University of Groningen
Faculty of Economics and Business

Econometrics & Operations Research

Numerical Approximation of Integrals using Simulation

Author:
Jelmer Wieringa

Supervisor:
dr. T. Kantarci

Bachelor's Thesis
Econometrics & Operations Research

Supervisor: dr. T. Kantarci
Second assessor: dr. Z. Méder

University of Groningen

Numerical Approximation of Integrals using Simulation

Author:
Jelmer Wieringa

Student Number - Program:
S4541979 - BSc EOR

December 24, 2024

Abstract

Integration problems arise frequently in econometrics, because integrals are often analytically intractable. A collection of algorithms, called Monte Carlo integration methods, can be used to approximate one- and multi-dimensional integrals. This paper reviews the underlying theory of Monte Carlo integration and accompanies the theory with applications. Our target audience includes practitioners and first-time learners, as the literature often presents only theoretical concepts, lacking pedagogically accessible and intuitive applications. The principle of Monte Carlo integration is introduced in the special cases of a one-dimensional integral and multi-dimensional integrals with independent probability density functions. Moreover, variance reduction methods improve the efficiency of the Monte Carlo integration estimator. However, in practice, we cannot generate samples from the probability density function of interest which is a necessity for the Monte Carlo integration technique. The Markov chain Monte Carlo algorithms present a solution to this problem. We focus on the Metropolis-Hastings algorithm. Finally, an econometric application involving a multi-dimensional integral is approximated by the Metropolis-Hastings algorithm and the Ergodic Theorem.

Contents

1 Introduction

Integration problems are ubiquitous in the fields of econometrics, data science, operations research and actuarial science. These fields share intersecting applications of integration that include but are not limited to calculating probabilities, moments, posterior distributions, maximum likelihood estimates, marginal probability density functions. However, there also exist many field-specific integration problems. In practice, these integrals are often intractable or even unsolvable. Fortunately, there exist techniques to approximate integrals numerically. In the literature, numerical integration is sometimes referred to as quadrature.

These numerical methods can be partitioned into two classes: deterministic and stochastic numerical integration. The former class involves all approximation techniques of integrals that do not entail simulation (Cameron and Trived, 2005). Classical deterministic quadratures are Riemann summation, Simpson's method and the trapezoidal rule. However, these techniques suffer from the 'curse of dimensionality' meaning that the number of integrand evaluations increases exponentially with the number of dimensions. Therefore, these methods become infeasible for multi-dimensional integrals which are often of our interest (Brooks, Gelman, Jones, and Meng, 2011). The latter class uses simulation to approximate integrals. Monte Carlo methods constitute a large portion of this class. Monte Carlo methods refer to a collection of algorithms that utilize pseudo-random sampling to obtain numerical results, which are integrals in our consideration. An advantage of Monte Carlo methods, next to the absence of the 'curse of dimensionality', is the utilization of the probabilistic nature of integrals. This refers to exploring the integrand's regions of importance by exploiting the information provided by the probability density function associated with the integrals (Robert and Casella, 2010).

Monte Carlo methods were formally introduced for the first time when Stanislaw Ulam proposed a solution for neutron diffusion problems to the fellow mathematician John von Neumann in 1947. This proposal was inspired by the first programmable, electronic digital computer, called ENIAC, which was recently developed. The implementation of these methods to model thermonuclear fusion reactions and neutron diffusion directly influenced the development of the hydrogen bomb at the Los Alamos National Laboratory. Nicholas Metropolis, a physicist who also worked at Los Alamos, suggested to assign the name 'Monte Carlo' to these methods that revolutionized computational mathematics. Monte Carlo refers to the well-known casino in Monaco that was frequently visited by Ulam's uncle (Metropolis, 1987). However, in the 1930s, the physisct Enrico Fermi already used similar techniques to study neutron transport, but he did not publish it. Furthermore, the first application of this technique was introduced by the Buffon needle problem in the 18^{th} century (Shonkwiler and Mendivil, 2024).

This paper presents a survey that introduces the basics of Monte Carlo integration. In particular, we start with explaining the main principle of Monte Carlo integration which is rewriting the integrand in terms of a probability density function and some other function. This procedure leads to another expression for the integral of interest that involves an expectation operator. Only integrals that can be rewritten in terms of a common probability density function are presented in this part. Next, a technique, called importance sampling, that can possibly reduce the variance of an integral's Monte Carlo estimator is presented. This technique is frequently used in practice. However, it is often infeasible to rewrite a multi-dimensional integral in terms of some probability density function. This is especially the case when there is some dependency which implies that the multivariate density function does not simplify to a product of univariate density functions. Hence, a method that enables us to sample from a complicated probability density function is needed. The most common such method is called Markov chain Monte Carlo. This method utilizes a type of stochastic

processes called Markov chains. Therefore, we first introduce Markov chains in both discrete- and continuous-time. Next, an econometric application that involves both Bayesian statistics and generalized linear models is given.

Moreover, as a by-product of this study, stand-alone files are created. In particular, we create three files, where in each we explain the components that build the theory of Monte Carlo integration. We take two broad audiences into consideration, the scientific community and first-time learners. The latter is important because it can be demanding to understand these concept both theoretically and practically. We explain the theory in detail, and we demonstrate all the theory in `MATLAB` by means of relevant examples.

`MATLAB` is the software of choice. We avoid commonly used programming languages, such as python and R, because the aim of this survey is to be pedagogically beneficial for a large audience. `MATLAB` comes forward in this respect because the syntax of the language is relatively simple and it can directly refer to the theory.

Well-known references regarding these Monte Carlo integration methods are Train (2009), Rubinstein (2016), Robert and Casella (2004), Robert and Casella (2010), and Chib and Greenberg (1969). These references either mainly focus on the theory part, or they exclude most of the details of the theory and primarily apply the results. Therefore, neither of these references, nor any available book on the subject, provides a complete survey of the basics of Monte Carlo integration with extensive theory and applications parts. In particular, it can be challenging for first-time learners to truly understand Monte Carlo integration accompanied by a Markov chain Monte Carlo algorithm. We fill in this gap with simple examples for each topic we cover.

First, Monte Carlo integration is explained for one-dimensional integrals. In addition, we show that this results in an unbiased estimator of the integral and we analyse the performance of this estimator analytically. After some examples that assess the relevant Monte Carlo estimator's performance, the logic of Monte Carlo integration is extended to multi-dimensional integrals in the special case of independence. Second, a variance reduction method called importance sampling is considered. Furthermore, an alternative importance sampling technique, called weighted importance sampling, which has greater practical applicability is explained. Both techniques are demonstrated by examples. Third, two common techniques to generate non-uniform samples are explained. These techniques are the inverse transformation method and rejection sampling. The latter has subtle similarities with the proposal mechanism of Markov chain Monte Carlo algorithms. Fourth, Markov chains together with the Markov chain variant of the strong law of large numbers are explained. Understanding these concepts is necessary to thoroughly comprehend the mechanism of Markov chain Monte Carlo algorithms. In particular, the most general Markov chain Monte Carlo algorithm, called Metropolis-Hastings algorithm, is considered. Finally, the Metropolis-Hastings algorithm is applied to calculate a predictive posterior probability of a Bayesian linear regression.

2 Monte Carlo Integration

2.1 Theory

Suppose we have an integrable function $\phi : \Omega \subseteq \mathbb{R} \to \mathbb{R}$ that is bounded on some $A \subseteq \Omega$. Integrability of a function means that the integral of ϕ exists and is finite: $\int |\phi| \, d\mu < \infty$, where μ is a measure. Continuity is a sufficient condition for integrability of real-valued functions. For completeness, the mathematical framework that will be used is the probability space $(\Omega, \mathcal{B}(\Omega), \mathbb{P})$, where $\mathcal{B}(\Omega)$ denotes the Borel σ-algebra on Ω and $\mathbb{P}$ some probability measure (Bierens, 2005). Knowledge of measure theory is not necessary for understanding this survey, but it is presented for mathematical rigorosity (Durrett, 2019).

Our goal is to approximate the following definite integral over some domain of integration $A \subseteq \Omega$:

$$\int_{A \subseteq \Omega} \phi(x) \, dx. \tag{1}$$

The objective of the Monte Carlo integration principle is to rewrite the integral in equation (1) in terms of the expectation of some random variable $g(X)$, where g is a real-valued function. We denote random variables by capital letters, in this case $X \equiv X(\omega) : \Omega \to \mathbb{R}$ denotes a random variable on the probability space $(\Omega, \mathcal{B}(\Omega), \mathbb{P})$. This rewriting is done by splitting the integrand ϕ into two functions, a probability density function (PDF) $f_X : \Omega \to \mathbb{R}$ and some $g : \mathbb{R} \to \mathbb{R}$. A function f_X is a PDF if $f_X(\omega) \geq 0$, $\forall \omega \in \Omega$, and $\mathbb{P}(\Omega) = \int_\Omega f_X(\omega) \, d\omega = 1$. Therefore, we aim to recognize a function f_X, which satisfies the PDF conditions, in ϕ and define the remaining part of ϕ as the function g. Furthermore, common PDFs f_X are preferred, because it is easy to generate sample from them. Common PDFs refer to PDFs of well-known distributions such as the uniform, exponential and normal distribution. The Monte Carlo integration principle is given by:

$$\int_A \phi(x) \, dx = \int_{\mathbb{R}} g(x) \cdot f_X(x) \, dx =: \mathbb{E}_{f_X}[g(X)]. \tag{2}$$

Observe that the domain of integration changes from A to $\mathbb{R}$, because the expectation of a real-valued continuous random variables involves an integral over $\mathbb{R}$ by definition. The PDF subscript f_X in $\mathbb{E}_{f_X}$ is sometimes omitted when it is clear from the context.

For example, if $A := [a, b] \subseteq \mathbb{R}$ with $a < b$, then the integral in equation (1) can be rewritten using a random variable X with a uniform distribution on the interval $[a, b]$.

$$\int_a^b \phi(x) \, dx = (b - a) \cdot \int_{\mathbb{R}} \phi(x) \cdot \frac{1}{b - a} \mathbb{I}\{x \in [a, b]\} \, dx = (b - a) \cdot \mathbb{E}[\phi(X)], \tag{3}$$

where the indicator function $\mathbb{I}\{E\} = \mathbb{I}\{x \in E\}$ of an event E equals 1 if the event is true, and zero otherwise.

However, the expectation of $g(X)$ can not be always calculated analytically. Hence, the Monte Carlo method, following Metropolis and Ulam (1949), is used to approximate $\mathbb{E}_{f_X}[g(X)]$. First, an independent identically distributed (i.i.d.) random sample $X_1, ..., X_{n \in \mathbb{N}}$ is generated from the PDF f_X. Second, the Monte Carlo integration estimator, which is the empirical average of $g(X)$, is calculated by:

$$\bar{g}_n := \frac{1}{n} \sum_{i=1}^{n} g(X_i). \tag{4}$$

But how can we be sure that this is a good estimator for the expectation $\mathbb{E}_{f_X}[g(X)]$? The Strong Law of Large Numbers (SLLN) ensures us that this is the case for a sufficiently large sample size (Robert and Casella, 2010).

Theorem 2.1. (Strong Law of Large Numbers) *(Durrett, 2019) Let $Y_1, Y_2, \ldots$ be pairwise i.i.d. random variables with $\mathbb{E}[|Y_i|] < \infty$, $\forall i \in \mathbb{N}$. Then:*

$$\frac{1}{n}\sum_{i=1}^{n} Y_i \xrightarrow{a.s.} \mathbb{E}[Y_i] \quad \iff \quad \mathbb{P}\left(\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} Y_i = \mathbb{E}[Y_i]\right) = 1. \tag{5}$$

This theorem informs us that for an i.i.d. sample with finite expectation, the empirical average converges to true mean as the sample size n goes to ∞. We can apply this to our problem, because composing a random variable X_i by a function g creates another random variable $g(X_i)$. Hence, Monte Carlo integration presents $\bar{g}_n$ as an approximation of $\int_{A\subseteq\Omega} \phi(x)\,dx$ when the sample size n is sufficiently large:

$$\bar{g}_n := \frac{1}{n}\sum_{i=1}^{n} g(X_i) \xrightarrow{a.s.} \mathbb{E}_{f_X}[g(X)] = \int_A \phi(x)\,dx. \tag{6}$$

Moreover, $\bar{g}_n$ turns out to be a 'good' approximation of $\int_A \phi(x)\,dx$. The same is true for multi-dimensional integrals in the special case of independence as will be demonstrated at the end of this section.

Next, we want to assess the performance of the Monte Carlo integration estimator $\bar{g}_n$.

Theorem 2.2. *$\bar{g}_n$ is an unbiased estimator, with variance:*

$$\mathrm{Var}_{f_X}[\bar{g}_n] = \frac{\mathrm{Var}_{f_X}[g(X)]}{n}, \tag{7}$$

assuming $\mathrm{Var}_{f_X}[g(X)]$ exists. Hence, the speed of convergence of Monte Carlo integration estimator is $\mathcal{O}(n^{-\frac{1}{2}})$.

Proof: $\bar{g}_n$ is an unbiased estimator of $\mathbb{E}_{f_X}[g(X_i)]$, because:

$$\mathbb{E}_{f_X}[\bar{g}_n] = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{f_X}[g(X_i)] \overset{i.d.}{=} \frac{1}{n}\cdot n \cdot \mathbb{E}_{f_X}[g(X_i)] = \mathbb{E}_{f_X}[g(X_i)], \tag{8}$$

where the first equality holds by linearity of expectation. Therefore, the Monte Carlo estimator $\bar{g}_n$ is an unbiased estimator for the integral of interest, i.e.: $\mathbb{E}_{f_X}[\bar{g}_n] = \int_A f(x)\,dx$. Next, the variance of $\bar{g}_n$ can be rewritten as follows:

$$\mathrm{Var}_{f_X}[\bar{g}_n] \overset{ind.}{=} \frac{1}{n^2}\sum_{i=1}^{n} \mathrm{Var}_{f_X}[g(X_i)] \overset{i.d.}{=} \frac{1}{n^2}\cdot n \cdot \mathrm{Var}_{f_X}[g(X_i)] = \frac{\mathrm{Var}_{f_X}[g(X_i)]}{n} \tag{9}$$

Notice that $\mathbb{E}_{f_X}[g(X_i)] = \mathbb{E}_{f_X}[g(X)]$ and $\mathrm{Var}_{f_X}[g(X_i)] = \mathrm{Var}_{f_X}[g(X)]$, because X_i and X have the same distribution. Lastly, by Robbins central limit theorem (Robbins, 1948), we obtain:

$$\begin{aligned}\frac{\bar{g}_n - \mathbb{E}_{f_X}[\bar{g}_n]}{\sqrt{\mathrm{Var}_{f_X}[\bar{g}_n]}} \xrightarrow{d} \mathcal{N}(0,1) \quad \iff \quad \bar{g}_n &\xrightarrow{d} \mathcal{N}\left(\mathbb{E}_{f_X}[\bar{g}_n], \mathrm{Var}_{f_X}[\bar{g}_n]\right) \\ &\overset{d}{=} \mathcal{N}\left(\int_A f(x)\,dx, \left[\frac{\mathrm{Var}_{f_X}[g(X)]}{\sqrt{n}}\right]^2\right),\end{aligned} \tag{10}$$

where $\mathbb{E}_{f_X}[\bar{g}_n] < \infty$ because ϕ is assumed to be integrable. Hence, the standard deviation of the Monte Carlo integration estimator $\bar{g}_n$ decreases at the rate of $\mathcal{O}(n^{-1/2})$. ■

In practice, the variance of $\bar{g}_n$ is not known, but it can be estimated by the sample variance:

$$\widehat{\mathrm{Var}_{f_X}[\bar{g}_n]} = \frac{1}{n} \cdot \left[\frac{1}{n-1} \sum_{i=1}^{n} (g(x_i) - \bar{g}_n)^2 \right] = \frac{1}{n(n-1)} \sum_{i=1}^{n} (g(x_i) - \bar{g}_n)^2, \tag{11}$$

where $\frac{1}{n-1} \sum_{i=1}^{n} (g(x_i) - \bar{g}_n)^2$ is the unbiased estimator of $\mathrm{Var}_{f_X}[g(X)]$, and x_i denotes a realization of X_i. $\widehat{\mathrm{Var}_{f_X}[\bar{g}_n]}$ is an estimator itself, and it turns out to be unbiased:

$$\mathbb{E}\left[\widehat{\mathrm{Var}_{f_X}[\bar{g}_n]}\right] = \frac{1}{n}\mathrm{Var}_{f_X}[g(X)] = \mathrm{Var}_{f_X}[\bar{g}_n], \tag{12}$$

where the last equality holds by equation (9).

Another measure for accuracy of an estimator is the mean squared error (MSE) (Dodge, 2008):

$$MSE(\bar{g}_n) := \mathbb{E}_{f_X}\left[\left(\int_A f(x)\,dx - \bar{g}_n \right)^2 \right] = \mathrm{Var}_{f_X}[\bar{g}_n] + \mathrm{Bias}(\bar{g}_n)^2, \tag{13}$$

where $\mathrm{Bias}(\bar{g}_n) := \mathbb{E}_{f_X}[\bar{g}_n] - \int_A f(x)\,dx$. Hence, when the estimator is unbiased, MSE equals the variance of $\bar{g}_n$. So, in the case of Monte Carlo integration, the MSE and variance are equivalent. Moreover, this measure will also be used for more advanced techniques introduced in subsequent sections.

2.2 Applications

Example 2.1: Identity Function
One of the most simple integrals is the integral of $\phi(x) = x$ over the interval $[0, 1]$:

$$\int_0^1 x\,dx = \left[\frac{1}{2}x^2\right]_{x=0}^{x=1} = \frac{1}{2}. \tag{14}$$

This integral can be solved analytically, but we are going to approximate it by Monte Carlo integration. We choose the most simple PDF called the standard uniform distribution for this example since ϕ is the identity map. The standard uniform PDF is a common choice because it is relatively simple to implement. Draw a random variable X from a uniform distribution over $[0, 1]$, i.e. $X \sim \mathrm{Unif}(0, 1)$. The corresponding PDF $f_X : [0, 1] \to \{0, \frac{1}{1-0} = 1\}$ is given by $f_X(x) = \mathbb{I}\{x \in [0, 1]\}$. Hence, we can rewrite the integral by using f_X:

$$\int_0^1 x\,dx = \int_{-\infty}^{\infty} x \cdot \mathbb{I}\{x \in [0,1]\}\,dx = \int_{-\infty}^{\infty} x \cdot f_X(x)\,dx = \mathbb{E}[X]. \tag{15}$$

This can be estimated by the following algorithm:

Algorithm: Pseudo-Code for **Monte Carlo Integration**

1: Input: Sample size n
2: **for** $i = 1, ..., n$
3: Sample $X_i \sim f_X$
4: Compute $g(X_i)$
5: **end**
6: Compute the estimator $\frac{1}{n} \sum_{i=1}^{n} g(X_i)$

Using the random number generating function `random` in MATLAB, we draw a random sample of size `n` from a uniform distribution over $[0, 1]$. Next, the empirical average of the sample is calculated using the `mean` function.

```
rng('default')   % Control random number generator

tic
n = 1000;   % Sample size
% Generate n realizations from Unif(0,1)
StandardUniform_Realization = random('Uniform',0,1,[n 1]);
Mean_Value = mean(StandardUniform_Realization);   % Empirical average
MC_Integral_Estimate = Mean_Value;   % Integral estimate
toc

MSE = 1/(n*(n-1))*sum((StandardUniform_Realization - 0.5).^2);
```

`MC_Integral_Estimate` produces a value of approximately `0.489`, which has MSE of approximately `8.035e-05`. This represents a reasonable estimate of the integral in equation (15), while using a sample size of only one thousand. A more intuitive understanding of Monte Carlo integration is presented through a visual representation of the process.

```
%% Creating a plot
plot(StandardUniform_Realization,'ro','MarkerFaceColor','red')
% 'ro' specificies red circles without connecting lines
hold on;
% Selecting points below the (0,0)-(1,1) diagonal
Below_Diagonal = StandardUniform_Realization < (1:n)'/n;
% Colour points below diagonal as green filled circles ('go')
plot(find(Below_Diagonal),StandardUniform_Realization(Below_Diagonal), ...
     'go','MarkerFaceColor','green')
Grey = [0.5 0.5 0.5];  % [0.5 0.5 0.5] yields a grey color
plot([0 n], [0 1],'Color',Grey,'LineWidth',2)
title('Monte Carlo Integration $\phi(x)=x$', ...
     'FontSize',27,'Interpreter','latex');
xlabel('n','FontSize',21,'Interpreter','latex');
ylabel('Unif$(0,1)$ Realization','FontSize',21,'Interpreter','latex');
hold off;
```

Monte Carlo integration approximates the area under $\phi(x) = x$ on the interval $[0, 1]$ by generating random points in the rectangle $[0, n] \times [0, 1]$ and multiplying the proportion of points that lie below the line by the total area of the square. See Figure 1. The points below the graph of the identity function $\phi(x) = x$ are coloured green, covering approximately half the area of the square. ▶

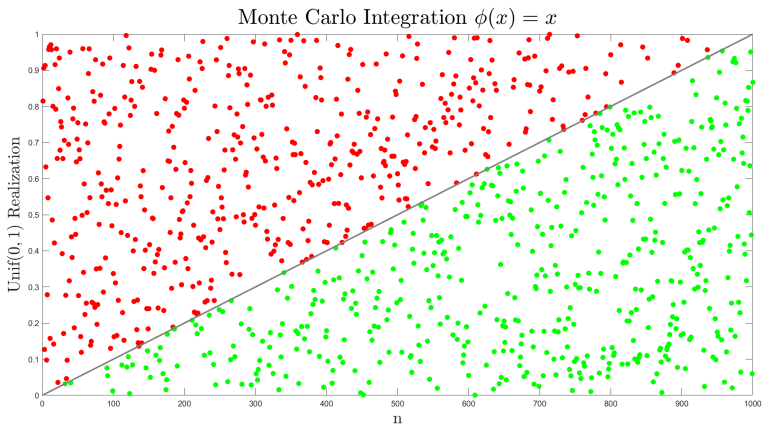

Figure 1

Example 2.2: Gaussian Integral
Arguably, one of the most well-known integrals is the definite Gaussian integral, which has a closed-form solution.

$$\int_{-\infty}^{\infty} e^{-x^2}\, dx = \sqrt{\pi} \approx 1.773. \tag{16}$$

Notice that $\frac{1}{\sqrt{\pi}}$ term in the normalization constant of the normal PDF is because of this integral.

