $O(1/\sqrt{N})$, which is the typical Monte Carlo convergence rate. To gain one decimal place of accuracy, we need approximately 100 times more samples.

By the Central Limit Theorem, for large N we have the asymptotic distribution:

$$\sqrt{N}(\hat{\pi}_N - \pi) \xrightarrow{d} \mathcal{N}(0, \pi(4-\pi))$$

This provides approximate confidence intervals:

$$\hat{\pi}_N \pm z_{\alpha/2}\sqrt{\frac{\pi(4-\pi)}{N}}$$

Monte Carlo Principle

This example demonstrates the fundamental Monte Carlo approach: reformulate a deterministic problem (computing π) as the expectation of a random variable, then estimate that expectation using sample averages.

4 Estimating Integrals

Monte Carlo integration transforms the problem of computing definite integrals into a sampling problem, making it particularly powerful for high-dimensional integration where traditional methods fail.

Consider the problem of estimating the integral $\ell = \int_a^b f(x)\,dx$. The key insight is to rewrite this integral as an expectation. If $X \sim \text{Uniform}(a, b)$, then:

$$\int_a^b f(x)\,dx = (b-a)\mathbb{E}[f(X)]$$

This reformulation allows us to estimate the integral using sample averages. Given N independent samples $X_1, \ldots, X_N \sim \text{Uniform}(a, b)$, our Monte Carlo estimator is:

$$\hat{\ell}_N = \frac{b-a}{N}\sum_{i=1}^{N} f(X_i)$$

> **Geometric Interpretation**
>
> We're approximating the area under $f(x)$ by averaging function values at random points and scaling by interval width.

4.1 Statistical Properties

The Monte Carlo integration estimator possesses several important statistical properties. First, it is unbiased:

$$\mathbb{E}[\hat{\ell}_N] = \frac{b-a}{N}\sum_{i=1}^{N}\mathbb{E}[f(X_i)] = \frac{b-a}{N}\sum_{i=1}^{N}\int_a^b f(x)\frac{1}{b-a}\,dx = \ell$$

The variance of our estimator is:

$$\text{Var}(\hat{\ell}_N) = \frac{(b-a)^2}{N}\cdot\sigma_f^2$$

where $\sigma_f^2 = \text{Var}(f(X))$ is the variance of $f(X)$ under the uniform distribution.

! Key Properties

- **Standard Error**: $\text{SE} = \frac{(b-a)\sigma_f}{\sqrt{N}}$
- **Convergence Rate**: $O(1/\sqrt{N})$ regardless of dimension
- **CLT**: $\sqrt{N}(\hat{\ell}_N - \ell) \xrightarrow{d} \mathcal{N}(0, (b-a)^2\sigma_f^2)$

4.2 Function Characteristics and Performance

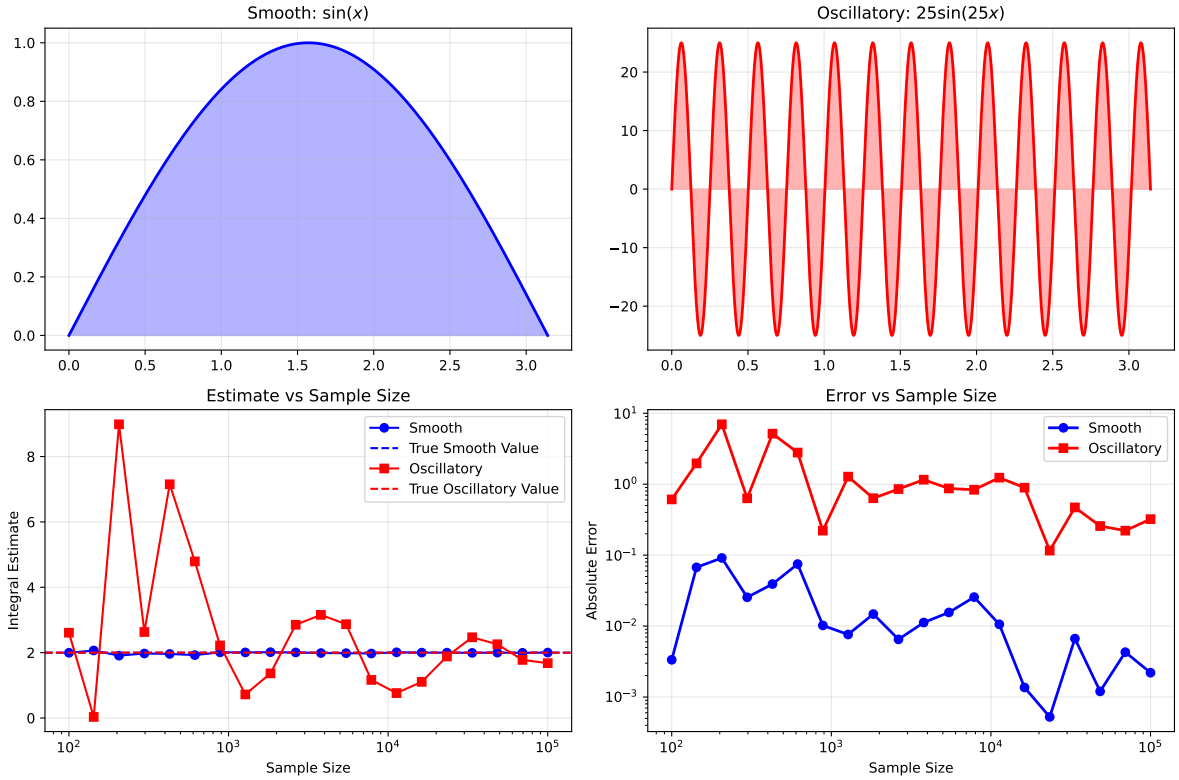
Monte Carlo integration's performance depends heavily on the function's characteristics. While the convergence rate is theoretically independent of function complexity, practical performance varies significantly with function smoothness and oscillatory behavior.

Functions with high-frequency oscillations present particular challenges for Monte Carlo methods because random sampling may miss important features or fail to adequately capture rapid variations. This example compares Monte Carlo integration performance on two trigonometric functions with dramatically different oscillation frequencies, illustrating how function characteristics affect integration accuracy and convergence behavior.

4.2.1 Example: Low vs. High Frequency Functions

The following example integrates two functions over the interval $[0, \pi]$:

- **Smooth decay**: $f_1(x) = \sin(x)$ — a smooth, slowly varying function
- **Oscillatory**: $f_2(x) = 25\sin(25x)$ — a rapidly oscillating function with many periods



	Function	True Value	Mean Est	Error	Std Dev	Std Ratio
0	Smooth	2.0	2.0008	0.0008	0.0086	-
1	Oscillatory	2.0	1.9919	0.0081	0.6073	71.006457

The contrast between the smooth $\sin(x)$ and rapidly oscillating $25\sin(25x)$ demonstrates why integration difficulty varies. The high-frequency function has many more sign changes and local extrema that random sampling must capture.

The log-log error plot reveals that while both functions follow the expected $O(1/\sqrt{n})$ convergence rate, the high-frequency function consistently maintains higher absolute errors across all sample sizes. The variance ratio indicates that the oscillatory function's variance is around 70 times higher than the smooth function's variance, illustrating how oscillations increase uncertainty in Monte Carlo estimates.

4.2.2 Remarks

- **Smooth functions** are well-suited for Monte Carlo integration, converging quickly with relatively few samples

- **Highly oscillatory functions** present challenges, requiring larger sample sizes and exhibiting higher variance
- For oscillatory integrals, consider **alternative approaches** such as:
  - Stratified sampling to ensure adequate coverage of oscillation periods
  - Importance sampling with distributions that account for function behavior
  - Specialized quadrature methods designed for oscillatory integrals
  - Transformation techniques to reduce oscillations

This example demonstrates why understanding your function's characteristics is crucial for effective Monte Carlo integration.

4.3 Multidimensional Integration

One of Monte Carlo integration's greatest advantages becomes apparent in higher dimensions. While traditional numerical integration methods (like the trapezoidal rule or Simpson's rule) suffer from the "curse of dimensionality"—requiring exponentially more grid points as dimensions increase—Monte Carlo methods maintain their $O(1/\sqrt{n})$ convergence rate regardless of dimension.

In d dimensions, a grid-based method with m points per dimension requires m^d total evaluations. For even modest accuracy in high dimensions, this becomes computationally prohibitive. Monte Carlo integration, however, uses the same number of random samples regardless of dimension, making it the method of choice for high-dimensional problems in physics, finance, and machine learning.

4.3.1 Example: Gaussian-like Function in Multiple Dimensions

The following example demonstrates Monte Carlo integration of the function $f(\mathbf{x}) = e^{-\|\mathbf{x}\|^2}$ over the unit hypercube $[0, 1]^d$ for dimensions $d = 1, 2, 3, 4, 5$.

The example reveals several important properties of multidimensional integration:

1. **Dimension-Independent Convergence**: The relative error plots show that Monte Carlo integration maintains similar convergence behavior across all dimensions. While there may be slight variations in the error magnitudes, the overall convergence rate (slope on the log-log plot) remains consistent, demonstrating Monte Carlo's fundamental advantage over grid-based methods that suffer from the curse of dimensionality.

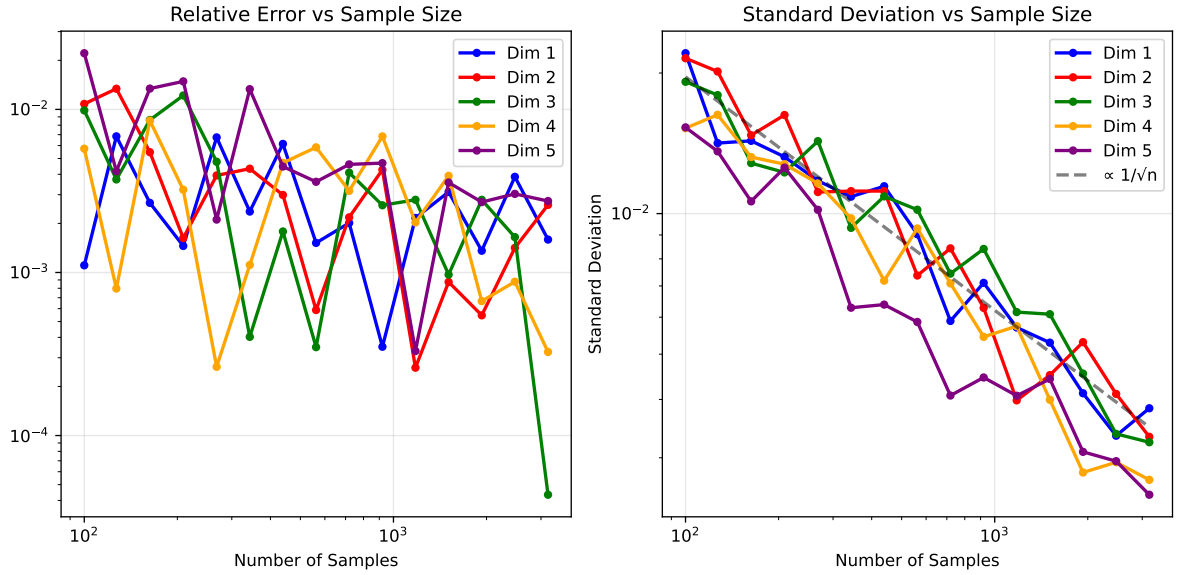


Figure 4.1: Monte Carlo advantage in higher dimensions

2. **Standard Deviation Scaling**: The standard deviation plot confirms the theoretical $1/\sqrt{n}$ convergence rate across all dimensions. The dashed reference line shows this theoretical scaling, and we can observe that all dimensions follow approximately the same convergence pattern, regardless of the integral's magnitude or the problem's dimensionality.

3. **Computational Efficiency**: Each dimension uses exactly the same number of function evaluations at each sample size, showcasing how Monte Carlo avoids the exponential scaling that plagues deterministic methods. While a grid-based approach would require n^d evaluations (where d is the dimension), Monte Carlo maintains constant computational cost per sample regardless of dimension.

4. **Robust Performance**: The similar behavior across dimensions demonstrates that Monte Carlo integration remains reliable and predictable even as the problem complexity increases, making it indispensable for high-dimensional problems in computational physics, Bayesian inference, and financial modeling.

This dimension-independent behavior makes Monte Carlo integration the method of choice when traditional quadrature rules become computationally infeasible in high-dimensional spaces.

When to Use Monte Carlo Integration

Advantages:

- **High dimensions**: Performance doesn't degrade with dimension
- **Complex domains**: Easy to handle irregular integration regions
- **Rough functions**: Works with discontinuous or highly oscillatory functions

Disadvantages:

- **Slow convergence**: $O(1/\sqrt{N})$ rate
- **Random error**: Introduces stochastic uncertainty
- **Low dimensions**: Often outperformed by deterministic methods

Practical Considerations:

- **Sample size**: Need large N for high precision
- **Function smoothness**: Smoother functions $\rightarrow$ lower variance $\rightarrow$ better estimates
- **Variance reduction**: Techniques like antithetic variables can improve efficiency
- **Error estimation**: Use sample variance to estimate uncertainty

4.4 Conclusion

Monte Carlo integration transforms integration into a sampling problem, making it invaluable for high-dimensional problems where traditional quadrature fails. While it converges slowly, its dimension-independent performance and ability to handle complex functions make it essential for modern computational statistics and machine learning applications.

Part II

Sampling Techniques

In this chapter, we will discuss how to sample from a probability distribution. Sampling from a probability distribution is a fundamental problem in statistics and machine learning. It is used in various applications like Monte Carlo methods, Bayesian inference, and reinforcement learning.

To understand what it means to sample from a probability distribution, let's consider a simple example. Let X be a discrete random variable that takes values $1, 2, \ldots, n$ with probabilities $p_1, p_2, \ldots, p_n$. To sample from this distribution, we want to generate a random variable X such that $\mathbb{P}(X = i) = p_i$ for all $i = 1, 2, \ldots, n$ i.e. we want to select a random integer i with probability p_i. If we generate enough samples $x_1, x_2, \ldots, x_N$ from this distribution, then the fraction of samples that are equal to i will be approximately equal to p_i for large N.

More formally, let $(\Omega, \mathcal{F}, P)$ be a probability space and let $X : \Omega \to \mathcal{X}$ be a random variable with distribution P_X. Sampling from the distribution of X means generating independent realizations $x_1, x_2, \ldots, x_n$ such that each x_i has the same distribution as X. By the strong law of large numbers, we have

$$\lim_{n\to\infty} \frac{1}{n} \sum_{j=1}^{n} f(x_j) = \mathbb{E}[f(X)]$$

almost surely for any measurable function f such that $\mathbb{E}[|f(X)|] < \infty$.

The fundamental challenge in sampling is transforming uniform random numbers, which are readily available from pseudorandom number generators, into samples from arbitrary probability distributions. Most computational environments provide uniform random number generators that produce sequences $U_1, U_2, \ldots$ where each U_i is approximately uniformly distributed on $[0, 1]$ and the sequence has good statistical properties.

For discrete distributions, if X takes values $\{x_1, x_2, \ldots, x_k\}$ with probabilities $\{p_1, p_2, \ldots, p_k\}$, then one approach is to partition the interval $[0, 1]$ into subintervals of lengths $p_1, p_2, \ldots, p_k$ and return x_i if the uniform random number falls in the i-th subinterval.

For continuous distributions with cumulative distribution function $F(x) = \mathbb{P}(X \leq x)$, the inverse transform method provides an elegant solution when F^{-1} exists and is computable. If $U \sim \text{Uniform}(0, 1)$, then $X = F^{-1}(U)$ has distribution function F. This follows because

$$\mathbb{P}(F^{-1}(U) \leq x) = \mathbb{P}(U \leq F(x)) = F(x)$$

for any x in the support of X.

When direct transformation methods are not available, more sophisticated techniques are required. Rejection sampling generates candidates from a proposal distribution and accepts them with a probability proportional to the target density. Markov chain Monte Carlo methods construct a Markov chain whose stationary distribution is the target distribution. Importance sampling generates samples from one distribution but weights them to approximate expectations under another distribution.

The quality of sampling algorithms is measured by several criteria. Correctness ensures that the generated samples actually follow the target distribution. Efficiency measures the computational cost per sample. For iterative methods like MCMC, convergence properties determine how quickly the algorithm approaches the target distribution. The choice of sampling method depends on the specific distribution, the required accuracy, and computational constraints.

In the following chapters, we will examine these sampling techniques in detail, providing both theoretical foundations and practical algorithms for implementation.

5 Random Number Generators

In the problem on estimating the value of π using a Monte Carlo method, we encountered three main principles of Monte Carlo simulations:

1. All Monte Carlo simulations require a **source of random numbers**. In our case, we used `np.random.rand()` to generate random numbers. The choice of random number generator can have a significant impact on the quality of the simulation. A poor random number generator can lead to biased results.
2. Our simulation result could be described as **generating samples from a distribution** X_N (more precisely generating samples from the pdf of X_N) whose mean is $\pi/4$. This is a general principle in Monte Carlo simulations: we generate samples from a distribution whose mean (or some other property) we want to estimate.
3. It is not enough to find the mean of the distribution. We also need to **estimate the uncertainty** in our estimate. A large uncertainty means that we need to run the simulation multiple times to get a reliable estimate.

We will keep these principles in mind and revisit them from time to time as we explore more advanced Monte Carlo simulations. For now, let's focus on the first principle: generating random numbers.

5.1 Pseudo-random number generators

A **random number generator** is a function that produces a sequence of numbers that meet certain statistical requirements for randomness. *True* random number generators are based on physical processes that are fundamentally random, such as radioactive decay or thermal noise. Such systems are useful in cryptography and other applications where true randomness is important for security.

Below is an example of a true random number generator based on lava lamps called the "wall of entropy". The lava lamps are used to generate random bits, which are then combined to generate random keys for encryption.

There are less exotic ways to generate random numbers. Computer chips have a hardware random number generator that uses thermal noise to generate random bits.

However, true random number generators are slow and expensive. These are not useful for simulations as they are not reproducible. Instead, we use **pseudo-random number generators** (PRNGs) to generate random numbers for simulations.

A **pseudo-random number generator** (PRNG) is a random number generator that produces a sequence of numbers that are not truly random, but are generated by a deterministic algorithm. The sequence of numbers produced by a PRNG is completely determined by the seed: if you know the seed, you can predict the entire sequence of numbers.

The reason for using PRNGs in simulations is to be able to reproduce the results. If you run a simulation with a given seed, you should get the same results every time. This is important for testing, debugging, and for sharing results with others.

We test the quality of a PRNG by running statistical tests on the sequence of numbers it generates. A good PRNG should produce numbers that are indistinguishable from true random numbers. It is said to fool the statistical tests of randomness.

PRNGs need to satisfy several statistical requirements to be useful in simulations. The most important requirements are:

1. **Uniformity**: The numbers generated should be uniformly distributed between 0 and 1.
2. **Independence**: The numbers generated should be independent of each other. Knowing one number should not give you any information about the next number.
3. **Speed**: The PRNG should be fast. Generating random numbers is a common operation in simulations, so the PRNG should be as fast as possible.

These are all difficult requirements to satisfy simultaneously. In practice, most PRNGs are imperfect. You have to choose a PRNG that is appropriate for your application. You have to decide which statistical tests of randomness are most important for your application, and choose a PRNG that satisfies those tests.

For example,

1. **Linear congruential generators** are simple and fast, but they have some statistical problems, as you'll see in the exercises. These are good enough if you only need a few random numbers. These were the first PRNGs to be widely used and have stayed popular for a long time because of their simplicity.
2. The **Mersenne Twister** is a widely-used PRNG that is fast and has good statistical properties. It is the default PRNG in many programming languages, including Python. However, it is not suitable for cryptographic applications.
3. **Cryptographically secure PRNGs** are designed to be secure against cryptographic attacks. They are slower than other PRNGs, but they are necessary for applications where security is important.

5.1.1 Cycle length

In practice, PRNGs generate a random integer between 0 and M for some large number M, and then divide by M to get a random number between 0 and 1. The **period** or **cycle length** of a PRNG is the number of random numbers it can generate before it starts repeating itself. A good PRNG should have a long cycle length.

However, relying solely on the cycle length to determine the quality of a PRNG is a mistake. A PRNG can have a long cycle length and still have poor statistical properties. For example, a PRNG that generates the sequence

$$1, 2, 3, 4, 5, \ldots, M-1, M, 1, 2, 3, 4, 5,$$

has a cycle length of M, but it is a terrible PRNG!

5.1.2 Linear Congruential Generator

A **linear congruential generator (LCG)** is an algorithm that yields a sequence of pseudo-randomized *integers* using a simple recurrence relation. The generator is defined by the recurrence relation:

$$X_{n+1} = (aX_n + c) \mod m \tag{5.1}$$

where: - X_n is the sequence of pseudo-randomized numbers - a is the multiplier - c is the increment

The above equation generates a sequence of integers between 0 and $m-1$. To generate a sequence of random numbers between 0 and 1, we divide the sequence by m:

$$r_n = \frac{X_n}{m} \tag{5.2}$$

LCGs are simple to implement and are computationally efficient. However, as you'll see in the homework, they have some statistical problems. The modulus m is the largest integer that the generator can produce. The modulus m is usually a power of 2, which makes the modulo operation fast.

The **Hull-Dobell Theorem** states that an LCG will have a full period for all seed values if and only if:

1. c and m are relatively prime,
2. $a-1$ is divisible by all prime factors of m,
3. $a-1$ is a multiple of 4 if m is a multiple of 4.

In the special case when m is a power of 2, the Hull-Dobell Theorem simplifies to:

1. c is odd,
2. a is congruent to 1 modulo 4.

It is in fact enough to take $c = 1$. Thus when the modulus is a power of 2, a full period LCG will be of the form:

$$X_{n+1} = (aX_n + 1) \mod 2^b, \tag{5.3}$$

with $a \equiv 1 \mod 4$.

5.2 Statistical test of randomness

As mentioned earlier, we test the quality of a PRNG by running statistical tests on the sequence of numbers it generates. The more tests a PRNG passes, the better it is. There exist many "tests suites" that are used to evaluate the quality of PRNGs such as the Diehard tests, the TestU01 suite, and the NIST Statistical Test Suite.

We'll look at the following three simple statistical tests of randomness from the NIST Statistical Test Suite:

1. **Chi-square test**: The chi-square test checks whether the observed frequency of the sequence is consistent with the expected frequency. If the sequence is truly random, then the observed frequency should be consistent with the expected frequency.
2. **Mono-bit test**: The mono-bit test checks whether the number of 0s and 1s in the sequence is approximately equal. If the sequence is truly random, then the number of 0s and 1s should be roughly equal.
3. **Runs test**: The runs test checks whether the number of runs of 0s and 1s in the sequence is consistent with a random sequence. A run is a sequence of consecutive 0s or 1s. If the sequence is truly random, then the number of runs of 0s and 1s should be consistent with a random sequence.

For each of these tests, we'll set up the null hypothesis and the alternative hypothesis as

- H_0: The sequence is random.
- H_1: The sequence is not random.

We'll use the p-value to determine whether to reject the null hypothesis. If the p-value is less than the significance level α, we reject the null hypothesis. If the p-value is greater than α, we fail to reject the null hypothesis.

5.2.1 Chi-square test

The chi-square test is a one-sided statistical test that measures how well a sample of data matches a theoretical distribution. The chi-square test is used to test whether the observed data is consistent with the expected data. The test statistic is given by:

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i} \tag{5.4}$$

where: - O_i is the observed frequency of the i-th bin - E_i is the expected frequency of the i-th bin - k is the number of bins

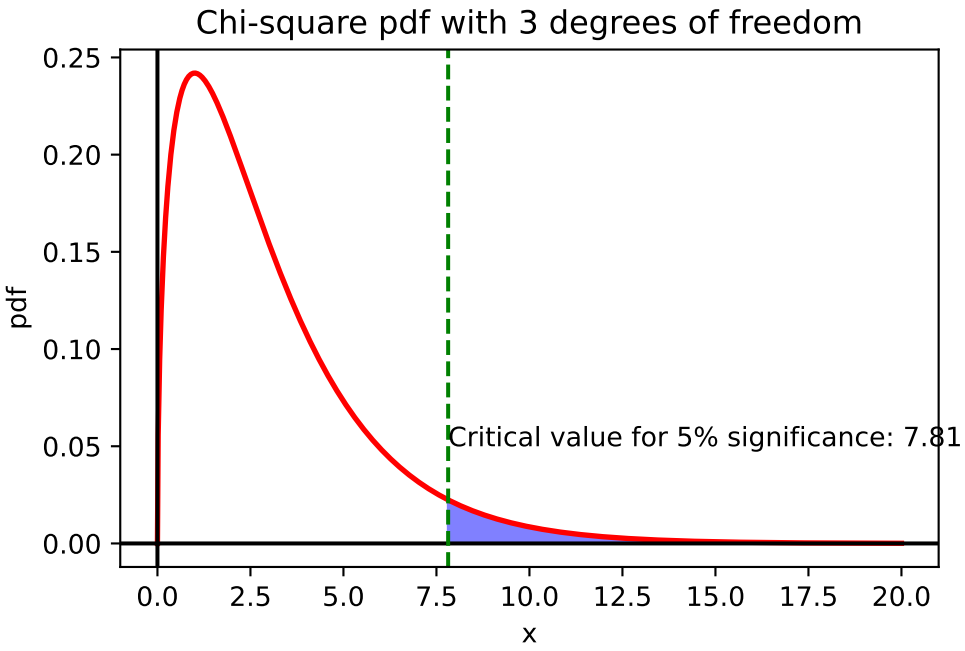
In our case of testing the randomness of a sequence of random numbers, we divide the interval $[0, 1]$ into k bins and count the number of random numbers that fall into each bin. The expected frequency of each bin is n/k, where n is the total number of random numbers.

The chi-square test is used to test the null hypothesis that the observed data is consistent with the expected data. If the chi-square test statistic is large, then the null hypothesis is rejected.

The critical value of the chi-square test statistic depends on the number of degrees of freedom. The degrees of freedom is given by $k - 1$. In python, you can use the `scipy.stats.chi2.ppf` function to perform the chi-square test.

The chi-square test is sensitive to the number of bins k. If k is too small, then the test may not be sensitive enough to detect deviations from the expected distribution. If k is too large, then the test may be too sensitive and may detect deviations that are not significant.

```
To find the critical value, we find $x$ such that $P(X > x) = 0.05$ for a chi-square distribu
If your chi-square value is greater than the critical value, you can reject the null hypothes
```



5.2.2 Mono-bit test

The mono-bit test is a two-sided statistical test that checks whether the number of 0s and 1s in the sequence is approximately equal. In a uniform binary sequence, roughly half the bits are 0s and half the bits are 1s.

Let X_i be the i-th bit in the sequence. Then X_i is a Bernoulli random variable with probability $p = 0.5$. Because the rv's X_i are i.i.d., by the *central limit theorem*, their average

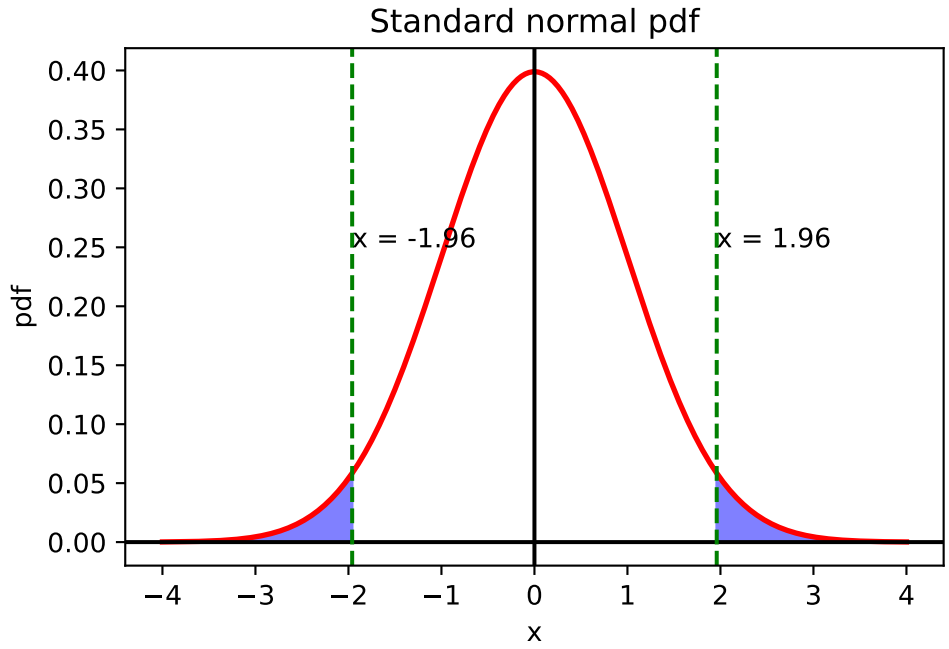
$$X = \frac{X_1 + X_2 + \cdots + X_k}{k} \tag{5.5}$$

approaches a normal distribution with mean 0.5 and variance $1/(4k)$ as k approaches infinity. We can hence use the z-test to test the null hypothesis that the sequence is random. The z-test statistic is given by

$$Z = \frac{X - 0.5}{\sqrt{1/(4k)}}. \tag{5.6}$$

The critical value of the z-test statistic depends on the significance level α. In python, you can use the `scipy.stats.norm.ppf` function to perform the z-test. Note that because this is a two-sided test, we reject the null hypothesis if $Z > z_{\alpha/2}$ or $Z < -z_{\alpha/2}$ where $z_{\alpha/2}$ is the critical value of the z-test statistic at the significance level $\alpha/2$.

To find the critical value, we find x such that $P(-x < X < x) = 0.95$ for a standard norma
If your z-score is outside of the critical values, you can reject the null hypothesis.



5.2.3 Runs test

The Wald-Wolfowitz runs test is a two-sided statistical test that checks whether the number of runs of 0s and 1s in the sequence is consistent with a random sequence. A run is a sequence of consecutive 0s or 1s. In a random sequence, the number of runs of 0s and 1s should be consistent with a random sequence.

For example, in the sequence 0011100110, there are 5 runs: 00, 111, 00, 11, 0. The runs test checks whether the number of runs of 0s and 1s is consistent with a random sequence.