To perform Monte Carlo integration for the definite Gaussian integral, we draw X from a standard normal distribution: $X \sim \mathcal{N}(0,1)$. We choose a standard normal PDF because it contains an e^{-x^2} term. Hence, X has the density function $f_X(x) : \mathbb{R} \to \mathbb{R}_{>0}$ defined by $f_X(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$. Then, we rewrite the integral by utilizing f_X, noting the presence of a similar exponential term.

$$\begin{aligned}
\int_{-\infty}^{\infty} e^{-x^2}\, dx &= \int_{-\infty}^{\infty} e^{-\frac{1}{2}x^2 - \frac{1}{2}x^2} \cdot \sqrt{\frac{2\pi}{2\pi}}\, dx \\
&= \int_{-\infty}^{\infty} \left[\sqrt{2\pi} e^{-\frac{1}{2}x^2}\right] \cdot \left[\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}\right] dx \\
&= \sqrt{2\pi} \int_{-\infty}^{\infty} \left[e^{-\frac{1}{2}x^2}\right] \cdot f_X(x)\, dx \\
&= \sqrt{2\pi} \cdot \mathbb{E}\left[e^{-\frac{1}{2}X^2}\right].
\end{aligned} \tag{17}$$

n normally distributed random numbers can be generated by the `random` function. The empirical average can then be calculated after applying the quadratic exponentiation $e^{-\frac{1}{2}X^2}$ to the generated random numbers.

```
rng('default')   % Control random number generator

tic
n = 1000;    % Sample size
% Generate n realizations from N(0,1)
StandardNormal_Realizations = random('Normal',0,1,[n 1]);

Exponential_Quadratic = exp(-0.5*StandardNormal_Realizations.^2);
```

```
Mean_Value = mean(Exponential_Quadratic);    % Expecation
MC_Integral_Estimate = sqrt(2*pi)*Mean_Value;    % Integral estimate
toc

MSE = 1/(n*(n-1))*sum((sqrt(2*pi).*Exponential_Quadratic ...
     - sqrt(pi)).^2);
```

`MC_Integral_Estimate` produces an estimate of approximately `1.788`, which has a `MSE` of approximately `4.817e-04`. The convergence of the estimate and its MSE can be visualized to gain insight into the underlying process.

```
%% Convergence plots
% Plotting the convergence of the MC estimate:
Estimate_Convergence = cumsum(sqrt(2*pi).*Exponential_Quadratic)./(1:n)';
plot(1:n, Estimate_Convergence,'LineWidth',2,'Color','blue')
hold on;
yline(sqrt(pi),'--g','LineWidth',2)
title('Convergence of Estimate','FontSize',34,'Interpreter','latex')
xlabel('n','FontSize',21,'Interpreter','latex')
ylabel('Estimate','FontSize',21,'Interpreter','latex')
legend({'Monte Carlo Estimate', '$$\sqrt{\pi}$$'}, ...
     'FontSize',21,'Location','northeast','Interpreter','latex');
hold off;

% Plotting the error convergence of the estimate:
Sample_Variance = cumsum((sqrt(2*pi).*Exponential_Quadratic ...
     - sqrt(pi)).^2)./(0:n-1)';
MSE_Convergence = Sample_Variance./(1:n)';
plot(1:n,MSE_Convergence,'LineWidth',2,'Color','blue')
hold on;
Orange = [240 100 10]/256;
plot(1:n, 1./sqrt(1:n),'LineWidth',2,'Color',Orange)
title('MSE of Monte Carlo Estimate','FontSize',34,'Interpreter','latex')
xlabel('n','FontSize',21,'Interpreter','latex')
ylabel('MSE','FontSize',21,'Interpreter','latex')
legend({'MSE','$$n^{-1/2}$$'},'FontSize',21,'Location','northeast', ...
     'Interpreter','latex');
hold off;
```

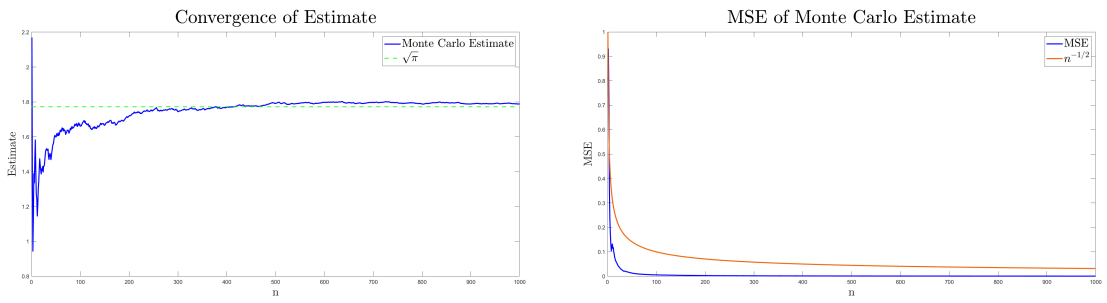

Figure 2

The left-hand side plot in Figure 2 illustrates the convergence of the Monte Carlo estimate as the sample size grows. Notably, the estimate is already reasonably accurate with a sample

size of just 250. Furthermore the right-hand side plot in Figure 2 depicts the behaviour the MSE as the sample size increases. The MSE remains below the graph of $n^{-1/2}$, because MSE and variance equals in Monte Carlo integration, which convergences at a rate of $\mathcal{O}(n^{-1/2})$. ▶

Example 2.3: Prime Counting

One of the most important conjectures in mathematics is the Riemann hypothesis (Riemann, 1859). It has important implications for the distribution of prime numbers, which is, to this day, not completely understood. However, the prime number theorem gives us the asymptotic distribution of primes.

Let us define the prime-counting function by $\pi(x)$, where $x \in \mathbb{R}$. For example, $\pi(3) = \#\{2,3\} = 2$, where $\#$ denotes the cardinality of a set, and $\pi(10) = 4$. Further, let us define the offset logarithmic integral by $Li(x) := \int_2^x \frac{dy}{\log(y)}$ for $x \in \mathbb{R}_{\geq 2}$. Then, the prime number theorem informs us the following (Jameson, 2003):

$$\pi(x) \sim Li(x) \quad \iff \quad \lim_{x\to\infty} \frac{\pi(x)}{Li(x)} = 1, \quad x \in \mathbb{R}_{\geq 2}. \tag{18}$$

This means that $Li(x)$ and $\pi(x)$ are asymptotically equivalent. Or in other words, roughly speaking, that the probability that x is a prime is $\frac{1}{\log(x)}$. Therefore, $Li(x)$ is considered as a 'good' approximation of the number of primes up to and including x.

We restrict the prime counting function to positive integers as domain: $\pi\,|_{\mathbb{N}}$. Draw $Y \sim \text{Unif}(2,n)$, where $n \in \mathbb{N}_{>2} := \{2,3,...\}$. The corresponding PDF $f_Y : \mathbb{R} \to \{0, \frac{1}{n-2}\}$ of Y is $f_Y(y) = \frac{1}{n-2} \cdot \mathbb{I}\{y \in [2,n]\}$. $Li(n)$ can be rewritten by utilizing f_X:

$$\begin{aligned} Li(n) = \int_2^n \frac{dy}{\log(y)} &= \int_{-\infty}^{\infty} \frac{1}{\log(y)} \cdot \frac{n-2}{n-2} \cdot \mathbb{I}\{y \in [2,n]\}\, dy \\ &= (n-2) \cdot \int_2^n \frac{1}{\log(y)} \cdot f_Y(y)\, dy \\ &= (n-2) \cdot \mathbb{E}\Big[\frac{1}{\log(y)}\Big]. \end{aligned} \tag{19}$$

Generate n uniformly distributed random numbers on $[2,n]$ by the `random` function. The empirical average can then be calculated after applying the inverse logarithm $\frac{1}{\log(y)}$ to the generated random numbers.

```
rng('default')   % Control random number generator

tic
n_Values = 100:100:10000;   % Sequence from 100 to 10,000 by steps of 100
n = 1000;   % Sample size
% Next, create empty vectors to store values:
MC_Log_Integral = zeros(size(n_Values));
Prime_Counts = zeros(size(n_Values));
% Remark: Using length() instead of size() creates a matrix

for i = 1:length(n_Values)
     N = n_Values(i);
     % Generate n realizations from Unif(2,N)
     Uniform_N_Realization = random('Uniform',2,N,[n 1]);
     Log_Transformation = 1./log(Uniform_N_Realization);
```

```
    Mean_Value = mean(Log_Transformation);
    MC_Log_Integral(i) = (N-2)*Mean_Value;
    Prime_Counts(i) = length(primes(N));
end
toc
```

This results in two arrays that are difficult to interpret. Hence, we create a plot that visualizes both arrays.

```
%% Plotting the Monte Carlo Integration estimates
plot(n_Values,MC_Log_Integral,'-og','MarkerFaceColor','green')
hold on;
Light_Blue = [0 0.5 1];   % Plotting true prime counts
stem(n_Values,Prime_Counts,'Marker','none','LineWidth',1.5, ...
    'Color',Light_Blue)

title(['Monte Carlo Estimation of Logarithmic Integral ' ...
    'for Prime Counting'],'FontSize',34,'Interpreter','latex')
xlabel('$n$','FontSize',21,'Interpreter','latex')
ylabel('$\#$ primes up to integer n','FontSize',21,'Interpreter','latex')
legend({'MC Log Integral','Prime Counts $\pi$'}, ...
    'Location','northwest','FontSize',30,'Interpreter','latex');
hold off;
```

Figure 3 illustrates the close approximation of the number primes up to 10,000 by the offset logarithmic integral, where the integral is estimated by means of Monte Carlo integration. ▶

Monte Carlo Estimation of Logarithmic Integral for Prime Counting

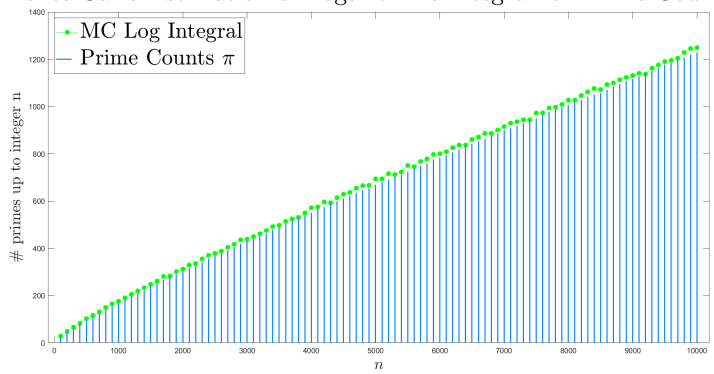

Figure 3

2.3 Multi-Dimensional Integrals

Generalizing the Monte Carlo integration technique to multiple integrals over a d-dimensional space is relatively straightforward. Suppose we have an integrable function $\phi : \Omega \subseteq \mathbb{R}^d \to \mathbb{R}$. Let $\boldsymbol{x} = (x_1, ..., x_d)^T$ be a d-tuple in $\Omega \subseteq \mathbb{R}^d$, and let $\boldsymbol{X} = (X_1, ..., X_d)^T$ be a random vector.

Our goal is to approximate the following definite multiple integral over some domain of integration $A := A_1 \times ... \times A_d \subseteq \Omega$:

$$\int_A \phi(\boldsymbol{x})\, d\boldsymbol{x} = \int_{A_d} \cdots \int_{A_1} \phi(x_1, ..., x_d)\, dx_1 \cdots dx_d, \tag{20}$$

by approximating $\mathbb{E}_{f_{\boldsymbol{X}}}[g(\boldsymbol{X})]$.

The SLLN in Theorem 2.1 also holds for random vectors. Hence, we can use the same procedure of Monte Carlo integration for one-dimensional integrals. First, generate a sample of n i.i.d. random vectors $\boldsymbol{X}_1, ..., \boldsymbol{X}_n$, where each random vector $\boldsymbol{X}_i = (X_{i1}, ..., X_{id})^T$ is drawn from the joint PDF $f_{\boldsymbol{X}}(x_1, ..., x_d)$. Second, the empirical average of $g(\boldsymbol{X}_1), ..., g(\boldsymbol{X}_n) \in \mathbb{R}$ is calculated by:

$$\bar{g}_n := \frac{1}{n}\sum_{i=1}^{n} g(\boldsymbol{X}_i), \tag{21}$$

which converges to $\mathbb{E}_{f_{\boldsymbol{X}}}[g(\boldsymbol{X})]$ as $n \to \infty$ by the SLLN. Therefore, Monte Carlo integration for multi-dimensional integrals can be summarized by:

$$\bar{g}_n := \frac{1}{n}\sum_{i=1}^{n} g(\boldsymbol{X}_i) \xrightarrow{a.s.} \mathbb{E}_{f_{\boldsymbol{X}}}[g(\boldsymbol{X})] = \int_{\mathbb{R}} g(\boldsymbol{x}) \cdot f_{\boldsymbol{X}}(\boldsymbol{x})\, d\boldsymbol{x} = \int_A \phi(\boldsymbol{x})\, d\boldsymbol{x}, \tag{22}$$

where $\boldsymbol{X}_i \sim f_{\boldsymbol{X}}(\boldsymbol{x})$, $\forall i \in \{1, ..., n\}$. A generalized version of Theorem 2.2 also holds in the case of multiple integrals.

However, since the all the random variables X_j in a random vector $\boldsymbol{X}$ are independent, the joint PDF $f_{\boldsymbol{X}}$ can be written as a product of univariate PDFs (Blitzstein and Hwang, 2019):

$$f_{\boldsymbol{X}}(\boldsymbol{x}) := f_{X_1,...,X_d}(x_1, ..., x_d) = f_{X_1}(x_1) \cdot ... \cdot f_{X_d}(x_d) = \prod_{j=1}^{d} f_{X_j}(x_j), \tag{23}$$

where $f_{X_j}(x_j)$ denotes the marginal/univariate PDF of the j^{th} entry of the random vector $\boldsymbol{X}$. For example, when using a uniform PDF, it is easier to handle each integral separately with a univariate PDF than using a joint PDF. Hence, we can rewrite $\mathbb{E}_{f_{\boldsymbol{X}}}[g(\boldsymbol{X})]$:

$$\begin{aligned}
\mathbb{E}_{f_{\boldsymbol{X}}}[g(\boldsymbol{X})] &= \int_{\mathbb{R}} \cdots \int_{\mathbb{R}} g(\boldsymbol{x}) \cdot [f_{X_1}(x_1) \cdot ... \cdot f_{X_d}(x_d)]\, dx_1 \cdots dx_d \\
&= \int_{\mathbb{R}} \cdots \int_{\mathbb{R}} f_{X_d}(x_d) \cdot ... \cdot f_{X_2}(x_2) \int_{\mathbb{R}} g(\boldsymbol{x}) \cdot f_{X_1}(x_1)\, dx_1 \cdots dx_d \\
&= \int_{\mathbb{R}} \cdots \int_{\mathbb{R}} f_{X_d}(x_d) \cdot ... \cdot f_{X_3}(x_3) \cdot \int_{\mathbb{R}} \mathbb{E}_{f_{X_1}}[g(x_1, x_2, ..., x_d)] \cdot f_{X_2}(x_2)\, dx_2 \cdots dx_d \\
&= \mathbb{E}_{f_{X_d}} \circ ... \circ \mathbb{E}_{f_{X_2}} \circ \mathbb{E}_{f_{X_1}}[g(\boldsymbol{X})] \in \mathbb{R}.
\end{aligned} \tag{24}$$

When g can be written as a product, i.e. $g(\boldsymbol{x}) = \prod_{j=1}^{d} g(x_j)$, then equation (24) simplifies to

$$\begin{aligned}
\mathbb{E}_{f_{\boldsymbol{X}}}[g(\boldsymbol{X})] &= \int_{\mathbb{R}} \cdots \int_{\mathbb{R}} \prod_{j=1}^{d} g(x_j) \cdot f_{X_j}(x_j)\, dx_1 \cdots dx_d \\
&\overset{(\star)}{=} \prod_{j=1}^{d} \int_{\mathbb{R}} g(x_j) \cdot f_{X_j}(x_j)\, dx_j,
\end{aligned} \tag{25}$$

where equality $(\star)$ holds by Fubini's theorem (Durrett, 2019).

Observe that equations (24) and (25) only provide an approximation technique in special cases. Generally, for complicated multi-dimensional integrals, it is not possible to use the approximation technique described in equation (22) because it is often infeasible to directly draw from a multi-dimensional PDF $f_{\boldsymbol{X}}(\boldsymbol{x})$. In section 4, a class of algorithms is described that enables us to indirectly generate samples from $f_{\boldsymbol{X}}(\boldsymbol{x})$. However, when the PDF $f_{\boldsymbol{X}}(\boldsymbol{x})$ in

equation (22) is a 'common' multi-dimensional PDF from which we can directly draw samples, then equation (22) can be used. An example of a common multi-dimensional PDF $f_{\boldsymbol{X}}(\boldsymbol{x})$, that does not satisfy equation (23) but from which we can directly draw samples, is a multivariate normal PDF with a covariance matrix that has non-zero off-diagonal elements.

Furthermore, let us reconsider the convergence rate $\mathcal{O}(n^{-\frac{1}{2}})$ from Theorem 2.2. This means that in order to halve the standard deviation of the estimator, the sample size n needs to be quadrupled: $\mathcal{O}(\frac{1}{\sqrt{4n}}) = \frac{1}{2}\,\mathcal{O}(n^{-\frac{1}{2}})$. In contrast, approximating the d-dimensional integral in equation (20) by deterministic methods, such as Riemann summation, yields a standard deviation of order $\mathcal{O}(n^{-\frac{1}{d}})$. See Figure 4 for some examples of the sequence $n^{-\frac{1}{d}}$.

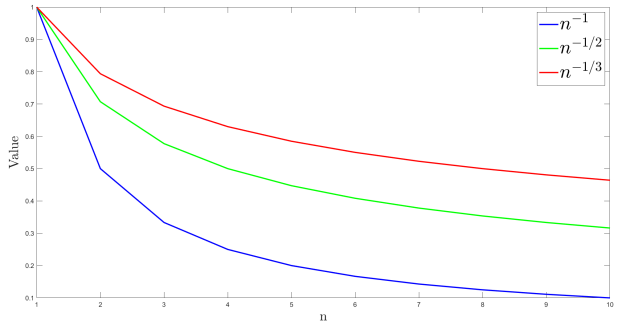

Figure 4

There is a curse of dimensionality with these deterministic methods, because $\mathcal{O}(n^{-\frac{1}{d}})$ depends on the dimension d. This is however not the case with Monte Carlo integration, because $\mathcal{O}(n^{-\frac{1}{2}})$ does not depend on d. Therefore, Monte Carlo integration is a leading and competitive technique for high-dimensional integrals, but it is not the best choice for low-dimensional integrals (Shonkwiler and Mendivil, 2024). However, it turns out that $\mathcal{O}(n^{-\frac{1}{2}})$ is not the best convergence rate that can be attained. In fact, as will be demonstrated in section 3, it exhibits one of the slowest convergence rates amongst sensible integration algorithms.

Example 2.4: Profit Calculation
Consider an energy supplier that exclusively offers electricity and gas. The supplier aims to calculate its profit π for the previous year. Suppose the demand for gas fluctuates per season, while the electricity demand stays constant. For simplicity, assume that initial price for both electricity and gas is €5 per day. And that there are hundred units available for each per day.

Let x and y be the quantities sold of the products gas and electricity per day, respectively. The profit function $\pi : \mathbb{R}^3 \to \mathbb{R}$ at a particular point in time t is given by:

$$\pi(x, y, t) = [P(x, y, t) - C(x, y, t)] \cdot Q(x, y, t), \tag{26}$$

where P, C, and Q are the price, cost and demand function, respectively.

Firstly, the demand function $Q : \mathbb{R}^3 \to \mathbb{R}$ is given by:

$$\begin{aligned} Q(x, y, t) &= [\text{Initial Demand} - \text{Future Demand Reduction}] \cdot (\text{Growth of Demand}) \\ &= [80 - 0.05x \cdot \rho - 0.08y] \cdot e^{\frac{t}{1000}}, \end{aligned} \tag{27}$$

where $\rho := \frac{1}{3}\cos\left(\frac{2\pi t}{365} - \frac{\pi}{6}\right) + 1$ denotes the seasonal effect term. This results in higher demand during the winter months compared to the summer months. Since Q depends both on x and y, it represents the total number of units sold.

Secondly, the price function $P : \mathbb{R}^3 \to \mathbb{R}$ is given by:

$$\begin{aligned} P(x, y, t) &= \text{Price} \cdot \text{Inflation} - \text{Price Sensitivity} \\ &= 5 \cdot \left(1 + \frac{0.02}{365}\right)^t - \frac{1}{200}x - \frac{1}{300}y. \end{aligned} \tag{28}$$

Thirdly, the cost function $C : \mathbb{R}^3 \to \mathbb{R}$ is given by:

$$\begin{aligned} C(x, y, t) &= [\text{Fixed Costs} + \text{Variable Costs}] \cdot (\text{Time Adjustment Factor}) \\ &= [2 + 0.015x + 0.01y] \cdot \left(1 + \frac{t}{1000}\right). \end{aligned} \tag{29}$$

The supplier's goal is to approximate last year's profit:

$$\Pi := \int_0^{365} \int_0^{100} \int_0^{100} \pi(x, y, t)\, dx\, dy\, dt \tag{30}$$

by Monte Carlo integration.

Firstly, define random variables $T \sim \text{Unif}(0, 365)$ with PDF $f_T(t) = \frac{1}{365} \cdot \mathbb{I}\{t \in [0, 365]\}$, and $X, Y \sim \text{Unif}(0, 100)$ with PDF $f_X(x) = f_Y(y) = \frac{1}{100} \cdot \mathbb{I}\{x, y \in [0, 100]\}$. Hence, Π can be rewritten as the expectation of $\pi(X, Y, T)$:

$$\Pi = \underbrace{[365 \cdot 100 \cdot 100]}_{\text{Volume } V} \cdot \int_0^{365} \int_0^{100} \int_0^{100} \pi(x, y, t) \cdot f_X(x) \cdot f_Y(y) \cdot f_T(t)\, dx\, dy\, dt. \tag{31}$$

Secondly, generate n samples $\theta_i := (X_i, Y_i, T_i)$, $i \in \{1, ..., n\}$. Consequently, Π can be approximated for sufficiently large n:

$$\Pi \approx \frac{V}{N} \sum_{i=1}^{N} \pi(\theta_i). \tag{32}$$

```
rng('default')   % Control random number generator

tic
n = 1000;   % Sample size
Volume_V = 365*100*100;

% Firstly, generate random samples:
X_Sample = round(100*rand(n,1));
Y_Sample = round(100*rand(n,1));
T_Sample = round(365*rand(n,1));

% Secondly, calulate profit pi for all samples:
Seasonal_Effect = (1/3)*cos((2*pi/365).*T_Sample - pi/6) + 1;
Growth = exp(T_Sample./1000);
Q_Function = (80 - 0.05.*X_Sample.*Seasonal_Effect - 0.08.* ...
     Y_Sample).*Growth;
Inflation = (1 + 0.02/365).^T_Sample;
P_Function = 5.*Inflation - (1/200).*X_Sample - (1/300)*Y_Sample;
C_Function = (2 + 0.015*X_Sample + 0.01*Y_Sample).*(1 + T_Sample./1000);

```

```
Pi_Function = (P_Function - C_Function).*Q_Function;

% Finally, calulate the total profit (big pi):
Pi = Volume_V*mean(Pi_Function);
toc
```

`Pi` produces a profit of approximately €`2.534e+08` for the energy supplier. ▶

3 Importance Sampling

In Theorem 2.2, we showed that the standard deviation of a Monte Carlo integration estimator decreases at the rate of $\mathcal{O}(n^{-\frac{1}{2}})$. Sometimes $\mathcal{O}(\cdot)$ denotes the rate at which the error of an estimator decreases, but since the Monte Carlo integration estimator is unbiased, the two definitions of $\mathcal{O}(\cdot)$ are equivalent because its variance and MSE are equal. However, a more efficient estimator can be obtain since $\mathcal{O}(n^{-\frac{1}{2}})$ is not the best rate that can be achieved for a Monte Carlo integration estimator. Variance reduction techniques refer to a collection of procedures that can be used to obtain a higher precision of the estimates for a given simulation without increasing the sample size n.

Importance sampling (IS) is one of the most widely used variance reduction techniques. The introduction of this concept in statistics and econometrics is commonly credited to Kloek and van Dijk (1978). Importance sampling involves choosing a proposal distribution which favours important samples, i.e. which focusses on regions that are important for an integral. Regions of importance for an integral refer to regions with relatively high probability density. Applying this technique often leads to dramatic variance reductions (Rubinstein, 2016).

Importance sampling can be used to reduce the variance in Monte Carlo integration. The aim is to choose a proposal PDF, p_X, such that the variance of the new estimator that favours important samples is smaller than the variance of the standard Monte Carlo integration estimator $\bar{g}_n$:

$$\frac{1}{n}\text{Var}_{p_X}\left[g(X)\frac{f_X(X)}{p_X(X)}\right] < \frac{1}{n}\text{Var}_{f_X}\left[g(X)\right] = \text{Var}_{f_X}\left[\bar{g}_n\right]. \tag{33}$$

3.1 Theory

The main objective is to reduce the variance when approximating the integral in equation (1) by using the Monte Carlo integration technique:

$$\int_A \phi(x)\,dx = \int_A g(x)\cdot f_X(x)\,dx = \mathbb{E}_{f_X}[g(X)] =: \theta_{MC}. \tag{34}$$

Let us call f_X the target PDF. The main idea of importance sampling is as follows:

$$\begin{aligned}
\int_A \phi(x)\,dx &= \int_{\mathbb{R}} g(x)\cdot f_X(x)\,dx \\
&= \int_{\mathbb{R}} \frac{g(x)\cdot f_X(x)}{p_X(x)}\cdot p_X(x)\,dx \\
&= \mathbb{E}_{p_X}\left[\frac{g(X)\cdot f_X(X)}{p_X(X)}\right] \\
&=: \mathbb{E}_{p_X}\left[g(X)\cdot W(X)\right] =: \theta_{IS} \\
&= \mathbb{E}_{f_X}\left[g(X)\right],
\end{aligned} \tag{35}$$

where $W(X) := \frac{f_X(X)}{p_X(X)}$ is called the likelihood ratio. In stochastic calculus and measure theory, this random variable is better known as the Radon-Nikodym derivative. Additionally, observe that θ_{MC} and θ_{IS} are different representations of the integral in equation (1), i.e. $\theta_{MC} = \theta_{IS}$.