Consider a binary sequence of length k. Define a random variable

$$Z_i = \begin{cases} 1 & \text{if the } i+1^{st} \text{ bit is different from the } i^{th} \text{ bit} \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

One can check that the number of runs in the sequence is given by the random variable

$$R = 1 + \sum_{i=1}^{k-1} Z_i. \tag{5.8}$$

Note that the random variables Z_i are independent. Hence,

$$\mathrm{E}[R] = 1 + \sum_{i=1}^{k-1} \mathrm{E}[Z_i]$$
$$\mathrm{Var}[R] = \sum_{i=1}^{k-1} \mathrm{Var}[Z_i].$$

The calculation of this expectation and variance is left as an exercise.

We assume that for large k the number of runs is approximately normally distributed. We can hence use the z-test to test the null hypothesis that the sequence is random. The z-test statistic is given by

$$Z = \frac{R - E[R]}{\sigma[R]}. \tag{5.9}$$

We reject the null hypothesis if $Z > z_{\alpha/2}$ or $Z < -z_{\alpha/2}$ where $z_{\alpha/2}$ is the critical value of the z-test statistic at the significance level $\alpha/2$. Note that here we use $\alpha/2$ instead of α because this is a two-sided test.

5.2.4 Spectral test

The previous tests fail to detect some of the problems with generating vectors using LCGs. One test for detecting these problems is the spectral test. The spectral test uses the Fast Fourier Transform (FFT) to analyze the spectral properties of the sequence. This test is beyond the scope of this class but you can see some examples in the homework. (This would be a good topic for a project!)

5.2.5 Final remarks

Note that *by definition*, a PRNG is not "random" in an absolute sense. The PRNG used in Python, the Mersenne Twister, is sufficiently random for most applications involving simulations. However, it is not suitable for cryptographic applications as it is possible to predict the entire sequence of numbers if you know a limited set of numbers. For cryptographic applications, you should use a cryptographically secure PRNG whose future numbers cannot be predicted easily from past numbers.

6 Discrete Distributions

6.1 Discrete Case

Let's consider a simple example where X is a discrete random variable that takes values $1, 2, 3$ with probabilities $0.2, 0.3, 0.5$ respectively. To sample from this distribution, we can use the following algorithm:

1. Generate a random number u from the uniform distribution $U(0, 1)$.
2. If $u \leq 0.2$, set $X = 1$.
3. If $0.2 < u \leq 0.5$, set $X = 2$.
4. If $0.5 < u \leq 1$, set $X = 3$.
5. Return X.

This algorithm can be easily extended to the case where X takes n values. This algorithm can be interpreted as a special case of the inverse transform sampling method described below.

6.1.1 Binomial Distribution

Let X be a random variable with binomial distribution with parameters n and p. The probability mass function of X is given by

$$\text{Binomial}(x) = \binom{n}{x} p^x (1-p)^{n-x}, \quad x = 0, 1, \ldots, n.$$

One way to sample from the binomial distribution is to treat it as a discrete distribution and use the above algorithm. However, we can use the Bernoulli distribution to sample from the binomial distribution.

If $Y_1, Y_2, \ldots, Y_n$ are independent random variables with Bernoulli distribution with parameter p, then the random variable $X = Y_1 + Y_2 + \ldots + Y_n$ has binomial distribution with parameters n and p. This gives us a simple algorithm to sample from the binomial distribution:

1. Generate n random numbers $u_1, u_2, \ldots, u_n$ from the uniform distribution $U(0, 1)$.
2. Set $Y_i = 1$ if $u_i \leq p$ and $Y_i = 0$ otherwise for $i = 1, 2, \ldots, n$.
3. Compute $X = Y_1 + Y_2 + \ldots + Y_n$.
4. Return X.

7 Inverse Transform Sampling

Consider a continuous random variable X with probability density function $f(x)$. Let U be a random variable with uniform distribution $U(0,1)$.

Theorem 7.1. ***(Inverse Transform Sampling)****: Let $F(x)$ be the cumulative distribution function of X. If $F(x)$ is strictly increasing and continuous, then the random variable $Y = F^{-1}(U)$ has the same distribution as X.*

Proof. Let $F(x)$ be the cumulative distribution function of X. The cumulative distribution function of U is given by

$$\mathbb{P}(U \leq u) = u.$$

Since $F(x)$ is strictly increasing and continuous, it has an inverse $F^{-1}(u)$.

The cumulative distribution function of $Y = F^{-1}(U)$ is given by

$$\begin{aligned} \mathbb{P}(Y \leq y) &= \mathbb{P}(F^{-1}(U) \leq y) \\ &= \mathbb{P}(U \leq F(y)) \\ &= F(y). \end{aligned}$$

Thus, Y has the same distribution as X. ■

The inverse transform sampling method can be used to sample from any probability distribution X for which we can compute the cumulative distribution function $F(x)$ and its inverse $F^{-1}(u)$. The algorithm to sample from a probability distribution using the inverse transform sampling method is as follows:

1. Generate a random number u from the uniform distribution $U(0,1)$.
2. Compute $x = F^{-1}(u)$.
3. Return x.

Even when $F(x)$ is not strictly increasing, we can still use the inverse transform sampling method by using the generalized inverse of $F(x)$. The generalized inverse of $F(x)$ is defined as

$$F^{-1}(u) = \inf\{x : F(x) \geq u\}.$$

You can think of inf as min for simplicity. □

7.1 Exponential Distribution

Let X be a random variable with exponential distribution with rate parameter $\lambda > 0$. The probability density function of X is given by

$$\text{Exp}(\lambda) = \lambda e^{-\lambda x}, \quad x \geq 0.$$

The cumulative distribution function of X is given by

$$F(x) = 1 - e^{-\lambda x}, \quad x \geq 0.$$

The inverse of the cumulative distribution function is given by

$$F^{-1}(u) = -\frac{1}{\lambda} \log(1 - u).$$

Hence, the inverse transform sampling method can be used to sample from the exponential distribution.

7.2 Weibull Distribution

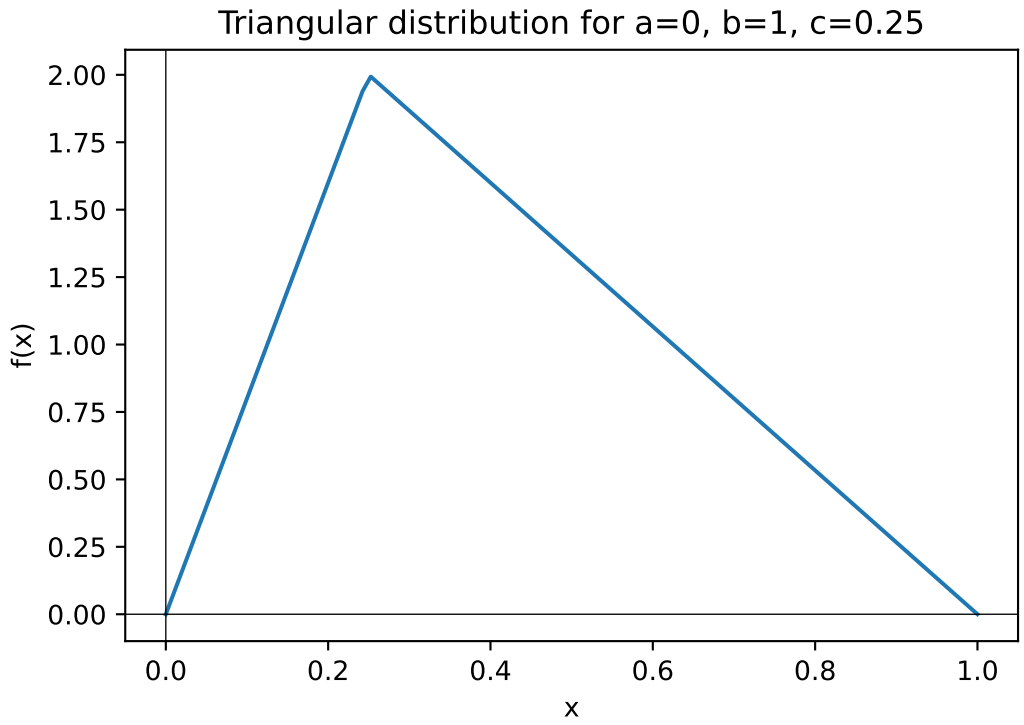
The Weibull distribution is a generalization of the exponential distribution. Let X be a random variable with Weibull distribution with shape parameter $k > 0$ and scale parameter $\lambda > 0$. The probability density function of X is given by

$$f(x) = \frac{k}{\lambda} \left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}, \quad x \geq 0.$$

The inverse transform sampling method can be used to sample from the Weibull distribution.

7.3 Triangular Distribution

Let X be a random variable with triangular distribution supported over the interval $[a, b]$ with maximum value at $c \in [a, b]$. We can use the inverse transform sampling method to sample from the triangular distribution.



7.4 Normal Distribution

The standard normal distribution with mean 0 and variance 1 is given has the probability density function

$$N(0, 1) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}.$$

It is not possible to sample from the normal distribution using the inverse transform sampling method because the cumulative distribution function of the normal distribution does not have a closed-form inverse. However, there are other methods to sample from the normal distribution. One such method is the Box-Muller transform. The Box-Muller transform is based on the

following idea. Consider a 2D random variable (X, Y) with standard normal distribution. The joint probability density function of (X, Y) is given by

$$f(x, y) = \frac{1}{2\pi} e^{-(x^2+y^2)/2}, \quad -\infty < x, y < \infty.$$

Let $R = \sqrt{X^2 + Y^2}$ and $\Theta = \arctan(Y/X)$ be the polar coordinates of (X, Y). Then notice that θ is uniformly distributed in $[0, 2\pi]$. We can calculate the cdf of R as follows:

$$\begin{aligned}
\mathbb{P}(R \le r) &= \mathbb{P}(X^2 + Y^2 \le r^2) \\
&= \int_{x^2+y^2 \le r^2} f(x, y)\, dx\, dy \\
&= \int_{x^2+y^2 \le r^2} e^{-(x^2+y^2)/2}\, dx\, dy \\
&= \int_0^{2\pi} \int_0^r \frac{1}{2\pi} e^{-r^2/2} r\, dr\, d\theta \\
&= 1 - e^{-r^2/2}.
\end{aligned}$$

We can invert this to get the inverse cdf of R:

$$F_R^{-1}(u) = \sqrt{-2\log(1-u)}.$$

We can summarize the above discussion in the following theorem.

Theorem 7.2. ***(Box-Muller Transform)****: Let U_1, U_2 be independent random variables with uniform distribution $U(0, 1)$. Let $R = \sqrt{-2\log U_1}$ and $\Theta = 2\pi U_2$. Then, the random variables $X = R\cos(\Theta)$ and $Y = R\sin(\Theta)$ are independent and have standard normal distribution.*

Note that we are using U_1 instead of $1 - U_1$ in the formula for R. This is because $1 - U_1$ is also uniformly distributed in $[0, 1]$.

This gives us the following algorithm to sample from the normal distribution:

1. Generate two random numbers u_1, u_2 from the uniform distribution $U(0, 1)$.
2. Compute $R = \sqrt{-2\log u_1}$ and $\Theta = 2\pi u_2$.
3. Compute $X = R\cos(\Theta)$ and $Y = R\sin(\Theta)$.
4. Return X (or Y).

7.5 Poisson Distribution

The Poisson distribution with parameter $\lambda > 0$ is a discrete distribution that models the number of events occurring in a fixed interval of time or space, where λ is the average rate of events. In unit time T, the expected number of events is λT. The probability mass function of the Poisson distribution is given by

$$\text{Pois}(n) = \frac{e^{-\lambda}\lambda^n}{n!}, \quad n \in \mathbb{N}.$$

The pmf of the Poisson distribution measures the probability of observing n events in time T.

7.5.1 Relation between Poisson and Binomial Distribution

The Poisson distribution can be approximated by the binomial distribution when the number of trials n is large and the probability of success p is small. Let X be a random variable with binomial distribution with parameters n and p. As $n \to \infty$ and $p \to 0$ such that $\lambda = np$ remains constant, the pmf of X converges to the pmf of the Poisson distribution with parameter λ. This gives us a simple algorithm to sample from the Poisson distribution:

1. Set $X = 0$.
2. Choose n be a large integer (something like $n > 10\lambda$).
3. Set $p = \lambda/n$.
4. Generate a X according to the binomial distribution with parameters n and p.

This is a fast method to sample from the Binomial approximation to the Poisson distribution and is good when λ is small. For large λ, the method described below is more efficient as the number of trials n required for the binomial distribution to approximate the Poisson distribution becomes very large.

7.5.2 Relation between Poisson and Exponential Distribution

We exploit the relation between the Poisson distribution and the exponential distribution to sample from the Poisson distribution. When events occur at a constant rate λ, the time between events follows an exponential distribution with rate parameter λ. More precisely,

Theorem 7.3. ***(Inter-arrival Times)****:*

Let $X_1, X_2, ...$ be independent random variables with exponential distribution with rate parameter λ. Define

$$N = \max\{n : X_1 + X_2 + \cdots + X_n \leq 1\}.$$

Then, N has Poisson distribution with parameter λ.

The proof of this requires us to understand the relation between exponential and gamma distributions. We will skip the proof for now. The Poisson-Exponential connection gives us a simple algorithm to sample from the Poisson distribution:

1. Set $S = 0$ and $N = 0$.
2. While True:
    1. Generate a random number $x \sim \text{Exp}(\lambda)$.
    2. Set $S = S + x$.
    3. If $S > 1$, return N.
    4. Else, set $N = N + 1$.

7.6 Beta Distribution

The Beta distribution is a continuous distribution defined on the interval $[0, 1]$. Let X be a random variable with Beta distribution with parameters $\alpha > 0$ and $\beta > 0$. The probability density function of X is given by

$$f(x) = cx^{\alpha-1}(1-x)^{\beta-1}, \quad 0 \leq x \leq 1,$$

where $c = \frac{(\alpha+\beta-1)!}{(\alpha-1)!(\beta-1)!}$ is the normalizing constant. (Note that when α and β are not integers, we use Γ function to as a generalization of the factorial function.)

This is a simple function supported over the interval $[0, 1]$. The Beta distribution is used as a prior distribution in Bayesian statistics. When α and β are non-zero, the cdf of the Beta distribution does not have a closed-form expression. However, we can use the inverse transform sampling method to sample from the Beta distribution. Instead, we can use the Beta-Order Statistics connection to sample from the Beta distribution.

Definition 7.1. (Order Statistics): Let $X_1, X_2, \ldots, X_n$ be independent and identically distributed random variables. The k-th order statistic, denoted $X_{(k)}$, is the k-th smallest value among $X_1, X_2, \ldots, X_n$.

Theorem 7.4. ***(Beta-Order Statistics)****: Let $U_1, U_2, \ldots, U_n$ be independent random variables with uniform distribution $U(0,1)$. Then the random variable $X = U_{(k)}$ has Beta distribution with parameters $\alpha = k$ and $\beta = n - k + 1$.*

Proof. We'll work out a partial proof of the theorem. Let $X = U_{(k)}$. The cumulative distribution function of X is given by

$$\begin{aligned} F(x) &= \mathbb{P}(U_{(k)} \leq x) \\ &= \mathbb{P}(\text{at least } k \text{ variables among } U_1, U_2, \ldots, U_n \text{ are less than } x) \\ &= \sum_{i=k}^{n} \binom{n}{i} x^i (1-x)^{n-i}. \end{aligned}$$

We differentiate both sides to get the probability density function of X:

$$\begin{aligned} f(x) &= \frac{d}{dx} F(x) \\ &= \sum_{i=k}^{n} \binom{n}{i} \frac{d}{dx} x^i (1-x)^{n-i} \\ &= \sum_{i=k}^{n} \binom{n}{i} i x^{i-1} (1-x)^{n-i} - \binom{n}{i} (n-i) x^i (1-x)^{n-i-1}. \end{aligned}$$

The rest of the proof involves checking that the higher terms in the alternating sum cancel out and only the first term with $i = k$ remains. ■ □

This theorem gives us a simple algorithm to sample from the Beta distribution:

1. Generate n random numbers $u_1, u_2, \ldots, u_n$ from the uniform distribution $U(0,1)$.
2. Sort the numbers in increasing order $u_{(1)} \leq u_{(2)} \leq \ldots \leq u_{(n)}$.
3. Return $u_{(k)}$.

7.7 Mixture Distributions

A mixture distribution is a probability distribution that is formed by taking a weighted sum of two or more probability distributions. Let X be a random variable that is a mixture of distributions $f_1(x), f_2(x), \ldots, f_n(x)$ with weights $w_1, w_2, \ldots, w_n$. The probability density function of X is given by

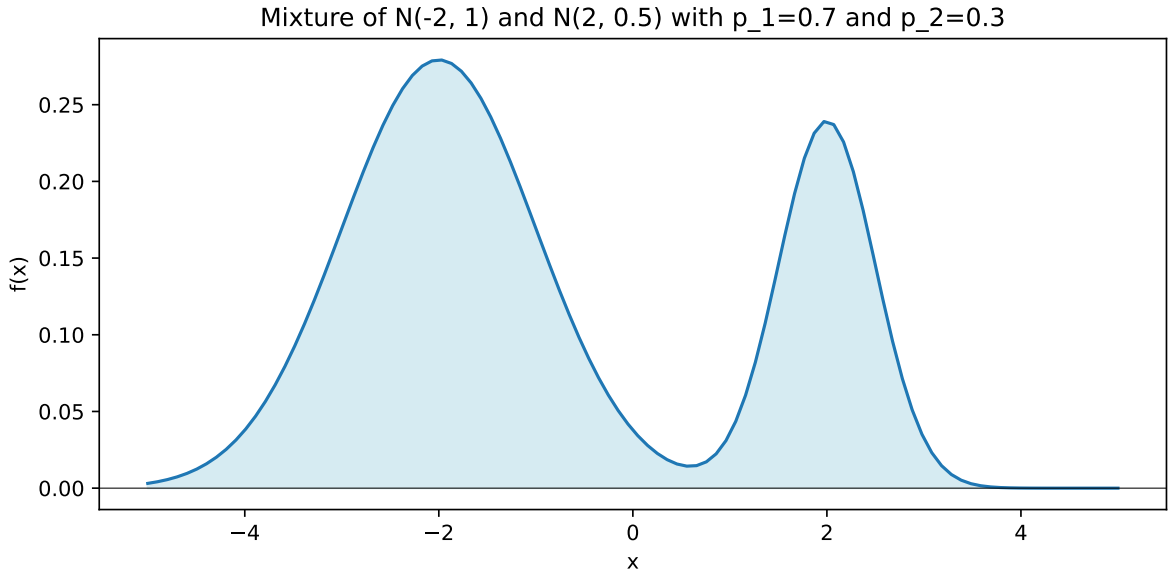
$$f(x) = w_1 f_1(x) + w_2 f_2(x) + ... + w_n f_n(x).$$

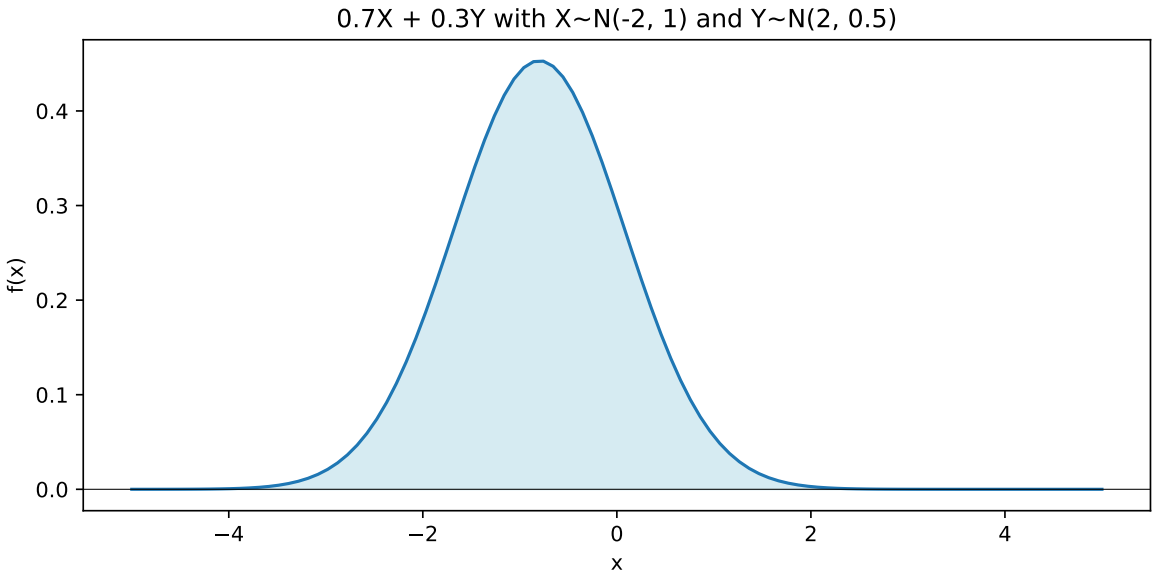
Mixture distributions are used to model complex distributions that cannot be modeled by a single distribution. We can sample from a mixture distribution by sampling from the component distributions and then taking a weighted sum of the samples $X = Y_1$ with probability w_1, $X = Y_2$ with probability w_2, ..., $X = Y_n$ with probability w_n.

To sample from a mixture distribution, we can use the following algorithm:

1. Sample from the discrete distribution $[w_1, w_2, ..., w_n]$ to select a component distribution.
2. Sample from the selected component distribution.
3. Return the sample.

Note that this is not the same as constructing a linear combination of the component distributions. For example, if $X_1 \sim N(\mu_1, \sigma_1^2)$ and $X_2 \sim N(\mu_2, \sigma_2^2)$ are independent, then $w_1 X_1 + w_2 X_2$ is is a normal distribution with mean $w_1\mu_1 + w_2\mu_2$ and variance $w_1^2\sigma_1^2 + w_2^2\sigma_2^2$. This is not the same as a mixture of two normal distributions.


Mixture of N(-2, 1) and N(2, 0.5) with p_1=0.7 and p_2=0.3
0.25
0.20
0.15
0.10
0.05
0.00
f(x)
−4
−2
0
2
4
x


0.7X + 0.3Y with X~N(-2, 1) and Y~N(2, 0.5)
0.4
0.3
0.2
0.1
0.0
f(x)
−4
−2
0
2
4
x

8 Other Distributions

The Beta distribution is a continuous distribution defined on the interval $[0, 1]$. Let X be a random variable with Beta distribution with parameters $\alpha > 0$ and $\beta > 0$. The probability density function of X is given by

$$f(x) = cx^{\alpha-1}(1-x)^{\beta-1}, \quad 0 \leq x \leq 1,$$

where $c = \frac{(\alpha+\beta-1)!}{(\alpha-1)!(\beta-1)!}$ is the normalizing constant. (Note that when α and β are not integers, we use Γ function to as a generalization of the factorial function.)

This is a simple function supported over the interval $[0, 1]$. The Beta distribution is used as a prior distribution in Bayesian statistics. When α and β are non-zero, the cdf of the Beta distribution does not have a closed-form expression. However, we can use the inverse transform sampling method to sample from the Beta distribution. Instead, we can use the Beta-Order Statistics connection to sample from the Beta distribution.

Definition 8.1. (Order Statistics): Let $X_1, X_2, \ldots, X_n$ be independent and identically distributed random variables. The k-th order statistic, denoted $X_{(k)}$, is the k-th smallest value among $X_1, X_2, \ldots, X_n$.

Theorem 8.1. ***(Beta-Order Statistics)****: Let $U_1, U_2, \ldots, U_n$ be independent random variables with uniform distribution $U(0, 1)$. Then the random variable $X = U_{(k)}$ has Beta distribution with parameters $\alpha = k$ and $\beta = n - k + 1$.*

Proof. We'll work out a partial proof of the theorem. Let $X = U_{(k)}$. The cumulative distribution function of X is given by

$$\begin{aligned} F(x) &= \mathbb{P}(U_{(k)} \leq x) \\ &= \mathbb{P}(\text{at least } k \text{ variables among } U_1, U_2, \ldots, U_n \text{ are less than } x) \\ &= \sum_{i=k}^{n} \binom{n}{i} x^i (1-x)^{n-i}. \end{aligned}$$

We differentiate both sides to get the probability density function of X:

$$\begin{aligned} f(x) &= \frac{d}{dx}F(x) \\ &= \sum_{i=k}^{n} \binom{n}{i} \frac{d}{dx} x^i (1-x)^{n-i} \\ &= \sum_{i=k}^{n} \binom{n}{i} i x^{i-1}(1-x)^{n-i} - \binom{n}{i}(n-i)x^i(1-x)^{n-i-1}. \end{aligned}$$

The rest of the proof involves checking that the higher terms in the alternating sum cancel out and only the first term with $i = k$ remains. ■ □

This theorem gives us a simple algorithm to sample from the Beta distribution:

1. Generate n random numbers $u_1, u_2, \ldots, u_n$ from the uniform distribution $U(0, 1)$.
2. Sort the numbers in increasing order $u_{(1)} \leq u_{(2)} \leq \ldots \leq u_{(n)}$.
3. Return $u_{(k)}$.

8.1 Mixture Distributions

A mixture distribution is a probability distribution that is formed by taking a weighted sum of two or more probability distributions. Let X be a random variable that is a mixture of distributions $f_1(x), f_2(x), \ldots, f_n(x)$ with weights $w_1, w_2, \ldots, w_n$. The probability density function of X is given by

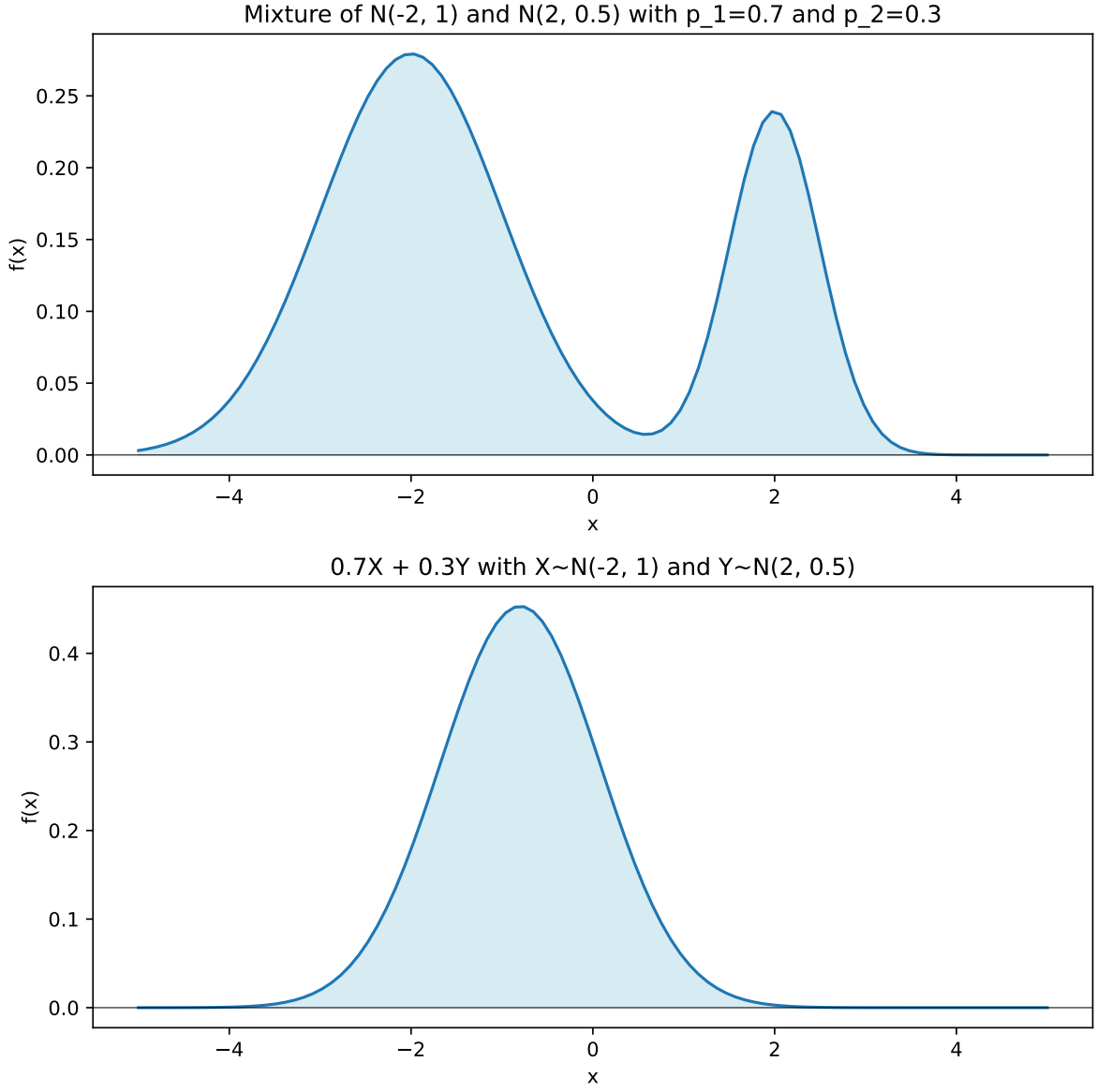
$$f(x) = w_1 f_1(x) + w_2 f_2(x) + \ldots + w_n f_n(x).$$

Mixture distributions are used to model complex distributions that cannot be modeled by a single distribution. We can sample from a mixture distribution by sampling from the component distributions and then taking a weighted sum of the samples $X = Y_1$ with probability w_1, $X = Y_2$ with probability w_2, $\ldots$, $X = Y_n$ with probability w_n.

To sample from a mixture distribution, we can use the following algorithm:

1. Sample from the discrete distribution $[w_1, w_2, \ldots, w_n]$ to select a component distribution.
2. Sample from the selected component distribution.
3. Return the sample.

Note that this is not the same as constructing a linear combination of the component distributions. For example, if $X_1 \sim N(\mu_1, \sigma_1^2)$ and $X_2 \sim N(\mu_2, \sigma_2^2)$ are independent, then $w_1 X_1 + w_2 X_2$ is is a normal distribution with mean $w_1\mu_1 + w_2\mu_2$ and variance $w_1^2\sigma_1^2 + w_2^2\sigma_2^2$. This is not the same as a mixture of two normal distributions.



9 Rejection Sampling

The **accept-reject Method**, also called **rejection sampling**, is a simple and general technique for generating random variables. It is based on the idea of sampling from a simple distribution and then rejecting the samples that are not in the desired distribution. This method is particularly useful when the desired distribution is difficult to sample from directly, but it is easy to evaluate the density function of the distribution.

For the accept-reject method, we need to recall the following definitions: Let X and Y be discrete random variables.

The **joint distribution** of X and Y is given by the probability mass function

$$f_{X,Y}(x, y) = \mathbb{P}(X = x, Y = y).$$

The **marginal distribution** of X is given by the probability mass function

$$f_X(x) = \mathbb{P}(X = x) = \sum_y f_{X,Y}(x, y).$$

The **conditional probability** of X given Y is defined as

$$f_{X|Y}(x|y) = \mathbb{P}(X = x|Y = y) = \frac{f_{X,Y}(x, y)}{f_Y(y)}.$$

When X and Y are continuous random variables, the definitions are similar, but we replace the probability mass functions with probability density functions.

The **accept-reject method** is based on the following theorem:

Theorem 9.1. ***Theorem (Marginal of Uniform):*** *Let $p(x)$ be a probability distribution. Let X, Y be two random variables having the joint distribution*

$$f_{X,Y}(x,y) = \begin{cases} 1, & \text{if } 0 \leq y \leq p(x), \\ 0, & \text{otherwise.} \end{cases} \tag{9.1}$$

Then the marginal distribution of X is given by $p(x)$.

Proof. The marginal distribution of X is given by

$$f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x,y)dy = \int_0^{p(x)} dy = p(x).$$

■ □

Thus in order to sample from a distribution $p(x)$, we want to come up with a way to sample from the joint distribution given by (Equation 9.1). The accept-reject method is a way to do this.

9.1 Accept-Reject Method v1

Suppose $p(x)$ is a probability distribution from which we want to sample. Suppose further that $p(x)$ is supported over the interval $[a, b]$ and is bounded by M, i.e., $p(x) \leq M$ for all $x \in [a, b]$.

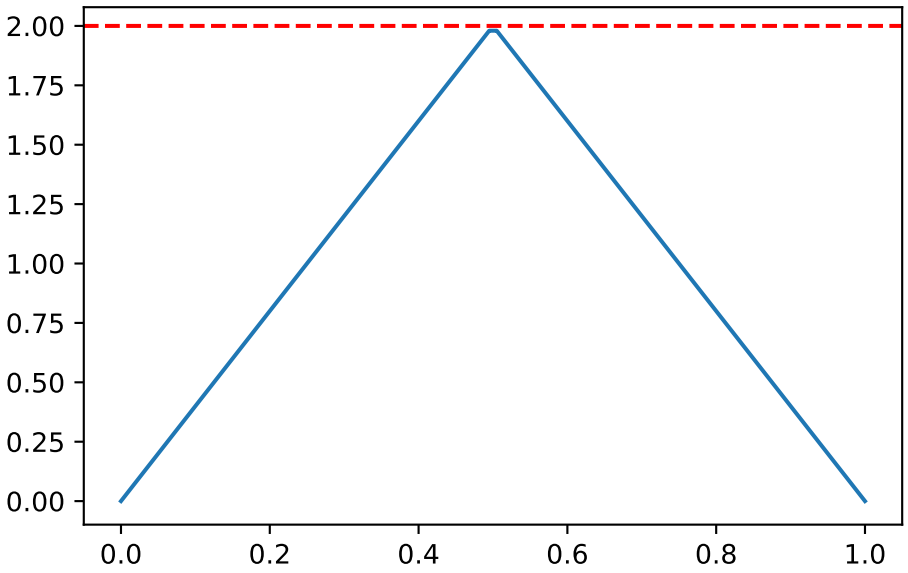
The simplest version of the accept-reject method is as follows:

1. Sample x uniformly from $[a, b]$.
2. Sample y uniformly from $[0, M]$.
3. If $y \leq p(x)$, return x; otherwise, go back to step 1.

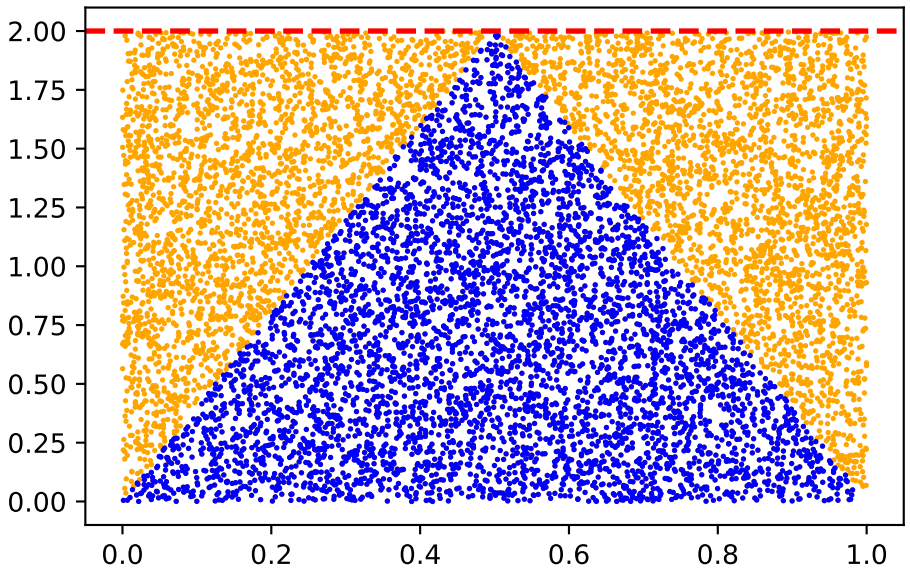
Example 9.1. Consider the triangular distribution

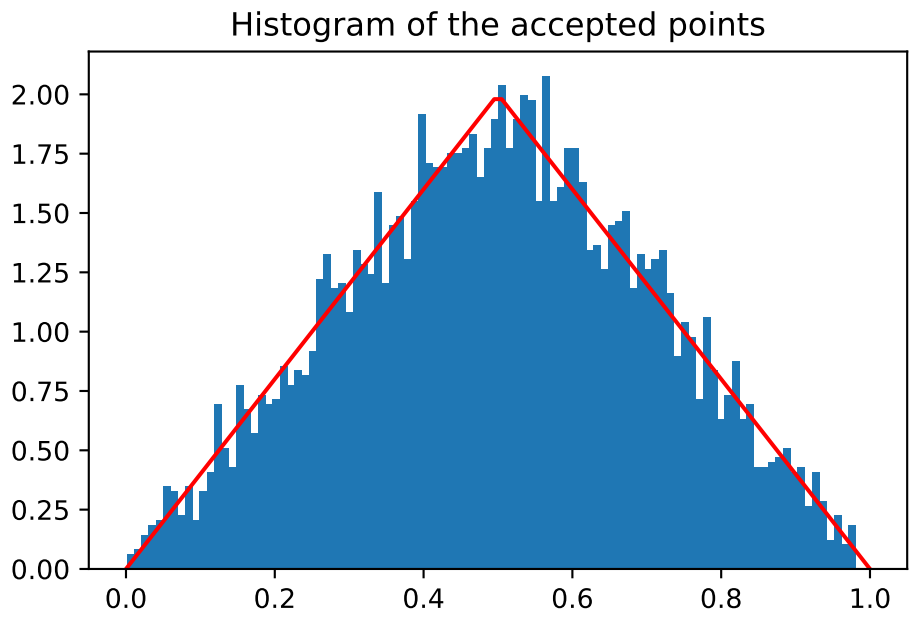
$$p(x) = \begin{cases} 4x, & \text{if } 0 \leq x \leq 0.5, \\ 4(1-x), & \text{if } 0.5 \leq x \leq 1 \\ 0, & \text{otherwise.} \end{cases} \tag{9.2}$$

We can use the accept-reject method to sample from this distribution. The density function is supported over $[0, 1]$ and is bounded by $M = 2$.



Efficiency = num accepted / num generated = 0.50 = 1/2.00





9.1.1 Efficiency

Note that unlike the methods we have seen so far, the accept-reject method is probabilistic. The method generates uniformly distributed samples in the rectangle of area $M(b-a)$, where M is the bound on $p(x)$ and $[a, b]$ is the support of $p(x)$. But because $p(x)$ is a probability distribution, the area under the curve is 1. Thus the efficiency of the accept-reject method is given by

$$\text{Efficiency} = \frac{1}{M(b-a)}.$$

If M is large i.e. the probability distribution has a large peak, then the efficiency of the accept-reject method is low.

9.2 Accept-Reject Method v2

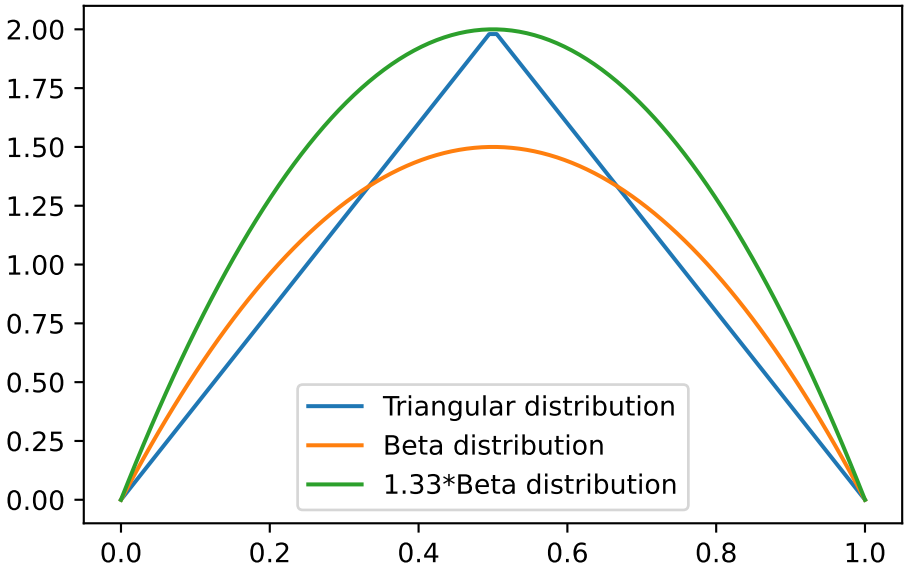
A better version of the accept-reject method is obtained by replacing the enveloping rectangle with a enveloping curve. The closer the enveloping curve is to the distribution, the higher the efficiency of the method.

Definition 9.1. Majorizing Function: Let $p(x)$ be a probability distribution. A function $g(x)$ is said to majorize $p(x)$ if $g(x) \geq p(x)$ for all x in the support of $p(x)$.

In the accept-reject jargon, we call $p(x)$ the **target distribution** and $g(x)$ the **proposal distribution**. Note that a probability distribution can never majorize another probability distribution as the area under the curve is 1. But we can scale the proposal distribution by a constant M such that $Mg(x)$ majorizes $p(x)$.

Example 9.2. Consider the triangular distribution given by (Equation 9.2). We saw that we can use the enveloping rectangle with $M = 2$ to sample from this distribution. The Beta distribution Beta$(2, 2)$ majorizes the triangular distribution with constant $M = 4/3$.

$$\text{Beta}(2, 2) = 6x(1 - x), \quad x \in [0, 1]. \tag{9.3}$$



In order to run the accept-reject method, we need to sample uniformly from the region between the graph of $Mg(x)$ and the x-axis. This can be done using the following theorem:

Theorem 9.2. ***Majorizing Function:*** *Let $g(x)$ be a probability distribution and let M be a constant. Let X and Y be two random variables such that $X \sim g(x)$ and $(Y|X = x) \sim U(0, Mg(x))$. Then the joint distribution of X and Y is uniform over the region between the graph of $Mg(x)$ and the x-axis.*

Proof. The conditional probability of Y given X is given by

$$f_{Y|X}(y|x) = \begin{cases} \frac{1}{Mg(x)}, & \text{if } 0 \leq y \leq Mg(x), \\ 0, & \text{otherwise.} \end{cases}$$

Hence, the joint distribution of X and Y is given by

$$f_{X,Y}(x,y) = f_{Y|X}(y|x)g(x) = \begin{cases} \frac{1}{M}, & \text{if } 0 \leq y \leq Mg(x), \\ 0, & \text{otherwise.} \end{cases}$$

Hence, the joint distribution is uniform over the region between the graph of $Mg(x)$ and the x-axis.

Note that we are seeing the constant $1/M$ instead of 1 as the area under the curve $y = Mg(x)$ is M and not 1. □

This gives us a way to sample from the majorizing function $g(x)$. We can then use the accept-reject method to sample from the target distribution $p(x)$. The algorithm is as follows:

1. Sample x from $g(x)$.
2. Sample y uniformly from $[0, Mg(x)]$.
3. If $y \leq p(x)$, return x; otherwise, go back to step 1.

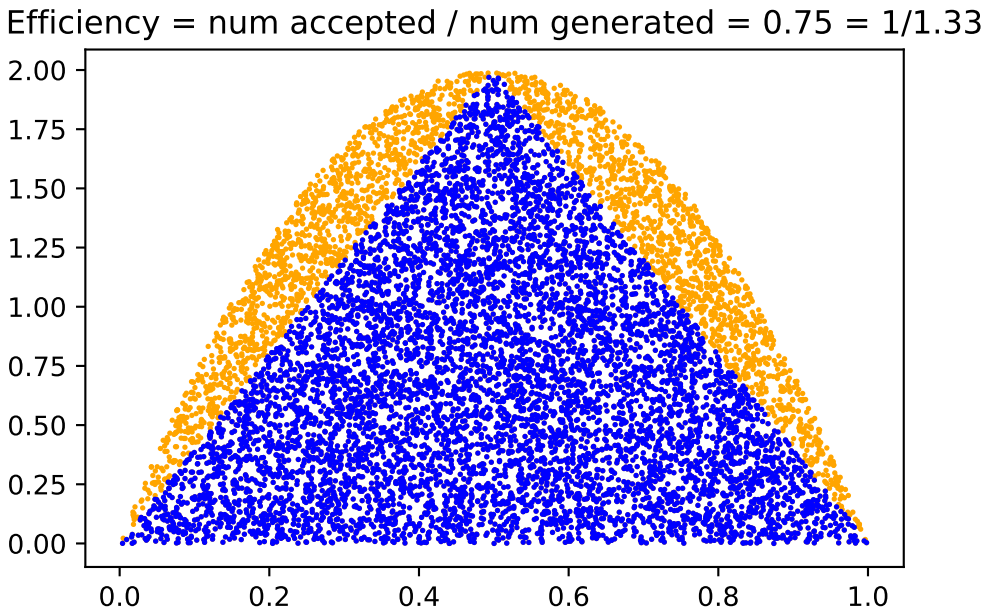
Steps 2 and 3, are often rewritten as:

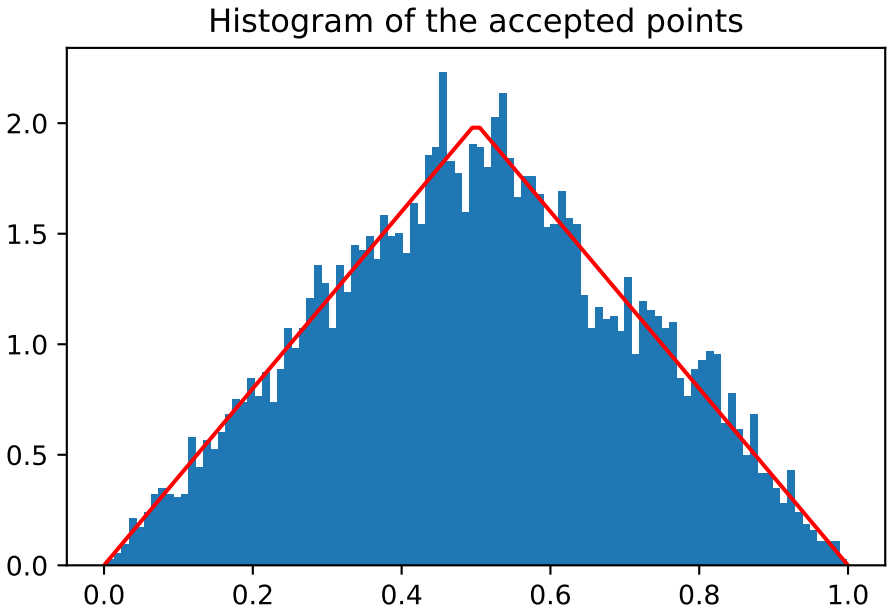
2. Sample u uniformly from $[0, 1]$.
3. If $u \leq p(x)/(Mg(x))$, return x; otherwise, go back to step 1.

Furthermore, taking a ratio of two small numbers can lead to numerical instability. We can avoid this by taking the logarithm of the ratio. The third step of the algorithm becomes:

3. If $\log(u) \leq \log(p(x)) - \log(Mg(x))$, return x; otherwise, go back to step 1.

Example 9.3. Consider the triangular distribution given by (Equation 9.2). We saw that the Beta distribution $\text{Beta}(2, 2)$ majorizes the triangular distribution. We can use the accept-reject method to sample from the triangular distribution using the Beta distribution as the proposal distribution. This improves the efficiency of the method from 1/2 to 3/4.





9.3 Normalizing Constant

We often find ourselves in a situation where we know the probability density function f up to a normalizing constant. For example, the gamma distribution has the density function

$$p(x) = cx^{\alpha-1}e^{-x/\beta}, \tag{9.4}$$

where c is a normalizing constant. We can compute c by integrating $p(x)$ and setting the integral to 1. But as it turns out we do not need to know c to sample from the distribution. We can use the accept-reject method to sample from the distribution without knowing the normalizing constant. For this we need the following theorem:

Theorem 9.3. ***Normalizing Constant:*** *Let $p(x)$ be any function with a finite integral c. Let X, Y be two random variables having the joint distribution*

$$f_{X,Y}(x,y) = \begin{cases} \frac{1}{c}, & \text{if } 0 \le y \le p(x), \\ 0, & \text{otherwise.} \end{cases} \tag{9.5}$$

Then the marginal distribution of X is given by $p(x)/c$.

Proof. The marginal distribution of X is given by

$$f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x,y)dy = \int_0^{p(x)} \frac{1}{c}dy = \frac{p(x)}{c}.$$

■ □

Thus sampling from a uniform distribution over the graph of $p(x)$ and then finding the marginal distribution of X gives us a sample from the distribution $p(x)/c$. This is a powerful result as it allows us to sample from a distribution without knowing the normalizing constant.

If we majorize $cp(x)$ with a proposal distribution $Mg(x)$, we can use the accept-reject method to sample from the distribution $p(x)/c$ without knowing the normalizing constant. The efficiency in this case will be given by

$$\text{Efficiency} = \frac{\text{area under the graph of } p(x)}{\text{area under the graph of } Mg(x)} = \frac{c}{M}.$$

Note that there is no loss in efficiency due to the unknown normalizing constant as if we can majorize $p(x)$ with $Mg(x)$, we can majorize the normalized proposal density $p(x)/c$ by $Mg(x)/c$, giving us the same efficiency.

9.4 Final remarks

If the proposal distribution is not close to the target distribution, then the efficiency of the method is low. In higher dimensions, it is difficult to come up with a good proposal distribution.

In higher dimensions, we need more sophisticated methods to sample from the target distribution such as Markov Chain Monte Carlo (MCMC) methods. We'll see some of these methods such as Metropolis-Hastings and Gibbs sampling in the future.

Adaptive rejection sampling is another method that tries to find a good proposal distribution by using the samples generated so far. The idea is to use the samples to build a piecewise linear majorizing function. This function is then used as the proposal distribution for the accept-reject method. Because the cdf of a piecewise linear function is a quadratic function, it can be inverted easily. The method works for log-concave densities but has been extended to more general densities. This would be a good topic for a project.

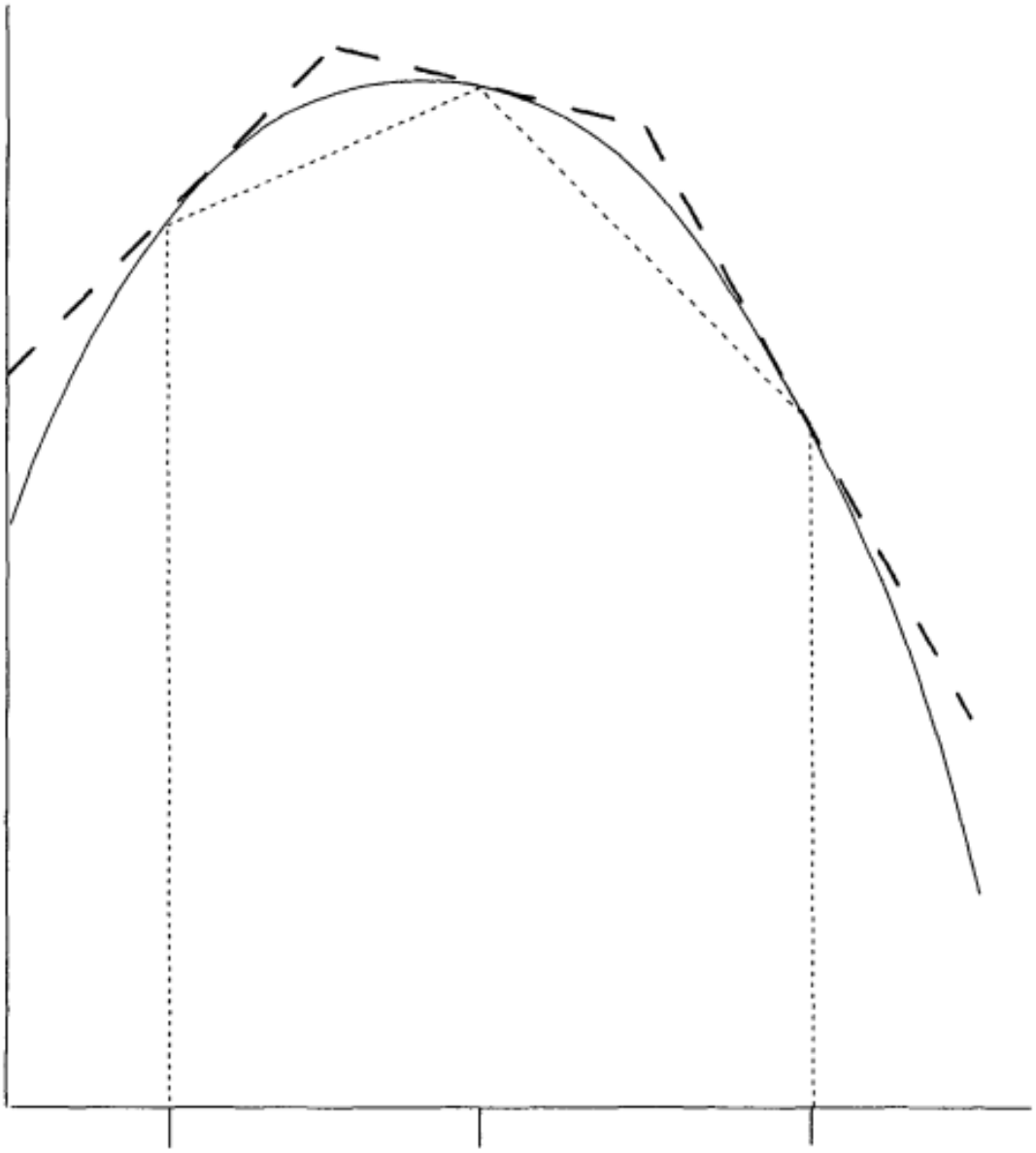


Figure 9.1: Gilks, W. R., & Wild, P. (1992). Adaptive Rejection Sampling for Gibbs Sampling. Journal of the Royal Statistical Society. Series C (Applied Statistics), 41(2), 337–348. https://doi.org/10.2307/2347565

10 Gibbs Sampling

Gibbs Sampling is a MCMC algorithm that is used to sample from a joint distribution using conditional distributions. For now, we'll focus on sampling in 2D, but the algorithm generalizes to higher dimensions.

The Gibbs sampling algorithm for sampling from a joint distribution $f_{X,Y}(X, Y)$ is as follows:

1. Start with some initial values X_0 and Y_0.
2. For $i = 1, 2, \ldots, N$
    1. Sample $X_i \sim f_{X|Y}(\cdot\,|Y_{i-1})$.
    2. Sample $Y_i \sim f_{Y|X}(\cdot\,|X_i)$.
3. Return the (X_0, Y_0), (X_1, Y_1), ..., (X_N, Y_N).

The algorithm generates a sample path of length N of the Markov Chain as described in Section 10.1. If needed, we can discard the initial samples to ensure that the Markov Chain has converged to the stationary distribution.

10.0.1 Joint Exponential Distribution

Let X and Y be two random variables with the truncated exponential distribution:

$$f_{X,Y}(x, y) = ce^{-\lambda xy} \text{ for } 0 \le x \le D_1, 0 \le y \le D_2,$$

where c is the normalization constant. We want to sample from the joint distribution $f_{X,Y}$. One can show that the conditionals are,

$$f_{X|Y}(x|y_0) = c_{y_0} e^{-\lambda x y_0} \text{ for } 0 \le x \le D_1$$
$$f_{Y|X}(y|x_0) = c_{x_0} e^{-\lambda y x_0} \text{ for } 0 \le y \le D_2,$$

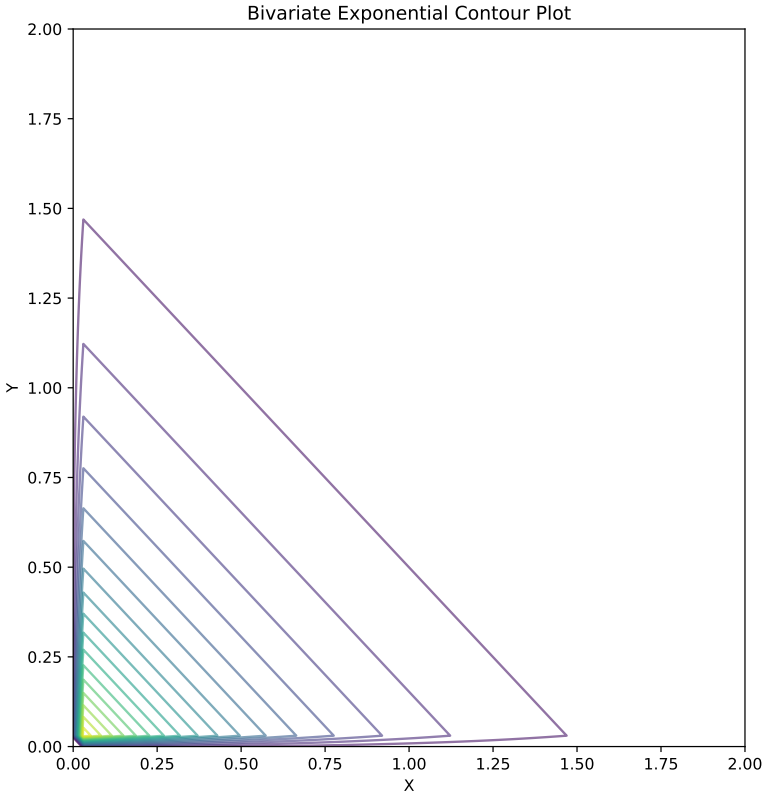
where c_{y_0} and c_{x_0} are the normalization constants. We can easily sample from the two conditionals $f_{X|Y}$ and $f_{Y|X}$ using the inverse method function (even for truncated distributions). The Gibbs algorithm becomes

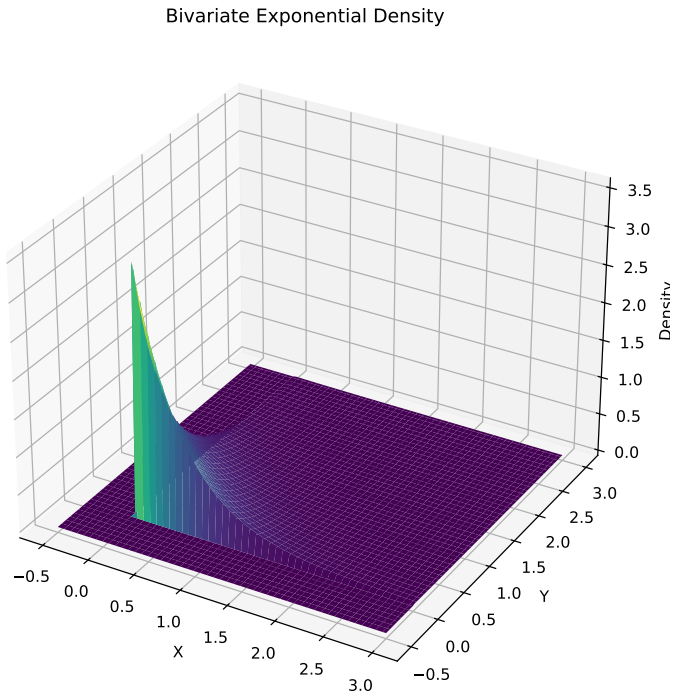
1. Start with some initial values X_0 and Y_0.