Hence, importance sampling rewrites the integral into the expectation of another random variable with a different PDF. Essentially, it changes the weights of the weighted average. Moreover, it could make the sample generation process easier. If f_X is a challenging PDF to draw from, then we could choose a proposal PDF p_X from which it is easier to generate samples. Another solution to this problem, called Markov chain Monte Carlo, is explained in

the section 4. Therefore, importance sampling yields two advanatages: a more efficient estimator of the integral in equation (1), and a procedure that enables us to perform Monte Carlo integration when it is difficult to generate samples from f_X.

An important assumption that must be made to ensure inclusion of the entire image of the integrand $\phi(A)$ is the following:

$$Supp(g \cdot f_X) \subseteq Supp(p_X). \tag{36}$$

This ensures that the support of the proposal PDF p_X covers that of $g \cdot f_X$. Otherwise, some parts of the integration domain might not be utilized which truncates the integral, leading to a biased result.

It turns out that drawing from $f_X(x)$ is equivalent to first drawing from $p_X(x)$ and then applying the transformation $\frac{f_X(y)}{p_X(y)}$ (Train, 2009).

Proposition 3.1.

$$\left\{ \textit{The set of weighted draws } \frac{f_X(X)}{p_X(X)} : X \sim p_X \right\} = \{ \textit{The set of draws from } f_X \} \tag{37}$$

Proof: Let F_X and P_X be the cumulative distribution functions (CDFs) of a random variable X. Proving that the CDFs F_X and P_X are equivalent is sufficient.

Pick some $\delta \in Supp(p_X)$ such that $|\delta| < \infty$. Rewriting the weighted CDF, i.e. the expectation of the likelihood ratio on $[-\infty, \delta]$, yields:

$$\begin{aligned} \int_{\mathbb{R}} \frac{f_X(x)}{p_X(x)} \cdot \mathbb{I}\{x \leq \delta\} \cdot p_X(x)\, dx &= \int_{-\infty}^{\delta} \frac{f_X(x)}{p_X(x)} p_X(x)\, dx \\ &= \int_{-\infty}^{\delta} f_X(x)\, dx = F_X(\delta). \end{aligned} \tag{38}$$

■

The importance sampling fundamental identity (Robert and Casella, 2010) gives an unbiased importance sampling (or likelihood ratio) estimator for any proposal PDF p_X that satisfies equation (36):

Theorem 3.1. *Let $X \sim p_X$ and let $X_1, ..., X_n \sim p_X$ be an i.i.d. random sample with realizations $x_1, ..., x_n$ such that $\mathbb{E}_{p_X}[|X_i|] < \infty$, $\forall i \in \{1, ..., n\}$. Then, an unbiased estimator of θ_{IS} is:*

$$\hat{\theta}_{IS} := \frac{1}{n} \sum_{i=1}^{n} g(x_i) \cdot \frac{f_X(x_i)}{p_X(x_i)} \xrightarrow{a.s.} \mathbb{E}_{p_X} \left[\frac{g(X) \cdot f_X(X)}{p_X(X)} \right] = \theta_{IS}. \tag{39}$$

Proof: $\hat{\theta}_{IS}$ is an unbiased estimator of θ_{IS} because:

$$\begin{aligned} \mathbb{E}_{p_X}\left[\hat{\theta}_{IS}\right] &= \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{p_X}\left[g(X_i) \cdot \frac{f_X(X_i)}{p_X(X_i)} \right] \qquad \text{[Linearity of expectation]} \\ &= \frac{1}{n} \sum_{i=1}^{n} \int_{\mathbb{R}} g(x_i) \cdot \frac{f_X(x_i)}{p_X(x_i)} \cdot p_X(x_i)\, dx \\ &= \frac{1}{n} \sum_{i=1}^{n} \int_{\mathbb{R}} g(x_i) \cdot f_X(x_i)\, dx \\ &= \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{f_X}\left[g(X_i)\right] \overset{i.d.}{=} \frac{1}{n} \cdot n \cdot \mathbb{E}_{f_X}\left[g(X)\right] = \mathbb{E}_{f_X}\left[g(X)\right] = \theta_{IS}. \end{aligned} \tag{40}$$

Furthermore, the almost surely convergence in equation (39) holds by the SLLN. ■

Observe that the Monte Carlo integration estimator in equation (4) is a special case of the importance sampling estimator in equation (39), if $p_X \overset{d}{=} f_X$ and $W(X) := \frac{f_X(X)}{p_X(X)} \equiv 1$.

Moreover, observe that the variance of the Monte Carlo integration estimator can be rewritten in terms of θ_{IS}:

$$\begin{aligned} \text{Var}[\bar{g}_n] = \frac{1}{n}\text{Var}_{f_X}\left[g(X)\right] &= \frac{1}{n} \cdot \left[\mathbb{E}_{f_X}\left[g(X)^2\right] - \mathbb{E}_{f_X}\left[g(X)\right]^2\right] \\ &= \frac{1}{n} \cdot \left[\int_{\mathbb{R}} g(x)^2 \cdot f_X(x)\, dx - \theta_{IS}^2\right], \end{aligned} \tag{41}$$

where $X \sim f_X(x)$. The variance of the importance sampling estimator can be written in a similar fashion.

Proposition 3.2.

$$\text{Var}_{p_X}\left[\hat{\theta}_{IS}\right] = \frac{1}{n}\text{Var}_{p_X}\left[g(X) \cdot W(X)\right] = \frac{1}{n} \cdot \left[\int_{\mathbb{R}} g(x)^2 \cdot W(x)^2 \cdot p_X(x)\, dx - \theta_{IS}^2\right]. \tag{42}$$

The unbiased estimator of the variance of $\hat{\theta}_{IS}$ is:

$$s_{IS}^2 := \widehat{\text{Var}_{p_X}\left[\hat{\theta}_{IS}\right]} = \frac{1}{n-1}\sum_{i=1}^{n}\left[g(x_i) \cdot W(x_i) - \hat{\theta}_{IS}\right]^2. \tag{43}$$

Proof: Equation (42) follows by the definition of variance. Next, let $\sigma^2 := \text{Var}_{p_X}\left[g(X_i) \cdot W(X_i)\right]$. The sample variance s_{IS}^2 is unbiased because:

$$\begin{aligned} \mathbb{E}_{p_X}\left[s_{IS}^2\right] &= \frac{1}{n-1} \cdot \mathbb{E}_{p_X}\left[\sum_{i=1}^{n}\left(g(X_i) \cdot W(X_i) - \hat{\theta}_{IS}\right)^2\right] \\ &= \frac{1}{n-1} \cdot \mathbb{E}_{p_X}\left[\sum_{i=1}^{n} g^2(X_i) \cdot W^2(X_i) - 2\hat{\theta}_{IS} \cdot \sum_{i=1}^{n} g(X_i) \cdot W(X_i) + \sum_{i=1}^{n}\hat{\theta}_{IS}^2\right] \\ &= \frac{1}{n-1} \cdot \mathbb{E}_{p_X}\left[\sum_{i=1}^{n} g^2(X_i) \cdot W^2(X_i) - n \cdot \hat{\theta}_{IS}^2\right] \\ &= \frac{1}{n-1} \cdot \sum_{i=1}^{n}\mathbb{E}_{p_X}\left[g^2(X_i) \cdot W^2(X_i)\right] - \frac{n}{n-1} \cdot \mathbb{E}_{p_X}\left[\hat{\theta}_{IS}^2\right] \\ &= \frac{1}{n-1}\sum_{i=1}^{n}\left(\sigma^2 + \mathbb{E}_{p_X}\left[g(X_i) \cdot W(X_i)\right]^2 \cdot\right) - \frac{n}{n-1} \cdot \left(\text{Var}_{p_X}\left[\hat{\theta}_{IS}\right] + \mathbb{E}_{p_X}\left[\hat{\theta}_{IS}\right]^2\right) \\ &= \frac{1}{n-1} \cdot \left(n \cdot \sigma^2 + n \cdot \mathbb{E}_{p_X}\left[g(X_i) \cdot W(X_i)\right] - n \cdot \frac{n}{n^2} \cdot \sigma^2 - n \cdot \mathbb{E}_{p_X}\left[g(X_i) \cdot W(X_i)\right]\right) \\ &= \frac{1}{n-1} \cdot \left((n-1) \cdot \sigma^2\right) = \sigma^2 := \text{Var}_{p_X}\left[\hat{\theta}_{IS}\right]. \end{aligned} \tag{44}$$

■

Observe that the mean squared error of $\hat{\theta}_{IS}$ is equal to its variance, because $\hat{\theta}_{IS}$ has no bias, i.e. it is an unbiased estimator, by Theorem 3.1.

To restate the goal of this section, we want to obtain a smaller variance for the Monte Carlo integration estimator. This is achieved when the variance of the importance sampling

estimator is smaller than the variance of the Monte Carlo estimator. Or equivalently, when their difference is positive:

$$0 < \mathrm{Var}_{f_X}\left[g(X)\right] - \mathrm{Var}_{p_X}\left[g(X)\cdot W(X)\right] = \int_{\mathbb{R}} g^2(x)\left[1 - \frac{f_X(x)}{p_X(x)}\right]\cdot f_X(x)\,dx. \tag{45}$$

Since $g, f_X \geq 0$, we should select a proposal PDF p_X such that:

$$\begin{cases} p_X(x) > f_X, & \text{if } g^2(x)f_X(x) \text{ is large;} \\ p_X(x) < f_X, & \text{if } g^2(x)f_X(x) \text{ is small.} \end{cases} \tag{46}$$

Hence, we want p_X to take on large (small) values where the integrand ϕ attains large (small) values. In other words, importance sampling involves choosing a proposal PDF which favours 'important values'. These important values are characterized by their relative size (dwarf or giant) in the image of ϕ, i.e. $\phi(A)$. This motivates the term 'importance sampling'.

In particular, the goal of this section can be stated more formally as minimizing the variance of $\hat{\theta}_{IS}$ with respect to p_X (Rubinstein, 2016):

$$\min_{p_X} \mathrm{Var}_{p_X}\left[g(X)\cdot W(X)\right]. \tag{47}$$

Fortunately, there exist a PDF p_X that minimizes the variance such that the minimum value of zero is attained.

Theorem 3.2. *The PDF that is the minimizer of optimization problem (47) is:*

$$p_X^*(x) = \frac{|g(x)|\cdot f_X(x)}{\int_{\mathbb{R}} |g(x)|\cdot f_X(x)\,dx}. \tag{48}$$

This minimizer PDF p_X^* is often called the optimal importance sampling PDF.

Proof: See Robert and Casella (2004). ■

Working with absolute values is often inconvenient, but the optimal importance sampling PDF in equation (48) simplifies when $g(x) \geq 0$, $\forall x \in \mathbb{R}$.

Corollary 3.1. *If $g \geq 0$, then:*

$$p_X^*(x) = \frac{g(x)\cdot f_X(x)}{\int_{\mathbb{R}} g(x)\cdot W(x)\cdot p_X(x)\,dx} = \frac{g(x)\cdot f_X(x)}{\theta_{IS}}. \tag{49}$$

For this simplified optimal importance sampling PDF, the variance of $\hat{\theta}_{IS}$ with respect to p_X^* equals zero.

Theorem 3.3.

$$\mathrm{Var}_{p_X^*}\left[\hat{\theta}_{IS}\right] \overset{(1)}{=} \mathrm{Var}_{p_X^*}\left[g(X)\cdot W^*(X)\right] \overset{(2)}{=} \mathrm{Var}_{p_X^*}\left[\theta_{IS}\right] \overset{(3)}{=} 0, \tag{50}$$

with likelihood ratio $W^*(x) := \frac{f_X(x)}{p_X^*(x)}$.

Proof:

$$
\begin{aligned}
\mathrm{Var}_{p_X^*}\left[\hat{\theta}_{IS}\right] &= \mathrm{Var}_{p_X^*}\left[\frac{1}{n}\sum_{i=1}^{n} g(X_i)\cdot W^*(X_i)\right] \\
&\overset{ind.}{=} \frac{1}{n}\sum_{i=1}^{n}\mathrm{Var}_{p_X^*}\left[g(X_i)\cdot W^*(X_i)\right] \\
&\overset{i.d.}{=} \frac{1}{n}\cdot n\cdot \mathrm{Var}_{p_X^*}\left[g(X)\cdot W^*(X)\right] \overset{(1)}{=} \mathrm{Var}_{p_X^*}\left[g(X)\cdot W^*(X)\right] \\
&= \mathrm{Var}_{p_X^*}\left[g(X)\cdot \frac{f_X(X)}{p_X^*(X)}\right] \\
&= \mathrm{Var}_{p_X^*}\left[g(X)\cdot f_X(X)\cdot \frac{\theta}{g(X)\cdot f_X(X)}\right] \\
&\overset{(2)}{=} \mathrm{Var}_{p_X^*}\left[\theta\right] = \mathrm{Var}_{p_X^*}\left[\mathbb{E}_{f_X}\left[g(X)\right]\right] \overset{(3)}{=} 0.
\end{aligned}
\tag{51}
$$

■

However, in general, the implementation of the optimal importance sampling PDF in Corollary (3.1) is problematic. The difficulty lies in the fact that we need to know θ_{IS} in order to determine p_X^*, but this is precisely the quantity we are trying to estimate. Brandimarte (2014) suggests that we should choose a proposal PDF p_X which approximates the product $g(x)\cdot f_X(x)$ as close as possible. Thus, the real challenge is selecting a proposal PDF p_X that yields a significant reduction in the variance of the estimate. There is no guarantee that the variance will be reduced, because a 'bad' choice for p_X could result in an increase in variance. Hence, caution must be taken when choosing a proposal PDF p_X.

An importance sampling estimator with infinite variance should be avoided. $\mathrm{Var}_{p_X}\left[\hat{\theta}_{IS}\right] < \infty$ if $\mathrm{Var}_{p_X}\left[g(X)\cdot W(X)\right] = \mathbb{E}_{p_X}\left[g^2(X)\cdot W^2(X)\right] - \theta_{IS}^2 < \infty$. We know that $\theta_{IS}^2 < \infty$ since we assumed that f is integrable, i.e. $\int_A |f(x)|\,dx < \infty$. Hence, it remains to show that:

$$
\mathbb{E}_{p_X}\left[g^2(X)\cdot W^2(X)\right] = \int_{\mathbb{R}} g^2(x)\cdot \frac{f_X(x)}{p_X(x)}\cdot p_X(x)\,dx < \infty. \tag{52}
$$

Since $\int_A |f(x)|\,dx < \infty$, we know that $\int_{\mathbb{R}} g(x)\cdot f_X(x)\,dx$ is bounded. Therefore, we want that the likelihood ratio $\frac{f_X(x)}{p_X(x)}$ is bounded for all $x\in\mathbb{R}$. Suppose the likelihood ratio is not bounded, i.e. there exists an $x_0 \in \mathrm{int}\left\{Supp(f_X)\cap Supp(p_X)\right\}$ such that $\lim_{x\to x_0}\left|\frac{f_X(x)}{p_X(x)}\right| = \infty$. This is the case if $\lim_{x\to x_0} p_X(x) = 0$ and $\lim_{x\to x_0} f_X(x) \neq 0$ which implies that p_X has 'lighter tails' than f_X. Hence, we want p_X to have 'heavier tails' than f_X. Figure 5 shows that the standard Cauchy PDF has heavier tails than the standard normal PDF. Therefore, the standard normal is not a good proposal PDF if we have a standard Cauchy target PDF. More generally, if the likelihood ratio is unbounded, the weights $\frac{f_X(x_i)}{p_X(x_i)}$ vary widely giving too much importance to a few values x_i.

Geweke (1989) proposes a sufficient condition for finite variance of $\hat{\theta}_{IS}$:

$$
\frac{f_X(x)}{p_X(x)} < M\in\mathbb{R},\ \forall x\in\mathbb{R} \qquad \& \qquad \mathrm{Var}_{f_X}\left[g(X)\right] < \infty. \tag{53}
$$

It is important to note that although the finite variance constraint in equation (52) is not necessary for the convergence of $\hat{\theta}_{IS}$, importance sampling performs poorly when (Robert and Casella, 2004):

$$
\mathbb{E}_{f_X}\left[\frac{f_X(x)}{p_X(x)}\right] = \int_{\mathbb{R}} \frac{f_X(x)}{p_X(x)}\cdot f_X(x)\,dx = +\infty. \tag{54}
$$

Poor performance refers to both the estimator's behaviour and the comparison with Monte Carlo integration estimator. Equation (54) gives us a test that detects possible poor performance. However, equation (53) prevents the scenario described in equation (54) from occurring.

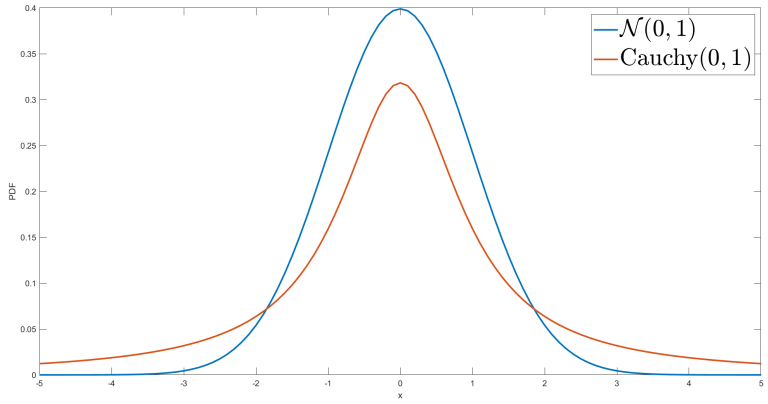

Figure 5

An overview of the conditions for choosing a proposal PDF p_X is given by:

IS.1.	$Supp(g \cdot f_X) \subseteq Supp(p_X)$
IS.2.	p_X is easy to draw from
IS.3.	p_X matches the product $g(x) \cdot f_X(x)$ as much as possible
IS.4.	$\frac{f_X(x)}{p_X(x)} < M \in \mathbb{R},\ \forall x \in \mathbb{R} \quad \& \quad \mathrm{Var}_{f_X}[g(X)] < \infty$
IS.5.	p_X has heavier tails than f_X

Moreover, the algorithm for calculating the importance sampling estimator is:

Algorithm: Pseudo-Code for **Importance Sampling**

Input: Sample size n
for $i = 1, ..., n$
    Sample $X_i \sim p_X$
    Compute $g(X_i)$
    Compute likelihood ratio $W(X_i) = \frac{f_X(X_i)}{p_X(X_i)}$
end
Compute the estimator $\hat{\theta}_{IS} = \frac{1}{n}\sum_{i=1}^{n} g(X_i) \cdot W(X_i)$

However, there are some difficulties. Condition IS.5. requires prior knowledge of the shape of f_X. And condition IS.4. is quite restrictive. Fortunately, there exists an alternative to the importance sampling estimator $\hat{\theta}_{IS}$ in Theorem 3.1. It addresses the finite variance issue and it generally results in a more stable estimtor. This alternative estimator is called the auto-normalizing or weighted importance sampling estimator:

$$\hat{\theta}_{IS,w} := \frac{\sum_{i=1}^{n} g(x_i) \cdot w(x_i)}{\sum_{i=1}^{n} w(x_i)} \quad \text{with} \quad w(x_i) := \frac{\tilde{f}_X(x_i)}{\tilde{p}_X(x_i)} = \frac{\frac{1}{c_f} f_X(x_i)}{\frac{1}{c_p} p_X(x_i)}. \tag{55}$$

The new likelihood ratios $w(x_i)$ are interpreted as weights of the random sample $\{X_i\}_{i=1}^n$. In contrast to Monte Carlo integration, where all samples contribute equally, samples here are assigned different weights. In contrast to the sum of likelihood ratios $\sum_{i=1}^n w(x_i)$ that is probably not equal to one, the new weights $\frac{w(x_i)}{\sum_{i=1}^n w(x_i)}$ must sum to one.

This estimator can also be used when f_X and p_X are only known up to some constants $c_f, c_p \in \mathbb{R}$, respectively. This can be a powerful estimator when dealing with a challenging integrand ϕ or when applying Bayesian statistics.

Auto-normalizing importance sampling produces an estimator of θ_{IS} with finite variance but a slight bias. However, this bias converges to zero as the sample size n increases.

Theorem 3.4. *Let $X \sim p_X(x)$ and let $X_1, ..., X_n \sim p_X(x)$ be an i.i.d. random sample with $\mathbb{E}_{p_X}[|X_i|] < \infty$, $\forall i \in \{1, ..., n\}$. Then, the auto-normalizing importance sampling estimator of θ_{IS} is:*

$$\hat{\theta}_{IS,w} := \frac{\sum_{i=1}^n g(x_i) \cdot w(x_i)}{\sum_{i=1}^n w(x_i)} \xrightarrow{a.s.} \frac{\mathbb{E}_{p_X}[g(X) \cdot w(X)]}{\mathbb{E}_{p_X}[w(X)]} = \theta_{IS} \tag{56}$$

where x_i denotes the realization of X_i.

Proof: By the SLLN:

$$\begin{aligned}
\mathbb{E}_{p_X}\left[\hat{\theta}_{IS,w}\right] = \frac{\frac{1}{n}\sum_{i=1}^n g(x_i) \cdot w(x_i)}{\frac{1}{n}\sum_{i=1}^n w(x_i)} &\xrightarrow{a.s.} \frac{\mathbb{E}_{p_X}[g(X) \cdot w(X)]}{\mathbb{E}_{p_X}[w(X)]} \\
&= \frac{\int_{\mathbb{R}} g(x) \cdot w(x) \cdot p_X(x)\, dx}{\int_{\mathbb{R}} w(x) \cdot p_X(x)\, dx} \\
&= \frac{\frac{c_f}{c_p} \cdot \int_{\mathbb{R}} g(x) \cdot \frac{\tilde{f}_X(x)}{\tilde{p}_X(x)} \cdot p_X(x)\, dx}{\frac{c_f}{c_p} \cdot \int_{\mathbb{R}} \frac{\tilde{f}_X(x)}{\tilde{p}_X(x)} \cdot p_X(x)\, dx} \\
&= \frac{\int_{\mathbb{R}} g(x) \cdot \frac{f_X(x)}{p_X(x)} \cdot p_X(x)\, dx}{\int_{\mathbb{R}} f_X(x)\, dx} = \theta_{IS}.
\end{aligned} \tag{57}$$

■

Theorem 3.5. *$\hat{\theta}_{IS,w}$ is a biased estimator of θ_{IS}, but the bias disappears as $n \to \infty$*

Proof: Liu (2001) provides a reasonable approximation of $\hat{\theta}_{IS,w}$'s expectation:

$$\mathbb{E}_{p_X}\left[\hat{\theta}_{IS,w}\right] \approx \theta_{IS} \underbrace{-\frac{1}{n}\text{Cov}_{p_X}[w(X), g(X) \cdot w(X)] + \frac{1}{n}\theta_{IS}\text{Var}_{p_X}[w(X)]}_{\approx \text{Bias}\left(\hat{\theta}_{IS,w}\right) \xrightarrow{n\to\infty} 0}. \tag{58}$$

■

Agapiou, Papaspiliopoulos, Sanz-Alonso, and Stuart (2017) show that the bias and MSE of $\hat{\theta}_{IS,w}$ decreases with rate $\mathcal{O}(n^{-1})$ which is faster than the rate, $\mathcal{O}(n^{-\frac{1}{2}})$, at which the standard deviation of the Monte Carlo integration estimator decreases.

A notion of sample variance is needed to be able to calculate the variance of $\hat{\theta}_{IS,w}$ in practice.