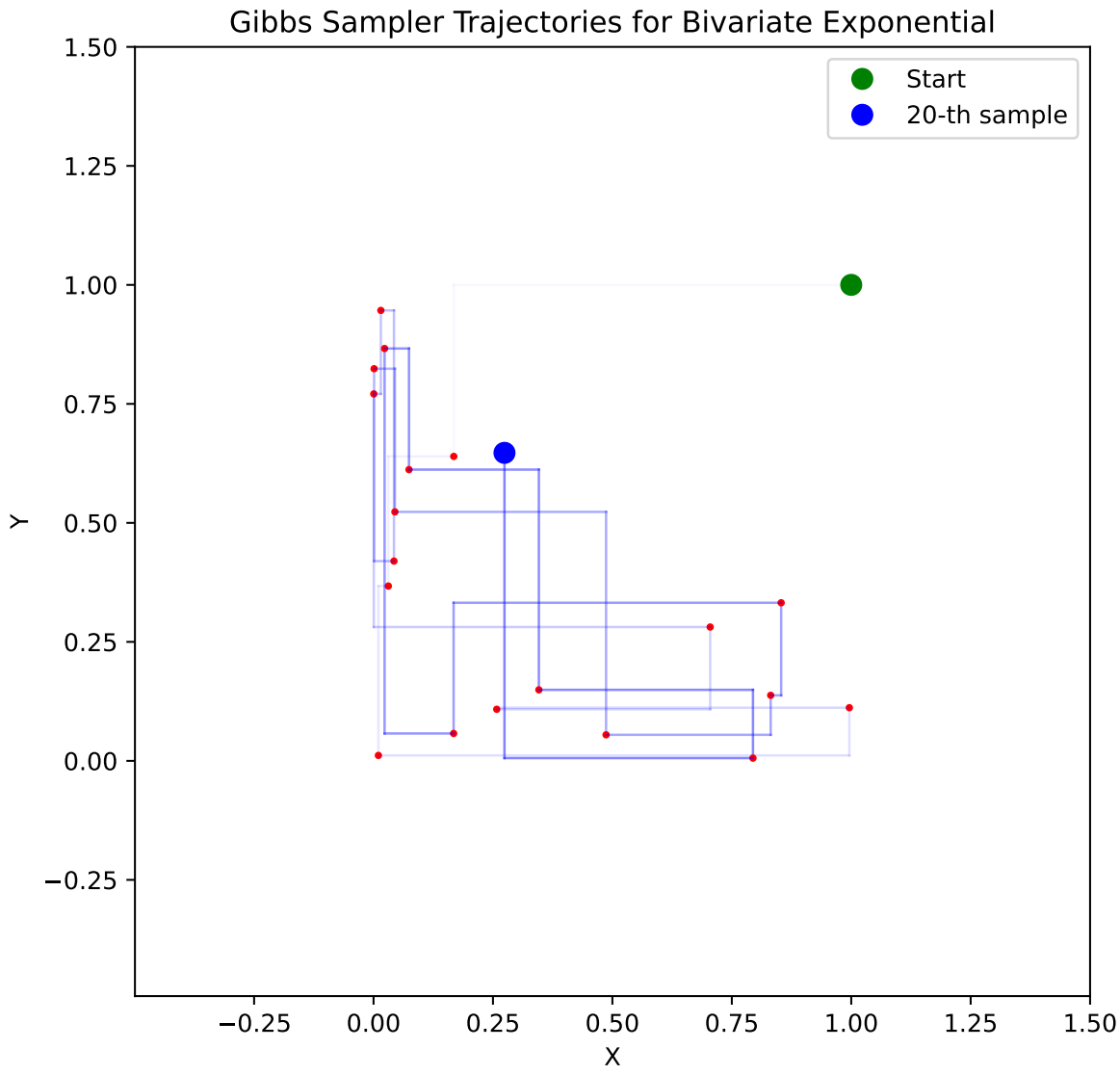
2. For $i = 1, 2, \ldots, N$

    1. Sample $X_i \sim (\text{Exp}(\lambda Y_{i-1})$ restricted to $[0, D_1])$ using the inverse transform method.
    2. Sample $Y_i \sim (\text{Exp}(\lambda X_i)$ restricted to $[0, D_2])$ using the inverse transform method.

3. Return the (X_0, Y_0), (X_1, Y_1), ..., (X_N, Y_N).


Bivariate Exponential Contour Plot
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Y
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
X


Bivariate Exponential Density
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Density
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
X
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
Y



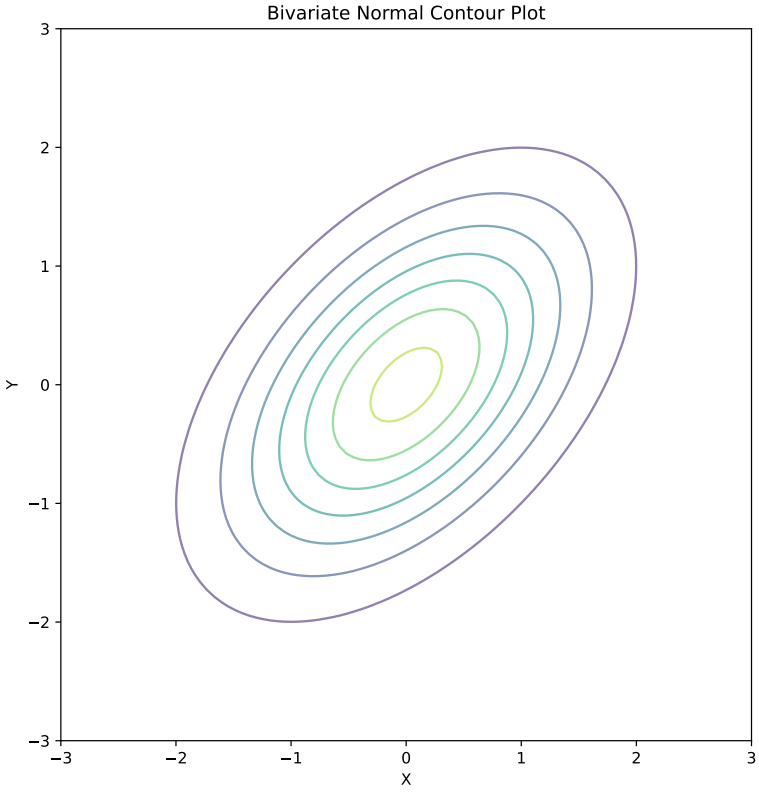
10.0.2 Bivariate Normal Distribution

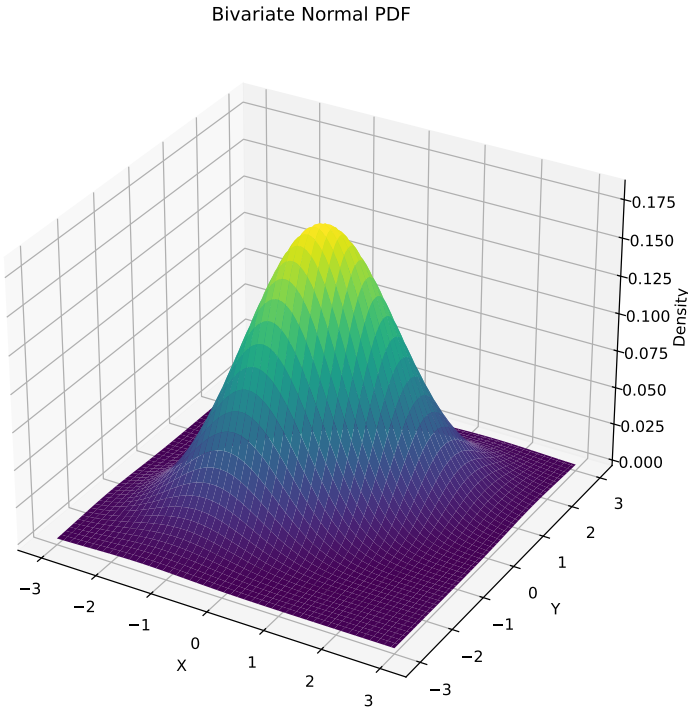
Suppose (X, Y) has a bivariate normal distribution with mean $(0, 0)$ and covariance matrix $\Sigma = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$. We want to sample from the joint distribution $f_{X,Y}$. One can show that

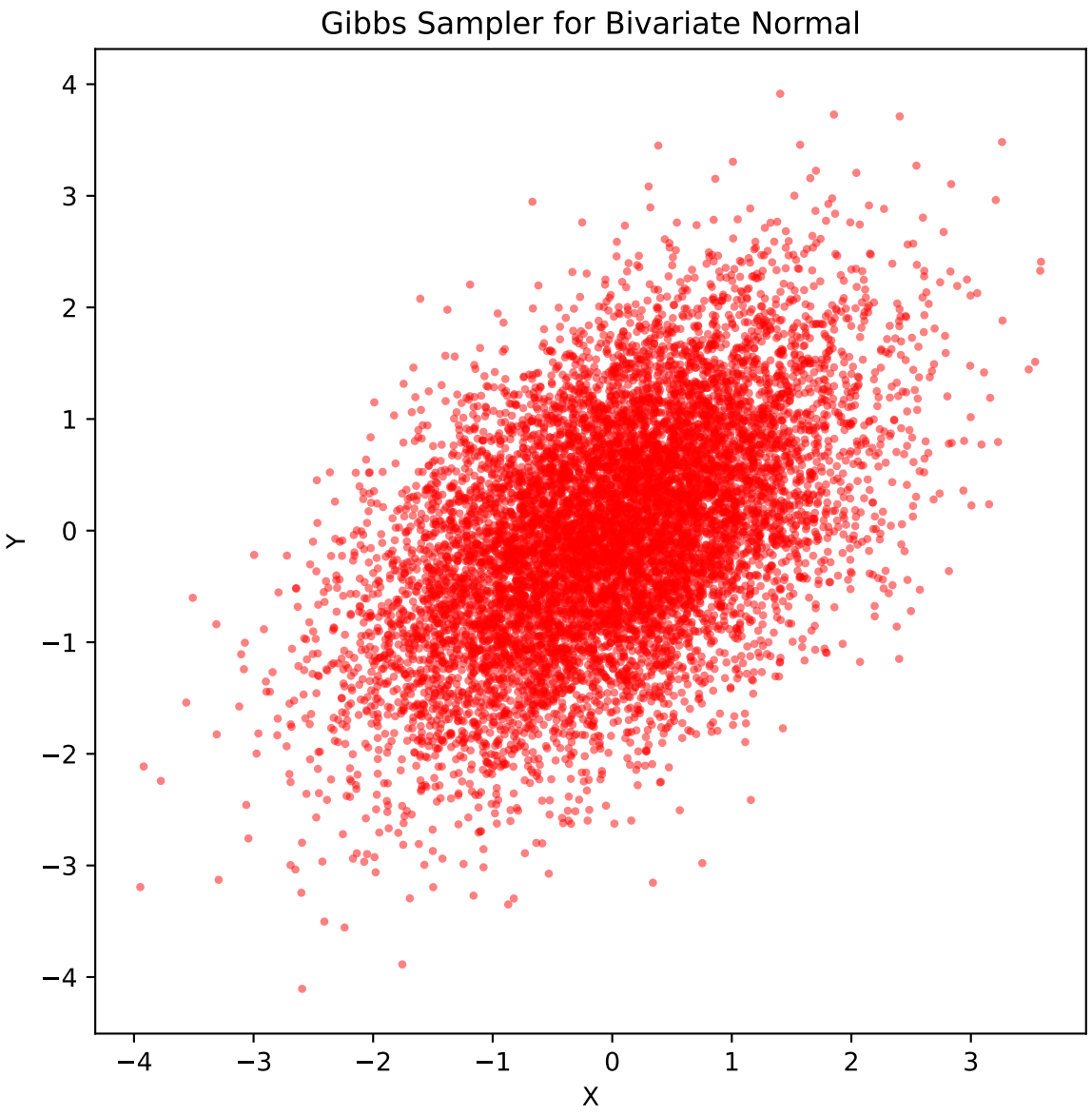
$$(X|Y = y_0) \sim \mathcal{N}(\rho y_0, 1 - \rho^2)$$
$$(Y|X = x_0) \sim \mathcal{N}(\rho x_0, 1 - \rho^2).$$

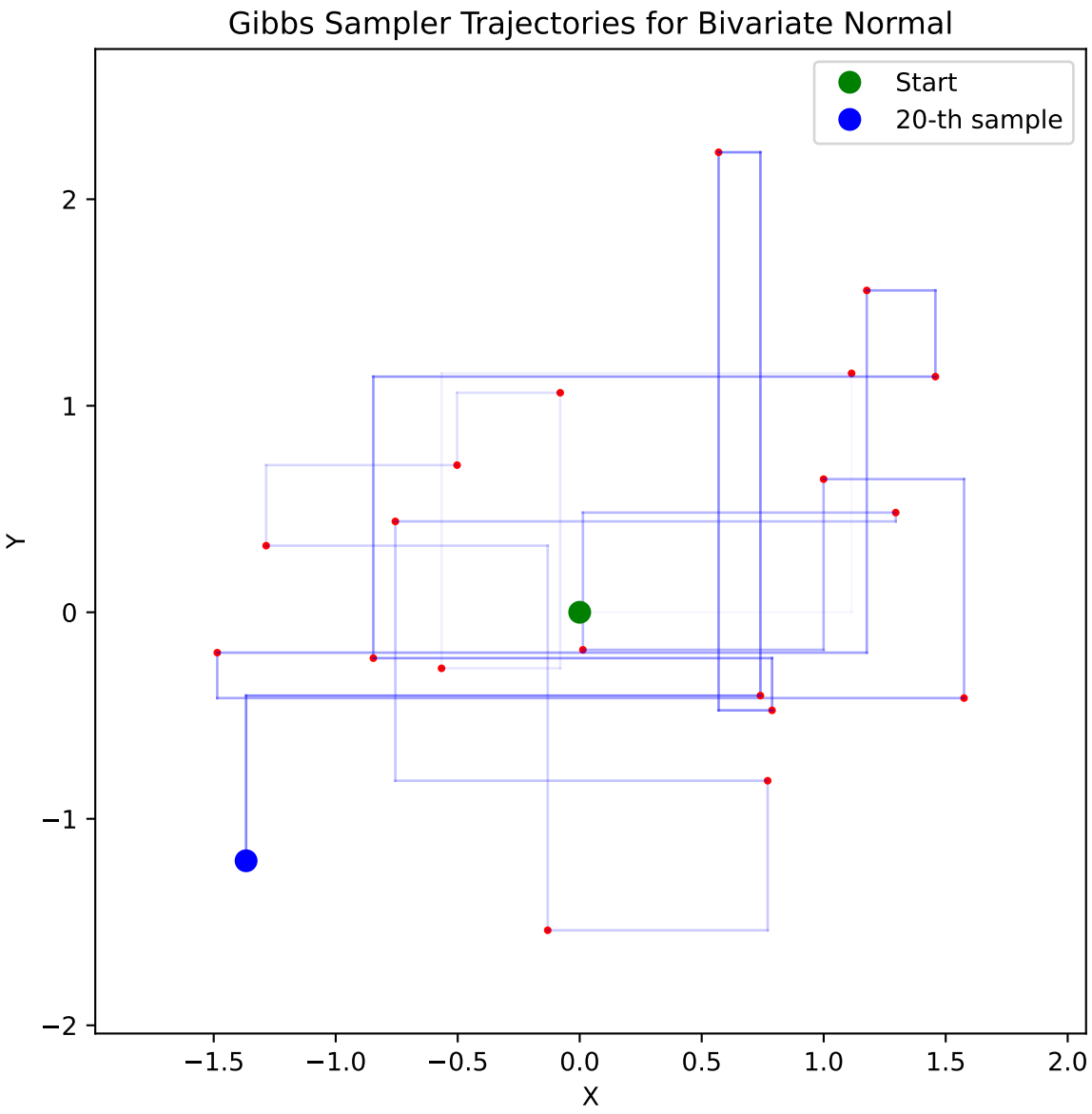
We can sample from the two conditionals $f_{X|Y}$ and $f_{Y|X}$ using the Box-Muller method. The Gibbs algorithm becomes

1. Start with some initial values X_0 and Y_0.
2. For $i = 1, 2, \ldots, N$
    1. Sample $X_i \sim \mathcal{N}(\rho Y_{i-1}, 1 - \rho^2)$.
    2. Sample $Y_i \sim \mathcal{N}(\rho X_i, 1 - \rho^2)$.
3. Return the (X_0, Y_0), (X_1, Y_1), ..., (X_N, Y_N).


Bivariate Normal Contour Plot
Y
X
3
2
1
0
−1
−2
−3


Bivariate Normal PDF
Density
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
X
Y


Gibbs Sampler for Bivariate Normal
4
3
2
1
0
−1
−2
−3
−4
Y
−4
−3
−2
−1
0
1
2
3
X



10.1 Markov Chain

The Gibbs sampling algorithm generates a Markov Chain whose state space is the product space of the state spaces of the individual variables $\Omega = \Omega_X \times \Omega_Y$. In the above examples, the state space is $[0, D_1] \times [0, D_2]$ for the exponential distribution and $\mathbb{R}^2$ for the bivariate normal distribution. The transition matrix of the Markov Chain in discrete case is given by

$$P\left(\begin{bmatrix} x \\ y \end{bmatrix}, \begin{bmatrix} x' \\ y' \end{bmatrix}\right) = \mathbb{P}(X_{i+1} = x' | Y_i = y)\mathbb{P}(Y_{i+1} = y' | X_i = x').$$

In the continuous case, the transition *kernel* is given by

$$K\left(\begin{bmatrix} x \\ y \end{bmatrix}, \begin{bmatrix} x' \\ y' \end{bmatrix}\right) = f_{X|Y}(x|y) f_{Y|X}(y|x').$$

Theorem 10.1. *The Gibbs sampling algorithm generates a Markov Chain with the transition matrix P as described above. The joint distribution $f_{X,Y}$ is a stationary distribution of the Markov Chain. Hence, if the Markov Chain converges to the stationary distribution, the samples generated by the Gibbs algorithm will be distributed according to $f_{X,Y}$.*

We'll provide a proof of the above theorem in the next section.

11 Metropolis–Hastings Algorithm

Metropolis–Hastings algorithm is a Markov Chain Monte Carlo (MCMC) method used to sample from a probability distribution. It is a type of a rejection sampling algorithm where we generate a sequence of samples from a target distribution by proposing a new sample and accepting or rejecting it based on a certain criterion. The algorithm is widely used in Bayesian statistics, statistical physics, and machine learning.

We'll start a few examples to understand the algorithm and then discuss the general algorithm.

11.1 Random Walks Metropolis–Hastings Algorithm

Consider a region Ω in $\mathbb{R}^n$. Suppose we want to sample uniformly from this region. One way to do this is through rejection sampling. We envelope the region in simple shape like a cube and sample uniformly from the cube. If the sample lies in the region Ω, we accept it, otherwise we reject it. The efficiency of this method depends on the ratio of the volume of the cube to the volume of the region Ω. If the region is highly non-convex, this ratio can be very small and the rejection sampling can be very inefficient.

An alternative method is to use random walks. We start at a point x_0 in the region Ω and take a random step in a random direction. If the new point x_1 is in the region Ω, we accept it, otherwise we stay at the point x_0. We repeat this process to generate a sequence of points. This method is more efficient than rejection sampling for highly non-convex regions.

The rationale behind this is that if a point is in the region Ω, then a point near it is also likely to be in the region. We can use this idea to sample from a probability distribution. The random walk Metropolis–Hastings algorithm for generating N samples from a region Ω is as follows:

1. Start at a point x_0 in the region Ω.
2. For $i = 0, 1, \ldots, N-1$:
    1. Generate a random step σx.
    2. Compute the new point y **near** x_i.
    3. If y is in the region Ω,
        1. Set $x_{i+1} = y$.

4. Else,
    1. Set $x_{i+1} = x_i$.
3. Return the sequence of points $x_0, x_1, \ldots, x_{N-1}$.

This is an example of a Metropolis–Hastings algorithm. The acceptance criterion is that the *new point y should be in the region Ω.*

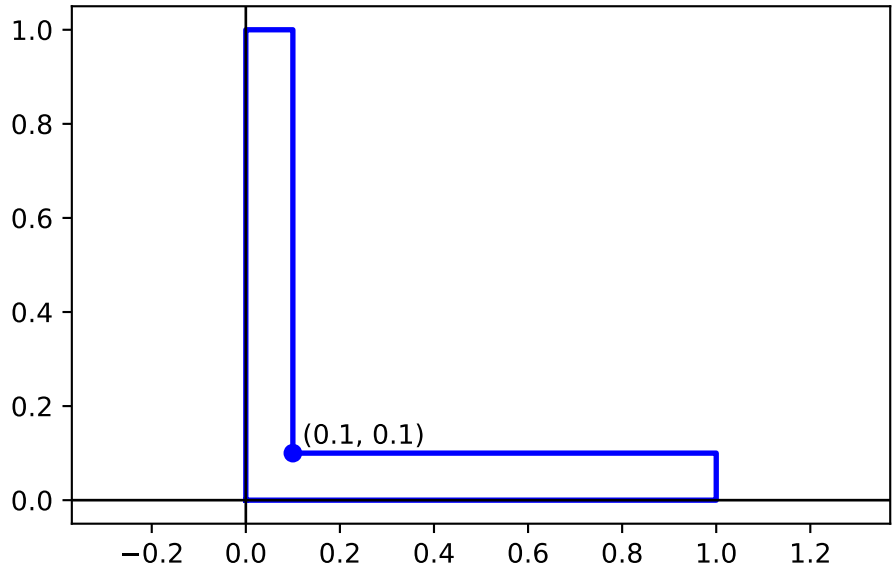
11.1.1 Proposal Distribution

The missing piece in the above algorithm is the method to generate a new point y near the current point x_i. This is done using a **proposal distribution**. The proposal distribution can be almost of any form, but it should be easy to sample from. The choice of the proposal distribution is crucial for the efficiency of the algorithm. A good proposal distribution should be able to explore the region Ω efficiently.

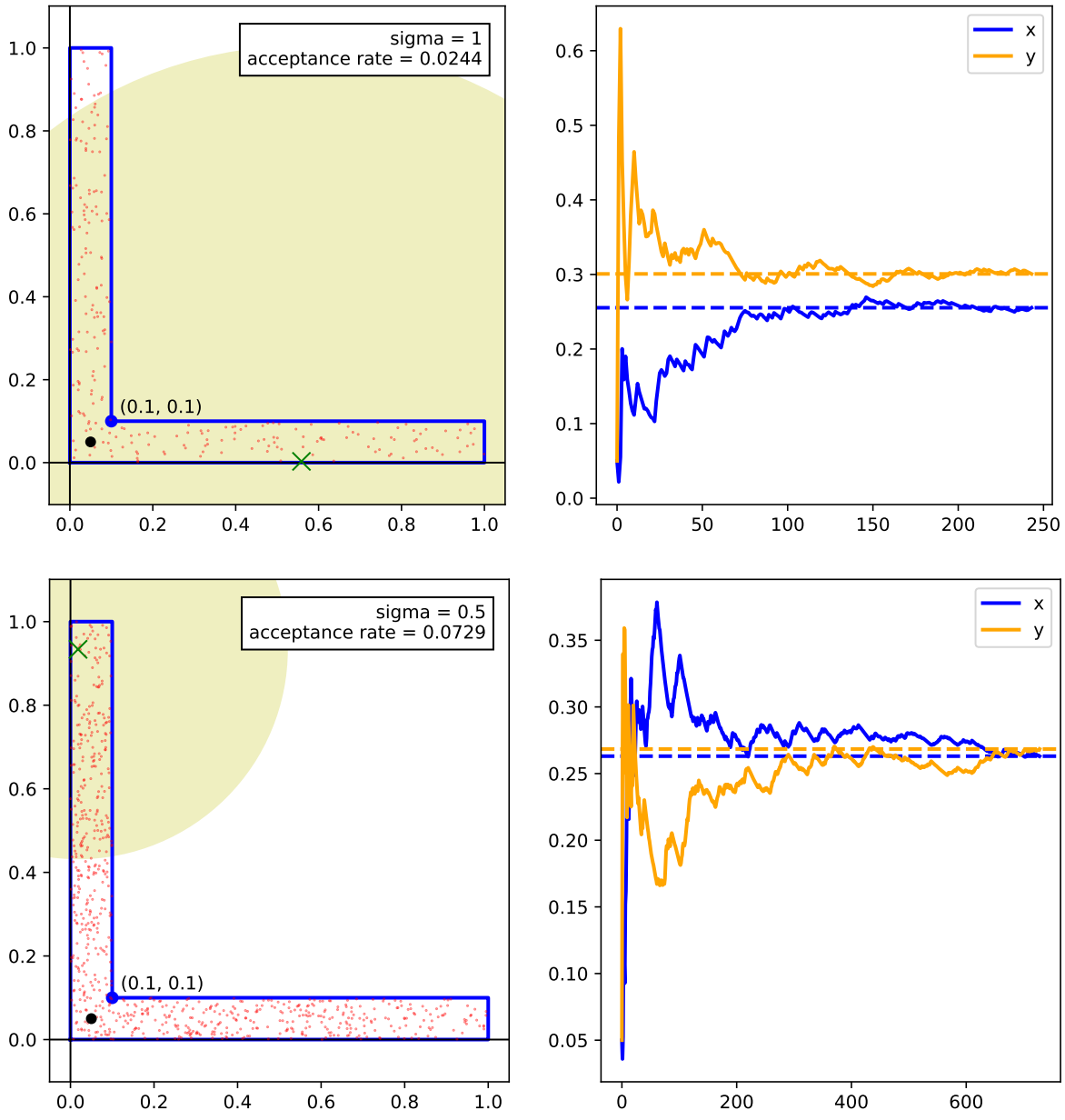
In the case of random walks, there are two common choices for the proposal distribution:

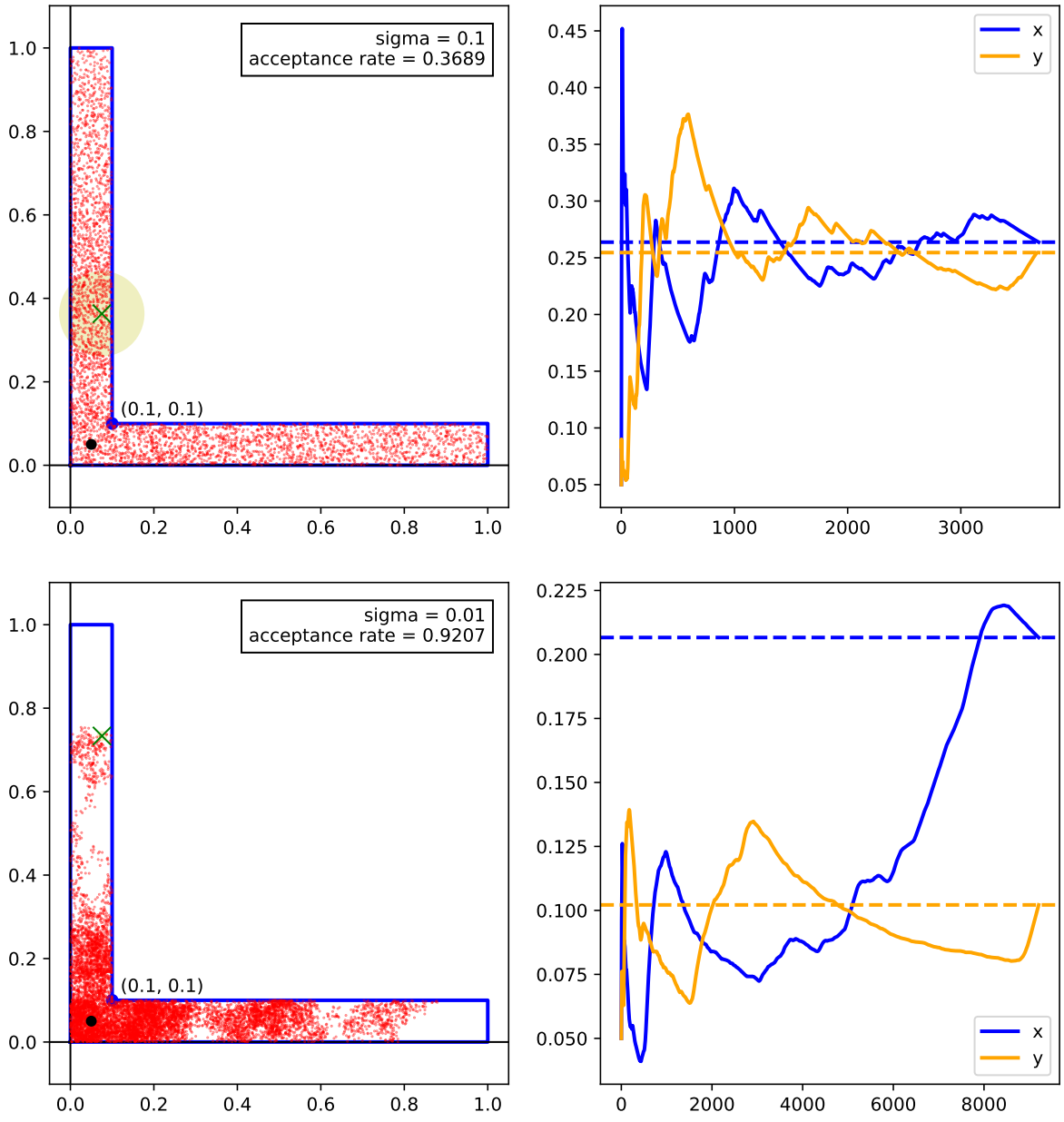
1. **Gaussian proposal**: We sample a new point from a Gaussian distribution centered at the current point x_i with a certain variance.
2. **Uniform proposal**: We sample a new point from a uniform distribution in a neighborhood of the current point x_i.

Example 11.1. Consider the L-shaped region in $\mathbb{R}^2$ as shown below. This is a non-convex region and rejection sampling would only provide an efficiency of $0.1 + 0.1 - 0.01 = 0.18$. We'll use the Metropolis–Hastings algorithm to sample from this region with higher efficiency.



The images below show scatter plots for the samples generated using the Metropolis–Hastings algorithm with four different choices of the Gaussian proposal distribution with standard deviations 1, 0.5, 0.2, and 0.01. The plots to the right are the running averages of the x and y coordinates. The scatter plot and the running averages provide a visual representation of the convergence of the samples to the target distribution.





11.1.2 Choosing the Proposal Distribution

Notice that there is a tradeoff between the acceptance rate and the rate of convergence. In the above example,

- The proposal distribution with $\sigma = 1$ has the lowest acceptance rate but the samples are well spread out and the underlying markov chain converges quickly.
- The proposal distribution with $\sigma = 0.01$ has the highest acceptance rate but the samples are clustered and the underlying markov chain converges slowly. In the above simulation, even after 10^5 samples (most of which are accepted), the samples are still clustered.

Both of these are undesirable. We need to choose a proposal distribution that has a good balance between the acceptance rate and the rate of convergence.

We see that for $\sigma = 0.5$, the samples are well spread out and the underlying markov chain converges quickly to the target distribution. This is a good choice for the proposal distribution. However, the acceptance rate is 0.07 which is lower than the acceptance rate for naive rejection sampling AND the generated samples are correlated. It is better to use rejection sampling than to use this proposal distribution.

For $\sigma = 0.2$, the acceptance rate is 0.3 which is better than the acceptance rate for naive rejection sampling. The samples are well spread out and the underlying markov chain converges quickly, although not as quickly as for $\sigma = 0.5$. For this example, $\sigma = 0.2$ is a good choice for the proposal distribution. We can fine tune this parameter to get a better acceptance rate, if needed.

11.2 Metropolis–Hastings Algorithm

The general algorithm for the Metropolis–Hastings algorithm is an algorithm for generating a sequence of samples from a probability distribution $p(x)$.

11.2.1 Structure of Metropolis–Hastings Algorithm

The above example illustrates the general structure of the Metropolis–Hastings algorithm for sampling from a probability distribution $p(x)$. The algorithm is as follows:

1. **Initialization**: Start at a point x_0.
2. For $i = 0, 1, \ldots, N-1$:
    1. Generate a proposal y close to x_i.
    2. Evaluate the acceptance ratio $\alpha(y|x_i)$.
    3. Accept y with probability $\alpha(y|x_i)$.
        1. If accepted, set $x_{i+1} = y$.
        2. Else, set $x_{i+1} = x_i$.
3. Return the sequence of points $x_0, x_1, \ldots, x_{N-1}$.

11.2.2 Proposal Distribution

For the MH algorithm, we need to specify a **proposal distribution** for generating "nearby points". This is a joint distribution $q(x, y)$ for two random variables X and Y. However, for running the algorithm we only need the conditional distribution $q(y|x)$. As such, it's more common to say that the proposal distribution is the conditional distribution $q(y|x)$. We think of x as the current point and y as the proposed point so that the proposal distribution $q(y|x)$ is *the distribution of the proposed point y given the current point x.*

In the Example 11.1, the proposal distribution $q(y|x)$ was a Gaussian distribution centered at x with a chosen variance,

$$q(y|x) = \mathcal{N}(y; x, \sigma^2 I).$$

The proposal distribution can be any distribution that is easy to sample from. The choice of the proposal distribution is crucial for the efficiency of the algorithm.

11.2.3 Acceptance Criterion

Once we generate a proposal y, we need to evaluate an acceptance criterion for w. The acceptance criterion is based on the ratio of the target distribution $p(x)$ and the proposal distribution $q(y|x)$. We calculate the acceptance ratio as

$$\alpha(y|x) = \min\left\{\frac{p(y)}{p(x)} \cdot \frac{q(x|y)}{q(y|x)}, 1\right\}.$$

The complete Metropolis–Hastings algorithm is as follows:

1. **Initialization**: Start at a point x_0.
2. For $i = 0, 1, \ldots, N-1$:
    1. Generate a proposal y from the proposal distribution $q(y|x_i)$.
    2. Compute the acceptance ratio
    $$\alpha(y|x) = \min\left\{\frac{p(y)}{p(x)} \cdot \frac{q(x|y)}{q(y|x)}, 1\right\}.$$
    3. Accept y with probability $\alpha(y|x)$:
        - If y is accepted, set $x_{i+1} = y$.
        - Otherwise, set $x_{i+1} = x_i$.
3. Return the sequence of points $x_0, x_1, \ldots, x_{N-1}$.

We can expand the algorithm further as follows:

1. **Initialization**: Start at a point x_0.
2. For $i = 0, 1, \ldots, N-1$:
    1. Generate a proposal $y \sim q(y|x_i)$.
    2. Compute
    $$\alpha_1 = p(y) \cdot q(x_i|y), \quad \alpha_2 = p(x_i) \cdot q(y|x_i).$$
    3. If $\alpha_1 \geq \alpha_2$,
        1. Set $x_{i+1} = y$.
    4. Else,
        1. Generate a random number $u \sim \text{Uniform}(0, 1)$.
        2. If $u < \alpha_1/\alpha_2$,
            1. Set $x_{i+1} = y$.
        3. Else,
            1. Set $x_{i+1} = x_i$.
3. Return the sequence of points $x_0, x_1, \ldots, x_{N-1}$.

11.3 Symmetric Proposal Distribution

Often we use a symmetric proposal distribution $q(y|x) = q(x|y)$. In this case, the acceptance ratio simplifies to

$$\alpha(y|x) = \min\left\{\frac{p(y)}{p(x)}, 1\right\}.$$

The Metropolis algorithm is a special case of the Metropolis–Hastings algorithm where the proposal distribution is symmetric. The Metropolis algorithm is widely used in practice. The algorithm is as follows:

1. **Initialization**: Start at a point x_0.
2. For $i = 0, 1, \ldots, N-1$:
    1. Generate a proposal $y \sim q(y|x_i)$.
    2. If $p(y) \geq p(x_i)$,
        1. Set $x_{i+1} = y$.
    3. Else,
        1. Generate a random number $u \sim \text{Uniform}(0, 1)$.
        2. If $u < p(y)/p(x_i)$,
            1. Set $x_{i+1} = y$.

3. Else,
    1. Set $x_{i+1} = x_i$.
3. Return the sequence of points $x_0, x_1, \dots, x_{N-1}$.

Example 11.2. If $p(x)$ is a uniform distribution over the region Ω, then the Metropolis algorithm reduces to the random walk Metropolis–Hastings algorithm. In this case, the acceptance ratio becomes

$$\alpha(y|x) = \begin{cases} 1, & \text{if } y \in \Omega, \\ 0, & \text{if } y \notin \Omega. \end{cases}$$

This means that we always accept a proposal y if it is in the region Ω and reject it otherwise. This is equivalent to the random walk Metropolis–Hastings algorithm in Section 11.1.

Several other algorithms, such as rejection sampling, random walks on graphs, and Gibbs sampling, can be viewed as special cases of the Metropolis–Hastings algorithm.

11.4 Metropolis Markov Chain

If the target distribution $p(x)$ is defined over the sample space Ω, then the Metropolis–Hastings algorithm generates a Markov chain with state space Ω.

For simplicity,

1. we will analyze the Metropolis algorithm with a symmetric proposal distribution $q(y|x) = q(x|y)$, and
2. we will assume that the target distribution $p(x)$ is defined over a finite sample space Ω.

The transition matrix for the Markov chain is given by

$$\mathbb{P}(X = b|X = a) = \begin{cases} q(b|a) & \text{if } p(b) \geq p(a) \text{ and } a \neq b, \\ \frac{p(b)}{p(a)} \cdot q(b|a) & \text{if } p(b) < p(a) \text{ and } a \neq b, \\ q(a|a) + \sum\limits_{b \in \Omega, p(b) < p(a)} q(b|a) \cdot \left(1 - \frac{p(b)}{p(a)}\right) & \text{if } a = b. \end{cases}$$

One can check that this defines a valid transition matrix by showing that the sum of the transition probabilities for each state is 1.

$$
\begin{aligned}
&\sum_{b\in\Omega} \mathbb{P}(X=b|X=a) \\
&= \sum_{p(b)\geq p(a), a\neq b} q(b|a) + \sum_{p(b)<p(a)} \frac{p(b)}{p(a)} \cdot q(b|a) + q(a|a) + \sum_{p(b)<p(a)} q(b|a) \cdot \left(1 - \frac{p(b)}{p(a)}\right) \\
&= \sum_{b\in\Omega} q(b|a) \\
&= 1.
\end{aligned}
$$

11.4.1 Detailed Balance Equations

Theorem 11.1. *The Metropolis algorithm is reversible with respect to the target distribution $p(x)$. This means that the Markov chain defined by the Metropolis algorithm satisfies the detailed balance equations:*

$$p(a) \cdot \mathbb{P}(X=b|X=a) = p(b) \cdot \mathbb{P}(X=a|X=b).$$

Proof. We only need check the detailed balance equations for the case when $a \neq b$. The case when $a = b$ is trivial. We make two cases:

1. **Case 1**: $p(a) \neq p(b)$.

Without loss of generality, assume that $p(a) < p(b)$. Then we have

$$
\begin{aligned}
p(a) \cdot \mathbb{P}(X=b|X=a) &= p(a) \cdot q(b|a) \\
&= p(a) \cdot q(a|b) \\
&= p(b) \cdot q(a|b) \frac{p(a)}{p(b)} \\
&= p(b) \cdot \mathbb{P}(X=a|X=b).
\end{aligned}
$$

2. **Case 2**: $p(a) = p(b)$. In this case, we have

$$
\begin{aligned}
p(a) \cdot \mathbb{P}(X=b|X=a) &= p(a) \cdot q(b|a) \\
&= p(b) \cdot q(a|b) \\
&= p(b) \cdot \mathbb{P}(X=a|X=b).
\end{aligned}
$$

Thus, in both cases, we have shown that the detailed balance equations hold. This completes the proof.

□

12 Variance Reduction

12.0.1 Markov Chain Monte Carlo (MCMC)

When the random variables X_i are not independent, the variance of the sum is given by

$$\mathrm{Var}(\widehat{\ell}) = \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{Var}(f(X_i)) + \frac{2}{N^2}\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}\mathrm{Cov}(f(X_i), f(X_j)).$$

There are n^2 terms in the covariance sum, and in general, we cannot guarantee that the covariance is small. So the above confidence interval is not valid. In the case of ergodic Markov chains, the central limit theorem for Markov chains allows us to estimate the CI as

$$\left[\widehat{\ell} - z_{1-\alpha/2}\frac{S_{eff}}{\sqrt{N_{eff}}}, \widehat{\ell} + z_{1-\alpha/2}\frac{S_{eff}}{\sqrt{N_{eff}}}\right],$$

where N_{eff} is the effective sample size. We will not discuss the details of effective sample size here.

Example: Estimating π. In the first example we saw for estimating π, we used the Monte Carlo method to estimate π by simulating random points in a square and counting the number of points that fall within a circle. The estimator we used was

$$\widehat{\ell} = \frac{4}{N}\sum_{i=1}^{N} I(X_i),$$

where $I(X_i)$ is an indicator function that is 1 if the point X_i is inside the circle and 0 otherwise. The variance of this estimator is given by

$$\mathrm{Var}(\widehat{\ell}) = \frac{16}{N}\left(\frac{\pi}{4}\left(1-\frac{\pi}{4}\right)\right).$$

Example: Estimation of Rare-Event Probabilities. Consider estimation of the tail probability $\ell = P(X > \gamma)$ of some random variable X for a large number γ. We can use the following estimator:

$$\widehat{\ell} = \frac{1}{N}\sum_{i=1}^{N} I_{>\gamma}(X_i),$$

where $I_{>\gamma}(X_i)$ is an indicator function that is 1 if $X_i > \gamma$ and 0 otherwise. The variance of this estimator is given by

$$\mathrm{Var}(\hat{\ell}) = \frac{1}{N}\left(\ell(1-\ell)\right).$$

The relative width of the confidence interval is given by

$$\text{Relative Width} = \frac{2z_{1-\alpha/2}\sqrt{\ell(1-\ell)}}{\hat{\ell}\sqrt{N}} \approx \frac{2z_{1-\alpha/2}}{\sqrt{N}}\sqrt{\frac{1-\ell}{\ell}} \approx \frac{2z_{1-\alpha/2}}{\sqrt{N\ell}}$$

When ℓ is small, the relative width of the confidence interval is large. This means that we need a large number of samples to get a good estimate of ℓ.

Example: Magnetization in a 2D Ising Model. In a previous worksheet, we used Gibbs sampling and Metropolis–Hastings sampling to estimate the magnetization of a 2D Ising model. We used the simple estimator

$$\hat{M} = \frac{1}{N}\sum_{i=1}^{N} M(\sigma_i),$$

where $M(\sigma_i)$ is the magnetization of the i-th sample.

The samples σ_i are not independent, and so the variance of the estimator will be

$$\mathrm{Var}(\hat{M}) = \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{Var}(M(\sigma_i)) + \frac{2}{N^2}\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}\mathrm{Cov}(M(\sigma_i), M(\sigma_j)),$$

which leads to much larger confidence intervals than we would expect from independent samples.

12.1 Importance Sampling

Importance sampling is a method to reduce the variance of an estimator by changing the distribution from which we sample. The idea is to reduce the number of low probability events that contribute to the variance of the estimator.

For example, when estimating the tail probability $\ell = P(X > \gamma)$, if ℓ is small, then most of the samples will be in the region $X \leq \gamma$, which contributes little to the estimate. However, we cannot just sample from the tail of the distribution as this would provide us no information about the rest of the distribution. Importance sampling allows us to sample from tail but then "fix" the estimate by weighting the samples appropriately.

Definition: Importance Sampling. Let X be a random variable with probability density function (pdf) $p(x)$, and let $q(x)$ be a proposal pdf such that $q(x) > 0$ for all x in the support of $p(x)$. The importance sampling estimator of $\ell = E[f(X)]$ is given by

$$\hat{\ell} = \frac{1}{N} \sum_{i=1}^{N} f(X_i) \frac{p(X_i)}{q(X_i)},$$

where $X_1, X_2, \ldots, X_N$ are i.i.d. samples from the distribution with pdf $q(x)$.

For clarity, we'll denote the estimator in **?@eq-crude-estimator** as $\hat{\ell}_{crude}$ and the importance sampling estimator as $\hat{\ell}_{IS}$.

Suppose $N = 1$ so that the estimator is given by

$$\hat{\ell}_{IS} = f(X) \frac{p(X)}{q(X)}.$$

Note that here $X \sim q(x)$ and NOT $p(x)$. We can check that this estimator is unbiased:

$$\begin{aligned} E_q[\hat{\ell}_{IS}] &= E_q \left[f(X) \frac{p(X)}{q(X)} \right] \\ &= \int f(x) \frac{p(x)}{q(x)} q(x) dx \\ &= \int f(x) p(x) dx \\ &= E[f(X)] \\ &= \ell. \end{aligned}$$

However,

$$\mathrm{Var}(\hat{\ell}_{IS}) \neq \mathrm{Var}(\hat{\ell}_{crude}).$$

This allows us to reduce the variance of the estimator by choosing $q(x)$ appropriately.

Suppose $f(X)$ is a non-negative function. Then if we choose

$$q(x) \propto p(x) f(x),$$

then the importance sampling estimator for $N = 1$

$$\hat{\ell}_{IS} = f(X) \frac{p(X)}{q(X)}$$

is a constant and has zero variance! When H is not non-negative, we can show that

$$q(x) \propto p(x)|f(x)|$$

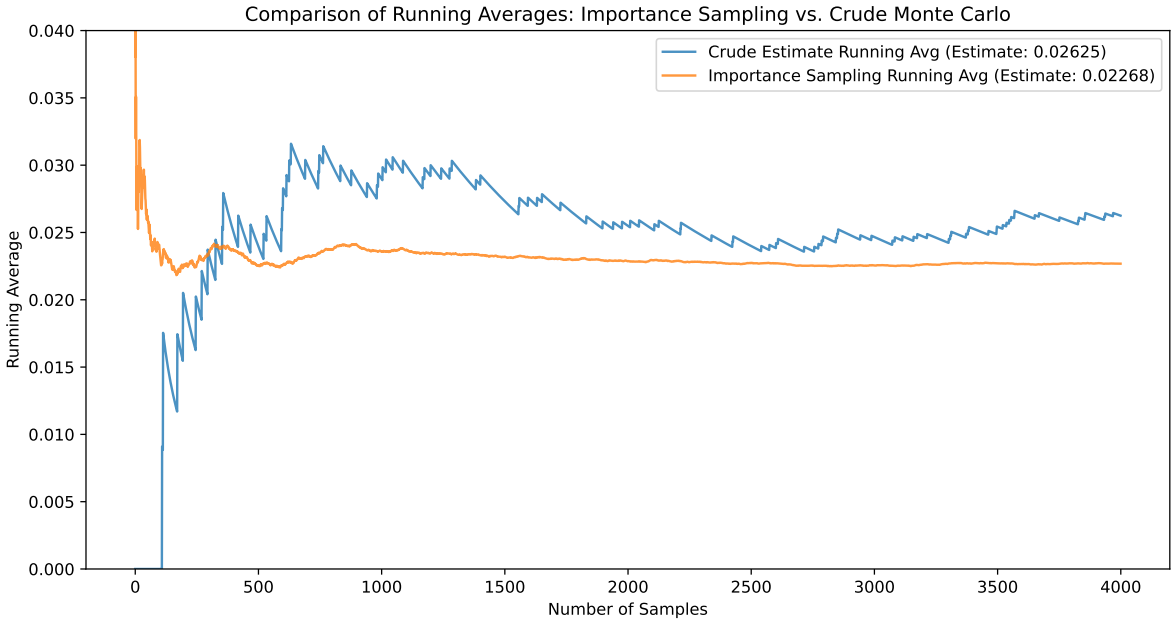
minimizes the variance of the estimator $\hat{\ell}_{IS}$.

However, note that our goal is to estimate $\ell = E[f(X)]$, which means that we do not know $f(x)$ in advance. So we cannot choose this $q(x)$ in advance. Even if we could, we might not be able to sample from $q(x)$ easily. In practice, we choose $q(x)$ to be a distribution that is easy to sample from and that is "close" to $p(x)$ in some sense.

Example: Importance Sampling for Rare Events. Consider the problem of estimating the tail probability $\ell = P(X > \gamma)$ for a random variable X with standard normal distribution. We can use importance sampling to estimate this probability by choosing a proposal distribution $q(x)$ that is concentrated in the tail region. One such distribution is the exponential distribution with parameter λ, which has pdf

$$q(x) = \lambda e^{-\lambda(x-2)}, \quad x \geq 2.$$

The plots below show the running averages of the crude and importance sampling estimators for $N = 2000$ samples. The importance sampling estimator is much more stable and converges to the true value of ℓ much faster than the crude estimator.




```
Variance of Importance Sampling Estimate: 0.00000
Relative Error of Importance Sampling Estimate: 0.01233
Variance of Crude Monte Carlo Estimate: 0.00001
Relative Error of Crude Monte Carlo Estimate: 0.09630
```

12.1.1 Remarks

1. The optimal choice of the proposal distribution $q(x)$ is not always easy to find. Even if we can find it, we may not be able to sample from it easily. For importance sampling algorithm, we need to be able to sample from $q(x)$. Often, we use a distribution that is easy to sample from and that is "close" to $|f(x)|p(x)$ in some sense.
2. Unlike rejection sampling and MCMC methods, for importance sampling we need to know the normalizing constant of the proposal distribution $q(x)$ in order to compute the weights. This means that we are fairly limited in the choice of $q(x)$. Some common choices are the exponential distribution, the normal distribution, and the uniform distribution, and a mixture of these distributions.
3. In order for the estimator to be well-defined, we need to ensure that $q(x) > 0$ for all x in the support of $f(x)p(x)$. As in the case of rare-event estimation, the support of $f(x)p(x)$ may be very small compared to the support of $p(x)$. We only need $q(x)$ to be positive in this smaller region.

12.2 Antithetic and Control Random Variates

In this section, we will discuss two methods of variance reduction that are based on the idea of using correlated random variables: **antithetic variables** and **control variates**. Recall that the variance of a sum of two random variables is given by

$$\text{var}(X + Y) = \text{var}(X) + \text{var}(Y) + 2\text{cov}(X, Y).$$

Oftentimes, having correlated random variables in undesirable as it reduces to an increase in variance and a decrease in the effective sample size. However, in some cases, we can use this correlation to our advantage.

12.2.1 Antithetic Variates

Antithetic variates are pairs of random variables that are negatively correlated. If we can find an estimator that uses sums to two antithetic random variables, we can reduce its variance.

Consider the example of estimating the integral from **?@sec-estimating-integrals**. Suppose $f(x)$ is a monotonic function over $[a, b]$. Then you'll show on the homework that if $X \sim U(a, b)$ then

$$\text{cov}(f(X), f(a + b - X)) \leq 0.$$

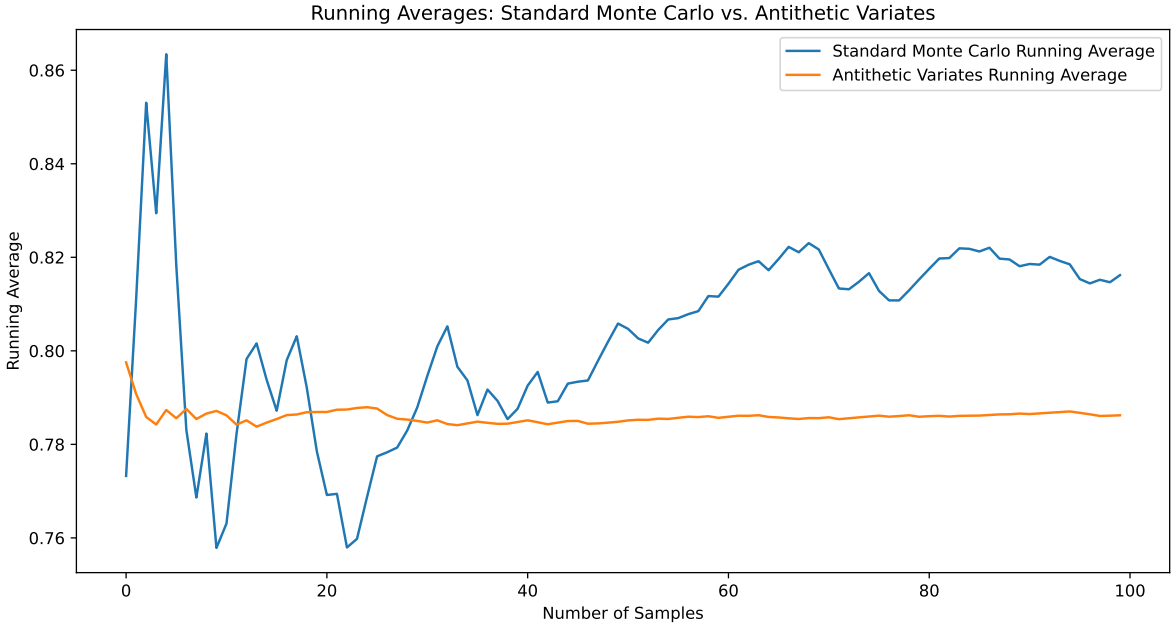
We can see intuitively why this is happening - if $f(x)$ is increasing the $f(b-x)$ is decreasing and vice versa, and hence the two are negatively correlated. In this case, we can reduce the variance of the crude estimator by instead using

$$\hat{\ell}_{anti} = \frac{(b-a)}{N} \sum_{i=1}^{2N} (f(X) + f(a+b-X))$$

Note that if $X \sim U(a,b)$ then so is $b-X$ and so $\hat{\ell}_{anti}$ is an unbiased estimator.

Theorem 12.1. $var(\hat{\ell}_{anti}) \leq var(\hat{\ell}_{crude})$.

Example 12.1. Example: Antithetic Variates. Consider the problem of estimating the integral $\ell = \int_0^1 (1+x^2)^{-1} dx$ using antithetic variates. The plots below show the running averages of the crude and antithetic variate estimators for $N = 100$ samples. The antithetic variate estimator achieves a $50x$ reduction in variance compared to the crude estimator.




```
Standard Monte Carlo Estimate: 0.795002, Variance: 1.270417e-04
Antithetic Variates Estimate: 0.786217, Variance: 1.951806e-06
Variance Reduction Factor: 65.09x
```

12.2.2 Control Variates

Control random variables are examples of random variables that are positively correlated. In this case, the difference

$$\text{var}(X - Y) = \text{var}(X) + \text{var}(Y) - 2\text{cov}(X, Y)$$

will have lower variance. Let $X \sim p(x)$. Suppose we want to estimate $\ell = \mathbb{E}[f(x)]$ for some function $f(x)$. We can use a control variate $Y = h(X)$ for some function $h(x)$ such that

1. $\mathbb{E}[h(X)]$ is known, say $\mathbb{E}[h(X)] = h_0$.
2. $h(x)$ is strongly positively correlated with $f(x)$, i.e., $\text{cov}(f(X), h(X)) \gg 0$.

Then we can use the control variate estimator

$$\hat{\ell}_{\text{CV}} = \frac{1}{n} \sum_{i=1}^{n} [f(X_i) - \beta(h(X_i) - h_0)]$$

where β is a constant. It is easy to see that $\hat{\ell}_{\text{CV}}$ is an unbiased estimator of ℓ. The variance of the control variate estimator is given by

$$\begin{aligned}
\text{var}(\hat{\ell}_{\text{CV}}) &= \frac{1}{n} \left[\text{var}(f(X)) + \beta^2 \text{var}(h(X)) - 2\beta \text{cov}(f(X), h(X))\right] \\
&= \frac{1}{n} \left[\text{var}(f(X)) + \beta \text{var}(h(X)) \left[\beta - 2\frac{\text{cov}(f(X), h(X))}{\text{var}(h(X))}\right]\right]
\end{aligned}$$

By choosing $\beta < 2\frac{\text{cov}(f(X), h(X))}{\text{var}(h(X))}$, we can reduce the variance of the control variate estimator. In practice, it is not easy to calculate the covariance between $f(X)$ and $h(X)$, so we

1. Pick a control variate $h(X)$ that is strongly correlated with $f(X)$, and whose expectation is known.
2. Experiment with different values of β to find the one that minimizes the variance of the control variate estimator.

Example 12.2. In the example below, we estimate the integral of xe^{-x} over the interval $[0, 1]$ using control function $g(x) = x$. We know that $\mathbb{E}[g(X)] = \frac{1}{2}$ so the estimator is given by

$$\hat{\ell}_{\text{CV}} = \frac{1}{n} \sum_{i=1}^{n} \left[f(X_i) - \beta(g(X_i) - \frac{1}{2})\right]$$

where β is a constant and $X_i \sim \text{Uniform}(0, 1)$ are i.i.d. We choose $\beta \approx 0.35$ which minimizes the variance of the control variate estimator. This results in an 8-fold reduction in variance compared to the naive estimator.

Part III

Applications

13 Stochastic Differential Equations

In this module, we will look at the Euler-Maruyama method for solving stochastic differential equations. Our focus will be on understanding numberical stability and convergence of the method.

Recall that the most basic differential equation, which is at the foundation of theory of ordinary differential equations, is the first order ordinary differential equation (ODE) of the form

$$\frac{dy}{dt} = \lambda y \tag{13.1}$$

where λ is a constant. The solution to this ODE is given by

$$y(t) = y(0)e^{\lambda t}. \tag{13.2}$$

We want to modify the ODE (Equation 13.1) to include noise

$$\frac{dy}{dt} = \lambda y + \text{ noise}.$$

To make sense of this, we'll make two changes:

1. We'll replace the function $y(t)$ with a stochastic process $X(t)$.
2. We'll model the noise as a Wiener process $W(t)$.

13.1 Stochastic Processes

A stochastic process is a time dependent random variable. More precisely, it is a function of the form

$$X(t, \omega) : \mathbb{R} \times \Omega \to \mathbb{R}$$

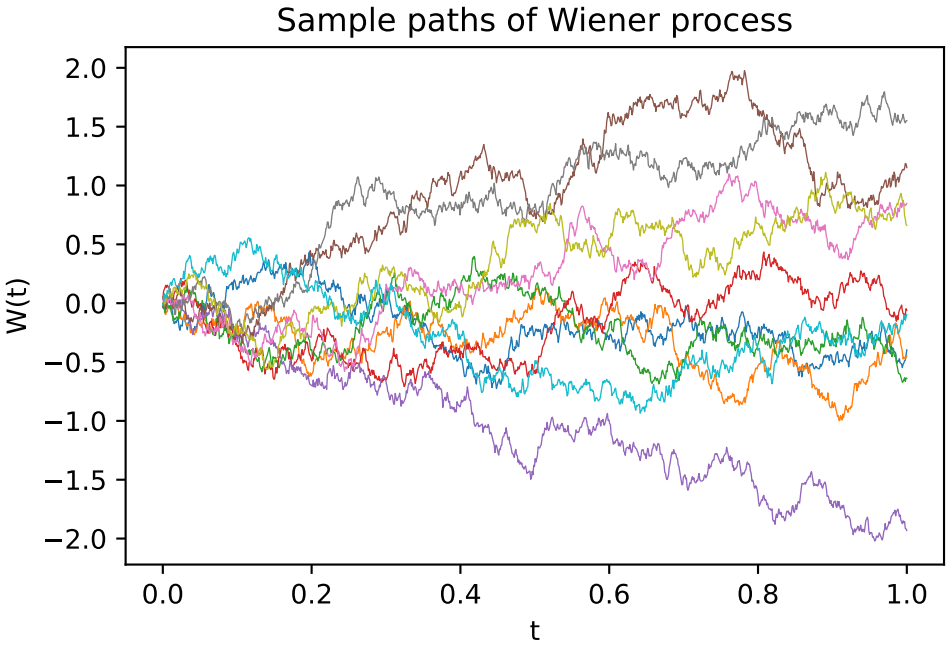
where Ω is the sample space. For example, if Ω is the set of gas molecules in a room, then $X(t, i)$ could be the position of the i-th molecule at time t. For at fixed t, $X(t, \cdot)$ is a random variable. For a fixed ω, $X(\cdot, \omega)$ is a function of time. This function, $X(\cdot, \omega)$ is called a **sample path** of the stochastic process.

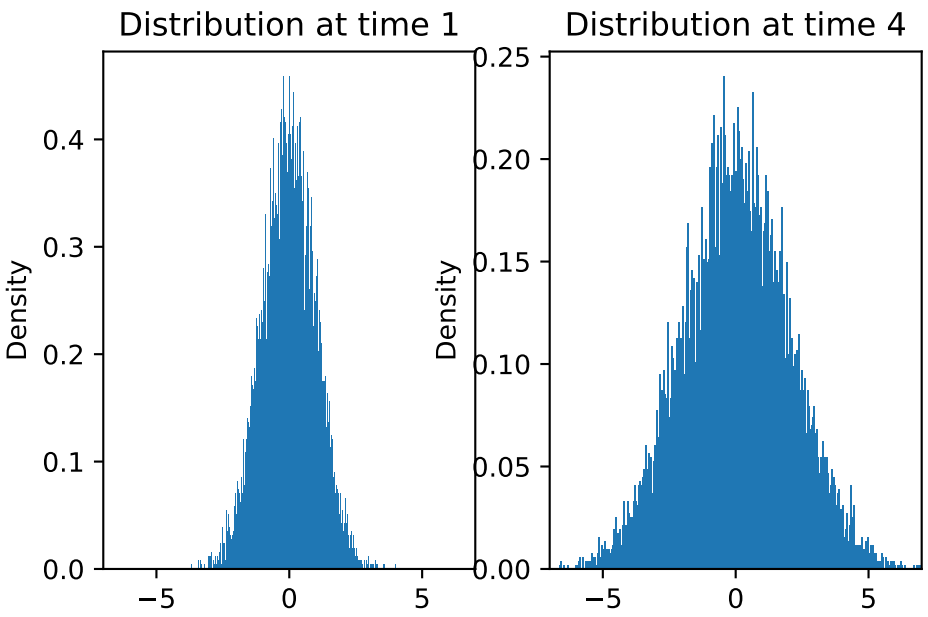
13.1.1 Wiener Process

A **Wiener process**, $W(t)$, also known as Brownian motion, is a stochastic process with the following properties:

1. $W(0) = 0$.
2. For $0 \leq s < t$, the increment $W(t) - W(s)$ is normally distributed with mean 0 and variance $t - s$.
3. The increments are independent.
4. The process is continuous but nowhere differentiable.