Theorem 3.6. *The sample variance of $\hat{\theta}_{IS,w}$ is:*

$$
\begin{aligned}
s^2_{IS,w} := \widehat{\mathrm{Var}_{p_X}\left[\hat{\theta}_{IS,w}\right]} &= \frac{n}{n-1} \cdot \frac{\sum_{i=1}^{n} w^2(x_i) \cdot \left[g(x_i) - \hat{\theta}_{IS,w}\right]^2}{\left[\sum_{i=1}^{n} w(x_i)\right]^2} \\
&= \frac{n}{n-1} \cdot \sum_{i=1}^{n} w_i^2 \cdot \left[g(x_i) - \hat{\theta}_{IS,w}\right]^2,
\end{aligned} \tag{59}
$$

where $w_i := \frac{w(x_i)}{\sum_{i=1}^{n} w(x_i)}$.

Proof: Cochran (1977) provides a technique to determine the sample variance of a ratio of a weighted arithmetic mean random variable that is directly applied. ■

w_i can be interpreted as probabilities. Observe the similarities with the sample variance of the regular importance sampling estimator in equation (43).

However, this does not encompass the entirety of importance sampling. More advanced techniques include sequential importance sampling, exponential tilting, sampling importance resampling. Furthermore, concepts in this section can be generalized to random vectors, analogous to the approach in subsection 2.3. Nevertheless, we will only apply importance sampling to standard integration problems.

3.2 Applications

Example 3.1: Tail Probability
Let Z be a standard normally distributed random variable: $Z \sim \mathcal{N}(0,1)$. Suppose we want to calculate the probability Z is greater than π which is an integral:

$$
\mathbb{P}(Z \geq \pi) = \int_{\pi}^{\infty} f_Z(z)\, dz = \int_{\pi}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}\, dz = \int_{-\infty}^{\infty} \mathbb{I}\{z \geq \pi\} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}\, dz =: \theta. \tag{60}
$$

```
%% True Integral Value
mu = 0; sigma = 1;
% We take the following value as the true value for the integral
True_Integral = 1 - cdf('Normal',pi,mu,sigma);  % Tail probability
```

In the code above, `True_Integral` gives the value of `8.4016e-04` for the integral in equation (60). Furthermore, Figure 6 depicts the region of interest.

```
%% Plotting the region of interest
x_Values = [-4:0.1:3.1, pi, 3.2:0.1:4];  % Sequence from -5 to 5 by 0.1
% Theoretical Standard Normal PDF
PD_Normal = makedist('Normal','mu',0,'sigma',1);
PDF_Normal = pdf(PD_Normal,x_Values);

plot(x_Values,PDF_Normal,'LineWidth',2,'Color','black')
hold on;
% Colouring the region under the curve after pi
AfterPi = x_Values >= pi;
patch([x_Values(AfterPi) flip(x_Values(AfterPi))], ...
     [zeros(size(PDF_Normal(AfterPi))) flip(PDF_Normal(AfterPi))], ...
     'magenta','EdgeColor','none')
xline(pi,'LineWidth',2,'Color','cyan')
text(3.2,0.15,'\pi','FontSize',27)
title('Region $Z \geq \pi$','fontsize',34,'Interpreter','latex')
xlabel('z','fontsize',21,'Interpreter','latex')
```

```
ylabel('PDF','fontsize',21,'Interpreter','latex')
legend({'$\mathcal{N}(0,1)$'},'Interpreter','latex','Location', ...
     'northeast','FontSize', 30);
hold off;

%% Zoom in on the coloured region in the tail
plot(x_Values,PDF_Normal,'LineWidth',2,'Color','black')
hold on;
% Coloring the region under the curve after pi
AfterPi = x_Values >= pi;
patch([x_Values(AfterPi) flip(x_Values(AfterPi))], ...
     [zeros(size(PDF_Normal(AfterPi))) flip(PDF_Normal(AfterPi))], ...
     'magenta','EdgeColor','none')
xline(pi,'LineWidth',2,'Color','cyan')
xlim([3 4])
text(3.15,0.004,'\pi','FontSize',27)
title('Zoomed-In Region $Z \geq \pi$', ...
     'fontsize',34,'fontweight','bold','Interpreter','latex')
xlabel('z','fontsize',21,'Interpreter','latex')
ylabel('PDF','fontsize',21,'Interpreter','latex')
legend({'$\mathcal{N}(0,1)$'},'Interpreter','latex','Location', ...
     'northeast','fontsize', 30);
hold off;
```

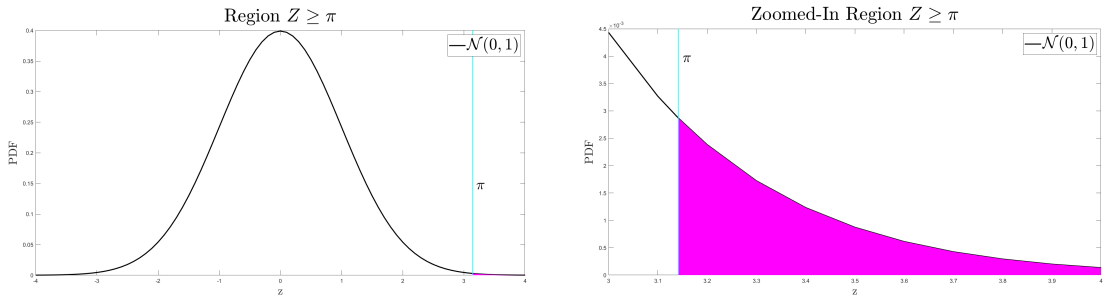

Figure 6

The integral in equation (60) can be estimated by Monte Carlo integration:

$$\mathbb{P}(Z \geq \pi) = \mathbb{E}_{f_Z}\left[\mathbb{I}\{Z \geq \pi\}\right] \approx \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\{Z_i \geq \pi\}, \tag{61}$$

where $Z_i \sim \mathcal{N}(0,1)$ for $i \in \{1, ..., n\}$. Since $\mathbb{I}\{Z_i \geq \pi\}$ is a Bernoulli random variable with very low success probability $\mathbb{P}(Z \geq \pi)$, the variance of $\mathbb{I}\{Z_i \geq \pi\}$ is approximately $\mathbb{P}(Z \geq \pi)\left[1 - \mathbb{P}(Z \geq \pi)\right] \approx \mathbb{P}(Z \geq \pi)$. But it turns out that there exist estimators of $\mathbb{P}(Z \geq \pi)$ that are more efficient, such estimators can be obtained by applying the importance sampling technique. We implement importance sampling with four distinct proposal PDFs. However, let us first approximate the integral in equation (60) as reference estimator.

```
%% Monte Carlo Integration
tic
n = 10000;    % Sample size
```

```
% Generate n realizations from N(0,1)
StandardNormal_Realization = random('Normal',mu,sigma,[n 1]);
% MC integral estimate: mean of indicator functions
MC_Integral_Estimate = mean(StandardNormal_Realization>=pi);
toc

% Variance of MC estimator (= MSE, by unbiasedness)
MSE_MCIntegration = 1/(n*(n-1))*sum( ...
     (StandardNormal_Realization>=pi - MC_Integral_Estimate).^2);
```

`MC_Integral_Estimate` is sensitive to sample size. In other words, when `rng( 'default' )` is not used, the value of `MC_Integral_Estimate` change a lot per simulation with `n = 10000`. Therefore, we will plot `MC_Integral_Estimate` and several importance sampling estimates across a sequence of sample sizes. This enables comparison and visualization of convergence of the estimators of interest.

First, by using condition IS.3., we observe that the exponential PDF with parameter 1 roughly matches the exponential term in $\mathbb{I}\{z \geq \pi\} \cdot \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}$. Therefore, we select the Expo(1) PDF as a proposal PDF. Let $Y_1 \sim \text{Expo}(1)$ with PDF $p_{Y_1}(z) = \mathbb{I}\{z > 0\} \cdot e^{-z}$. Rewriting the integral:

$$\begin{aligned}
\mathbb{P}(Z \geq \pi) = \int_\pi^\infty \frac{f_Z(z)}{p_{Y_1}(z)} p_{Y_1}(z)\, dz &= \frac{1}{\sqrt{2\pi}} \int_\pi^\infty \frac{e^{-\frac{1}{2}z^2}}{e^{-z}} \cdot \mathbb{I}\{z > 0\} \cdot e^{-z}\, dz \\
&= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^\infty \frac{e^{-\frac{1}{2}z^2}}{e^{-z}} \cdot \mathbb{I}\{z \geq \pi\} \cdot e^{-z}\, dz \\
&= \frac{1}{\sqrt{2\pi}} \cdot \mathbb{E}_{p_{Y_1}} \left[\frac{e^{-\frac{1}{2}Z^2}}{e^{-Z}} \cdot \mathbb{I}\{Z \geq \pi\} \right] \\
&= \mathbb{E}_{p_{Y_1}} \left[\frac{f_Z(Z)}{p_{Y_1}(Z)} \right].
\end{aligned} \tag{62}$$

Hence, an importance sampling estimator is $\hat{\theta}_{IS}^{\text{Expo}(1)} = \frac{1}{n} \sum_{i=1}^n \frac{f_Z(Z_i)}{p_{Y_1}(Z_i)}$ $= \frac{1}{n} \frac{1}{\sqrt{2\pi}} \sum_{i=1}^n e^{-\frac{1}{2}Z_i^2 + Z_i} \cdot \mathbb{I}\{Z_i \geq \pi\}$, where $Z_i \sim \text{Expo}(1)$.

```
%% Importance Sampling: Exponential Proposal PDF
tic
% Generate n realizations from Expo(1)
Exponential_Realization = random('Exponential',1,[n 1]);
% Indicator function: Z >= pi
Select_Exponential = Exponential_Realization( ...
     Exponential_Realization>=pi,1);
% IS integral estimate: mean of exponential quadratic
Transform_Exponential = exp(-0.5*Select_Exponential.^2)./exp( ...
     -Select_Exponential);
IS_Exponential = (1/sqrt(2*pi))*(sum(Transform_Exponential)/n);
toc

% Variance of IS estimator (= MSE, by unbiasedness)
MSE_Exponential = 1/(n-1)*sum( ...
     ((1/sqrt(2*pi))*Transform_Exponential - IS_Exponential).^2 );
```

Second, the above proposal density can be improved by truncating the PDF so that its support does not include values lower than π. This implies another proposal PDF:

$\mathcal{T}\text{Expo}(1,\pi)$, i.e. the Expo(1) PDF truncated by π. Let $Y_2 \sim \mathcal{T}\text{Expo}(1,\pi)$ with PDF $p_{Y_2}(z) = \left[\int_\pi^\infty e^{-z}\,dz\right]^{-1} \cdot \mathbb{I}\{\geq \pi\} \cdot e^{-z} = \mathbb{I}\{\geq \pi\} \cdot e^{-z+\pi}$, where $\int_\pi^\infty e^{-z}\,dz$ is the normalization constant. Unfortunately, the `random` function in `MATLAB` does not enable us to directly draw from this PDF. The inverse transformation method can be used to draw from p_{Y_2}. Applying this technique results in equivalent drawing from $P_X^{-1}(U) = \pi - \log(1-U)$, where $U \sim \text{Unif}(0,1)$.

Let us briefly explore the workings of the inverse transformation method. The main aim of this method is drawing from a CDF that is not straightforward to draw from. Packages for random number generation in programming languages only cover the most simple and widely-used PDFs and CDFs, but there exist uncountably many of them. Under some restrictive conditions, the inverse transformation method is the most natural technique that enables us to draw from a non-standard CDF. First, it requires a normalized distribution. Second, it requires knowledge about the exact expression for the CDF which is a problem when dealing with data for example. Third, it needs the analytical inverse of the CDF of interest. This is often problematic because CDFs are not always injective. For example, if a CDF evaluated at x equals one, then it is also equal to one at $x+\epsilon$ for any $\epsilon > 0$ since every CDF is strictly non-decreasing. However, we can relax the last condition by introducing the notion of the generalized inverse of the CDF F of a random variable X (Rubinstein, 2016):

$$F^{-1}(y) := \inf\{x : F(x) \geq y\}, \quad y \in [0,1]. \tag{63}$$

The generalized inverse of F is a subset of the pre-image of F such that F restricted to $F^{-1}(y)$ becomes bijective and hence invertible. And if F is bijective, then the generalized inverse is equal to the ordinary inverse. Now, we have all the ingredients to state the inverse transformation method.

Theorem 3.7. *To generate a random variable X with (normalized) CDF F, draw $U \sim$ Unif$(0,1)$ and calculate $X = F^{-1}(U)$.*

Proof: The proof is elegant since it only uses the generalized inverse of F and the observation that $\mathbb{P}(U \leq u) = \frac{u-0}{1-0} = u$.

$$\mathbb{P}(X \leq x) = \mathbb{P}(F^{-1}(U) \leq x) = \mathbb{P}(U \leq F(x)) = F(x) \quad \Longrightarrow \quad X = F^{-1}(U). \tag{64}$$

■

Let us apply the inverse transformation method to sample from $\mathcal{T}\text{Expo}(1,\pi)$. It can be shown that $P_{Y_2}(x) = \mathbb{P}(Y_2 \leq x) = \int_{-\infty}^x p_{Y_2}(t)\,dt = -e^{-x+\pi} + 1$. Then, solving $u = -e^{-x+\pi} + 1$ for x yields $x = \pi - \log(1-u)$. Hence, for $U \in \text{Unif}(0,1)$, $X = P_{Y_2}^{-1}(U) = \pi - \log(1-U)$. So, drawings from $P_{Y_2}^{-1}(U)$ are equivalent to drawings from P_{Y_2}. However, this technique is quite restrictive and it only works in the univariate case (Train, 2009).

We are now ready to return to our example. Performing the importance sampling technique with proposal PDF p_{Y_2} yields:

$$\begin{aligned}
\mathbb{P}(Z \geq \pi) &= \frac{1}{\sqrt{2\pi}} \int_\pi^\infty \frac{e^{-\frac{1}{2}z^2}}{e^{-z+\pi}} \cdot \mathbb{I}\{z > \pi\} \cdot e^{-z+\pi}\,dz \\
&= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^\infty \frac{e^{-\frac{1}{2}z^2}}{e^{-z+\pi}} \cdot \mathbb{I}\{z > \pi\} \cdot e^{-z+\pi}\,dz \\
&= \frac{1}{\sqrt{2\pi}} \cdot \mathbb{E}_{p_{Y_2}}\left[\frac{e^{-\frac{1}{2}Z^2}}{e^{-Z+\pi}}\right].
\end{aligned} \tag{65}$$

Hence, another importance sampling estimator is $\hat{\theta}_{IS}^{\mathcal{T}\text{Expo}(1,\pi)} = \frac{1}{n\sqrt{2\pi}} \sum_{i=1}^{n} e^{-\frac{1}{2}Z_i^2 + Z_i - \pi}$, where $Z_i \sim \mathcal{T}\text{Expo}(1,\pi)$.

```
%% Importance Sampling: Truncated Exponential Proposal PDF
tic
% Generate n realizations from Unif(0,1)
U = random('Uniform',0,1,[n 1]);
% Transform to CDF of truncated exponential
TruncatedExponential_Realization = pi - log(1-U);
% IS integral estimate: mean of exponential quadratic
Transform_Truncated = exp(-0.5*TruncatedExponential_Realization.^2 ...
    )./exp(-TruncatedExponential_Realization + pi);
% Mean function can be used because we do not have any deleted zeros
IS_TruncatedExponential = (1/sqrt(2*pi))*mean(Transform_Truncated);
toc

% Variance of MC estimator (= MSE, by unbiasedness)
MSE_TruncatedExponential = 1/(n-1)*sum( ...
    ((1/sqrt(2*pi))*Transform_Truncated - IS_TruncatedExponential).^2 );
```

Third, we try another proposal PDF, different from an exponential PDF, that matches the product $\mathbb{I}\{z \geq \pi\} \cdot \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}$ as much as possible. Consider a Gamma distribution. Let $Y_3 \sim \text{Gamma}(1,2)$ with PDF $p_{Y_3}(z) = \mathbb{I}\{z > 0\} \cdot \frac{z^{1-1}}{\Gamma(1)2^1} e^{-\frac{z}{2}} = \mathbb{I}\{z > 0\} \cdot \frac{1}{2} e^{-\frac{1}{2}z}$. This is somewhat similar to the product we aim to match. Performing the importance sampling technique with this proposal PDF p_{Y_3} yields:

$$\begin{aligned}
\mathbb{P}(Z \geq \pi) &= \int_{\pi}^{\infty} \frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}}{\frac{1}{2} e^{-\frac{1}{2}z}} \cdot \mathbb{I}\{z > 0\} \cdot \frac{1}{2} e^{-\frac{1}{2}z} \, dz \\
&= \int_{-\infty}^{\infty} \frac{2}{\sqrt{2\pi}} \frac{e^{-\frac{1}{2}z^2}}{e^{-\frac{1}{2}z}} \cdot \mathbb{I}\{z > \pi\} \cdot \frac{1}{2} e^{-\frac{1}{2}z} \, dz \\
&= \frac{\sqrt{2}}{\sqrt{\pi}} \cdot \mathbb{E}_{p_{Y_3}} \left[\frac{e^{-\frac{1}{2}z^2}}{e^{-\frac{1}{2}z}} \cdot \mathbb{I}\{z > \pi\} \cdot \mathbb{I}\{Z > \pi\} \right].
\end{aligned} \tag{66}$$

Hence, another importance sampling estimator is $\hat{\theta}_{IS}^{\text{Gamma}(1,2)} = \frac{1}{n} \frac{\sqrt{2}}{\sqrt{\pi}} \sum_{i=1}^{n} e^{-\frac{1}{2}Z_i^2 + \frac{1}{2}Z_i} \cdot \mathbb{I}\{Z_i > \pi\}$, where $Z_i \sim \text{Gamma}(1,2)$.

```
%% Importance Sampling: Gamma Proposal PDF
tic
% Generate n realizations from Gamma(1,2)
Gamma_Realization = random('Gamma',1,2,[n 1]);
% Indicator function: Z >= pi
Select_Gamma = Gamma_Realization(Gamma_Realization>=pi,1);
% IS integral estimate: mean of exponential quadratic
Transform_Gamma = exp(-0.5*Select_Gamma.^2)./exp(-0.5*Select_Gamma);
IS_Gamma = (sqrt(2)/sqrt(pi))*(sum(Transform_Gamma)/n);
toc

% Variance of MC estimator (= MSE, by unbiasedness)
MSE_Gamma = 1/(n-1)*sum( ...
    ((sqrt(2)/sqrt(pi))*Transform_Gamma - IS_Gamma).^2 );
```

Finally, we consider another new PDF. Let $Y_4 \sim \text{Weib}(2, \sqrt{2})$ with Weibull PDF $p_{Y_4}(z) = \mathbb{I}\{z > 0\} \cdot z e^{-\frac{1}{2}z^2}$. This PDF matches the product $\mathbb{I}\{z \geq \pi\} \cdot \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}$ almost perfectly. So, purely based upon condition IS.4., this seems to be the best proposal PDF so far. Performing the importance sampling technique with this proposal PDF p_{Y_4} yields:

$$\begin{aligned}
\mathbb{P}(Z \geq \pi) &= \frac{1}{\sqrt{2\pi}} \int_{\pi}^{\infty} \frac{e^{-\frac{1}{2}z^2}}{z e^{-\frac{1}{2}z^2}} \cdot \mathbb{I}\{z > 0\} \cdot z e^{-\frac{1}{2}z^2} \, dz \\
&= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \frac{1}{z} \cdot \mathbb{I}\{z > \pi\} \cdot z e^{-\frac{1}{2}z} \, dz \\
&= \frac{1}{\sqrt{2\pi}} \cdot \mathbb{E}_{p_{Y_4}} \left[\frac{1}{Z} \cdot \mathbb{I}\{Z > \pi\} \right].
\end{aligned} \tag{67}$$

This results in our last importance sampling estimator $\hat{\theta}_{IS}^{\text{Weib}(2,\sqrt{2})} = \frac{1}{n} \frac{1}{\sqrt{2\pi}} \sum_{i=1}^{n} \frac{1}{Z_i} \cdot \mathbb{I}\{Z_i > \pi\}$, where $Z_i \sim \text{Weib}(2, \sqrt{2})$.

```
%% Importance Sampling: Weibull Proposal PDF
tic
alpha = 2; beta = sqrt(2);
% Generate n realizations from Weibull(2,sqrt(2))
Weibull_Realization = random('Weibull',alpha,beta,[n 1]);
% Indicator function: Z >= pi
Select_Weibull = Weibull_Realization(Weibull_Realization>=pi,1);
% IS integral estimate: mean of 1/Z
Transform_Weibull = 1./Select_Weibull;
IS_Weibull = (1/sqrt(2*pi))*(sum(Transform_Weibull)/n);
toc

% Variance of MC estimator (= MSE, by unbiasedness)
MSE_Weibull = 1/(n-1)*sum( ...
     ((1/sqrt(2*pi))*Transform_Weibull - IS_Weibull).^2 );
```

We have gathered a Monte Carlo integration estimator and four importance sampling estimators for the integral in equation (60). Let us compare the proposal PDFs. The following code produces a visual comparison of all the PDFs considered, as shown in Figure 7.

```
%% Tails Comparison
x_Values = -5:0.1:10;  % Sequence from -5 to 10 by steps of 0.1

% Theoretical Standard Normal PDF
PD_Normal = makedist('Normal','mu',0,'sigma',1);
PDF_Normal = pdf(PD_Normal,x_Values);
% Theoretical Standard Exponential PDF
PD_Exponential = makedist('Exponential',1);
PDF_Exponential = pdf(PD_Exponential,x_Values);
% Theoretical Standard Truncated Exponential PDF
PD_Exponential_Truncated = truncate(PD_Exponential,pi,inf);
PDF_Exponential_Truncated = pdf(PD_Exponential_Truncated,x_Values);
% Theoretical Standard Weibull PDF
PD_Gamma = makedist('Gamma',1,2);
PDF_Gamma= pdf(PD_Gamma,x_Values);
% Theoretical Standard Weibull PDF
PD_Weibull = makedist('Weibull',2,sqrt(2));
PDF_Weibull= pdf(PD_Weibull,x_Values);
```

```
19
20 plot(x_Values,PDF_Normal,'LineWidth',2,'Color','black')
21 hold on;
22 plot(x_Values,PDF_Exponential,'LineWidth',2,'Color','blue')
23 plot(x_Values,PDF_Exponential_Truncated,'LineWidth',2,'Color','cyan')
24 plot(x_Values,PDF_Gamma,'LineWidth',2,'Color','magenta')
25 plot(x_Values,PDF_Weibull,'LineWidth',2,'Color','red')
26
27 title('Comparison of PDFs','FontSize',27,'Interpreter','latex')
28 xlabel('x','FontSize',21,'Interpreter','latex')
29 ylabel('PDF','FontSize',21,'Interpreter','latex')
30 legend({'$\mathcal{N}(0,1)$','Expo$(1)$','$\mathcal{T}$Expo$(1,\pi)$', ...
31      'Gamma$(1,2)$','Weibull$(2,\sqrt{2})$'}, ...
32      'Location','northeast','FontSize',30,'Interpreter','latex');
33 hold off;
```

In Figure 7, observe that all proposal PDFs satisfy condition IS.6. since each one has heavier tails than the standard normal target PDF. Moreover, conditions IS.1. and IS.2. are also satisfied because the standard normal PDF is positive on $\mathbb{R}$.