We can derive the properties of the Wiener process by taking the continuous limit of a random walk. Note that for all time t, the mean of $W(t)$ is 0. However the variance grows with time. Imagine a box of gas particles all starting at the origin without any initial velocity or external forces. As there is no external force or initial velocity, the center of mass of the gas particles will remain at the origin. However, the gas particles will spread out over time. The Wiener process models this spreading out of gas particles.





13.2 Stochastic Differential Equations

A **stochastic differential equation** (SDE) is an equation of the form

$$dX(t) = a(X(t), t)dt + b(X(t), t)dW(t)$$

where a and b are functions of $X(t)$ and t. The term $a(X(t), t)dt$ is the deterministic part of the equation and $b(X(t), t)dW(t)$ is the stochastic part. The solution to an SDE is a stochastic process $X(t)$. $W(t)$ is the Wiener process.

We write the equation in this form because the Wiener process is not differentiable. We interpret the equation as saying

$$X(t) - X(0) = \int_0^t a(X(s), s)ds + \int_0^t b(X(s), s)dW(s).$$

13.2.1 Ito vs Stratonovich Interpretation

We can try to define the above integrals in the usual sense using Riemann sums. The first integral can be written as the limit

$$\int_0^t a(X(s), s)ds = \lim_{n\to\infty} \sum_{i=0}^{n-1} a(X(t_i), t_i)\Delta t_i$$

where t_i is some point in the interval $[t_i, t_{i+1}]$ and $\Delta t = t/n$. Even though function $a(X(s), s)$ is a stochastic process, it is mathematically still just a function. The integral can be defined in the usual sense.

However, the second integral is more problematic. We can try to define it as

$$\int_0^t b(X(s), s)dW(s) = \lim_{n\to\infty} \sum_{i=0}^{n-1} b(X(t_i), t_i)(W(t_{i+1}) - W(t_i))$$

where t_i is some point in the interval $[t_i, t_{i+1}]$, $\Delta t = t/n$, and $W(t_{i+1}) - W(t_i)$ is normally distributed with mean 0 and variance Δt. The problem is that this integral does not converge in the usual sense. Where the sum converges depends on which point in the interval $[t_i, t_{i+1}]$ we choose to evaluate the integrand. This happens because the Wiener process is not differentiable. There are two commonly used interpretations of the integral:

1. **Ito Interpretation**: In this interpretation, we use the left end point of the interval to evaluate the integrand. This is the most common interpretation.
2. **Stratonovich Interpretation**: In this interpretation, we use the midpoint of the interval to evaluate the integrand.

It is possible to convert between the two interpretations using the Ito formula. We will assume the Ito interpretation in this module.

13.3 Geometric Brownian Motion

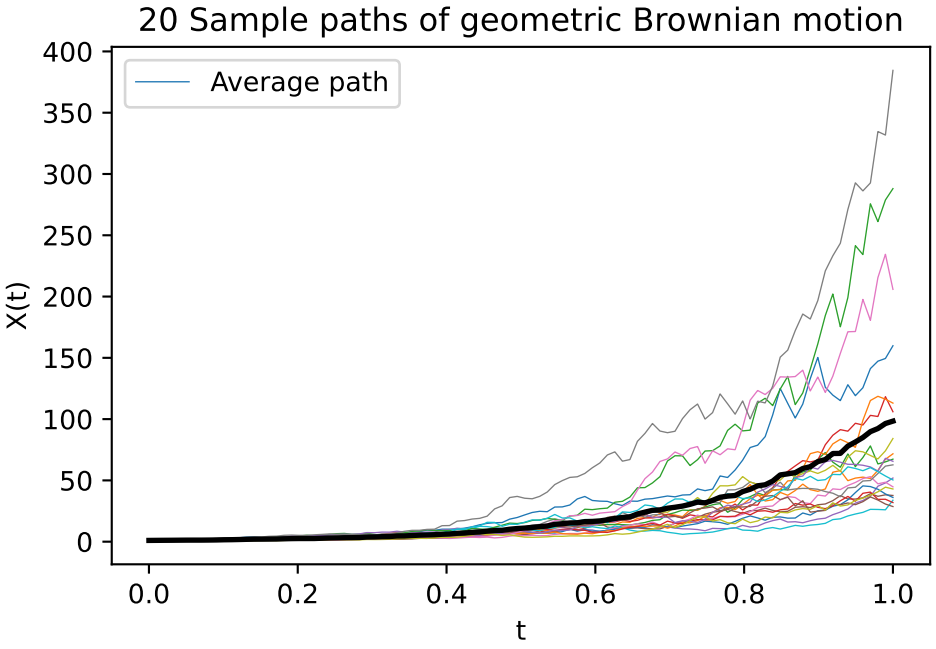
A simple example of a stochastic differential equation is the **geometric Brownian motion**. This is a model for the evolution of stock prices. The equation is

$$dX(t) = \mu X(t)dt + \sigma X(t)dW(t)$$

where μ is the drift and σ is the volatility. This is one of the few SDEs for which we can find an exact solution. The solution to the Ito version of the equation is

$$X(t) = X(0)e^{(\mu - \sigma^2/2)t + \sigma W(t)}. \tag{13.3}$$

This is a log-normal distribution whose mean is given by $X(0)e^{\mu t}$ and variance is given by $X(0)^2 e^{2\mu t}(e^{\sigma^2 t} - 1)$. Note that when $\sigma = 0$, this reduces to a standard ODE. Unlike (Equation 13.2), the solution to the SDE has a quadratic term $\sigma^2/2$ in the exponent. This term appears due to Ito's lemma, which is the stochastic analog of the chain rule.



13.4 Euler-Maruyama Method

Most SDEs do not have analytical solutions. We need to solve them numerically. Similar to ODEs, some simple regularity conditions on the coefficients a and b imply that the SDE has a unique solution. Most common SDEs satisfy these conditions. However, unlike ODEs, the solution to an SDE is a stochastic process. We can't just evaluate the solution at a few points to get an approximate solution. We need to generate several sample paths of the stochastic process to get an approximate solution.

The simplest method for solving SDEs is the **Euler-Maruyama method**. This is a stochastic analog of the Euler method for ODEs. The Euler-Maruyama method is a recursive method. Given the value of the stochastic process X_n at time t_n, we can find the value of the process at time $t_{n+1} = t_n + \Delta t$ using the formula

$$X_{n+1} = X_n + a(X_n, t_n)\Delta t + b(X_n, t_n)\Delta W_n$$

where $\Delta t = t_{n+1} - t_n$ and $\Delta W_n = W(t_{n+1}) - W(t_n) \sim \mathcal{N}(0, \Delta t)$. This results in a simple algorithm for solving SDEs:

1. Initialize X_0.
2. For $n = 0, 1, 2, \ldots, N-1$, do
    1. Generate a random number $\Delta W_n \sim \mathcal{N}(0, \Delta t)$.
    2. Compute $X_{n+1} = X_n + a(X_n, t_n)\Delta t + b(X_n, t_n)\Delta W_n$.
3. Return $X_0, X_1, \ldots, X_N$.

13.5 Convergence of the Euler-Maruyama Method

The EM method provides a numerical approximation to the solution of the SDE. We want to understand how good this approximation is.

Fix a time interval $[0, T]$. Divide the interval $[0, T]$ into N subintervals of length $\Delta t = T/N$. Let $t_i = i\Delta t$ so that $t_0 = 0$ and $t_N = T$. Let $X(t)$ be the analytical solution to the SDE at time t with initial condition $X(0)$. We first think of the EM method as generating a sequence of random variables $X_0, X_1, \ldots, X_N$ defined recursively by

$$\begin{aligned} X_0 &= X(0), \\ X_{i+1} &= X_i + a(X_i, t_i)\Delta t + b(X_i, t_i)\Delta W_i. \end{aligned}$$

where $\Delta W_i \sim \mathcal{N}(0, \Delta t)$.

Then we are interested in the question of how good the sequence $X_0, X_1, \ldots, X_N$ is as an approximation to the solution $X(t_0), X(t_1), \ldots, X(t_N)$. If the EM method is a good approximation method, we should get

$$X_i \to X(t_i) \text{ as } \Delta t \to 0.$$

However, as X_i and $X(t_i)$ are random variables, we need to be more precise about what we mean by convergence. There are two notions of convergence that we are interested in: weak convergence and strong convergence.

13.5.1 Weak Convergence

With the setup as above, we say that the EM method converges weakly to the solution over the interval $[0, T]$ if

$$\mathbb{E}[X_i] \to \mathbb{E}[X(t_i)] \text{ as } \Delta t \to 0.$$

Since we are interested in using the EM method to approximate the solution to the SDE, we need more than just convergence in expectation. We need to know how fast the method converges. We say that the EM method converges weakly to the solution over the interval $[0, T]$ with order p if

$$|\mathbb{E}[X_i] - \mathbb{E}[X(t_i)]| \leq C\Delta t^p$$

For some constant C that does not depend on Δt. Note that C is allowed to depend on T and $X(0)$. We often write this as

$$|\mathbb{E}[X_i] - \mathbb{E}[X(t_i)]| = O(\Delta t^p)$$

to emphasize the rate of convergence.

13.5.2 Strong Convergence

Strong convergence is the convergence of the sample paths. We say that the EM method converges strongly to the solution over the interval $[0, T]$ if

$$X_i \xrightarrow{a.s.} X(t_i) \text{ as } \Delta t \to 0.$$

Recall that convergence almost surely means that the probability of the event that the sequence X_i does not converge to $X(t_i)$ is 0. By Markov's inequality it is enough to say that the expected value of the distance between X_i and $X(t_i)$ goes to 0.

$$\mathbb{E}[|X_i - X(t_i)|] \to 0 \text{ as } \Delta t \to 0.$$

We say that the EM method converges strongly to the solution over the interval $[0, T]$ with order p if

$$\mathbb{E}[|X_i - X(t_i)|] = O(\Delta t^p)$$

In the homework assignment, you will derive the rates of convergence for the EM method for the geometric Brownian motion *empirically*. However, it is necessary to derive these rates theoretically as for arbitrary SDEs, it is not possible to write an exact analytical solution. The rates of convergence tell us how close the analytical solution is to the numerical solution.

Also, you'll only estimate p at the end of the interval i.e. $t = T$. This is technically not correct. We should find a p that works for each time step t_i. However, this is computationally expensive and the assumption is that error grows with time so that the error at the end of the interval is the largest.

13.6 Final remarks

The EM method has a slower rate of strong convergence compared to weak convergence. This is because even though the EM method is a natural extension of the Euler method for ODEs, there are second order terms in the Ito formula (chain rule for stochastic processes) that contribute to the error. This is fixed in the Milstein method, which is a second order method for solving SDEs. However the Milstein method is more computationally expensive compared to the EM method and has the same weak rate of convergence as the EM method.

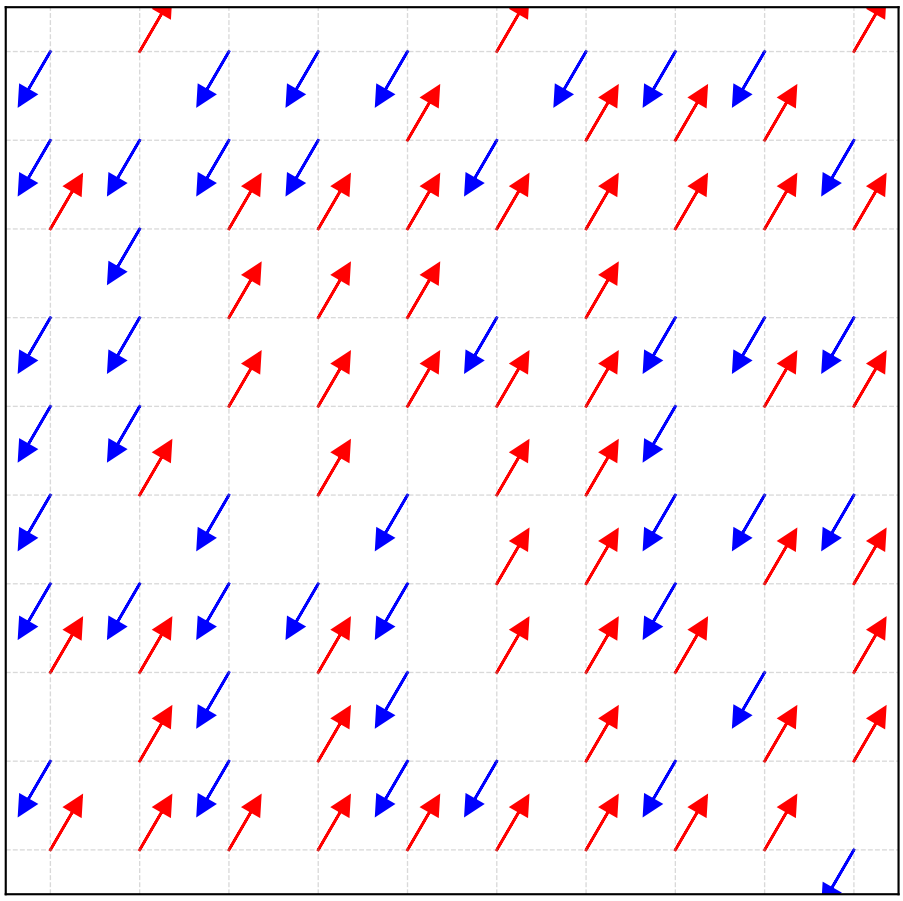
14 Ising Model

14.1 The Physical Ising Model

The Ising model is a simple model of ferromagnetism from statistical mechanics. The setup is as follows:

1. **Particles**: We have a lattice of N particles, each of which can be have one of two states of magnetization: spin up ($X = 1$) or spin down ($X = -1$). For simplicity, we assume the grid is "wrapped" so that the top and bottom edges are connected, and the left and right edges are connected. **This means that each site has four neighbors.** Such a grid is called a torus.
2. **Coupling constant**: Two adjacent particles have a tendency to have spins aligned. This tendency is quantified by a coupling constant J. If two adjacent particles have the same spin, the energy of the system is lowered by $-J$, and if they have opposite spins, the energy is raised by J. Any physical system tends to minimize its energy, so the system will tend to have neighboring particles with the same spin.
3. **External magnetic field**: For the same of simplicity we'll assume that there is no external force field.
4. **Temperature**: Temperature adds randomness to the system. At 0K, all the particles have their spins aligned. As the temperature increases, the particles start to flip their spins. The higher the temperature, the more likely it is for neighboring particles to have opposite spins.
5. **Magnetization**: If the system is in a state where most of the particles have the same spin, the system is said to be magnetized.

Next we create a mathematical model to describe the system.



14.2 The Mathematical Ising Model

We model the system using N^2 random variables $X_{i,j}$. Each $X_{i,j}$ is a binary random variable that takes the value 1 or -1. The configuration of the system is given by the N^2 length vector $\mathbf{X} = (X_{1,1}, X_{1,2}, \ldots, X_{N,N})$. There are 2^{N^2} possible configurations of the system i.e. the state space has size 2^{N^2}. We can imagine each configuration as being the vertex of a hypercube in N^2 dimensions with vertices $(\pm 1, \pm 1, \ldots, \pm 1)$. We'll let the variable σ denote a configuration of the system. So σ is a vector of length N^2 with entries in $\{-1, 1\}$.

In order to describe the system mathematically, we need to provide the joint probability distribution of the random variables $X_{i,j}$. This is done using the Hamiltonian of the system and the Boltzmann distribution.

14.2.1 Hamiltonian

The Hamiltonian of the system is given by

$$H(\sigma) = -J \sum_{(i,j) \text{ and } (i',j') \text{ are neighbors}} X_{i,j} X_{i',j'}$$

where the sum is over all pairs of neighboring particles. The Hamiltonian is just a measure of the energy of the system. Note that because J is positive, the energy is minimized when neighboring particles have the same spin.

14.2.2 Boltzmann Distribution

The probability of the system being in a particular configuration σ is given by the Boltzmann distribution:

$$f(\sigma) = \frac{e^{-H(\sigma)/T}}{Z}$$

where T is the temperature, and Z is the normalization constant called the partition function. This is the joint probability mass function of the random variables $X_{i,j}$. The partition function Z is not easy to compute, but thankfully we do not need it to sample from the joint distribution.

14.2.3 Low Temperature

If we plugin the partition function explicitly, we can see that the probability distribution becomes

$$f(\sigma) = \frac{e^{-H(\sigma)/T}}{\sum_{\sigma'} e^{-H(\sigma')/T}}.$$

As $T \to 0$, the term with the highest value of $-H(\sigma)$ will dominate the sum in the denominator. But the term with the highest value of $-H(\sigma)$ is the one that minimizes $H(\sigma)$. So,

$$\lim_{T \to \infty} f(\sigma) = \lim_{T \to \infty} \frac{e^{-H(\sigma)/T}}{e^{-H(\sigma)_{\min}/T}} = \lim_{T \to \infty} e^{(H(\sigma)_{\min} - H(\sigma))/T}.$$

If $H(\sigma)_{\min} = H(\sigma)$, then the probability of the system being in that state is 1 and 0 otherwise. This means that at low temperatures, the system will be in a state that minimizes the energy.

But as T increases, more and more high energy states become probable. So temperature can be thought of as adding variability to the system.

14.2.4 Magnetization

The magnetization of the system is given by

$$M(\sigma) = \frac{1}{N^2} \sum_{i,j} X_{i,j}$$

If the system is magnetized, then $M(\sigma)$ will be close to 1 or -1. Our goal is to find the expected value of the magnetization of the system. To do this we need to sample from the distribution $f(\sigma)$ and then compute the magnetization of each sample. The average of these magnetizations will be the expected magnetization of the system.

14.3 Gibbs Sampling

There are two natural ways of sampling from the distribution $f(\sigma)$: Metropolis-Hastings and Gibbs sampling. In the Metropolis-Hastings approach, we randomly choose a particle and flip its spin. We then accept or reject the new configuration based on the Metropolis-Hastings acceptance probability. In the Gibbs sampling approach, we update the entire grid of particles one at a time. We update each particle by sampling from the conditional distribution of that particle given the rest of the particles. In this notebook, we'll use Gibbs sampling.

14.3.1 Conditional Distribution

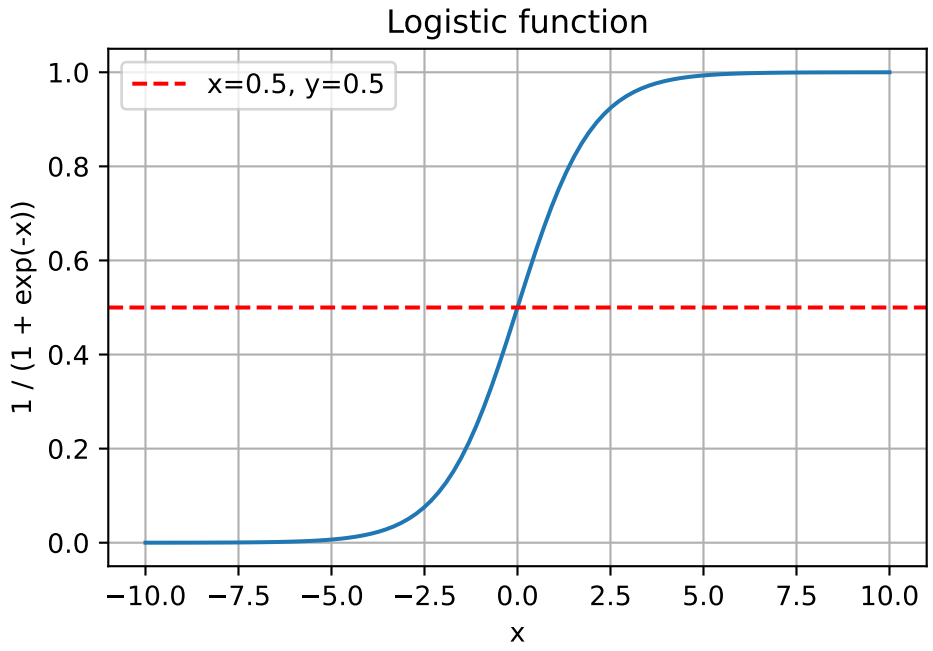
Consider a single spin $X_{i,j}$. Let $\hat{X}_{i,j}$ denote the set of spins at all sites except (i, j). We want to find the conditional distribution of $X_{i,j}$ given $\hat{X}_{i,j}$. We'll assume the following result without proof:

$$\begin{aligned} P(X_{i,j} = 1 | \hat{X}_{i,j}) &= \frac{1}{1 + \exp(-2h_{i,j}/T)}, \\ P(X_{i,j} = -1 | \hat{X}_{i,j}) &= 1 - P(X_{i,j} = 1 | \hat{X}_{i,j}). \end{aligned} \tag{14.1}$$

where T is the temperature of the system and $h_{i,j}$ is the effective field at site (i, j), given by

$$\begin{aligned} h_{i,j} &= J \sum_{(i',j') \in \text{neighbors}(i,j)} X_{i',j'} \\ &= J \left(X_{i-1,j} + X_{i+1,j} + X_{i,j-1} + X_{i,j+1} \right) \end{aligned}$$

The conditional distribution in Equation 14.1 is a logistic function. When $h_{i,j}$ is positive, the probability of $X_{i,j}$ being 1 is greater than 0.5, and when $h_{i,j}$ is negative, the probability of $X_{i,j}$ being 1 is less than 0.5. This makes sense because if the effective field is positive, most of the neighbors of (i, j) have spin 1, so it is likely that (i, j) will also have spin 1 and if the effective field is negative, most of the neighbors of (i, j) have spin -1, so it is likely that (i, j) will also have spin -1.



14.3.2 The Algorithm

1. Initialize the grid of spins X_i for $i = 1$ to N randomly.
2. For $i = 1$ to N
    1. For $j = 1$ to N
        1. Compute the effective field $h_{i,j}$.
        2. Generate a random number u from a uniform distribution.
        3. Set $X_{i,j}$ to 1 if $u < \dfrac{1}{1 + \exp(-2h_{i,j}/T)}$ and -1 otherwise.
3. Compute the magnetization of the system.
4. Repeat steps 2 and 3.

Equilibrium/Mixing Time:

Even when the grid size is modest, the dimensions of the state space is quite large, 2^{N^2}. This means that the Markov chain will take a long time to mix. Practically, this means that

the initial configuration might be far from the stationary distribution i.e. the system has not reached thermal equilibrium. In this case it is important to discard the initial samples to not skew the magnetization estimate.

One simple way to this is to run the Gibbs sampler until the running average of the magnetization stabilizes. At that point, we can start using the samples to estimate the magnetization.

14.4 Phase Transitions

It was discovered that the Ising model for a 2D grid exhibits a phenomenon called **phase transition**. If we slowly increase the temperature, we expect the magnetization to decrease. What is surprising is that the magnetization decreases smoothly until a certain temperature, after which it drops suddenly. This sudden drop is called a phase transition. The temperature at which this happens is called the **critical temperature**.

It is possible to calculate the critical temperature theoretically, in the limit when the size of the lattice $\to \infty$. This limiting critical temperature for a square lattice is given by

$$T_c = \frac{2J}{\log(1+\sqrt{2})}.$$

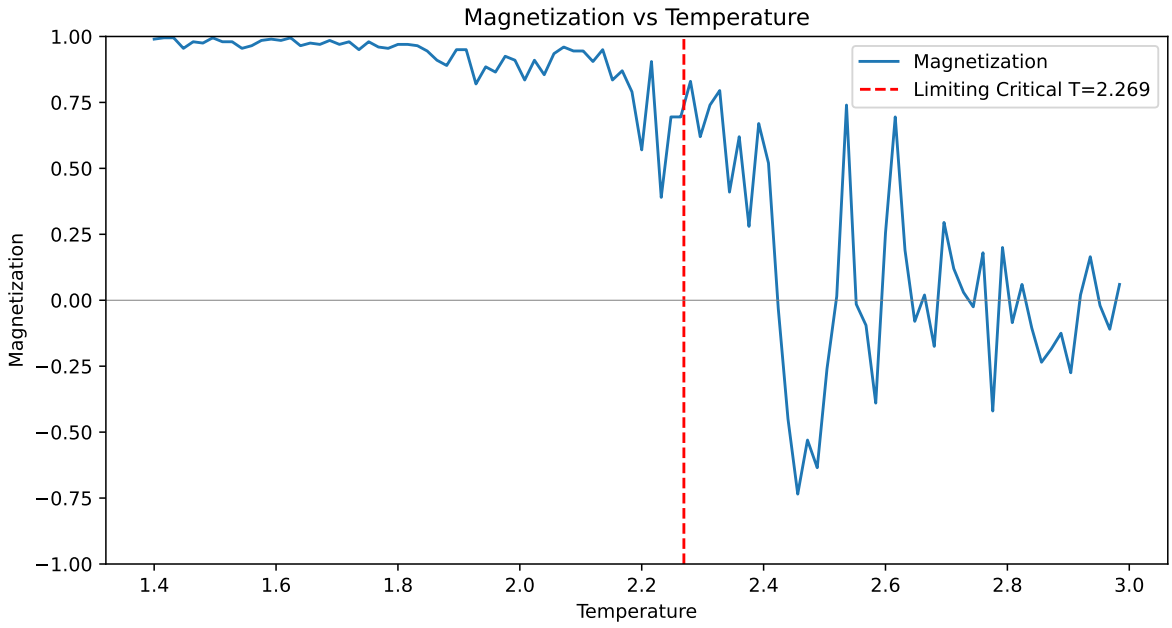
Of course, in our simulation, we'll be using finite sized lattices so we should expect some deviation from this result.

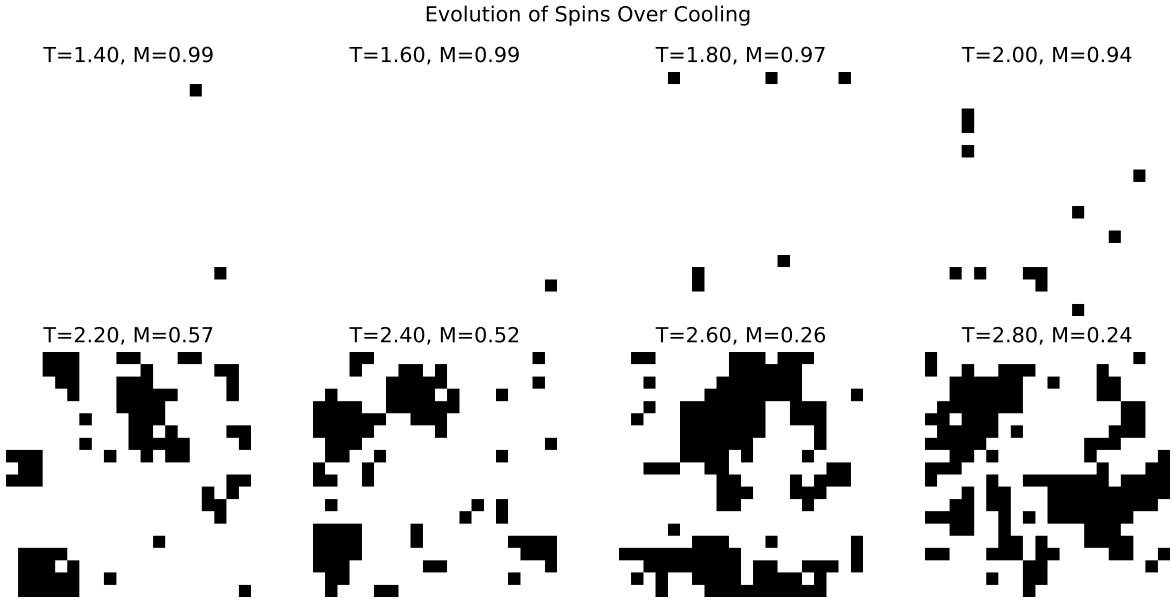
14.5 Simulation

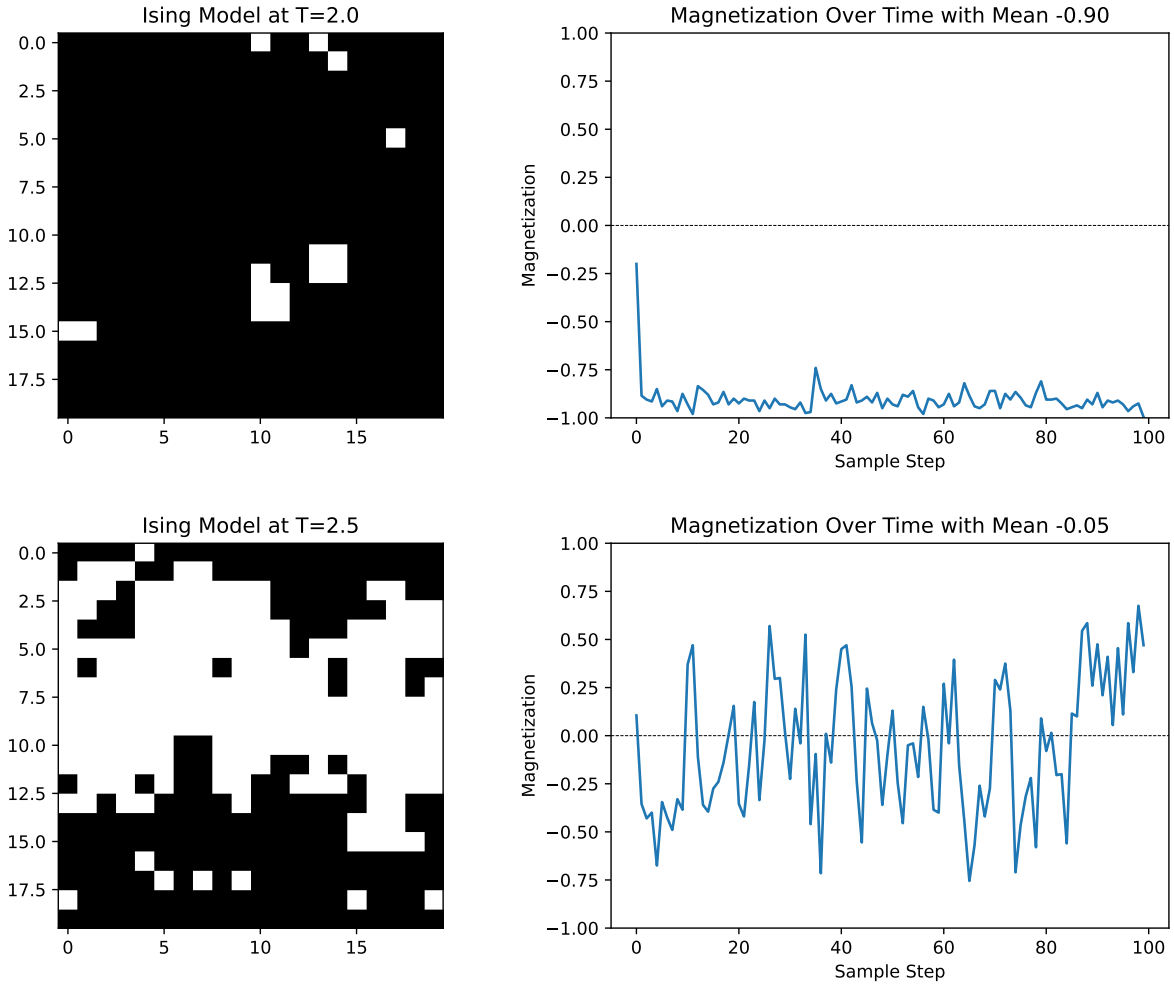
Below are the results of simulating the Ising model on a 20×20 grid for $J = 1$. The predicted critical temperature is $T_c = 2.269$.