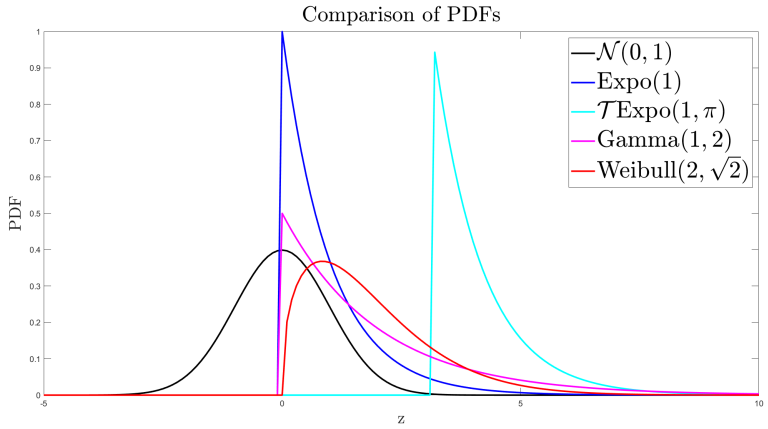

Figure 7

Condition IS.5. can also be checked by analysing the likelihood ratios. The following code produces a visual representation of the likelihood ratio for each of the four proposal PDFs.

```
1 %% Likelihood Ratio Comparison
2 plot(x_Values,PDF_Normal./PDF_Exponential,'LineWidth',2,'Color','blue')
3 hold on;
4 plot(x_Values,PDF_Normal./PDF_Exponential_Truncated, ...
5      'LineWidth',2,'Color','cyan')
6 plot(x_Values,PDF_Normal./PDF_Gamma,'LineWidth',2,'Color','magenta')
7 plot(x_Values,PDF_Normal./PDF_Weibull,'LineWidth',2,'Color','red')
8
9 title('Likelihood Ratios','FontSize',27,'Interpreter','latex')
10 xlim([0 5])
11 xlabel('z','FontSize',21,'Interpreter','latex')
12 ylabel('PDF','FontSize',21,'Interpreter','latex')
13 legend({'Expo$(1)$','$\mathcal{T}$Expo$(1,\pi)$', ...
```

```
    'Gamma$(1,2)$','Weibull$(2,\sqrt{2})$'}, ...
    'Location','northeast','FontSize',30,'Interpreter','latex');
hold off;
```

Figure 8 displays the likelihood ratios for all proposal PDFs. All execpt the likelihood ratio with the Weibull proposal PDF are bounded on $\mathbb{R}$:

$$\lim_{z \downarrow 0} \frac{f_Z(z)}{p_{Y_4}(z)} = \lim_{z \downarrow 0} \frac{1}{\sqrt{2\pi} \cdot z} = +\infty. \tag{68}$$

Therefore, the Weibull proposal PDF is the only one which does not satisfy condition IS.5. This implies that $\hat{\theta}_{IS}^{\text{Weib}(2,\sqrt{2})}$ is likely to not be a good estimator of the integral in equation (60).

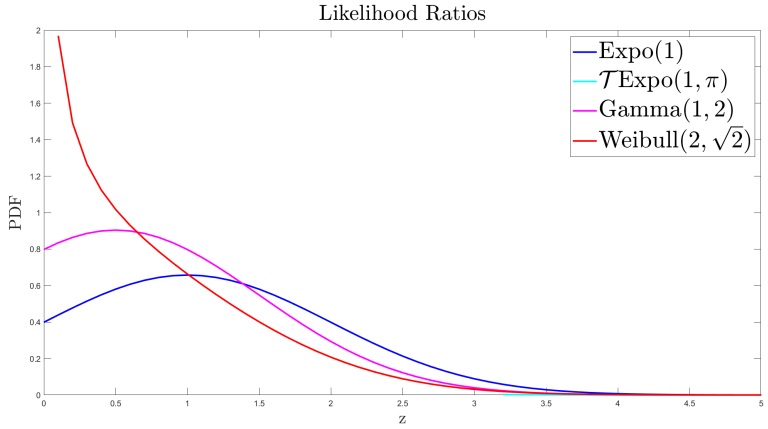

Figure 8

Let us plot all five estimators over a sequence of growing sample sizes to facilitate a comparison of their convergence behaviour and error.

```
%% Plot over sample size
rng('default')   % Control random number generator

tic
% Sequence of sample size from 1000 to 100,000 by steps of 1000
n_Values = 1000:1000:100000;
% Next, create empty vectors to store values:
MC_Integral_Estimates = zeros(size(n_Values));
IS_Exponential = zeros(size(n_Values));
IS_TruncatedExponential = zeros(size(n_Values));
IS_Gamma = zeros(size(n_Values));
IS_Weibull = zeros(size(n_Values));

MSE_MCIntegration = zeros(size(n_Values));
MSE_Exponential = zeros(size(n_Values));
MSE_TruncatedExponential = zeros(size(n_Values));
MSE_Gamma = zeros(size(n_Values));
MSE_Weibull = zeros(size(n_Values));

```

```
20
21 for i = 1:length(n_Values)
22      N = n_Values(i);
23
24      %%% MC Integration
25      % Generate n realizations from N(0,1)
26      StandardNormal_N_Realization = random('Normal',0,1,[N 1]);
27      % MC integral estimate: mean of indicator functions
28      MC_Integral_Estimates(i) = mean(StandardNormal_N_Realization>=pi);
29      % Variance (=MSE) of estimator
30      MSE_MCIntegration(i) = 1/(N-1)*sum( ...
31           (StandardNormal_N_Realization>=pi - ...
32           MC_Integral_Estimates(i)).^2 );
33
34      %%% IS: Exponential
35      % Generate n realizations from Expo(1)
36      Exponential_N_Realization = random('Exponential',1,[N 1]);
37      % Indicator function: Z >= pi
38      Select_N_Exponential = Exponential_N_Realization( ...
39           Exponential_N_Realization>=pi,1);
40      % IS integral estimate: mean of exponential quadratic
41      Transform_N_Exponential = exp(-0.5*Select_N_Exponential.^2 + ...
42           Select_N_Exponential);
43      IS_Exponential(i) = (1/sqrt(2*pi))*(sum(Transform_N_Exponential)/N);
44      % Variance (=MSE) of estimator
45      MSE_Exponential(i) = 1/(N-1)*sum( ...
46           ((1/sqrt(2*pi))*Transform_N_Exponential - ...
47           IS_Exponential(i)).^2 );
48
49      %%% IS: Truncated Exponential
50      % Generate n realizations from Unif(0,1)
51      U = random('Uniform',0,1,[N 1]);
52      % Transform to CDF of truncated exponential
53      TruncatedExponential_N_Realization = pi - log(1-U);
54      % IS integral estimate: mean of exponential quadratic
55      Transform_N_Truncated = exp(-0.5* ...
56           TruncatedExponential_N_Realization.^2 ...
57           + TruncatedExponential_N_Realization - pi);
58      % Mean function can be used because we do not have any deleted zeros
59      IS_TruncatedExponential(i) = (1/sqrt(2*pi))*mean( ...
60           Transform_N_Truncated);
61      % Variance (=MSE) of estimator
62      MSE_TruncatedExponential(i) = 1/(N-1)*sum( ((1/sqrt(2*pi))* ...
63           Transform_N_Truncated - IS_TruncatedExponential(i)).^2 );
64
65      %%% IS: Gamma
66      % Generate n realizations from Gamma(1,2)
67      Gamma_N_Realization = random('Gamma',1,2,[N 1]);
68      % Indicator function: Z >= pi
69      Select_N_Gamma = Gamma_N_Realization(Gamma_N_Realization>=pi,1);
70      % IS integral estimate: mean of exponential quadratic
71      Transform_N_Gamma = exp(-0.5*Select_N_Gamma.^2 + 0.5*Select_N_Gamma);
72      IS_Gamma(i) = (sqrt(2)/sqrt(pi))*(sum(Transform_N_Gamma)/N);
73      % Variance (=MSE) of estimator
```

```
     MSE_Gamma(i) = 1/(N-1)*sum( ...
          ((sqrt(2)/sqrt(pi))*Transform_N_Gamma - IS_Gamma(i)).^2 );

     %%% IS: Weibull
     alpha = 2; beta = sqrt(2);
     % Generate n realizations from Weibull(2,sqrt(2))
     Weibull_N_Realization = random('Weibull',alpha,beta,[N 1]);
     % Indicator function: Z >= pi
     Select_N_Weibull = Weibull_N_Realization( ...
          Weibull_N_Realization>=pi,1);
     % IS integral estimate: mean of 1/Z
     Transform_N_Weibull = 1./Select_N_Weibull;
     IS_Weibull(i) = (1/sqrt(2*pi))*(sum(Transform_N_Weibull)/N);
     % Variance (=MSE) of estimator
     MSE_Weibull(i) = 1/(N-1)*sum( ...
          ((1/sqrt(2*pi))*Transform_N_Weibull - IS_Weibull(i)).^2 );
end
toc
```

After running the code several times, the running time ranges between approximately 1 and 10 seconds. We can plot the convergence behaviour of all five estimators.

```
%%% Plot the estimates
plot(n_Values,MC_Integral_Estimates,'LineWidth',2,'Color','yellow')
hold on;
plot(n_Values,IS_Exponential,'LineWidth',2,'Color','cyan')
plot(n_Values,IS_TruncatedExponential,'LineWidth',2,'Color','magenta')
plot(n_Values,IS_Gamma,'LineWidth',2,'Color','green')
plot(n_Values,IS_Weibull,'LineWidth',2,'Color','blue')

title('Convergence of Estimators','FontSize',27,'Interpreter','latex')
yline(True_Integral,'LineWidth',2);
xlabel('Sample Size $n$','FontSize',21,'Interpreter','latex')
ylabel('Probability','FontSize',21,'Interpreter','latex')
legend({'MC Integration','IS $\sim$ Expo$(1)$', ...
     'IS $\sim$ $\mathcal{T}$Expo$(1,\pi)$','IS $\sim$ Gamma$(1,2)$', ...
     'IS $\sim$ Weibull$(2,\sqrt{2})$'}, ...
     'Location','northeast','FontSize',30,'Interpreter','latex');
hold off;
```

Figure 9 shows that $\hat{\theta}_{IS}^{\text{Weib}(2,\sqrt{2})}$ is a bad estimator for $\mathbb{P}(Z \geq \pi)$ because its point convergence is way off. Recall that the concept of convergence refers to the notion of tendency to a specific value. Thus, even though the Weibull$(2, \sqrt{2})$ PDF matched the product $g(z) \cdot f_Z(z)$ most closely, it results in the worst importance sampling estimator among the four proposal PDFs. We delete this estimator in Figure 9 to obtain a visual comparison of the convergence of the other estimators.

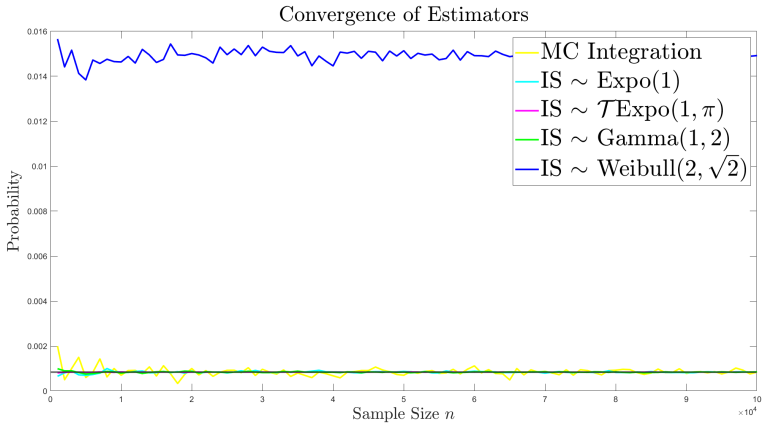

Figure 9

Figure 10 shows that the other importance sampling estimators convergence faster than the Monte Carlo integration estimator in the sense that their distance from the true fluctuates less. The Monte Carlo estimator continues to fluctuate significantly around the true value for relatively large sample sizes. Hence, the importance sampling estimators are better in terms of convergence, but it is hard to distinguish between them.

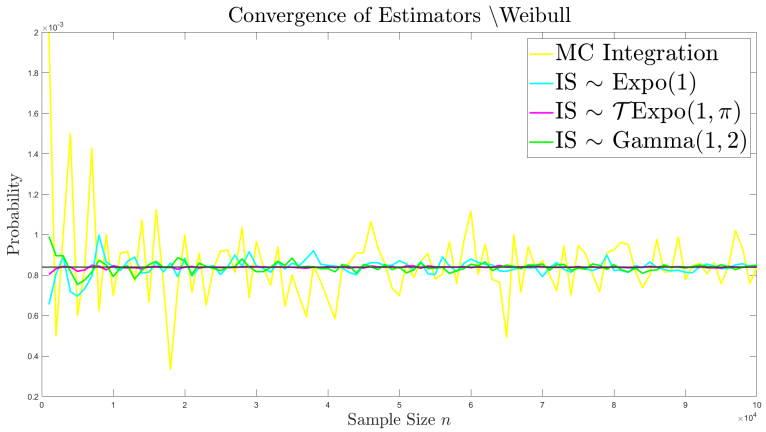

Figure 10

Let us plot the mean squared error of these importance sampling estimates over a sequence of growing sample sizes. Observe that the MSE equals the variance in this case since an importance sampling estimator is unbiased. We plot the MSE by using a logarithmic transformation to improve visibility. This does not influence the comparison since a logarithmic function is monotonically increasing.

```
%%% MSE plot
% Apply a logarithmic transformation to improve comparison
plot(n_Values,log(MSE_MCIntegration),'LineWidth',2,'Color','yellow')
hold on;
plot(n_Values,log(MSE_Exponential),'LineWidth',2,'Color','cyan')
plot(n_Values,log(MSE_TruncatedExponential), ...
```

```
     'LineWidth',2,'Color','magenta')
plot(n_Values,log(MSE_Gamma),'LineWidth',2,'Color','green')
plot(n_Values,log(MSE_Weibull),'LineWidth',2,'Color','blue')
Orange = [240 100 10]/256;
plot(n_Values,log(1./sqrt(n_Values)),'LineWidth',2,'Color',Orange)

title('$\log$(MSE) Comparison','fontsize',27,'Interpreter','latex')
xlabel('Sample Size $n$','FontSize',21,'Interpreter','latex')
ylabel('$\log$(MSE)','FontSize',21,'Interpreter','latex')
legend({'MC Integration','IS $\sim$ Expo$(1)$', ...
     'IS $\sim$ $\mathcal{T}$Expo$(1,\pi)$','IS $\sim$ Gamma$(1,2)$', ...
     'IS $\sim$ Weibull$(2,\sqrt{2})$','$\mathcal{O}(n^{-1/2})$'}, ...
     'Location','northeast','FontSize',30,'Interpreter','latex');
hold off;
```

Figure 11 displays the convergence of the MSE for all five estimators. Observe that all lie below the $n^{-1/2}$ curve. The Monte Carlo estimator and importance sampling estimator with Weibull proposal PDF posses approximately equivalent MSE behaviour. However, the Weibull importance sampling estimator is worse because its point estimate differs from the true value. Furthermore, the truncated exponential importance sampling estimator $\hat{\theta}_{IS}^{\mathcal{T}\text{Expo}(1,\pi)}$ has the lowest MSE among the five estimators. This makes it the best estimator in terms of convergence and MSE amongst all five estimators considered. Thus, by using conditions IS.1-5., we have found three importance sampling estimators that are better than the Monte Carlo integration estimator.

▶

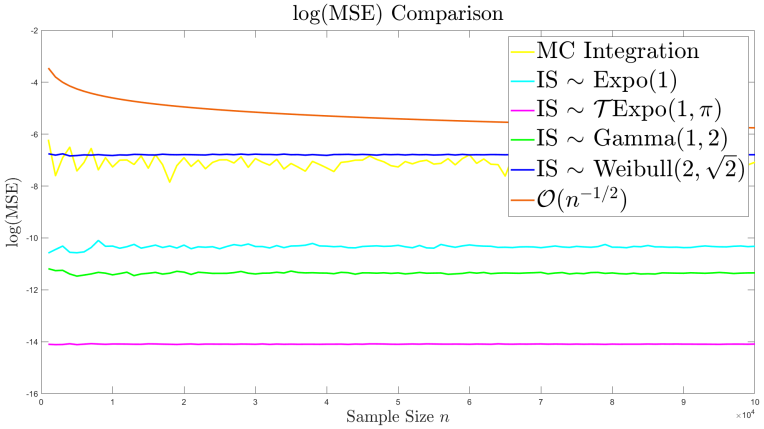

Figure 11

Example 3.2: Security Pricing
Suppose there are n securities for which prices are known. A new security, for example an option or asset, is introduced. This new security is defined by the random cash flow d to be obtained at the end of the period. Let one period duration be equal to a day, and let us denote R as the one-period risk-free return. What is the price of that new security at the end of the day?

One way to calculate the price P of a security with future pay-off random variable $d(X)$,

$X \sim f_X(x)$, and risk-free return rate R is by the discounted expectation (Luenberger, 2013):

$$P = \frac{\mathbb{E}[d(X)]}{R} = \frac{1}{R}\int_{-\infty}^{\infty} d(x) \cdot f_X(x)\, dx. \tag{69}$$

Suppose we know that the risk-free return rate is $R = 1.05$ and the random cash flow is characterized by the following function:

$$d : \mathbb{R} \to \mathbb{R}_{\geq 0}, \quad d(x) := \left| \sin\left(\frac{2\pi x}{21}\right) \cdot \cos\left(\frac{2\pi x}{e} x\right) \right|. \tag{70}$$

The random variable X follows a Gamma$(1.2, 0.5)$ distribution with PDF f_X. However, suppose that we can only determine the distribution up to some constant. In other words, we observe $X \sim \tilde{f}_X(x) := x^{0.2} \cdot e^{-\frac{1}{2}x} \propto f_X(x)$.

The left-hand side plot of Figure 12 displays the graph of the dividend function $d(x)$ without absolute values. This shows that the function has aperiodic wave behaviour which makes sense for random cash flows. Moreover, the right-hand side plot of Figure 12 also displays the trajectory of the dividend random variable over time. The dividends are non-negative and have spikes which could imitate realistic behaviour.

Since the distribution of $d(X)$ is only known up to a constant, we can use weighted importance sampling to compute the integral of interest in equation (69). Let us choose the Expo$(\frac{1}{2})$ as the proposal density: $p(x) = \frac{1}{2}e^{-\frac{1}{2}x}$. Hence, the weights are calculated as follows:

$$w(x_i) = \frac{\tilde{f}_X(x_i)}{p_X(x_i)} = \frac{x^{0.2} \cdot e^{-\frac{1}{2}x}}{\frac{1}{2}e^{-\frac{1}{2}x}} = 2x^{0.2}, \tag{71}$$

where x_i's are the realizations of a random sample $\{X_i\}_{i=1}^n$ drawn from p_X.

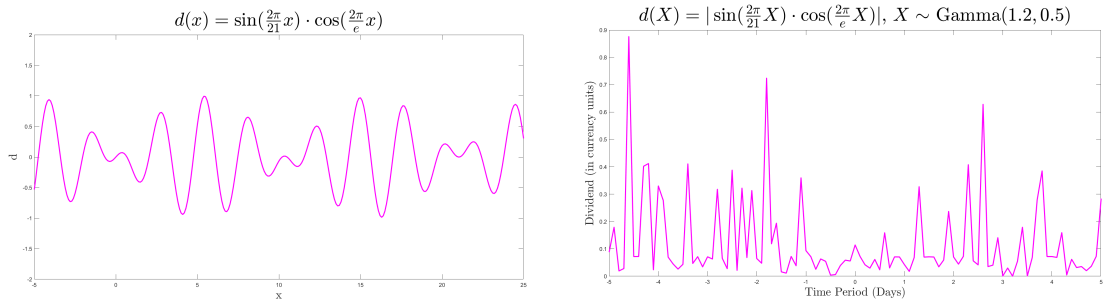

Figure 12

The algorithm for weighted importance sampling is, where $g := d$ and $\tilde{p} = p$ in this case:

Algorithm: Pseudo-Code for **Weighted Importance Sampling**

1: Input: Sample size n

2: **for** $i = 1, ..., n$

3: Sample $X_i \sim \tilde{p}$

4: Compute $g(X_i)$

5: Compute the weight $w(X_i) = \frac{\tilde{f}_X(X_i)}{\tilde{p}(X_i)}$

6: Compute the normalization constant $w_i = \frac{w(X_i)}{\sum_{i=1}^{n} w(X_i)}$

7: **end**

8: Compute the estimator $\hat{\theta}_{IS,w} = \frac{1}{n} \sum_{i=1}^{n} w_i \cdot g(X_i)$

The following code computes $\hat{\theta}_{IS,w}$ and its sample variance of different sample sizes for this example by the algorithm above.

```
rng('default')   % Control random number generator
tic
% Sequence of sample size from 1000 to 100,000 by steps of 1000
n_Values = 1000:1000:100000;
% Next, create empty vectors to store values:
WS_Exponential = zeros(size(n_Values));
SampleVariance_Exponential = zeros(size(n_Values));

for i = 1:length(n_Values)
     N = n_Values(i);

     %%% WS: Exponential
     % Generate n realizations from Expo(1)
     Exponential_N_Realization = random('Exponential',1,[N 1]);
     % Caculate g(X_i)
     g_N_Exponential = (1/1.05)*abs( ...
          sin( (2*pi)/21.*Exponential_N_Realization ) .* ...
          cos( (2*pi)/exp(1).*Exponential_N_Realization ));
     % Calculate weights
     w_N_Exponential = 2*Exponential_N_Realization.^(0.2);
     % Calculate normalization constants
     NormalizationConstants_N_Exponential = w_N_Exponential/sum( ...
          w_N_Exponential);
     % Calculate the weighted importance sampling estimator
     WS_Exponential(i) = sum(NormalizationConstants_N_Exponential.* ...
          g_N_Exponential);
     % Sample Variance
     SampleVariance_Exponential(i) = (N/(N-1))*sum( ...
          (NormalizationConstants_N_Exponential.^2) ...
          .*(g_N_Exponential-WS_Exponential(i)).^2 );
end
toc

%%% Convergence plot
plot(n_Values,WS_Exponential,'LineWidth',2,'Color','blue')
hold on;
title('Convergence of $\hat{\theta}_{IS,w}$', ...
     'FontSize',34,'Interpreter','latex')
```

```
xlabel('Sample Size $n$','FontSize',21,'Interpreter','latex')
legend('$\hat{\theta}_{IS,w} \sim$ Expo$(\frac{1}{2})$', ...
     'Location','northeast','FontSize',30,'Interpreter','latex');
hold off;

%%% Sample Variance plot
plot(n_Values,log(sqrt(SampleVariance_Exponential)), ...
     'LineWidth',2,'Color','blue')
hold on;
Orange = [240 100 10]/256;
plot(n_Values, log(1./sqrt(n_Values)),'LineWidth',2,'Color',Orange)
plot(n_Values, log(1./n_Values),'LineWidth',2,'Color','green')
title('Standard Deviation of $\hat{\theta}_{IS,w}$', ...
     'FontSize',34,'Interpreter','latex')
xlabel('Sample Size $n$','FontSize',21,'Interpreter','latex')
ylabel('$\log(s^2_{IS,w})$','FontSize',21,'Interpreter','latex')
legend({'SD $\hat{\theta}_{IS,w}$', ...
     '$\mathcal{O}(n^{-1/2})$','$\mathcal{O}(n^{-1})$'}, ...
     'Location','northeast','FontSize',30,'Interpreter','latex');
hold off;
```

The left-hand side plot of Figure 13 shows the weighted importance sampling estimates over an increasing sample size. At first glance, it seems that the estimates deviate a lot, but the variance is relatively small. The right-hand side plot of Figure 13 indicates that the sample standard deviation of $\hat{\theta}_{IS,w}$ decreases faster than the rate, $\mathcal{O}(n^{-\frac{1}{2}})$, at which the standard deviation of a Monte Carlo integration estimator decreases. Thus, the price of the new security at the end of the day is approximately 0.192 currency units. ▶

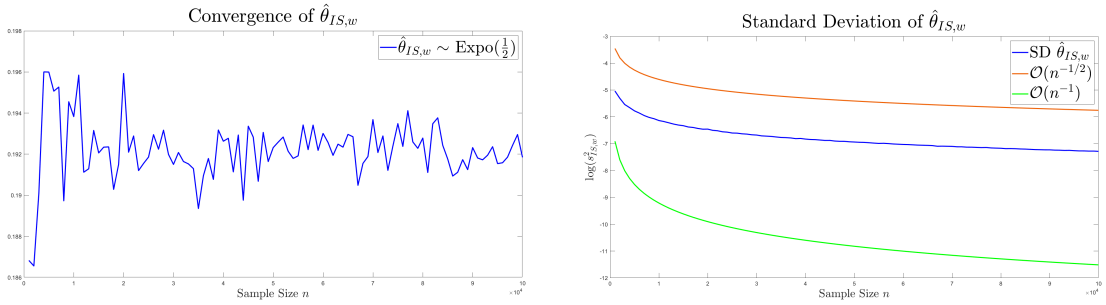

Figure 13

4 MCMC Integration

4.1 Motivation

Approximating the integral in equation (1) by Monte Carlo integration fails when there is an infinite variance problem or when it is difficult to sample from the target PDF f_X. Recall that the Monte Carlo integration technique works as follows:

$$\hat{\theta}_{MC} := \frac{1}{n}\sum_{i=1}^{n} g(X_i) \xrightarrow{n\to\infty} \int_A \phi(x)\,dx = \int_{\mathbb{R}} g(x)\cdot f_X(x)\,dx =: \mathbb{E}_{f_X}[g(X)] = \theta_{MC}, \tag{72}$$

where $X_i \sim f_X(x)$.