We can see that at $T = 1.5$, the system is magnetized and the magnetization is more or less constant and close to 1 or -1 over various runs. At $T = 2.5$, the system is not magnetized and the average magnetization is close to 0. It fluctuates wildly over various runs.

If we gradually increase the temperature from $T = 1$ to $T = 3$, we can see that the magnetization decreases smoothly until $T = 2$ after which it drops suddenly. This is the phase transition. Below the graph are the snapshots of the grid at different temperatures. One can see that at low temperatures, the grid is mostly monochromatic, but at high temperatures, the grid is more chaotic.


Magnetization vs Temperature
Magnetization
Limiting Critical T=2.269
Magnetization
Temperature
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
1.4
1.6
1.8
2.0
2.2
2.4
2.6
2.8
3.0


Evolution of Spins Over Cooling
T=1.40, M=0.99
T=1.60, M=0.99
T=1.80, M=0.97
T=2.00, M=0.94
T=2.20, M=0.57
T=2.40, M=0.52
T=2.60, M=0.26
T=2.80, M=0.24



14.6 Final Remarks

Not all graphs show a phase transition. The Ising model in 1D has no phase transition at any temperature. It was conjectured that the Ising model in 2D has no phase transition either, but this was proven wrong when an analytical solution was found by Lars Onsager in 1944. The Ising model in 3D also has a phase transition, but the critical temperature is not known exactly. Most critical temperatures are only known for limiting cases when the size of the lattice goes to infinity. For finite cases, Monte Carlo simulations are the only way to estimate the critical temperature.

15 M/M/n Queue

The M/M/n queue is a discrete stochastic process that models a queue with n servers. There are three parameters:

- n: the number of servers
- λ: the arrival rate of customers
- μ: the service rate of each server

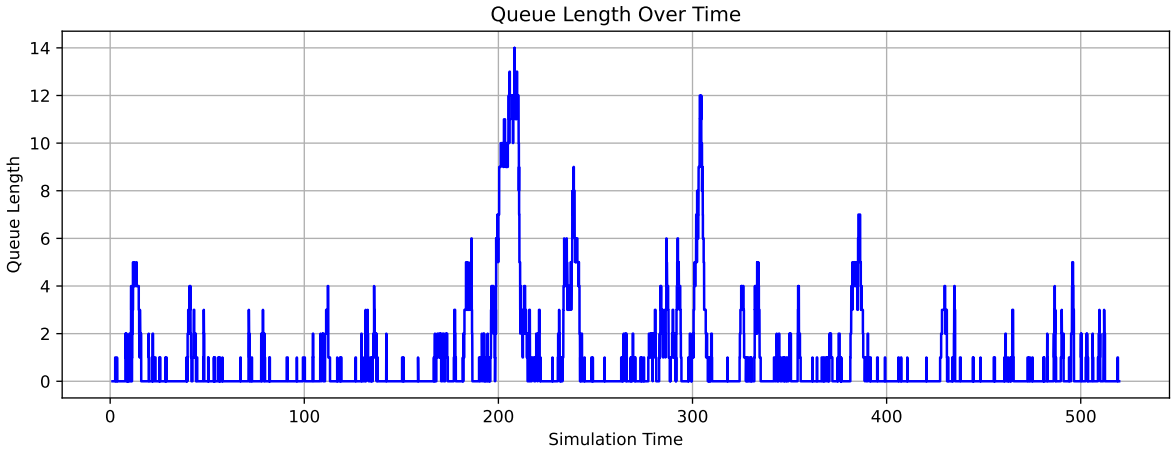
The "M" in M/M/n stands for "memoryless" or Markovian. The M/M/n queue models the following process: - customers arrive at the queue according to a Poisson process with rate λ. - Each customer is served by one of the n servers, and the service times are exponentially distributed with rate μ. - If all servers are busy, the customer waits in line until a server becomes available.

We are interested in testing the performance of the queue as function of the three parameters. There are various performance measures we can compute, including: - The average number of customers in the waiting line - The average time a customer spends waiting in line - The usage of the servers - The probability that a customer has to wait in line

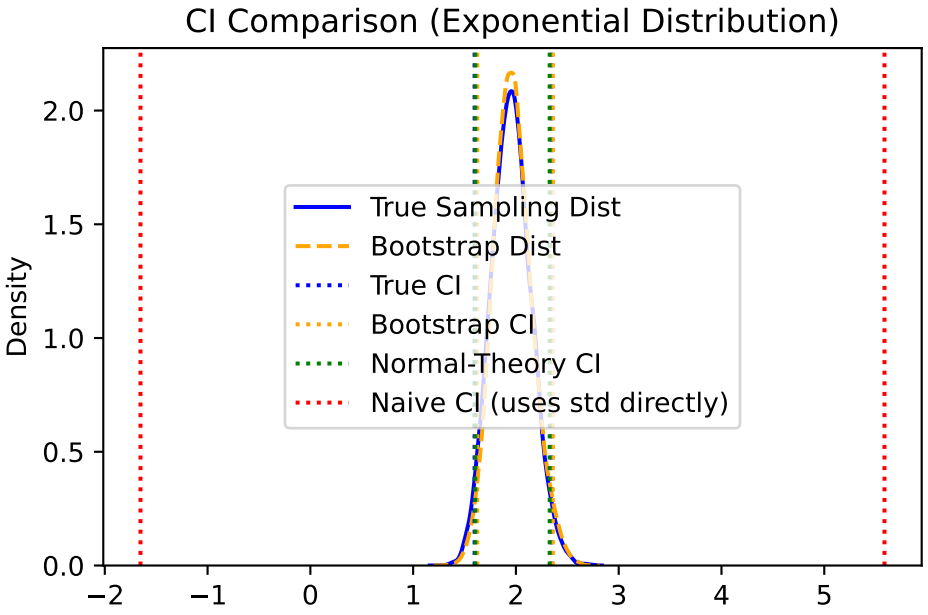
There are two approaches we can take to simulate an M/M/n queue

1. Process driven
2. Event driven

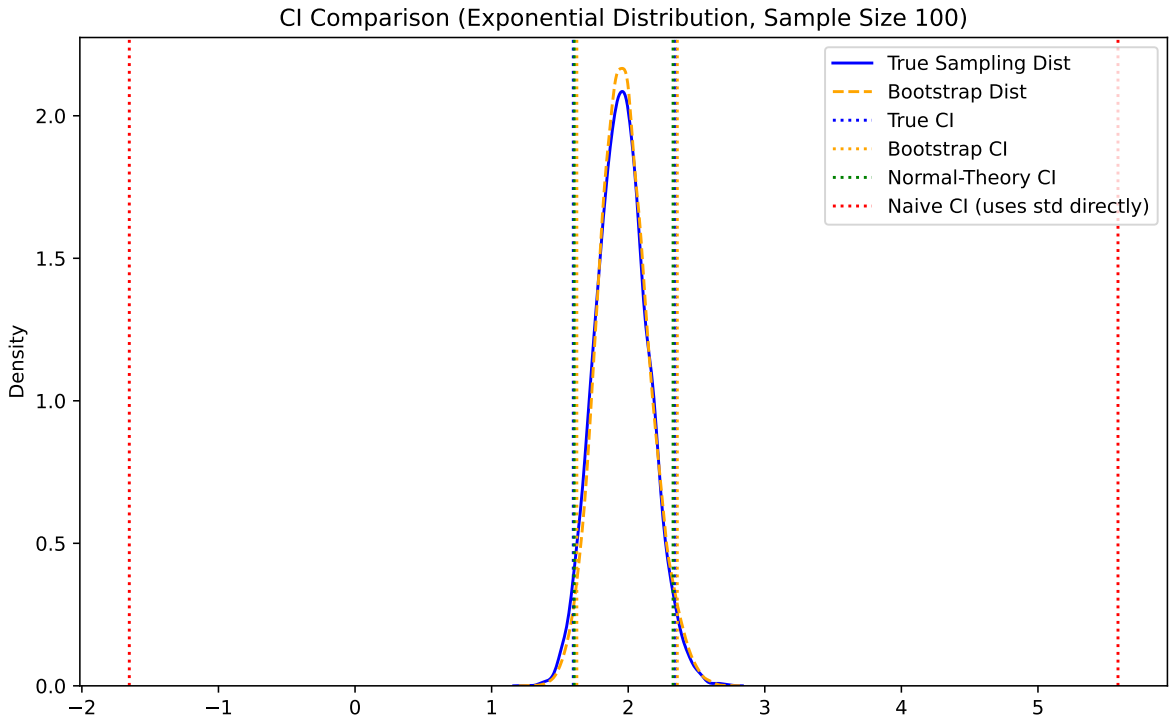
```
Simulated 1000 customers
Average wait time     : 0.4640
Average sojourn time  : 1.1262
Server utilization    : 0.6667 (theoretical)
```


Queue Length Over Time
Queue Length
Simulation Time
0
2
4
6
8
10
12
14
0
100
200
300
400
500

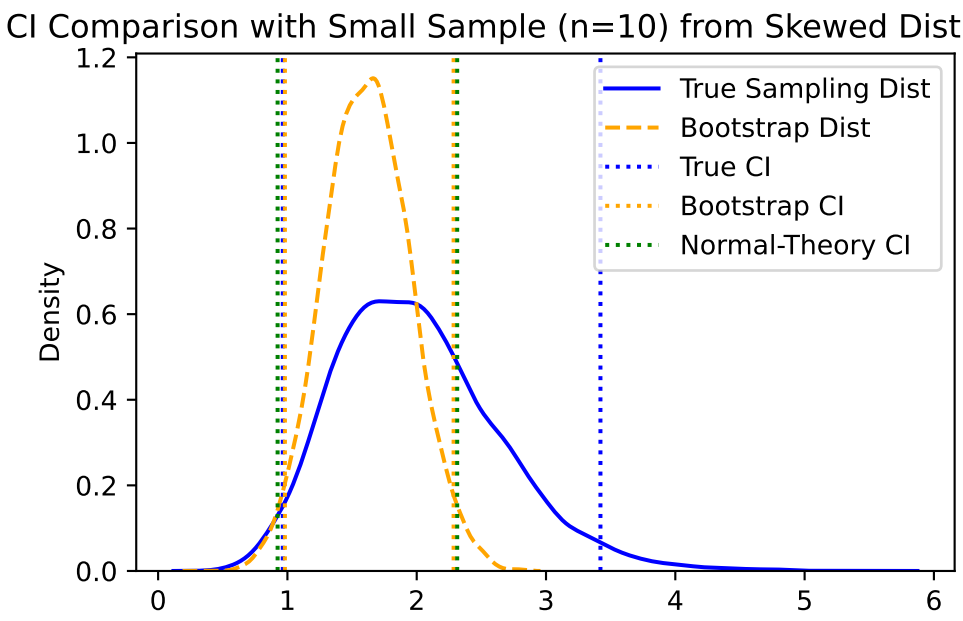
16 Bootstrap



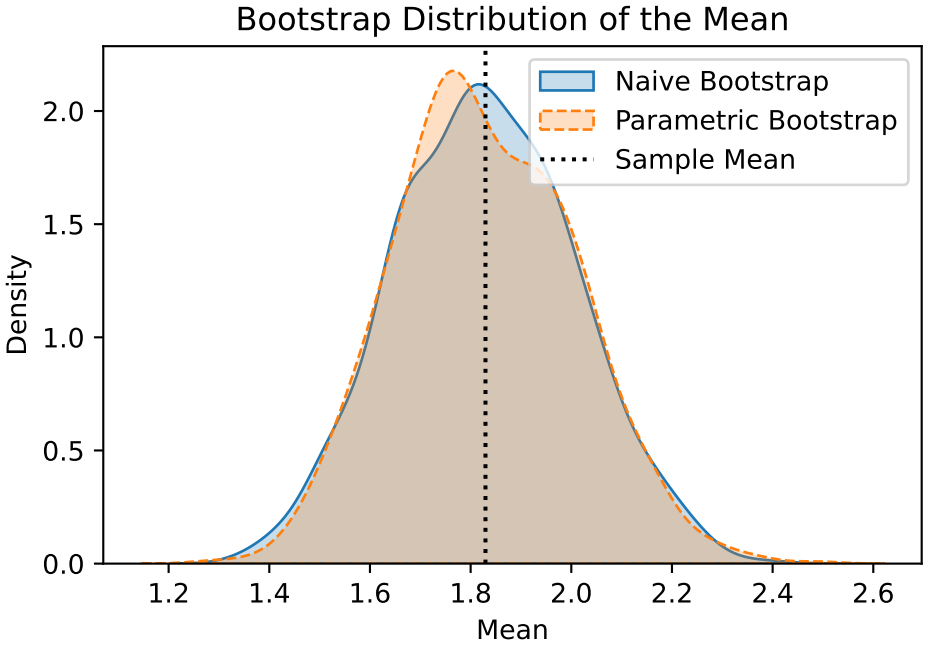

```
Sample Mean: 1.967
True CI: [1.5979144  2.34577445]
Bootstrap CI: [1.62388954 2.35938801]
Normal-Theory CI: [1.60463905 2.32850834]
```

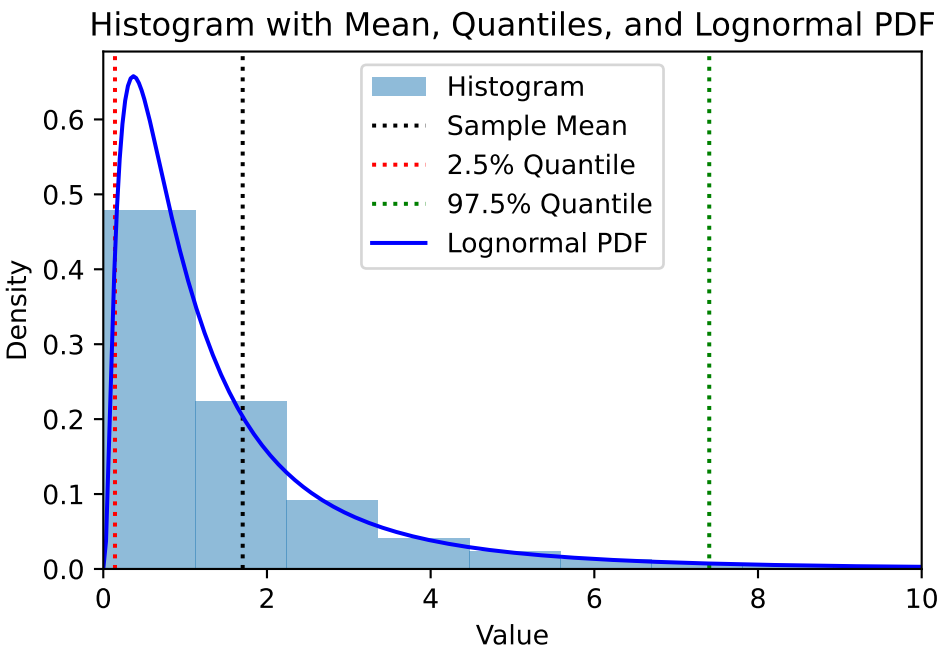



```
True CI: [1.5979144  2.34577445]
Bootstrap CI: [1.62388954 2.35938801]
Normal-Theory CI: [1.60463905 2.32850834]
Naive CI: [-1.65277273  5.58592012]
```




```
Sample Mean: 1.619
True CI:           [0.96432733 3.4209794 ]
Bootstrap CI:      [0.98059327 2.28526628]
Normal-Theory CI: [0.92412973 2.31300113]
```




Histogram with Mean, Quantiles, and Lognormal PDF
Histogram
Sample Mean
2.5% Quantile
97.5% Quantile
Lognormal PDF
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
2
4
6
8
10
Value

Part IV

Advanced Topics

17 Kalman Filters

18

Part V

Optimization

19

20

21

22 References

Rubinstein, R. Y., and D. P. Kroese. 2017. *Simulation and the Monte Carlo Method*. 3rd ed. USA: Wiley.

A Review of Probability Theory

This appendix reviews the mathematical foundations underlying Monte Carlo methods, covering probability spaces, random variables, estimation theory, and the fundamental limit theorems that justify Monte Carlo inference.

A.1 Probability Spaces and Random Variables

A **probability space** is a triple $(\Omega, \mathcal{F}, P)$ where Ω is the sample space, $\mathcal{F}$ is a σ-algebra on Ω, and $P : \mathcal{F} \to [0, 1]$ is a probability measure satisfying Kolmogorov's axioms.

A **random variable** is a measurable function $X : \Omega \to \mathbb{R}$ such that $\{X \leq x\} \in \mathcal{F}$ for all $x \in \mathbb{R}$. The **cumulative distribution function (CDF)** of X is:

$$F_X(x) = P(X \leq x) = P(\{\omega \in \Omega : X(\omega) \leq x\})$$

Continuous random variables have a **probability density function (PDF)** $f_X(x)$ such that:

$$F_X(x) = \int_{-\infty}^{x} f_X(t)\, dt \quad \text{and} \quad f_X(x) = \frac{dF_X(x)}{dx}$$

Discrete random variables with support $\{x_1, x_2, \ldots\}$ satisfy $P(X = x_i) = p_i$ where $\sum_i p_i = 1$.

Definition (Moments)

For a random variable X:
Expected value:

$$\mathbb{E}[X] = \begin{cases} \sum_i x_i \cdot P(X = x_i) & \text{if } X \text{ is discrete} \\ \int_{-\infty}^{\infty} x \cdot f_X(x)\, dx & \text{if } X \text{ is continuous} \end{cases}$$

Variance:

$$\text{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$$

A.2 Statistical Estimation and the Sample Mean

Consider i.i.d. observations $X_1, \ldots, X_N$ with common distribution F. We seek to estimate $\theta = \mathbb{E}[g(X)]$ for some measurable function $g : \mathbb{R} \to \mathbb{R}$ using the **sample mean estimator**:

$$\hat{\theta}_N = \frac{1}{N} \sum_{i=1}^{N} g(X_i) \tag{A.1}$$

Theorem (Properties of the Sample Mean Estimator)

The sample mean estimator $\hat{\theta}_N$ satisfies:

1. **Unbiasedness**: $\mathbb{E}[\hat{\theta}_N] = \theta$
2. **Variance**: $\mathrm{Var}(\hat{\theta}_N) = \frac{\sigma^2}{N}$ where $\sigma^2 = \mathrm{Var}(g(X))$
3. **Standard Error**: $\mathrm{SE}(\hat{\theta}_N) = \sigma/\sqrt{N}$
4. **Mean Squared Error**: $\mathrm{MSE}(\hat{\theta}_N) = \sigma^2/N$

For any estimator $\hat{\theta}$ of parameter θ, we define:

- **Bias**: $\mathrm{Bias}(\hat{\theta}) = \mathbb{E}[\hat{\theta}] - \theta$
- **Mean Squared Error**: $\mathrm{MSE}(\hat{\theta}) = \mathbb{E}[(\hat{\theta} - \theta)^2] = \mathrm{Var}(\hat{\theta}) + \mathrm{Bias}^2(\hat{\theta})$

An estimator sequence $\{\hat{\theta}_N\}$ is **consistent** if $\hat{\theta}_N \xrightarrow{P} \theta$ as $N \to \infty$, and **strongly consistent** if $\hat{\theta}_N \xrightarrow{a.s.} \theta$ as $N \to \infty$.

A.3 Fundamental Limit Theorems

The theoretical foundation of Monte Carlo methods rests on two cornerstone results from probability theory.

Theorem (Strong Law of Large Numbers)

Let $X_1, X_2, \ldots$ be i.i.d. random variables with $\mathbb{E}[|X_i|] < \infty$ and $\mathbb{E}[X_i] = \mu$. Then:

$$\frac{1}{N} \sum_{i=1}^{N} X_i \xrightarrow{a.s.} \mu \quad \text{as } N \to \infty$$

Theorem (Central Limit Theorem)

Let $X_1, X_2, \ldots$ be i.i.d. random variables with $\mathbb{E}[X_i] = \mu$ and $0 < \text{Var}(X_i) = \sigma^2 < \infty$. Then:

$$\frac{\sqrt{N}(\bar{X}_N - \mu)}{\sigma} \xrightarrow{d} \mathcal{N}(0,1) \quad \text{as } N \to \infty$$

where $\bar{X}_N = \frac{1}{N}\sum_{i=1}^{N} X_i$ and $\xrightarrow{d}$ denotes convergence in distribution.

Convergence notation: $\xrightarrow{P}$ denotes convergence in probability, $\xrightarrow{a.s.}$ denotes almost sure convergence, and $\xrightarrow{d}$ denotes convergence in distribution.

A.3.1 Monte Carlo Implications

For our estimator $\hat{\theta}_N = \frac{1}{N}\sum_{i=1}^{N} g(X_i)$ where $\theta = \mathbb{E}[g(X)]$:

1. **Consistency** (from SLLN): $\hat{\theta}_N \xrightarrow{a.s.} \theta$
2. **Asymptotic normality** (from CLT): $\sqrt{N}(\hat{\theta}_N - \theta) \xrightarrow{d} \mathcal{N}(0, \sigma^2)$ where $\sigma^2 = \text{Var}(g(X))$

A.4 Confidence Intervals and Convergence Analysis

Let $S_N^2 = \frac{1}{N-1}\sum_{i=1}^{N}(g(X_i) - \hat{\theta}_N)^2$ be the sample variance. By the CLT:

$$\frac{\hat{\theta}_N - \theta}{S_N/\sqrt{N}} \xrightarrow{d} \mathcal{N}(0,1)$$

An asymptotic $(1-\alpha)$-level confidence interval is:

$$\left[\hat{\theta}_N - z_{1-\alpha/2}\frac{S_N}{\sqrt{N}}, \quad \hat{\theta}_N + z_{1-\alpha/2}\frac{S_N}{\sqrt{N}}\right]$$

where $z_{1-\alpha/2} = \Phi^{-1}(1-\alpha/2)$ and Φ is the standard normal CDF.

The **absolute width** is $2z_{1-\alpha/2}\frac{S_N}{\sqrt{N}}$ and the **relative width** is $\frac{2z_{1-\alpha/2}S_N}{|\hat{\theta}_N|\sqrt{N}}$.

Remark (Monte Carlo Convergence Properties)

The sample mean estimator has standard error $\text{SE}(\hat{\theta}_N) = \sigma/\sqrt{N} = O(N^{-1/2})$, leading to several key properties:

1. **Square Root Law**: To halve the confidence interval width requires four times as many samples
2. **Dimension Independence**: The $O(N^{-1/2})$ convergence rate is independent of problem dimension for independent samples
3. **MCMC Caveat**: When using MCMC, dependence between samples can effectively reduce the number of independent observations, particularly in high dimensions

Conditions for Limit Theorems

Both the Law of Large Numbers and Central Limit Theorem require:

- **Independence**: Samples must be independent (or satisfy weaker mixing conditions)
- **Identical distribution**: Samples from the same distribution
- **Finite moments**: Finite mean for LLN, finite variance for CLT

When using MCMC, the independence assumption is violated, requiring analysis of effective sample size.

B Markov Chains

B.1 Definitions

In this module, we will learn about Markov Chains. A Markov Chain is a discrete-time stochastic process that satisfies the Markov property. The Markov property states that the future state of the process depends only on the current state and not on the sequence of events that preceded it. In other words, the future is conditionally independent of the past given the present.

Definition B.1. Markov Chain: A Markov chain is a sequence of random variables $X_0, X_1, X_2, ...$ satisfying the following properties:

1. **State Space**: The random variables X_i take values in a finite set Ω called the state space.
2. **Markov Property**: For all $n \geq 0$ and all states $i_0, i_1, ..., i_{n+1} \in \Omega$,

$$P(X_{n+1} = i_{n+1} \mid X_0 = i_0, X_1 = i_1, ..., X_n = i_n) = P(X_{n+1} = i_{n+1} \mid X_n = i_n).$$

3. **Time Homogeneity**: The transition probabilities $P(X_{n+1} = j \mid X_n = i)$ do not depend on n.

The transition matrix of a Markov chain is a square matrix whose (i, j)-th entry is the probability of transitioning from state i to state j in one time step.

Definition B.2. Transition Matrix: Let $X_1, X_2, X_3, ...$ be a Markov chain with state space $\Omega = \{1, 2, ..., N\}$. The transition matrix P of the Markov chain is an $N \times N$ matrix whose (i, j)-th entry is given by

$$P(i, j) = P(X_{n+1} = j \mid X_n = i).$$

Throughout this notebook, we'll let $X_0, X_1, X_2, ...$ be a Markov chain with state space $\Omega = \{1, 2, ..., N\}$ and transition matrix P.

We can represent a Markov chain by a directed graph called a **state diagram**. Each state is represented by a node, and the transition probabilities are represented by directed edges between the nodes. The transition matrix can be derived from the state diagram by assigning the transition probabilities to the corresponding entries of the matrix.

The probability distribution of the Markov chain at time n is a *row vector* π_n whose i^{th} entry is $\mathbb{P}(X_n = i)$ for each $i \in \Omega$.

Theorem B.1. *Let π_n be the probability distribution of the chain at time n. Then,*

$$\pi_{n+1} = \pi_n P.$$

And hence,

$$\pi_n = \pi_0 P^n.$$

Proof.

$$\begin{aligned}
\pi_{n+1}(j) &= \mathbb{P}(X_{n+1} = j) \\
&= \sum_{i \in S} \mathbb{P}(X_{n+1} = j \mid X_n = i)\mathbb{P}(X_n = i) \\
&= \sum_{i \in S} \pi_n(i)P(i, j) \\
&= \pi_n P(j).
\end{aligned}$$

□

B.2 Stationary Distribution

Definition B.3. A probability distribution π is called a **stationary distribution** of a Markov chain with transition matrix P if $\pi = \pi P$.

P is guaranteed to have an eigenvalue of 1 because it's row sum is 1 i.e.

$$P\vec{1} = \vec{1},$$

where $\vec{1}$ is the vector of all ones. Since, there is a right eigenvector corresponding to the eigenvalue 1, there will be a left eigenvector as well. The left eigenvector is a stationary distribution of the Markov chain.

It is not hard to see that every eigenvalue of P is less than or equal to 1 in magnitude. Suppose $\vec{v}$ is a left eigenvector of P corresponding to an eigenvalue λ. Let v_I be the largest component of $\vec{v}$ in magnitude. Then, we have

$$\begin{aligned} \lambda\vec{v} &= \vec{v}P \\ \implies \lambda v_I &= \sum_j v_j P(j, I) \\ &\leq \sum_j |v_j| P(j, I) \\ &\leq \sum_j |v_I| P(j, I) \\ &= |v_I| \end{aligned}$$

Thus, $|\lambda| \leq 1$. In particular, this means that the Frobenius norm of P is less than or equal to 1 and for all vectors $\vec{v}$, we have

$$\|\vec{v}\|_2 \leq \|\vec{v}P\|_2.$$

We are particularly interested in the case when there is exactly one eigenvector with eigenvalue of magnitude 1 which would then be the stationary distribution of the Markov chain.

B.3 Fundamental Theorem

Definition B.4. We say that a Markov Chain is **irreducible** if for every pair of states $i, j \in \Omega$, there exists an integer n such that $P^n(i, j) > 0$ i.e. it is possible to go from any state to any other state in a finite number of steps.

Definition B.5. We say that a state i is **aperiodic** if the greatest common divisor of the set $\{n \geq 1 : P^n(i, i) > 0\}$ is 1. A Markov chain is called **aperiodic** if all its states are aperiodic.

Note that sometimes we add a preliminary requirement that the set $\{n \geq 1 : P^n(i,i) > 0\}$ is non-empty. This condition is called **positive recurrence**. We'll assume this as a part of the definition of aperiodicity.

Definition B.6. A Markov chain is called **ergodic** if it is irreducible and aperiodic.

Theorem B.2. ***Fundamental Theorem of Markov Chains****: If a Markov chain is ergodic, then it has a unique stationary distribution* Π*. Moreover, in this case, for any initial distribution* π_0*, the distribution of the chain converges to* Π *as* $n \to \infty$ *i.e.*

$$\lim_{n\to\infty} \pi_0 P^n = \Pi,$$

for any initial distribution π_0*.*

B.4 Random Walk on Graphs

Our main example of a Markov chain is the random walk on a graph. Let $G = (V, E)$ be a graph with vertex set V and edge set E. Suppose you want to move from one vertex to another by following the edges of the graph. At each vertex, you choose an edge uniformly at random and move to the adjacent vertex. This process is called a random walk on the graph.

The random walk on G is a Markov chain with state space $\Omega = V$ and transition probabilities given by

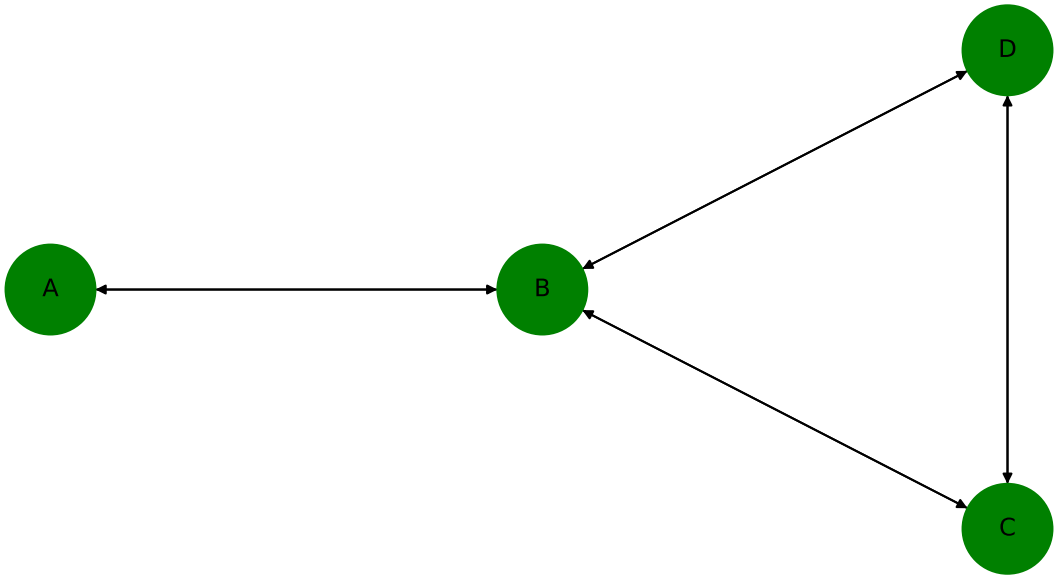
$$P(i,j) = \begin{cases} \frac{1}{\deg(i)} & \text{if } (i,j) \in E, \\ 0 & \text{otherwise,} \end{cases}$$

where $\deg(i)$ is the degree of vertex i i.e. the number of edges incident to i.

Example B.1. Consider a graph G with 4 vertices as shown below. The transition matrix of the random walk on G is given by

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1/3 & 0 & 1/3 & 1/3 \\ 0 & 1/2 & 0 & 1/2 \\ 0 & 1/2 & 1/2 & 0 \end{pmatrix}.$$

Suppose we start at vertex A. This means that the initial distribution is $\pi_0 = [1, 0, 0, 0]$. The n^{th} distribution π_n can be obtained by multiplying π_0 with the transition matrix P^n, as shown below.




```
Transition matrix of the Markov chain:
[[0.         1.         0.         0.        ]
 [0.33333333 0.         0.33333333 0.33333333]
 [0.         0.5        0.         0.5       ]
 [0.         0.5        0.5        0.        ]]


Example of evolution of a Markov chain:
State at time 0 : [1, 0, 0, 0]
State at time 1 : [0. 1. 0. 0.]
State at time 2 : [0.333 0.    0.333 0.333]
State at time 3 : [0.    0.667 0.167 0.167]
State at time 4 : [0.222 0.167 0.306 0.306]
State at time 5 : [0.056 0.528 0.208 0.208]
State at time 6 : [0.176 0.264 0.28  0.28 ]
State at time 7 : [0.088 0.456 0.228 0.228]
State at time 8 : [0.152 0.316 0.266 0.266]
State at time 9 : [0.105 0.418 0.238 0.238]
State at time 10 : [0.139 0.344 0.259 0.259]
```

In HW, you'll prove the following theorem about ergodicity of random walks on graphs.

Theorem B.3. *A random walk on a graph is*

1. *Irreducible if and only if the graph is connected.*
2. *Aperiodic if and only if the graph is not bipartite.*

If these conditions hold, then the stationary distribution of the random walk is given by

$$\Pi(i) = \frac{deg(i)}{2|E|},$$

where $deg(i)$ is the degree of vertex i and $|E|$ is the number of edges in the graph.

B.5 Mixing Time

Assume that the Markov chain is ergodic, and hence has a stationary distribution Π. We know that any initial distribution π_0 converges to Π as $n \to \infty$. The mixing time measures the rate of this convergence.

Definition B.7. The **total variation distance** between two probability distributions μ and ν on a finite state space Ω is defined as

$$\begin{aligned}\|\mu - \nu\|_{\text{TV}} &= \frac{1}{2}\sum_{i\in\Omega} |\mu(i) - \nu(i)| \\ &= \frac{1}{2}\|\mu - \nu\|_{L^1}.\end{aligned}$$

One can show that

$$\|\mu - \nu\|_{\text{TV}} = \sup\{|\mu(A) - \nu(A)| : A \subseteq \Omega\},$$

i.e. $\|\mu - \nu\|_{\text{TV}}$ is the maximum difference in the probability of any event under the two distributions.

We use the total variation distance to measure the distance between the distribution of the Markov chain at time n and the stationary distribution. The **mixing time** of the Markov chain is defined as the smallest N such that for the stationary distribution Π and any initial distributions π_0, we have

$$\|\pi_0 P^n - \Pi\|_{\text{TV}} \leq \frac{1}{4},$$

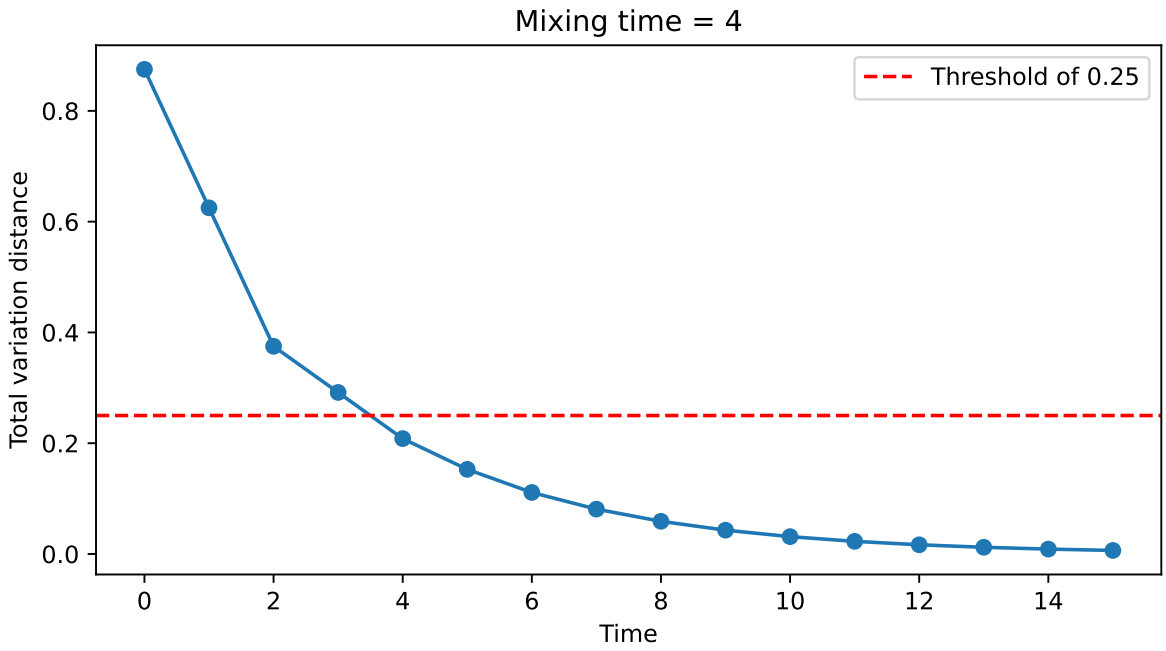
for all $n \geq N$. The constant $\frac{1}{4}$ is arbitrary and can be replaced by any other constant in $(0, 1)$. This will only change the value of the mixing time by a constant factor and not the order of magnitude.

When using Markov chains for sampling, we want the mixing time to be as small as possible. This ensures that the distribution of the chain is close to the stationary distribution after a small number of steps. We think of the time before the chain mixes as a transient phase - the chain has not yet reached equilibrium. This is a burn-in period where we discard the samples. The bigger the mixing time, the more the number of wasted samples in the burn-in phase.

Example. Consider Example B.1 again. If we start at the vertex A, by step 4 we have already reached the distribution $\pi_4 = [0.222, 0.167, 0.306, 0.306]$. The stationary distribution is $\Pi = [1/8, 4/8, 2/8, 2/8]$. The total variation distance between π_4 and Π is $\|\pi_4 - \Pi\|_{\text{TV}} = 0.21$. This is less than $1/4$, and hence the mixing time is 4.

This only computes the mixing time for the initial distribution $\pi_0 = [1, 0, 0, 0]$. In general, we need to compute the mixing time for all possible initial distributions. The mixing time is the maximum of these mixing times.

In practice, we either provide a theoretical bound on the mixing time or "visually" inspect the convergence of the chain to the stationary distribution. Running a simulation to compute the mixing time is computationally expensive and not commonly done.



B.5.1 Connection to Spectral Theory

Continuing the example from above, the matrix $I_4 - P$ is called the **normalized Laplacian matrix** of the graph G.

$$\mathcal{L} = I_4 - P = \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1/3 & 1 & -1/3 & -1/3 \\ 0 & -1/2 & 1 & -1/2 \\ 0 & -1/2 & -1/2 & 0 \end{pmatrix}$$

One can show that 0 is an eigenvalue of $\mathcal{L}$ and all eigenvalues are non-negative. The second smallest eigenvalue of $\mathcal{L}$ is called the **spectral gap** of the graph (which could be 0).

Spectral graph theory, in particular Cheeger inequalities, prove that there is an inverse relationship between the spectral gap of the graph and the mixing time of the random walk on the graph. The smaller the spectral gap, the larger the mixing time. You'll explore this connection in the homework.

B.6 Sampling from a Markov Chain

The algorithm to generate sample paths of length n of a Markov Chain is simple. Suppose P is the transition matrix of the Markov Chain with mixing time t. The algorithm is as follows:

1. Start at some initial state X_0.
2. For $i = 0, 1, \ldots, N-1$
   - Generate $X_{i+1} \sim P(X_i, \cdot)$.
3. Discard the first T samples and return $X_{T+1}, X_{T+2}, \ldots, X_N$.

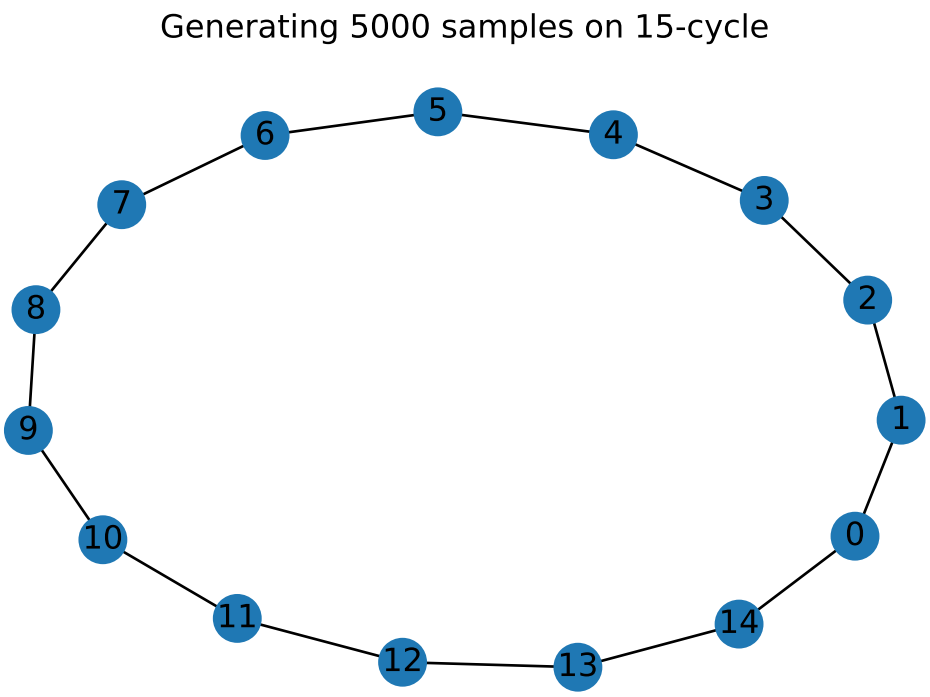
We can interpret this algorithm as generating $N - T$ samples from the stationary distribution of the Markov Chain. The number of samples discarded is called the **burn-in period**.

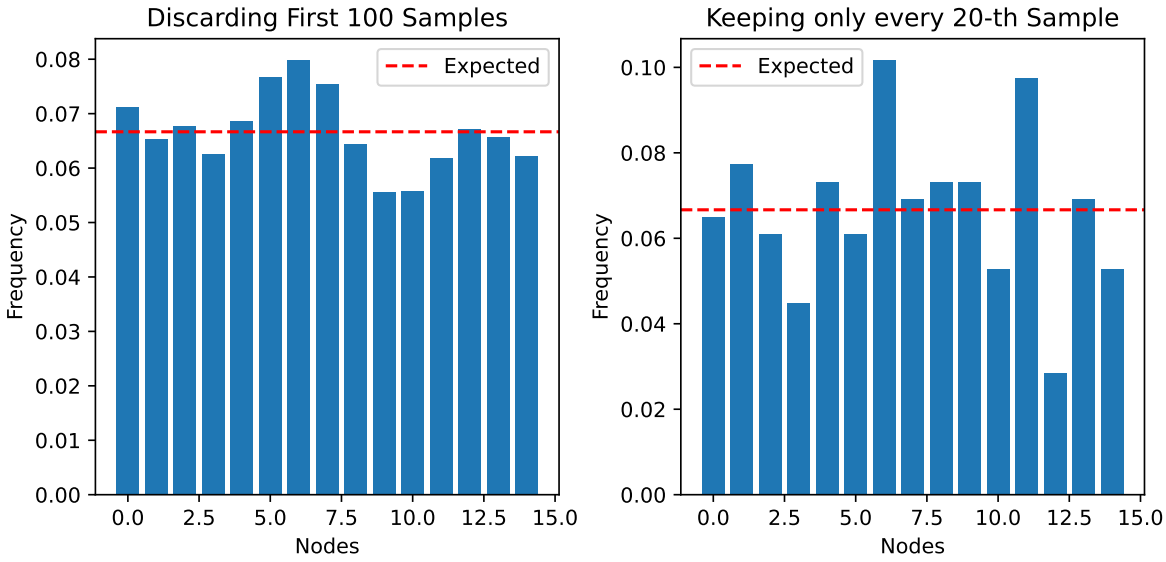
One big issue with this algorithm is that the samples are highly correlated. If independence is important, selecting every 40th sample may be beneficial. However, this leads to a lot of wasted samples and it might not get rid of all the correlation. In practice, it is better to generate a large number of samples than to "thin" the samples.

Example B.2. In the example below we generate a uniform distribution over $[0, 14]$ by generating a random walk over a cycle of length 15.

The autocorrelation plot below shows the correlation between samples at different lags i.e. X_i and X_{i+k} for different values of k. We can see that the correlation decreases as k increases and stabilizes around $k = 40$.

We can either discard the first 40 samples or select every 40-th sample to reduce the correlation. If independence is important, then we should select every 40-th sample. However, this leads to a lot of wasted samples and it might not get rid of all the correlation. This method is not preferred in practice. In practice, it is better to generate a large number of samples than to "thin" the samples.





B.7 Reversible Markov Chains

A Markov Chain is **reversible** with respect to a distribution π if the following holds:

$$\pi_i P(i,j) = \pi_j P(j,i) \quad \text{for all } i,j. \tag{B.1}$$

This is saying that the probability of transitioning from x to y is the same as the probability of transitioning from y to x. Equation B.1 is known as the **detailed balance equation**.

Theorem B.4. ***(Detailed Balance Equation).*** *If a Markov Chain is reversible with respect to a distribution π, then π is a stationary distribution of the Markov Chain.*

Proof. Suppose the Markov Chain is reversible with respect to π. Then,

$$\begin{aligned}(\pi P)_i &= \sum_j \pi_j P(j,i) \\ &= \sum_j \pi_i P(i,j) \\ &= \pi_i \sum_j P(i,j) \\ &= \pi_i.\end{aligned}$$

Thus, π is the stationary distribution of the Markov Chain. □

Equation B.1 is a sufficient but not necessary condition for π to be the stationary distribution of the Markov Chain. You can have a Markov Chain with a stationary distribution that is not reversible.

Note that we did not use any properties of the Markov Chain in the proof of the theorem, except that the row sum of the transition matrix is 1. A better way to phrase this theorem would be to say that "if the row sum of the transition matrix is 1 and the detailed balance equation holds, then π is an eigenvector of the transition matrix with eigenvalue 1."

Example B.3. Random Walks on Graphs. Consider a graph $G = (V, E)$ with vertices V and edges E. Let $P(i,j) = 1/d(i)$ if $(i,j) \in E$ and 0 otherwise, where $d(i)$ is the degree of vertex i. Then, the stationary distribution of the Markov Chain is $\pi_i = d(i)/2|E|$, where $|E|$ is the number of edges in the graph.

The Markov Chain is reversible with respect to π. Consider two vertices i and j. If $(i,j) \in E$, then

$$\begin{aligned}\pi_i P(i,j) &= \frac{d(i)}{2|E|} \cdot \frac{1}{d(i)} \\ &= \frac{1}{2|E|} \\ &= \frac{d(j)}{2|E|} \cdot \frac{1}{d(j)} \\ &= \pi_j P(j,i).\end{aligned}$$

If $(i,j) \notin E$, then $\pi_i P(i,j) = \pi_j P(j,i) = 0$.

Many Markov chains encountered in practice are reversible with respect to some distribution. It is much easier to check the detailed balance equation than to compute the stationary distribution directly. Moreover, reversible markov chains can be analyzed using spectral methods and we can find good bounds on their mixing time.

B.7.1 Reversibility and Symmetry

Theorem B.5. *If a Markov Chain is reversible with respect to a distribution π, then the matrix*

$$Q = diag(\sqrt{\pi})\ P\ diag(\sqrt{\pi^{-1}})$$

is symmetric. In particular, P is similar to the symmetric matrix Q and hence has real eigenvalues.

Proof. This is because

$$\begin{aligned}Q(i,j) &= \sqrt{\pi_i} P(i,j) \sqrt{\pi_j^{-1}} \\ &= \sqrt{\pi_i} \frac{\pi_j P(j,i)}{\pi_i} \sqrt{\pi_j^{-1}} \\ &= \sqrt{\pi_j} P(j,i) \sqrt{\pi_i^{-1}} \\ &= Q(j,i).\end{aligned}$$

Similarly, $Q(j,i) = \pi_j P(j,i)$. As the Markov Chain is reversible with respect to π, we get $Q(i,j) = Q(j,i)$. Thus, Q is symmetric. □

Now suppose $\mathbf{v}$ is a left eigenvector of Q with eigenvalue λ. Then,

$$\begin{aligned} \mathbf{v}Q &= \lambda\mathbf{v} \\ \mathbf{v}\text{diag}(\sqrt{\pi})\, P\, \text{diag}(\sqrt{\pi^{-1}}) &= \lambda\mathbf{v} \\ \implies \mathbf{v}\text{diag}(\sqrt{\pi})\, P &= \lambda\mathbf{v}\, \text{diag}(\sqrt{\pi}). \end{aligned}$$

Hence, $\text{diag}(\sqrt{\pi})\mathbf{v}$ will be an eigenvector of P with the same eigenvalue. As Q is symmetric, it has real eigenvalues and orthogonal eigenvectors. It is easier to do spectral analysis of Q and use that to deduce properties of P. For example, we can find the eigenvector corresponding to the largest eigenvalue of Q by solving the optimization problem

$$\mathbf{v} = \arg\max_{\mathbf{x}\neq 0} \frac{\mathbf{x}^T Q\mathbf{x}}{\mathbf{x}^T\mathbf{x}}.$$

Multiplying the above vector $\mathbf{v}$ by $\text{diag}(\sqrt{\pi})$ gives us the stationary distribution for P.

B.8 Transition Kernels

The sample spaces we encounter in MCMC methods are not discrete but continuous. Instead of a transition matrix, we use a **transition kernel** $K(x, y)$ that gives the "probability of transitioning from state x to state y". However, because the sample space is continuous, the probability of being in a particular state is zero. Instead, we use the **density** of the distribution at that point. The transition kernel satisfies the equation:

$$\mathbb{P}(X_{n+1} \in A \mid X_n = x) = \int_A K(x, y)\, dy.$$

All the properties of Markov Chains that we discussed earlier can be extended to transition kernels. The fundamental theorem of Markov Chains becomes

Theorem B.6. ***(Fundamental Theorem of Markov Chains for Kernels).*** *If a Markov Chain with transition kernel K is*

1. ***Irreducible**: For all x, y, there exists n such that $K^n(x, y) > 0$.*
2. ***Positive recurrent**: The expected return time to a state is finite.*
3. ***Aperiodic**: For all x, $\gcd\{n : K^n(x, x) > 0\} = 1$.*

Then, the Markov Chain has a unique stationary distribution Π *and for all initial distributions* π_0*,*

$$\int \pi_0(x) K^n(x, y)\, dx \xrightarrow[TV]{} \Pi(y) \quad as\ n \to \infty.$$

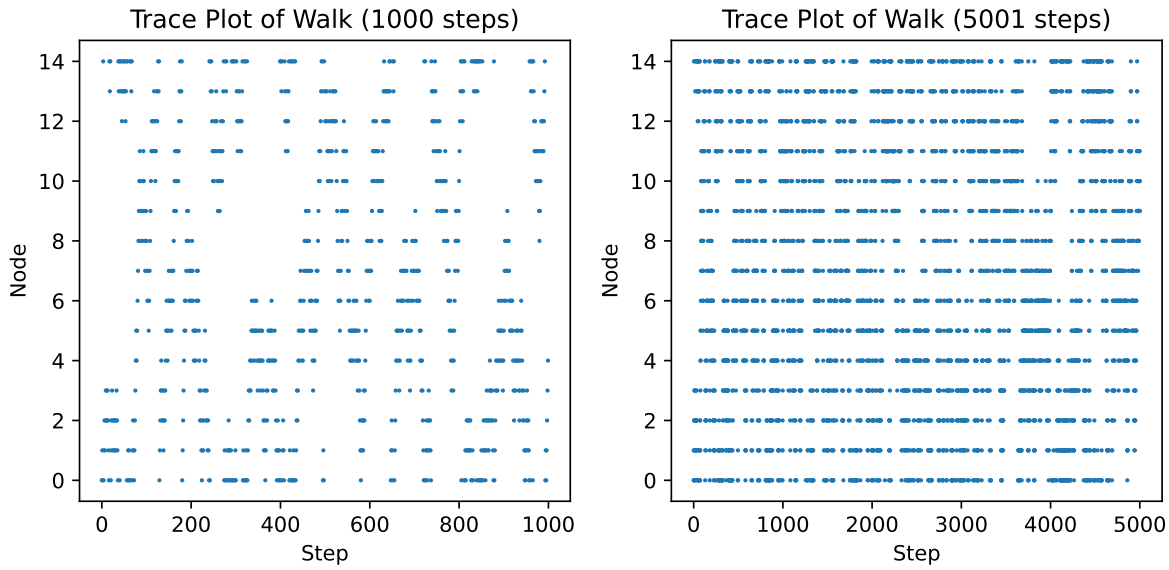
B.9 Analyzing Convergence of Markov Chains

In practice we need to decide how many samples to burn and how many samples to generate. We use various heuristics to decide this.

B.9.1 Trace Plots

A trace plot is simply a scatter plot of the samples generated by the Markov Chain. It is useful to see if the Markov Chain has converged. If the Markov Chain has converged, the trace plot should look like a cloud of points. If the Markov Chain has not converged, the trace plot will show a trend.

Below are the trace plots for Example B.2. We can see that the chain does not look uniform even after a 1000 samples but after 5000 samples it is starting to look uniform.

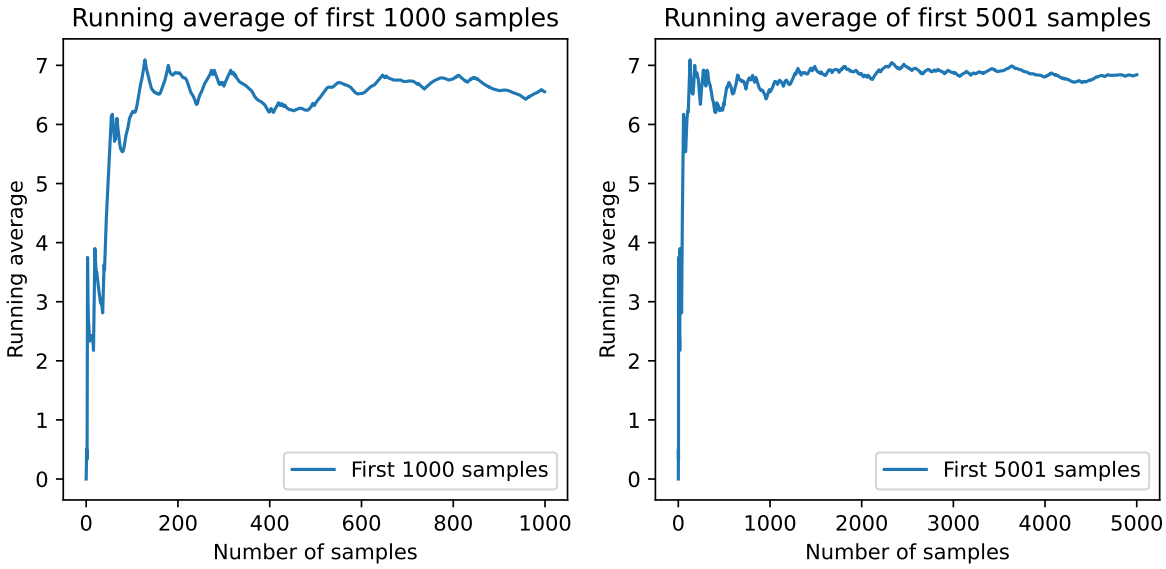


B.9.2 Running Average

The running average is the average of the first n samples. It is useful to see if the Markov Chain has converged. If the Markov Chain has converged, the running average should stabilize around the true mean. If the Markov Chain has not converged, the running average will show a trend.

Below is the running average plot for Example B.2. We can see that the running average is stabilizing around the true mean around 3000 samples.

Running averages can be deceptive and show stability even when the Markov Chain has not converged. It is better to use multiple diagnostics to check for convergence. They are better at telling when the Markov Chain has **not** converged than when it has converged.



B.9.3 Autocorrelation Function

One method for finding the burn-in period is to use the autocorrelation function. The **autocorrelation function** of a sequence of numbers $x = (x_0, x_1, \dots, x_n)$ at lag k is defined as

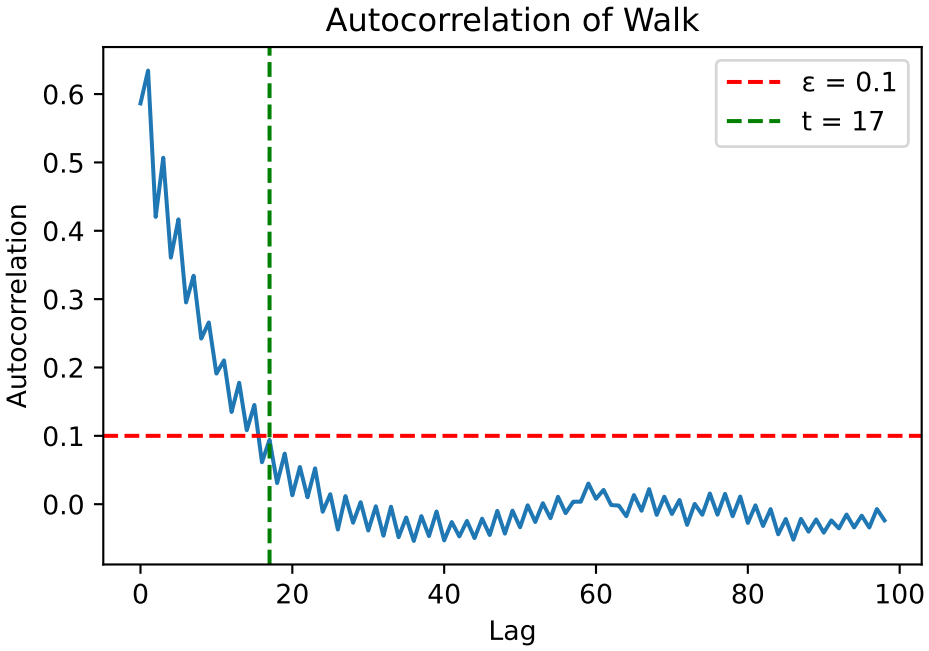
$$\text{ACF}(k) = \text{Corr}(x[k:], x[:-k])$$

where by $x[k:]$ we mean the subsequence $x_k, x_{k+1}, \dots, x_n$ and by $x[:-k]$ we mean the subsequence $x_0, x_1, \dots, x_{n-k}$. It is the correlation between the sequence x and the same sequence

shifted by k. One way to choose a burn-in period is to pick a threshold ϵ and find the smallest T such that $\text{ACF}(T) < \epsilon$. Then to be conservative choose a burn-in period of $2T$ or $3T$.

Below is the autocorrelation plot for Example B.2. We can see that the correlation decreases as k increases and drops below 0.1 around 17. So a burn-in period of 40 is a good conservative choice.

This is just one way to choose the burn-in period. There are many other methods and the choice depends on the application. We will stick to this method for the rest of the course.



C Hidden Markov Chains

D Particle Filter with Resampling

1. **Inputs:**
   - Observations $\{y_1, y_2, \ldots, y_T\}$
   - Number of particles N
   - Transition model $p(x_t \mid x_{t-1})$
   - Emission model $p(y_t \mid x_t)$
   - Initial distribution $p(x_1)$
2. **Initialization:** For $i = 1$ to N
   1. Set $x_0^{(i)} \leftarrow x_0$
   2. Set weight $w_1^{(i)} \leftarrow 1/N$
3. For $t = 1$ to T
   1. For $i = 1$ to N
      1. Sample $x_t^{(i)} \sim p(x_t \mid x_{t-1}^{(i)})$
      2. Compute weight $w_t^{(i)} \leftarrow w_{t-1}^{(i)} \cdot p(y_t \mid x_t^{(i)})$
   2. Normalize weights $w_t^{(i)} \leftarrow \frac{w_t^{(i)}}{\sum_{j=1}^{N} w_t^{(j)}}$
4. **Return:** Particles $\{x_t^{(i)}\}$ and weights $w_t^{(i)}$ for all t