In general, computers are great at sampling i.i.d. pseudo-random numbers from a uniform distribution. Pseudo-randomness refers to the process of generating a deterministic sequence of numbers which appear random, and they pass various statistical tests of randomness ensuring the i.i.d. condition. However, sampling from non-uniform PDFs is much more complicated. `random(·)` enables us to sample form a finite collection of common PDFs, but there are uncountably many PDFs. There exist techniques, such as inverse transformation method and rejection sampling, that use uniformly distributed samples to draw samples from a more complicated PDF. But these methods have restrictive conditions and fail for multi-dimensional PDFs (Guilhoto, 2017).

Consider multi-dimensional integrals. Suppose $\phi : \Omega \subseteq \mathbb{R}^d \to \mathbb{R}$, $d \in \mathbb{N}$, is an integrable function that is bounded on some $A \subseteq \Omega$. Notice that we only consider scalar-valued functions, i.e. functions from a vector space to a field ($\mathbb{Q}, \mathbb{R}, \mathbb{C}$, etc.). Hence, the integral in equation (72) generalizes to:

$$\int_A \phi(\boldsymbol{x})\,d\boldsymbol{x} = \int_{\mathbb{R}^d} g(\boldsymbol{x})\cdot f_{\boldsymbol{X}}(\boldsymbol{x})\,d\boldsymbol{x} =: \mathbb{E}_{f_{\boldsymbol{X}}}[g(\boldsymbol{X})], \tag{73}$$

where $f_{\boldsymbol{X}}$ is a d-dimensional PDF of the d-dimensional random vector $\boldsymbol{X} = (X_1, ..., X_d)^T$ and $\boldsymbol{x} = (x_1, ..., x_d)^T \in \Omega$.

Let $A := A_1 \times ... \times A_d \subseteq \Omega$. In section 2.3, we have seen that the integral in equation (73) simplifies when $\boldsymbol{X}$ is a vector of i.i.d. random variables X_i:

$$\begin{aligned}
\mathbb{E}_{f_{\boldsymbol{X}}}[g(\boldsymbol{X})] &= \int_{\mathbb{R}} \cdots \int_{\mathbb{R}} g(\boldsymbol{x})\cdot[f_{X_1}(x_1)\cdot ... \cdot f_{X_d}(x_d)]\; dx_1 \cdots dx_d \\
&= \int_{\mathbb{R}} \cdots \int_{\mathbb{R}} f_{X_d}(x_d)\cdot ... \cdot f_{X_2}(x_2) \int_{\mathbb{R}} g(\boldsymbol{x})\cdot f_{X_1}(x_1)\,dx_1 \cdots dx_d \\
&= \mathbb{E}_{f_{X_d}} \circ ... \circ \mathbb{E}_{f_{X_2}} \circ \mathbb{E}_{f_{X_1}}[g(\boldsymbol{X})] \in \mathbb{R}.
\end{aligned} \tag{74}$$

In this special case, inverse transformation method or rejection sampling could be used to generate samples from f_{X_j}, $j \in \{1, ..., d\}$, and Monte Carlo integration to approximate the univariate integrals $\int_{\mathbb{R}} g(\boldsymbol{x})\cdot f_{X_1}(x_1)\,dx_1$ and $\int_{\mathbb{R}} \mathbb{E}_{f_{X_{j-1}}}[g(\boldsymbol{X})]\cdot f_{X_j}(x_j)\,dx_j$ for $j \in \{2, ..., d\}$. However, this situation of independence is not always possible. Therefore, we want a more general technique for sampling from a multi-dimensional PDF that allows for dependencies.

Markov Chain Monte Carlo (MCMC) refers to a collection of algorithms that generate samples from a target distribution $f_{\boldsymbol{X}}$. The idea of MCMC is to generate a Markov chain that converges in distribution to $f_{\boldsymbol{X}}$, and use the Markov chain variant of the SLLN for random variables, called the Ergodic Theorem, to estimate an expectation of g with respect to $f_{\boldsymbol{X}}$. The most well-known and widely used MCMC algorithm is the Metropolis-Hastings algorithm. It was first developed by Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller (1953) and later generalized by Hastings (1970). However, there is some controversy regarding the credits

Metropolis received with respect to his contribution to the development of this algorithm. The Metropolis-Hastings algorithm is one of the ten most influential algorithms of the 20^{th} century according to Dongarra and Sullivan (2000).

The Metropolis-Hastings algorithm essentially works as follows. Suppose some tourists want to explore a city by sightseeing. Assume the tourists start at some central landmark in the city. After having visited this landmark, the tourists consider to move to a new landmark, i.e. a new state. The guide proposes a football stadium. If this proposal is popular, it will be accepted. Otherwise, the tourists prefer to stay at the current landmark. A landmark that is likely to be unpopular, for instance a supermarket, has low probability to be accepted. If this procedure is iterated many times, then a chain of states is created which is equivalent to a path through the city in this example. Hence, if this is done for a sufficient number of times, then the tourist will have experienced a good exploration of the city where popular landmarks are visited frequently. This is the intuition behind the Metropolis-Hastings algorithm, where the visited landmarks represent the states of a Markov chain and the map of a city represents the target PDF. Observe that every decision made is based upon the current landmark which introduces some dependency.

First, rejection sampling method is introduced, which is impractical for sampling from multi-dimensional PDFs, but it shares some similarities with MCMC regarding the choice of accepting future values. Second, discrete-time Markov chains are introduced together with the Markov chain variant of the strong law of large numbers. Third, the most important concepts of discrete-time Markov chains that are useful for MCMC algorithms are generalized to continuous-time Markov chain. Fourth, the Metropolis-Hastings algorithm is presented and applied to sampling from a bimodal univariate PDF. Lastly, the Metropolis-Hastings algorithm is applied to approximate a multi-dimensional integral that is of interest in econometrics.

4.2 Rejection Sampling

The two most well-known techniques to sample from a univariate non-uniform target PDF f_X are the inverse transformation method and the rejection sampling method. In example 3.1, the inverse transformation method was explained. This technique relies on setting the CDF $F_X(x)$ equal to U, and rewriting the equation in terms of U. When $U \sim \text{Unif}(0,1)$, $X = F_X^{-1}(U)$ enables us to draw from F_X by applying the function F_X^{-1} to the random variable U. If the normalized CDF F_X is known with existing (generalized) inverse, then the inverse transformation method is the best choice. In many instances, however, this is not the case. An sampling technique with less restrictive assumptions is rejection sampling, or also called accept-reject sampling.

The inverse transformation method belongs to the class of direct sampling techniques, because it directly deals with the CDF of the random variable X to be generated. Rejection sampling, due to von Neumann (1951), is an indirect sampling technique, meaning that a generated proposal random variable is accepted subject to passing a test (Robert and Casella, 2010). In comparison to the inverse transformation method, rejection sampling does not require the target PDF to be normalized, i.e. f_X can be known up to its normalization constant c_f:

$$\tilde{f}_X(x) = \frac{1}{c_f} f_X(x), \ \forall x \in \mathbb{R}. \tag{75}$$

Suppose we have the following unnormalized PDF, that is bounded on $[-3,3] \subsetneq \mathbb{R}$:

$$\tilde{f}_X(x) = e^{-\frac{1}{2}x^2} \cdot \left[\pi \cos^2(x) \cdot \sin^2(\pi x) + 1\right] \cdot \mathbb{I}\{x \in [-3,3]\}. \tag{76}$$

See Figure 14 for the graph of $\tilde{f}_X$. The idea of rejection sampling is to choose a proposal PDF p_X such that the area below its graph covers the area below the graph of f_X, i.e.:

$$\{(x,y) \ : \ 0 \leq y \leq \tilde{f}_X(x), \ x \in \mathbb{R}\} \subset \{(x,y) \ : \ 0 \leq y \leq p_X(x), \ x \in \mathbb{R}\} \tag{77}$$

The simplest choice for a proposal PDF is the uniform PDF over the support of $\tilde{f}_X$, that is $p_X \overset{d}{=} \text{Unif}(-3,3)$ in this case. See Figure 14. However, this PDF does not cover $\tilde{f}_X$ everywhere. Hence, the PDF needs to be scaled by a constant C:

$$C := \underbrace{\sup\{f(x) : x \in \mathbb{R}\}}_{c} \cdot (\max\{Supp(f)\} - \min\{Supp(f)\}) \, . \tag{78}$$

We want that the scaled uniform proposal PDF equals c, the maximal value f attains, on its support. Furthermore, the other term in C is the corresponding regularization constant. Figure 14 shows that $C \cdot p_X$ covers $\tilde{f}_X$.

```
%% Rejection Sampling Example
x_Values = -4:0.001:4;

%%% Tilde_f
Trigonometric_Part = pi*cos(x_Values).^2.*sin(pi*x_Values).^2 + 1;
Tilde_f = zeros(1,length(x_Values));
for i=1:length(x_Values)
     if abs(x_Values(i)) <= 3
          Tilde_f(i) = exp(-0.5*x_Values(i).^2).*Trigonometric_Part(i);
          else
          Tilde_f(i) = 0;
     end
end

PD_StandardUniform = makedist('Uniform',-3,3);
PDF_StandardUniform = pdf(PD_StandardUniform,x_Values);
c = max(Tilde_f);
C = c*(3-(-3));

%%% Plotting the functions with dotted lines
plot(x_Values,Tilde_f,'LineWidth',2,'Color','green')
hold on;
plot(x_Values,PDF_StandardUniform,'LineWidth',2,'Color','blue')
plot(x_Values,C*PDF_StandardUniform,'LineWidth',2,'Color','magenta')

title(['$\tilde{f}_X(x) = e^{-\frac{1}{2}x^2} \cdot \left[ ' ...
     '\pi \cos^2(x) \cdot \sin^2(\pi x) + 1 \right]$'], ...
     'FontSize',27,'Interpreter','latex')
xlabel('x','FontSize',21,'Interpreter','latex')
Grey = [0.7 0.7 0.7];
plot(-4:-3,[max(Tilde_f) max(Tilde_f)],'--','LineWidth',2,'Color',Grey)
text(-5.1,max(Tilde_f),'$C \cdot p_X(-1.2)$', ...
     'FontSize',21,'Interpreter','latex')
xline(-1.2,'--','LineWidth',2,'Color',Grey)
text(-1.3,-0.08,'-1.2','FontSize',10)
f_Value = Tilde_f((-1.2+4)/0.01);  % f(-1.2)

plot(-1.2,f_Value,'Marker','.','MarkerSize',27, ...
```

```
    'LineWidth',2,'Color','red');  % Colour intersection
legend({'$\tilde{f}_X$','Unif$(-3,3)$','Scaled Unif$(-3,3)$'}, ...
    'Location','northeast','FontSize',30,'Interpreter','latex');
hold off;
```

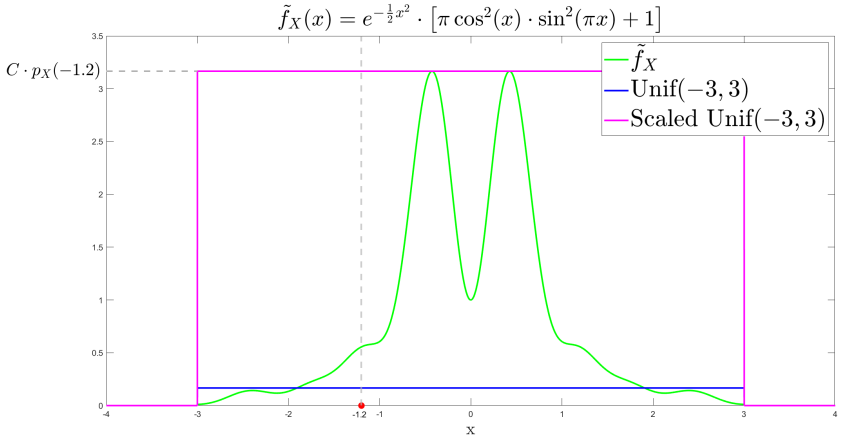

Figure 14

Now, we want to use $C \cdot p_X \overset{d}{=} C \cdot$ Unif$(-3,3)$ to sample from $\tilde{f}_X$. Consider $x = -1.2$, see Figure 14. $\tilde{f}_X$ evaluated at $x = -1.2$ yields a much smaller value than $C \cdot p_X(-1.2)$. Hence, let us consider a random variable $Y \sim \text{Unif}(0, C \cdot p_X(-1.2))$. If $Y \leq \tilde{f}_X(-1.2)$, then we accept $x = -1.2$. Otherwise, we reject it. This is the acceptance-rejection test of the rejection sampling method. Observe that in the neighbourhood around a peak of $\tilde{f}_X$, it is much more likely that x gets accepted. Conversely, it is much more likely that a value x gets rejected when $\tilde{f}_X$ takes on relatively small values in a neighbourhood around x.

The algorithm of rejection sampling is given by:

Algorithm: Pseudo-Code for **Rejection Sampling**

Input: The number of iterations n and scaling factor C
for $i = 1, ..., n$
    Generate $X \sim p_X$
    Generate $U \sim \text{Unif}(0, C \cdot p_X(X))$
    if $U \leq \tilde{f}_X(X)$ **then**
        Accept X
    end
end

Frequently, the rejection sampling algorithm is stated slightly different in terms of acceptance rate α:

    Generate $U \sim \text{Unif}(0, 1)$
    if $U \leq \frac{\tilde{f}_X(X)}{C \cdot p_X(X)} =: \frac{1}{C} \cdot \alpha$ **then**
        Accept X
    end

The two formulations of the algorithm are equivalent because a uniform random variable is invariant under linear transformation:

$$U \sim \text{Unif}(0, C \cdot p_X(X)) \quad \iff \quad C \cdot p_X(X) \cdot U \sim \text{Unif}(0, 1). \tag{79}$$

The latter formulation of the algorithm is in some sense analogous to the Metropolis-Hastings algorithms, because both use a somewhat similar acceptance criterium. However, as we will see later, the main difference next to the use of Markov chains, is that rejection sampling keeps rejected values while Metropolis-Hastings algorithm does not. This is less efficient in terms of computer power.

The following theorem ensures us that the rejection sampling algorithm creates samples from the target PDF.

Theorem 4.1. *The random variable X generated in the rejection sampling algorithm has the desired target PDF f_X.*

Proof: See Rubinstein (2016). ■

Theorem 4.1 works for every proposal PDF p_X that satisfies the following two conditions:

RS.1.	$Supp(\tilde{f}_X) \subseteq Supp(p_X)$
RS.2.	p_X covers $\tilde{f}_X$: $\exists$ a $C \in \mathbb{R}_{\geq 1}$ such that $\tilde{f}_X(x) \leq C \cdot p_X(x)$, $\forall x \in \mathbb{R}$.

Applying the rejection sampling algorithm to our example.

```
%%% Selecting points below Tilde_f
n=1000;
Uniform_Points = random('Uniform',-3,3,[n 2]);
ScaledUniform_Points = [Uniform_Points(:,1) (c/3)*abs( ...
Uniform_Points(:,2))];  % Merge first column and scaled second column

Red_Region = zeros(n,2);
Green_Region = zeros(n,2);
for j=1:n
     x_Coordinate = ScaledUniform_Points(j,1);
     y_Coordinate = ScaledUniform_Points(j,2);

     % Find index in x_Values for Tilde_f(x_Coordinate)
     Find_Index = (abs(min(x_Values)) + x_Coordinate) / 0.001;
     % int16() formats it so that it can be used as an input
     Index = int16(Find_Index)+1;  % Small round-off error

     if y_Coordinate <= Tilde_f(Index)
          % Colour the point green
          Green_Region(j,1) = x_Coordinate;
          Green_Region(j,2) = y_Coordinate;
          else
          % Colour the point red
          Red_Region(j,1) = x_Coordinate;
          Red_Region(j,2) = y_Coordinate;
     end
end

%%% Plotting the points below the scaled uniform PDF
plot(x_Values,Tilde_f,'LineWidth',2,'Color','green')
```

```
hold on;
plot(x_Values,C*PDF_StandardUniform,'LineWidth',2,'Color','magenta')
scatter(Red_Region(:,1),Red_Region(:,2), ...  % Plot the red points
     'MarkerEdgeColor','red','MarkerFaceColor','red')
scatter(Green_Region(:,1),Green_Region(:,2), ...  % Plot the green points
     'MarkerEdgeColor','green','MarkerFaceColor','green')

title('$\tilde{f}_X \ \ \& \ $ Scaled Uniform Proposal PDF', ...
     'FontSize',27,'Interpreter','latex')
xlabel('x','FontSize',21,'Interpreter','latex')
legend({'$\tilde{f}_X$','$C \cdot$Unif$(-3,3)$'}, ...
     'Location','northeast','FontSize',30,'Interpreter','latex');
hold off;
```

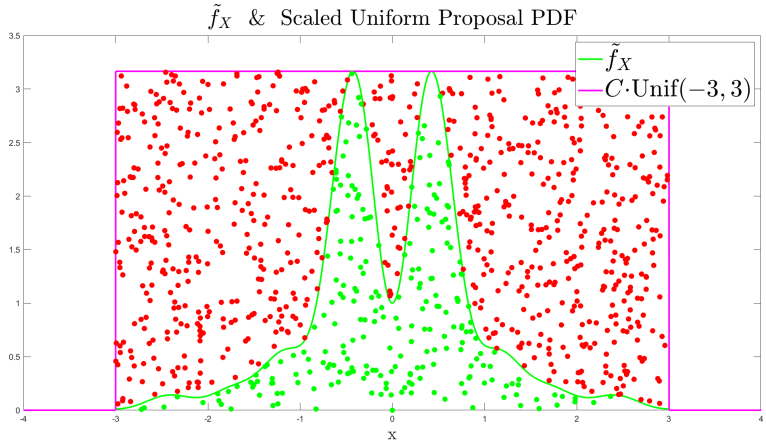

Figure 15

Figure 15 shows the outcome of the rejection sampling algorithm visually. Observe that many points are rejected, i.e. coloured red. This is because of the big difference in area under the graphs of p_X and $\tilde{f}_X$. The green points represent the accepted points. Since Theorem 4.1 holds for every proposal PDF p_X that satisfies both RS.1. and RS.2., we can choose another proposal PDF that covers $\tilde{f}_X$ with less difference in area. Define $p_X \overset{d}{=} \mathcal{N}(0,1)$. By trial-and-error, one can find that $C \equiv C_N = 9$ satisfies condition RS.2., see Figure 16.

```
%% New proposal PDF
n=1000;
PD_StandardNormal = makedist('Normal','mu',0,'sigma',1);
PDF_StandardNormal = pdf(PD_StandardNormal,x_Values);
C_N = 9;  % Constant that ensures that proposal PDF lies above target PDF

plot(x_Values,Tilde_f,'LineWidth',2,'Color','green')
hold on;
plot(x_Values,C_N*PDF_StandardNormal,'LineWidth',2,'Color','magenta')

title(['$\tilde{f}_X(x) = e^{-\frac{1}{2}x^2} \cdot \left[ ' ...
     '\pi \cos^2(x) \cdot \sin^2(\pi x) + 1 \right]$'], ...
     'FontSize',27,'Interpreter','latex')
```

```
xlabel('x','FontSize',21,'Interpreter','latex')
legend({'$\tilde{f}_X$','$9 \cdot \mathcal{N}(0,1)$'}, ...
     'Location','northeast','FontSize',30,'Interpreter','latex');
hold off;
```

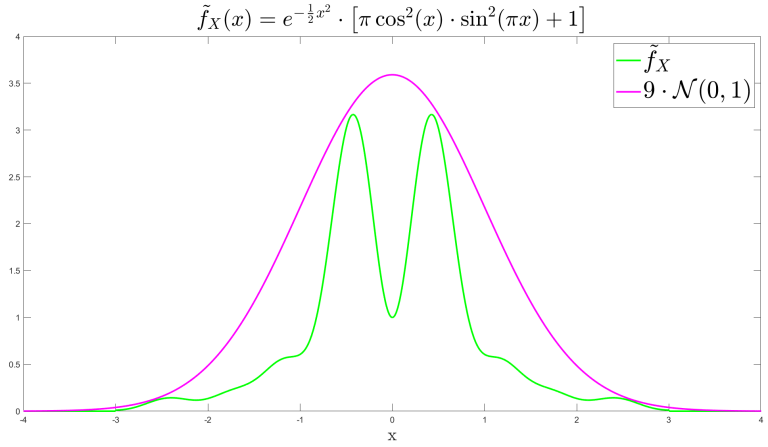

Figure 16

Applying the rejection sampling with proposal PDF $p_x \overset{d}{=} \mathcal{N}(0,1)$.

```
%%% Generate points the fill the region below the standard normal curve
rng('default')
StandardNormal_Realizations = random('Normal',0,1,[n 1]);  % x-coordinates
StandardNormal_Points = (1/sqrt(2*pi))*exp( ...  % y-coordinates
     -0.5*StandardNormal_Realizations.^2);
Uniform_Scaling = random('Uniform',0,1,[n 1]);  % Spread points vertically
c = C_N;

ScaledStandardNormal_Points = [
StandardNormal_Realizations c*Uniform_Scaling.*StandardNormal_Points];
scatter(ScaledStandardNormal_Points(:,1),ScaledStandardNormal_Points(:,2))

%%% Selecting points below Tilde_f
Red_Region = zeros(n,2);
Green_Region = zeros(n,2);
for k=1:n
     x_Coordinate = ScaledStandardNormal_Points(k,1);
     y_Coordinate = ScaledStandardNormal_Points(k,2);

     % Find index in x_Values for Tilde_f(x_Coordinate)
     Find_Index = (abs(min(x_Values)) + x_Coordinate) / 0.001;
     % int16() formats it so that it can be used as an input
     Index = int16(Find_Index)+1;  % Small round-off error

     if y_Coordinate <= Tilde_f(Index)
          % Colour the point green
          Green_Region(k,1) = x_Coordinate;
          Green_Region(k,2) = y_Coordinate;
```

```
            else
            % Colour the point red
            Red_Region(k,1) = x_Coordinate;
            Red_Region(k,2) = y_Coordinate;
        end
end

%%% Plotting the points below the scaled standard normal PDF
plot(x_Values,Tilde_f,'LineWidth',2,'Color','green')
hold on;
plot(x_Values,C_N*PDF_StandardNormal,'LineWidth',2,'Color','magenta')
scatter(Red_Region(:,1),Red_Region(:,2), ...  % Plot the red points
    'MarkerEdgeColor','red','MarkerFaceColor','red')
scatter(Green_Region(:,1),Green_Region(:,2), ...  % Plot the green points
    'MarkerEdgeColor','green','MarkerFaceColor','green')

title('$\tilde{f}_X \ \ \& \ $ Scaled Normal Proposal PDF', ...
    'FontSize',27,'Interpreter','latex')
xlabel('x','FontSize',21,'Interpreter','latex')
legend({'$\tilde{f}(x)$','$9 \cdot \mathcal{N}(0,1)$'}, ...
    'Location','northeast','FontSize',30,'Interpreter','latex');
hold off;
```

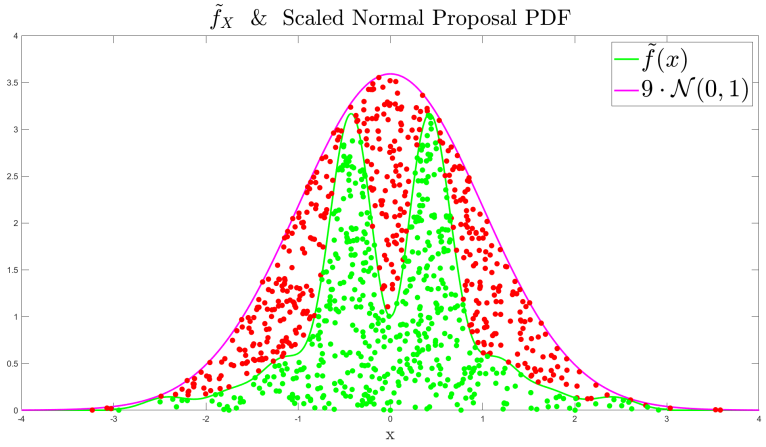

Figure 17

Rejection sampling can be generalized to simulate independent d-dimensional PDFs, but it suffers from a curse of dimensionality: the rejection rate rapidly increases as d increases. Therefore, it becomes impractical for large d. Hence, a better sampling technique is needed for approximating multi-dimensional integrals.

4.3 Discrete-Time Markov Chains

Markov chains were invented, or discovered, by the well-known Russian mathematician Andrey Markov who was a student of Pafnuty Chebyshev. Markov chains are a type of stochastic processes described by a stochastic matrix, which encodes the probabilities of the transitions between different states, and an initial distribution. Markov chains will not be explained in

depth. For those encountering this topic for the first time, references such as Levin and Peres (2017) and Ross (2019) are recommended.

Let $(\Omega, \mathcal{B}(\Omega), \mathbb{P})$ be a probability space.

Definition 4.1. *A stochastic process $\{X(t) : t \in T\}$ is a family of random variables $X_t : \Omega \to S$, where S is the state-space containing all states of the stochastic process and T is a time index set.*

T can be either countable or uncountably infinite. Countable means that it has an one-to-one correspondence, i.e. a bijective relation, with $\mathbb{N}$, for example: the alphabet $\{a, ..., z\}$ and the integers $\mathbb{Z}$. We restrict ourself to discrete-time processes by defining $T := \mathbb{N}$. If $\#S < \infty$, then we have a finite state space with labels $1, ..., N < \infty$. Additionally, we also restrict ourselves to finite state spaces, because a computer can only consider finitely many values.

Stochastic processes can be used to model real-life processes. For example, stochastic processes can be used as a probabilistic model to represent the price of one share of some security, such as NVIDIA, at the end of $n + 1 \in \mathbb{N}$ trading days. The simplest model would assume the random variables X_i, $i \in \{1, ..., n, n + 1\}$, to be independent. However, this is not a realistic model, because the price at the end of day $n + 1$ clearly depends on the price at the end of the previous days $n, n - 1, n - 2, ..., k \geq 1$. Taking all these dependencies into account would result in a very complicated model. But it might be reasonable to assume that the price at the end of day $n + 1$ only depends on the price at the end of day n. This is exactly what Markov chains do, because they posses the (first-order) Markov property (Ross, 2019).

Proposition 4.1. (Markov property) *A discrete-time stochastic process $\{X(t) := X_t : t \in \mathbb{N}\}$ with state space S is a discrete-time Markov chain, if:*

$$\mathbb{P}(X_{n+1} = i_{n+1} \mid X_n = i_n, ..., X_0 = i_0) = \mathbb{P}(X_{n+1} = i_{n+1} \mid X_n = i_n), \tag{80}$$

for time index $n \in \mathbb{N}$ and for states $i_0, ..., i_{n+1} \in S$.

In other words, a Markov chain is sequence of random variables such that a given state at time $n + 1$ only depends on its previous state but is conditionally independent of all other prior states. This can describe many real-world systems.

Furthermore, we want to define the evolution structure of the Markov chain. It turns out that a discrete-time Markov chain is completely determined by its initial distribution and transition matrix. First, we define an initial distribution: $\boldsymbol{\lambda} = (\lambda_i)_{i \in S} \in \mathbb{R}^{1 \times N}$, where $N := \#S$. λ is a probability measure on the state space S which defines a probability mass function (PMF) meaning that $\lambda_i \in [0, 1]$, $\forall i \in S$, and $\sum_{i \in S} \lambda_i = 1$. An initial distribution provides the probabilities that the Markov chain is in state $i \in S$ at initial time $t = 0$: $\mathbb{P}(X_0 = i) = \lambda_i$. Second, we define (one-step) transition probabilities from state i to j at time point n:

$$p_{ij,n} := \mathbb{P}\left(X_{n+1} = j \mid X_n = i\right) \tag{81}$$

Next to the discrete-time and finite state-space assumptions, we also assume that the Markov chain is homogenous. Homogeneity means that the transition probabilities do not depend on time, i.e. the Markov chain is time-invariant:

$$\mathbb{P}(X_{n+1} = j \mid X_n = i = \mathbb{P}(X_1 = i \mid X_0 = j) =: p_{ij}. \tag{82}$$

In other words, $p_{ij} = \mathbb{P}(X_{n+1} = i_{n+1} \mid X_n = i_n) = \mathbb{P}(X_n = i_n \mid X_{n-1} = i_{n-1}) = ...$ $= \mathbb{P}(X_1 = i_1 \mid X_0 = i_0)$. Hence, $p_{ij,n} = p_{ij}$. The transition probabilities between all states can

be stored in a matrix:

$$\mathbf{P} := \begin{bmatrix} p_{11} & \dots & p_{1N} \\ \vdots & \ddots & \vdots \\ p_{N1} & \dots & p_{NN} \end{bmatrix} = \begin{bmatrix} \mathbb{P}\left(X_1 = 1 \mid X_0 = 1\right) & \dots & \mathbb{P}\left(X_1 = N \mid X_0 = 1\right) \\ \vdots & \ddots & \vdots \\ \mathbb{P}\left(X_1 = 1 \mid X_0 = N\right) & \dots & \mathbb{P}\left(X_1 = N \mid X_0 = N\right) \end{bmatrix}. \tag{83}$$

This is a stochastic matrix, because it is a square matrix with only non-negative entries where all the rows must sum to one. We call this stochastic matrix $\mathbf{P}$ the transition matrix of the Markov chain. Observe that $\mathbf{P}$ is independent of time, which implies that the transition probabilities between two states at any time point only depends on the difference between the states.

So, we have obtained a complete description of a discrete-time homogenous Markov chain with finite state space.

Definition 4.2. *$\{X_i\}_{i\in S}$ is a discrete-time homogenous Markov chain with finite state space $S := \{1, ..., N\} \subset \mathbb{N}$ and transition matrix $\mathbf{P}$, if:*

- *X_0 has an initial distribution $\boldsymbol{\lambda} = (\lambda_i)_{i\in S} \implies \mathbb{P}(X_0 = i) = \lambda_i$;*
- *$\mathbb{P}(X_{n+1} = i_{n+1} \mid X_n = i_n, ..., X_0 = i_0) = p_{i_{n+1}, i_n} =: p_{ij}$.*

Example 4.1: Jumping Frog
A classical example for Markov chains considers a frog in a pond with two floating pond plants labelled 1 and 2. Hence, the state space is $S = \{1, 2\}$. Every day, the frog decides whether to jump to the other floating plant. Suppose there is a 50% chance that the frog sits on either plant 1 or 2 at day 0. Hence, the initial distribution is: $\boldsymbol{\lambda} = (\mathbb{P}(X_0 = 1) \;\; \mathbb{P}(X_0 = 2))$ $= (0.5 \;\; 0.5)$. When the frog sits on plant 1, there is a 40% chance that it will jump to plant 2. And when the frog sits on plant 2, there is a 80% chance that it will jump to plant 1. This results in the following transition matrix:

$$\mathbf{P} = \begin{bmatrix} 1 - 0.4 & 0.4 \\ 0.8 & 1 - 0.8 \end{bmatrix} = \begin{bmatrix} 0.6 & 0.4 \\ 0.8 & 0.2 \end{bmatrix} \tag{84}$$

▶

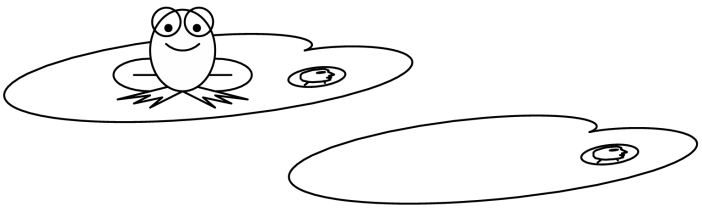

Figure 18: Jumping frog illustration: Levin and Peres (2017).

In addition to the structure of the discrete-time homogeneous Markov chain, we also want to describe the distribution of the Markov chain after n time periods. Let $\boldsymbol{\mu}_n \in \mathbb{R}^{1\times N}$ denote the distribution of the discrete-time homogenous Markov chain after n periods. First, we can

calculate the probability that the Markov chain is in some arbitrary state $j \in S$ after one period, using the law of total probability:

$$\begin{aligned}\mathbb{P}(X_1 = j) &= \sum_{i=1}^{N} \mathbb{P}(X_1 = j \mid X_0 = 1) \cdot \mathbb{P}(X_0 = i) \\ &= \sum_{i=1}^{N} p_{ij} \cdot \lambda_i = \sum_{i=1}^{N} \lambda_i \cdot p_{ij} = (\boldsymbol{\lambda}\mathbf{P})_j,\end{aligned} \tag{85}$$

where $(\boldsymbol{\lambda}\mathbf{P})_j$ denotes the j^{th} entry of $\boldsymbol{\lambda}\mathbf{P}$. Hence, the distribution of the Markov chain after one period is given by: $\boldsymbol{\mu}_1 = \boldsymbol{\lambda}\mathbf{P}$. In order to generalize this to a formula for $\boldsymbol{\mu}_n$, we need an important lemma about the n-step transition matrix.

Lemma 4.1. *The n-step transition matrix $\mathbf{P}^{(n)} = [\mathbb{P}(X_n = j \mid X_0 = i)]_{i,j \in S}$ is equal to the n^{th} power of the transition matrix $\mathbf{P}$: $\mathbf{P}^{(n)} = \mathbf{P}^n$.*

Proof: Pick two arbitrary states $i, j \in S$. Then, rewriting p_{ij} yields:

$$\begin{aligned}\mathbf{P}_{ij}^{(n)} = p_{ij}^{(n)} &= \mathbb{P}(X_n = j \mid X_0 = i) = \sum_{k=1}^{N} \mathbb{P}(X_n = j, X_1 = k \mid X_0 = i) \\ &\overset{(\star)}{=} \sum_{k=1}^{N} \mathbb{P}(X_n = j \mid X_1 = k, X_0 = i) \cdot \mathbb{P}(X_1 = k \mid X_0 = i) \\ &= \sum_{k=1}^{N} \mathbb{P}(X_1 = k \mid X_0 = i) \cdot \mathbb{P}(X_n = j \mid X_1 = k) \\ &= \sum_{k=1}^{N} p_{ik} \cdot p_{kj}^{(n-1)} = p_{ij} \cdot p_{ij}^{(n-1)} = \mathbf{P}_{ij} \cdot \mathbf{P}_{ij}^{(n-1)},\end{aligned} \tag{86}$$

where $(\star)$ uses Bayes' rule $\mathbb{P}(A \mid B) \cdot \mathbb{P}(B) = \mathbb{P}(A, B)$ with events $A, B \in \mathcal{A}$ such that $\mathbb{P}(B) \neq 0$. Hence, by mathematical induction (which is omitted here): $\mathbf{P}_{ij}^{(n)} = \mathbf{P}_{ij}^n$. ■

A generalization of the n-step transition matrix is given by the Chapman-Kolmogorov equation (Ross, 2019).

Theorem 4.2. (The Chapman-Kolmogorov Equation)
$\mathbf{P}^{(m+n)} = \mathbf{P}^m \cdot \mathbf{P}^n$, $\forall$ *time points m, n such that $m + n \leq N$.*

Proof: Pick two arbitrary states $i, j \in S$ and two arbitrary time points $m, n \in \mathbb{N}$ such that $m + n \leq N$.

$$\begin{aligned}\mathbf{P}_{ij}^{(m+n)} &= \mathbb{P}(X_{m+n} = j \mid X_0 = i) = \sum_{k=1}^{N} \mathbb{P}(X_{m+n} = j \mid X_n = k) \cdot \mathbb{P}(X_n = k \mid X_0 = i) \\ &= \sum_{k=1}^{N} \mathbf{P}_{ik}^{(m)} \cdot \mathbf{P}_{kj}^{(n)} = \mathbf{P}_{ij}^m \cdot \mathbf{P}_{ij}^n.\end{aligned} \tag{87}$$

■

Now, we have all ingredients to determine a general formula for the n-step distribution $\boldsymbol{\mu}_n$ of the Markov chain.

Theorem 4.3. *The PMF of the discrete-time homogenous Markov chain at time n is given by:*

$$\boldsymbol{\mu}_n = \lambda \mathbf{P}^n \in \mathbb{R}^{1\times N}, \tag{88}$$

where $\boldsymbol{\mu}_n(i) := (\boldsymbol{\mu}_n)_i = \mathbb{P}(X_n = i)$, $\forall$ *states* $i \in S$.

Proof: Equation (88) is obtained by reiteration: $\boldsymbol{\mu}_n = \boldsymbol{\mu}_{n-1}\mathbf{P} = (\boldsymbol{\mu}_{n-2}\mathbf{P})\mathbf{P} = \lambda\boldsymbol{\mu}_1\mathbf{P}^{n-1}$ $= \lambda\mathbf{P}^n$. ■

Example 4.2: Jumping Frog (cont.)
We can calculate the distribution of the Markov chain after one period $\boldsymbol{\mu}_1$:

$$\boldsymbol{\mu}_1 = \begin{pmatrix} 0.5 & 0.5 \end{pmatrix} \begin{bmatrix} 0.6 & 0.4 \\ 0.8 & 0.2 \end{bmatrix} = \begin{pmatrix} 0.7 & 0.3 \end{pmatrix} = \begin{pmatrix} \mathbb{P}(X_1 = 1) & \mathbb{P}(X_1 = 2) \end{pmatrix}. \tag{89}$$

Moreover, we can also calculate the distribution of the Markov chain for a collection of periods. Figure 19 shows the probabilities for being in state 1 and 2 for an increasing sequence of time points. Observe that after just six periods, the probability that the frog sits on plant 1 or 2 converges to $\frac{2}{3}$ and $\frac{1}{3}$, respectively. This distribution $\begin{pmatrix} \frac{2}{3} & \frac{1}{3} \end{pmatrix}$ is called the stationary distribution of the Markov chain. ▶

```
Lambda = [0.5 0.5];
P = [0.6 0.4; 0.8 0.2];

%%% Calculate the distrubtion mu_n for multiple n
tic
n = 20;  % Time steps
% Store the distribution vectors per column in a matrix
Mu = zeros(2,n);
for i = 1:n
     Mu(1:2,i) = Lambda*(P^i);  % Do not use '.*'
end
toc

%%% Plotting the distrubtion mu_n for multiple n
% Probability that the frog sits on plant 1
plot(0:n,[Lambda(1) Mu(1,:)],'-bo','MarkerFaceColor','magenta', ...
     'LineWidth',1,'MarkerSize',10)
hold on;
title('I\kern-0.15em P$(X_n = 1)$','FontSize',34,'Interpreter','latex')
xlabel('Time Period $n$','FontSize',21,'Interpreter','latex')
hold off;

% Probability that the frog sits on plant 1
plot(0:n,[Lambda(2) Mu(2,:)],'-bo','MarkerFaceColor','magenta', ...
     'LineWidth',1,'MarkerSize',10)
hold on;
title('I\kern-0.15em P$(X_n = 2)$','FontSize',34,'Interpreter','latex')
xlabel('Time Period $n$','FontSize',21,'Interpreter','latex')
hold off;
```

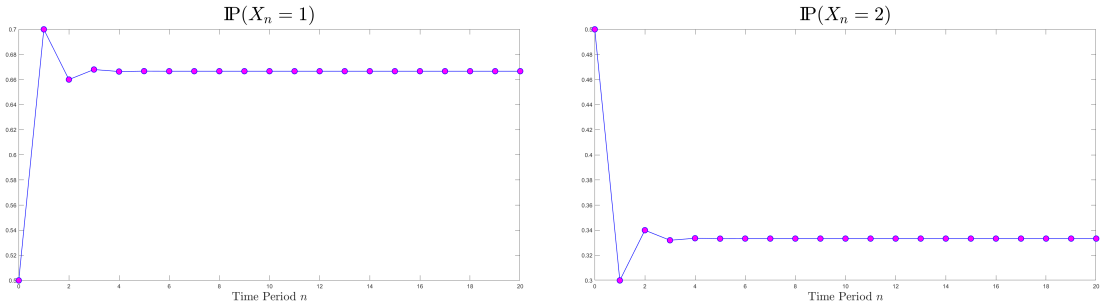

Figure 19

The formal definition of a stationary distribution for a discrete-time homogenous Markov chain is as follows (Guilhoto, 2017).

Definition 4.3. *A probability distribution $\boldsymbol{\pi}$ is called **stationary** in relation to a transition matrix $\mathbf{P}$, if the following equation holds:*

$$\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P} \quad \iff \quad \boldsymbol{\pi}(i) = \sum_{j \in S} \boldsymbol{\pi}(j)\mathbf{P}_{ji}, \ \forall i \in S, \tag{90}$$

where $\boldsymbol{\pi}(i) := \boldsymbol{\pi}_i \in \mathbb{R}$ denotes the i^{th} entry of the row vector $\boldsymbol{\pi}$.

This important type of distribution, which do not change after post-multiplication by the transition matrix, plays a crucial role in MCMC methods. In particular, we want to create a Markov chain with some specific stationary distribution in MCMC simulation. The fact that evolving further does not change the distribution implies the terminology: 'stationary'.

The detailed balance equation gives stronger condition than stationarity, but it enables us to verify whether or not a distribution is stationary.

Proposition 4.2. *Let $\mathbf{P}$ be the transition matrix of a discrete-time homogenous Markov chain. If a distribution $\boldsymbol{\pi}$ satisfies the detailed balance equation:*

$$\boldsymbol{\pi}_i\mathbf{P}_{ij} = \boldsymbol{\pi}_j\mathbf{P}_{ji}, \quad \forall i, j \in S, \tag{91}$$

then $\boldsymbol{\pi}$ is a stationary distribution of the Markov chain.

Proof: Suppose $\boldsymbol{\pi}$ satisfies the detailed balance equation (91). Pick two arbitrary states $i, j \in S$. Then:

$$\sum_{i \in S} \boldsymbol{\pi}_i\mathbf{P}_{ij} = \sum_{i \in S} \boldsymbol{\pi}_j\mathbf{P}_{ji} = \boldsymbol{\pi}_j \sum_{i \in S} \mathbf{P}_{ji} = \boldsymbol{\pi}_j \cdot 1 = \boldsymbol{\pi}_j. \tag{92}$$

Hence, $\boldsymbol{\pi}$ satisfies equation (90) and therefore is a stationary distribution. ■

Example 4.3: Jumping Frog (cont.)
We can check analytically that $\boldsymbol{\pi} = \begin{pmatrix} \frac{2}{3} & \frac{1}{3} \end{pmatrix}$ is indeed a stationary distribution:

$$\boldsymbol{\pi}\mathbf{P} = \begin{pmatrix} \frac{2}{3} & \frac{1}{3} \end{pmatrix} \begin{bmatrix} 0.6 & 0.4 \\ 0.8 & 0.2 \end{bmatrix} = \begin{pmatrix} \frac{6+4}{15} & \frac{4+1}{15} \end{pmatrix} = \begin{pmatrix} \frac{2}{3} & \frac{1}{3} \end{pmatrix} = \boldsymbol{\pi}. \tag{93}$$

Equivalently, it can be verified that all three distinct variantions of the detailed balance equation are satisfied. For example, $\boldsymbol{\pi}_1\mathbf{P}_{12} = \frac{2}{3} \cdot \frac{2}{5} = \frac{4}{15} = \frac{1}{3} \cdot \frac{4}{5} = \boldsymbol{\pi}_2\mathbf{P}_{21}$. ▶

Properties of Discrete-Time Markov Chains

In MCMC simulation, we need Markov chains that have the following characteristics: posses a stationary distribution, ensured convergence to stationary distribution, and uniqueness of their stationary distribution. In order to understand whether and why the above characteristic are satisfied, some class properties of states need to be introduced: irreducibility, recurrence, transience, and aperiodicity.

First, a Markov chain is irreducible if each state can be reached from any other state. This concept can be defined in multiple equivalent ways, we present two of them. We say that two states $i, j \in S$ communicate, i.e. $i \leftrightarrow j$, if there exists $n, m \in \mathbb{N}$ such that $p_{ij}^{(n)} = p_{ij}^{n} > 0$ and $p_{ji}^{(m)} = p_{ji}^{m} > 0$. The notion of communication is an equivalence relation. Furthermore, communication classes $C_k \subset S$ are pairwise distinct which partition the state space S: $S = \bigcup_{k \in S} C_k$. If state i is in two communication classes C_k and C_l, then $C_k = C_l$.

Definition 4.4. *A discrete-time homogenous Markov chain is irreducible, if one of the following equivalent conditions hold:*

- *For every pair of states $i, j \in S$, there exists a time point $n \in \mathbb{N}$ such that $p_{ij}^{n} = (\mathbf{P}^n)_{ij} > 0$.*
- *All states communicate, i.e. S is a communication class.*

When a Markov chain is reducible, we are not able to sample from the whole support. And the irreducibility property is also needed to ensure existence and uniqueness of the stationary distribution.

Second, a state $i \in S$ of Markov chain is recurrent if there is a positive probability of returning to state i any time in the future. Let $\tau_i := \inf_{n \in \mathbb{N}} \{t \geq 1 : X_n = i\}$ be the return time to some state i.

Definition 4.5. *A state i of a discrete-time homogenous Markov chain is recurrent, if one of the following equivalent conditions hold:*

- $\mathbb{P}(\textit{Markov chain returns to state } i \mid \textit{it starts in } i) = \mathbb{P}(\tau_i < \infty \mid X_0 = i) = 1$;
- $\mathbb{E}[\#\ \textit{visits in state } i] = \infty$;
- $\sum_{n=1}^{\infty} p_{ii}^{n} = \infty$.

If a state is not recurrent, then it is called transient. Furthermore, state i is positive recurrent if: $\mathbb{E}[\tau_i \mid X_0 = i] < \infty$, i.e. the expected waiting time for returning is finite. If a Markov chain is recurrent but not positive recurrent, it is called null recurrent.

Ross (2019) proves the equivalence of the above conditions. Recurrence is also a class property, because if state i is recurrent and $i \leftrightarrow j$ then state j is also recurrent. Hence, one state in some communication class being recurrent is a sufficient condition for recurrence of the whole class. In fact, all states in a finite irreducible Markov chain are positive recurrent.

Irreducibility and recurrence constitute a sufficient condition for the existence and uniqueness of the stationary distribution of a Markov chain (Akyildiz, 2022):

Theorem 4.4. $\mathbf{P}$ *has a unique stationary distribution* $\boldsymbol{\pi} \iff \mathbf{P}$ *is irreducible.*

Proof: Pick two arbitrary stationary distributions $\boldsymbol{\pi}$ and $\boldsymbol{\pi'}$. The idea is to use stationarity and irreducibility, which implies positive recurrence, to show that $\max_j \boldsymbol{\pi}_j / \boldsymbol{\pi}'_j = 1$. See Levin and Peres (2017) for all the details. ■

This result is noteworthy and insightful, but more conditions are needed to ensure actual convergence to the stationary distribution.

Third, a state $i \in S$ is periodic if the time between two visits of i contains some structure.

Definition 4.6. *A state i of a discrete-time homogenous Markov chain has period σ, if:* $\sigma = \gcd\{n \in \mathbb{N} \mid \mathbb{P}(X_n = i \mid X_0 = i) > 0\} = \gcd\{n \in \mathbb{N} \mid p_{ii}^n > 0\}$.

A brief summary of the notion of greatest common divider of two positive integers $a, b \in \mathbb{N}$: $\gcd(a, b)$ is the largest $r \in \mathbb{N}$ such that $r \mid a$ and $r \mid b$, where $r \mid a$ means that there exist an integer $q \in \mathbb{Z}$ such that $\frac{a}{r} = q$. And two integers $a, b \in \mathbb{N}$ are called co-prime, if $\gcd(a, b) = 1$. For example, $\gcd(12, 15) = 3$ and $\gcd(2, 7) = 1$. Moreover, a state i is aperiodic if all integers in $\{n \in \mathbb{N} \mid p_{ii}^n > 0\}$ are co-prime.

Lastly, a Markov chain is ergodic if all its states are positive recurrent and aperiodic.

Example 4.4: Jumping Frog (cont.)
First, this Markov chain is clearly irreducible, because $p_{12} = 0.4 > 0$ and $p_{21} = 0.8 > 0$. Hence, the state space $S = \{1, 2\}$ is a community class. Second, state 1 is recurrent, because: $\sum_{n=1}^{\infty} p_{11}^n = \sum_{n=1}^{5} p_{11}^n + \sum_{k=6}^{\infty} \frac{2}{3} = \infty$. And state 2 is also recurrent, since 1 and 2 communicate. Third, state 1 is aperiodic, because $p_{11}^2 > 0$ and $p_{11}^3 > 0$ while $\gcd(2, 3) = 1$ since they are both prime. Analogously, it can be shown that state 2 is also aperiodic. Lastly, this Markov chain is also positive recurrent, because $\#S < \infty$ and both states are recurrent. Hence, it is ergodic. ▶

So far, we have seen that discrete-time homogenous Markov chains have a stationary distribution if either Definition 4.3 or Proposition 4.2 hold. And by Theorem 4.4, it is the unique stationary distribution if the transition matrix is irreducible and positive recurrent. Lastly, we need a situation in which convergence to the unique stationary distribution is ensured.

Theorem 4.5. *Let $\{X_n : n \in \mathbb{N}\}$ be a discrete-time homogenous Markov chain with stationarity distribution $\boldsymbol{\pi}$.*
If the Markov chain is irreducible and ergodic, then $\boldsymbol{\mu}_n$ converges to the stationary distribution $\boldsymbol{\pi}$ as n grows arbitrarily large. This holds for any chosen initial state X_0.
In other words, the limit of the long-run proportions exist:

$$\lim_{n \to \infty} p_{ij}^n = \boldsymbol{\pi}_i, \quad \forall \boldsymbol{\lambda}, \forall i, j \in S. \tag{94}$$

Proof: Let d_{TV} be the total variation metric which measures the largest difference between two probabilities assigned to a single event $A \in \mathcal{A}$, i.e. it is a measure of distance between two PDFs. Guilhoto (2017) shows that the distance between $\boldsymbol{\mu}_n$ and $\boldsymbol{\pi}$ for any initial state X_0 is bounded above:

$$\max_{x_0 \in \Omega} d(\boldsymbol{\mu}_n, \boldsymbol{\pi}) \leq C \cdot \alpha^n, \tag{95}$$

for some constants $C \in \mathbb{R}_{>0}$ and $\alpha \in (0, 1)$. Since $\lim_{n \to \infty} C \cdot \alpha^n = 0$, convergence to the stationary distribution $\boldsymbol{\pi}$ is ensured. ■

Finally, we have a situation in which Monte Carlo integration and Markov chains are connected. The Ergodic Theorem is the most useful property of Markov chains needed for Markov chain Monte Carlo methods regarding integration. It is the Markov chain variant of the (Strong) Law of Large Numbers. The idea of the Ergodic Theorem for Markov chains is: 'time averages equal space averages'.

Theorem 4.6. (Ergodic Theorem) *Let $g : A \subseteq \mathbb{R}^d \to \mathbb{R}$ be the function of interest in equation (2). If $\{X_n\}$ is an irreducible Markov chain with stationary distribution $\boldsymbol{\pi}$, over state space A, then for any starting distribution $\boldsymbol{\lambda}$:*

$$\mathbb{P}\left(\lim_{n\to\infty} \frac{1}{n}\sum_{i=0}^{n-1} g(X_i) = \mathbb{E}_{\boldsymbol{\pi}}[g]\right) = 1 \quad \iff \quad \frac{1}{n}\sum_{i=0}^{n-1} g(X_i) \xrightarrow{a.s.} \mathbb{E}_{\boldsymbol{\pi}}[g]\,. \tag{96}$$

Proof: See Levin and Peres (2017).

Hence, the Ergodic Theorem informs us about the very important link between stationary distributions and expectations of functions whose domain is the state space of the Markov chain. This gives us a method to approximate the following integral by the Monte Carlo technique without directly drawing from f_X:

$$\int_A f(x)\,dx = \mathbb{E}_{f_X}[g(X)]. \tag{97}$$

Observe the peculiar vector notation in the expectation' subscript in equation (96). This vector notation denotes the distribution of the Markov chain. Hence, the expectation becomes a summation: $\mathbb{E}_{\boldsymbol{\pi}}[g] = \sum_{x\in A} g(x)\cdot\boldsymbol{\pi}(x) = \sum_{x\in A} g(x)\cdot\boldsymbol{\pi}_x \in \mathbb{R}$. However, since we only consider discrete-time Markov chains, g is only evaluated at a discretization of its domain A. The idea is in some sense similar to approximating an integral by a Riemann sum where the area is approximated by fitting in a finite number of vertical rectangles. But discretization is often a computationally expensive procedure (Guilhoto, 2017). The alternative approach uses continuous-time Markov chains to draw from a continuous PDF g by MCMC which consequently enables us to approximate the integral in equation (97).

4.4 Continuous-Time Markov Chain

All the concept above of discrete-time Markov chains in section 4.3 can be generalized to continuous-time with time set $T := \mathbb{R} \geq 0$. Especially, the existence of and convergence to a unique stationary distribution and the Ergodic Theorem remain practically unchanged. The continuous-time variant will be explicitly considered only for the most significant concepts.

In the continuous-time case, the probability of occurrence of a particular state i is zero. Therefore, the discrete-time Markov property $\mathbb{P}(X_{n+1} = i_{n+1} \mid X_n = i_n, ..., X_0 = i_0) = \mathbb{P}(X_{n+1} = i_{n+1} \mid X_n = i_n)$ changes to continuous-time Markov property:

$$\mathbb{P}(X_n \in M \mid X_n = i_n, ..., X_0 = i_0) = \mathbb{P}(X_n \in M \mid X_n = i_n) =: K(M \mid i_n), \tag{98}$$

for any $\mathbb{P}$-measurable set M. The continuous-time variant of the transition matrix $P = (p_{ij})_{i,j} = (\mathbb{P}(X_1 = j \mid X_0 = i))_{i,j}$ is the transition kernel K. K defines a PDF over the next-state, given the current state. Essentially, a transition probability changes from a PMF to a PDF. Hence, assuming time-homogeneity, it must satisfy (Akyildiz, 2022):

$$\int_S K(x \mid i)\,dx := \int_S K(\{x\} \mid i)\,dx = 1, \forall n \quad \& \quad K(\cdot \mid \cdot) \geq 0. \tag{99}$$

Observe the change in state space, a countable or uncountable infinite state space S instead of a finite discrete state space. For example, $S = \mathbb{R}$ in case of a normal transition kernel $K(M \mid x) \overset{d}{=} \mathcal{N}(x, \sigma^2)$.

Next, let λ denote the initial PDF. Stationarity is also an essential concept for MCMC. Recall that a PMF $\boldsymbol{\pi}$ of a discrete-time Markov chain is stationary if it satisfies $\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P}$. This condition is equivalent to $\boldsymbol{\pi}(i) = \sum_{j\in S\subset\mathbb{N}} \boldsymbol{\pi}(j)\mathbf{P}_{ji}, \forall i \in S$. We want to generalize this notion to a PDF K for a continuous-time Markov chain.

Definition 4.7. *A PDF π is stationary in relation to a transition kernel K, if the following equation holds:*

$$\pi(x) = \int_S K(x \mid y) \cdot \pi(y)\, dy \tag{100}$$

Observe the similarities with stationarity in the discrete-time case, while using the fact that an integral over a subset of the natural numbers $\mathbb{N}$ becomes a summation. A PDF π being stationary means that the kernel K operating on π yields the same PDF π. In fact, matrix multiplication and integration are both linear operators. Hence, stationarity implies that π is invariant under transformation, where the transformation is integration against the kernel in this case.

Moreover, the existence of a stationary PDF imposes the preliminary irreducibility constraint on K. This entails that, for any initial value $X(0) \sim \lambda$, the sequence $\{X(t) : t \in T\}$ has a positive probability of eventually reaching any region of the state space S. A sufficient condition for irreducibility of the continuous-time Markov chain is $K(x, a) > 0$, $\forall a \in S$ (Robert and Casella, 2010).

Analogous to the discrete-time case, PDF π is stationary if it satisfies the detailed balance equation.

Proposition 4.3. *Let K be the transition kernel of a continuous-time homogenous Markov chain. If a PDF π satisfies the detailed balance equation:*

$$K(y \mid x)\pi(x) = K(x \mid y)\pi(y), \quad \forall x, y \in S, \tag{101}$$

then π is a stationary distribution of the Markov chain.

Proof: Integrating both sides of equation (101) with respect to y yields:

$$\begin{aligned} \int_S K(y \mid x)\pi(x)\, dy &= \int_S K(x \mid y)\pi(y)\, dy \\ \pi(x) \cdot \int_S K(y \mid x)\, dy = \pi(x) &= \int_S K(x \mid y)\pi(y)\, dy \end{aligned} \tag{102}$$

■

Furthermore, the convergence theorem also holds for continuous-time Markov chains implying that it convergences to a unique stationary PDF when the Markov chain is irreducible and ergodic. Moreover, there are only minor changes in the Ergodic Theorem for continuous-time.

Theorem 4.7. (Ergodic Theorem: Continuous-Time) *Let $g : A \subseteq \mathbb{R}^d \to \mathbb{R}$ be the function of interest in equation (2). If $\{X_n\}$ is an irreducible continuous-time Markov chain with stationary PDF π, over state space A, then for any starting PDF λ:*

$$\mathbb{P}\left(\lim_{n\to\infty} \frac{1}{n}\sum_{i=0}^{n-1} g(X_i) = \mathbb{E}_\pi\left[g\right]\right) = 1 \quad \iff \quad \frac{1}{n}\sum_{i=0}^{n-1} g(X_i) \xrightarrow{a.s.} \mathbb{E}_\pi\left[g\right], \tag{103}$$

where $\mathbb{E}_\pi\left[g\right] = \int_A g(x) \cdot \pi(x)\, dx$.

So, equation (97) can be approximated by using the Ergodic Theorem when we can draw from the stationary PDF $\pi := f_{\boldsymbol{X}}$.

4.5 MCMC Algorithms

Markov chain Monte Carlo (MCMC) refers to a powerful collection of algorithms that enable us to directly sample from complicated PDFs, such as $f_{\boldsymbol{X}}$, using Markov chains. In other words, it is a strategy for generating samples $X_1, ..., X_n$ while exploring the state space S using a Markov chain mechanism. MCMC is used when it is difficult or impossible to draw from $f_{\boldsymbol{X}}$ directly, but $f_{\boldsymbol{X}}$ is known up to its normalization constant (Andrieu, de Freitas, Doucet, and Jordan, 2003).

The working principle of MCMC is as follows. Given a target PDF $f_{\boldsymbol{X}}$, we build a Markov transition kernel K with stationary distribution $\pi := f_{\boldsymbol{X}}$. Then, we generate a continuous-time Markov chain $\{X_i\}_{i \in S}$ using this transition kernel K so that the limiting distribution of $\{X(t)\}_{t \in T}$ is $f_{\boldsymbol{X}}$. Hence, integrals, like $\int_A \phi(\boldsymbol{x})\, d\boldsymbol{x}$, can be approximated according to the Ergodic Theorem by $\mathbb{E}_\pi[g]$ (Robert and Casella, 2010).

The principle for discrete-time Markov chains is analogous to this, the only difference is the use of a transition matrix instead of a transition kernel. This method can be used when the aim is to sample from a PMF, or when an integral is approximated by a discretization of the integration domain.

But how do we derive such transition kernels K that are universal in the sense that they are theoretically valid for every PDF $f_{\boldsymbol{X}}$? Fortunately, there exist methods that tackle this problem. The Metropolis-Hastings algorithm is one of the simplest such methods.

First, let us see how the Metropolis-Hastings algorithm works for sampling from a univariate PDF f_X.

Metropolis-Hastings Algorithm: Pseudo-Code for **Univariate PDFs**

Input: The number of iterations n and an initial value x_0
for $i = 0, ..., n-1$ % `In MATLAB, the first index takes value 1`
    Generate $U \sim \text{Unif}(0, 1)$
    Generate proposal $Y \sim p(y \mid x_i)$
    Calculate acceptance ratio $r \equiv r(x_i, y) := \frac{f_X(y) \cdot p(x_i|y)}{f_X(x_i) \cdot p(y|x_i)}$
    if $U \leq r$ **then**
        $X_{i+1} = y$ % `Accept`
    else
        $X_{i+1} = x_i$ % `Reject`
    end
end

The main principle of this algorithm is to choose propose a new state Y for the Markov chain $\{X_i\}_{i \in T}$, when in the current state x_i, based upon some acceptance probability. Performing this procedure for $n-1$ iterations results in a Metropolis-Hastings Markov chain that will converge to the target PDF f_X under some conditions. Hence, a subset of converged valued of the Metropolis-Hastings Markov chain $X_i : i \in T$ is a set of samples from f_X. Therefore, Metropolis-Hastings algorithm, and in fact every MCMC algorithm, is a indirect sampling method.

Observe the irregular-seeming usage of lowercase and capital notation. X_i denotes a random variable which is a function, and x_i denotes a realization of the function X_i. After time step i, the realization x_i of the random variable X_i is known. But the value that the Markov chain will take at time $i+1$ is not known at time i. In this algorithm, X_{i+1} will either equal the

deterministic value x_i or a realization of some proposal random variable Y. Additionally, PDFs evaluated at a deterministic value, like $f_X(y)$ and $p(y \mid x_i)$, are points in $\mathbb{R}^d$, where $d = 1$ in the univariate case. However, $f_X(Y)$ and $p(Y \mid X_i)$ are random variables.

Moreover, observe that the Metropolis-Hastings Markov chain draws values from a continuous state space S at discrete-time steps. This is because computers cannot continuously draw numbers. Therefore, we essentially use discrete approximations of continuous distributions. Hence, $T = \mathbb{N}$ and the transition probabilities are determined by the transition kernel of this algorithm.

Let us delve deeper into the steps that the Metropolis algorithm defines. First, the number of iterations n and the initial value of the Metropolis-Hastings Markov chain must be given. Second, a for-loop is created. In each loop, a standard uniform random variable U is generated together with a proposal random variable Y which is distributed according to some chosen proposal PDF p. Notice that $p(\cdot \mid x_i)$ is a conditional PDF which implies that the parameter space of p depends on the previous value of the Metropolis-Hastings Markov chain x_i. A common choice of the proposal PDF is a normal PDF $\mathcal{N}(x_i, \sigma^2)$, because it is symmetric and computers can relatively easily draw from it. Using the target PDF f_X and a chosen proposal PDF p, the acceptance ratio r is computed. Based upon this acceptance rate r, the generated proposal Y is accepted or rejected. If the proposal is accepted, then the value of the Metropolis-Hastings Markov chain at the next time step $i + 1$ equals the realization of the proposal Y. Otherwise, it remains x_i. Therefore, it can occur that the Metropolis-Hastings Markov chain is in the same state for a number of successive iterations. Since the sequence of x_i's is dependent on its previous state with some probability α, it a Markov chain.

This also indicates a difference with rejection sampling, where rejected values are discarded instead of kept. Furthermore, this also implies that the resulting samples from f_X are clearly dependent, because the decision for the next state's value depends on the current state x_i. This correlation shows the main difference with regular generation of samples, used in standard Monte Carlo integration, that yields independent samples.

The decision of accepting a proposal when $U \leq r$ might seem arbitrary. This condition, $U \leq r$, is only a practical trick for computational efficiency. It is theoretically equivalent to accepting the proposal with acceptance probability $\alpha \equiv \alpha(x_i, y) := \min\left\{1, \frac{f_X(y) \cdot p(x_i \mid y)}{f_X(x_i) \cdot p(y \mid x_i)}\right\}$. This equivalence holds because of the form of the standard uniform CDF:

$$U \sim \text{Unif}(0,1) \quad \Longrightarrow \quad \mathbb{P}(U \leq c) = \begin{cases} 0, & \text{if } c < 0 \\ c, & \text{if } c \in [0,1] \\ 1, & \text{if } c > 1 \end{cases}. \tag{104}$$

If $r \geq 1$, then $\alpha = 1$ because U only takes values in $[0, 1]$. Or if $r < 1$, then $\alpha = r < 1$ because $\mathbb{P}(U \leq r) = r$. The case where $f_X(y) = 0$ is omitted by one of the assumptions we will see in a moment. In addition, $\alpha := 0$ when $f_X(y) = f_X(x_i) = 0$ to avoid theoretical difficulties.

Furthermore, the choice of the acceptance rate r might also seem arbitrary. The ratio $\frac{f_X(y)}{f_X(x_i)}$ implies that regions of higher density are preferred to regions with lower density. This will become clearer in Example 4.5. Moreover, the ratio $\frac{p(x_i \mid y)}{p(y \mid x_i)}$ ensures that the algorithm compensates for asymmetries of the proposal PDF p. Without this ratio that compensates for asymmetries, the Markov chain will deviate from the target PDF f_X unless the proposal PDF p is symmetric. If the proposal PDF p is symmetric, then $\frac{p(x_i \mid y)}{p(y \mid x_i)} = 1$. For example, this is the case for $p(y \mid x_i) := \mathcal{N}(x_i, \sigma^2)$. A MCMC algorithm that only considers symmetric proposal PDFs $p(y \mid x_i) = p(x_i \mid y)$ is called the Metropolis algorithm. This is the original algorithm developed by Metropolis et al. (1953). Almost two decades later, a generalization that accounts for asymmetry by inserting the ratio $\frac{p(x_i \mid y)}{p(y \mid x_i)}$ was developed by Hastings (1970).

Other well-known basic MCMC algorithms are presented in Table 1 that shows the main differences between these algorithms (Robert and Casella, 2010).

Algorithm	**Proposal PDF** p	**Acceptance Probability** α	**Key Feature**
Metropolis	$p(y \mid x_i) = p(x_i \mid y)$	$\min\left\{1, \frac{f_X(y)}{f_X(x_i)}\right\}$	Symmetric proposal PDF
Metropolis-Hastings	$p(y \mid x_i)$	$\min\left\{1, \frac{f_X(y)\cdot p(x_i \mid y)}{f_X(x_i)\cdot p(y \mid x_i)}\right\}$	Arbitrary proposal PDF
Independent M-H	$p(y \mid x_i) = p(y)$	$\min\left\{1, \frac{f_X(y)\cdot p(x_i)}{f_X(x_i)\cdot p(y)}\right\}$	Proposal PDF independent of current state x_i
Random Walk M-H	$p(y \mid x_i) = x_i + \epsilon_i$	$\min\left\{1, \frac{f_X(y)}{f_X(x_i)}\right\}$	Random walk as proposal PDF
Gibbs Sampling	$f_X(y_i \mid \{y_j : 0 \leq j \leq i\} \cup \{x_j : j > i\})$	1	Directly samples from conditional target PDF f_X

Table 1: Comparison of MCMC Algorithms

All the algorithms in Table 1, except Metropolis-Hastings (M-H) algorithm, are a special case of the Metropolis-Hastings algorithm. In particular, even though Gibbs sampling belongs to a different class of MCMC algorithms, Robert and Casella (2004) show that it is a special case of the Metropolis-Hastings algorithm. Gibbs sampling was developed by Geman and Geman (1984). It is especially useful when the aim is to sample from a multi-dimensional target PDF $f_{\boldsymbol{X}}$, or when an arbitrary state space S, possibly different from $\mathbb{R}^d$, is considered (Rubinstein, 2016). Notice that the Gibbs sampling proposal PDF given in Table 1 is for the case $d = 1$. In addition, Table 1 indicates that the Metropolis-Hastings algorithm imposes relatively minimal requirements on the proposal PDF p.

However, we have not shown that the Metropolis-Hastings algorithm works. It is sufficient to show that f_X is a stationary PDF of the Metropolis-Hastings Markov chain. Proposition 4.3 informs us that this is the case if f_X satisfies the detailed balance equation.

Theorem 4.8. *The Metropolis-Hastings algorithm satisfies the detailed balance equation with respect to* f_X*:*

$$K(y \mid x) f_X(x) = K(x \mid y) f_X(y), \quad \forall x, y \in S, \tag{105}$$

where K *is the transition kernel of the Metropolis-Hastings algorithm.*

Proof: Robert and Casella (2004) show that the Metropolis-Hastings transition kernel K is defined as follows:

$$K(y \mid x) = \alpha(x, y) \cdot p(y \mid x) + [1 - h(x)] \cdot \delta_x(y), \quad \forall x, y \in S, \tag{106}$$

where δ_x is the Dirac delta function of x and $h(x) := \int_X \alpha(x, y) \cdot p(y \mid x)\, dy$. Next, Akyildiz (2022) shows that this kernel equals the detailed balance equation with respect to f_X. ■ Theorem 4.8 also holds in the case of a multi-dimensional target PDF $f_{\boldsymbol{X}}$.

Moreover, even though the Metropolis-Hastings algorithm imposes relatively minimal requirements on the target PDF f_X and the proposal PDF p, there are still some assumptions made. Fortunately, most assumptions are implicit and mainly used for theoretical purposes. All the assumption are summarized below.

MH.1.	x_0 is such that $f_X(x_0) > 0$
MH.2.	f_X & p known up to a constant independent of x_i
MH.3.	$Supp(f_X)$ is connected, i.e. it does not admit a partition (Sutherland, 2009)
MH.4.	$Supp(f_X) \subseteq \bigcup_{z \in Supp(f_X)} Supp(p(\cdot \mid z))$
MH.5.	f_X bounded
MH.6.	f_X is positive on every compact subset of $Supp(f_X)$
MH.7.	For $\epsilon > 0$, $p(y \mid x) > \epsilon$, whenever $\lvert x - y \rvert < \delta$

In practice, only the first condition MH.1. is checked. Condition MH.1. implies that $f_X(X_i) > 0$, $\forall i > 0$, which ensures that α is well-defined. Next, condition MH.2. assumes that f_X and p are known up to their normalization constant. The normalization constants have no influence because they disappear in the acceptance ratio. Moreover, conditions MH.3. and MH.4. are necessary to ensure that the entire support of f_X can be visited by the Metropolis-Hastings Markov chain. Lastly, conditions MH.5-7. are necessary to ensure that the stationary distribution f_X satisfies the conditions in the Ergodic Theorem 4.7. In particular, MH.7. implies irreducibility and aperiodicity of the Metropolis-Hastings Markov chain.

The assumptions above also reveal a condition on the universality property of the Metropolis-Hastings algorithm. This universality property refers to the fact that any conditional proposal PDF p on $Supp(f_X)$ can lead to the simulation of an arbitrary target PDF f_X. Assumptions MH.5-7. imply that if both $p(y \mid x_i)$ allows for moves in a neighbourhood of x_i with bounded radius (by δ) and f_X is such that $\alpha(x_i, y) > 0$ in this neighbourhood, then any part of the support of f_X can be visited with a sufficiently large number of steps. This results in the fact that p needs to rarely draw points in the high density regions of f_X in order for the universality property to hold (Robert and Casella, 2004).

In general, Monte Carlo integration with MCMC algorithms does not achieve the $\mathcal{O}(n^{-1/2})$ convergence rate, but it can still be shown to be consistent. Consistency refers to the fact that the estimator of the integral asymptotically converges to its true value (Henning, Osborne, and Kersting, 2022). Moreover, stationarity and ergodicity of f_X do not necessarily make a good MCMC algorithm, because consistency is a relatively weak property. Fortunately, there exists a diverse range of MCMC methods.

Chib and Greenberg (1969) give an expression for the variance of the Monte Carlo integral estimator using Metropolis-Hastings algorithm. This variance turns out to be larger than the variance of the Monte Carlo integral estimate using independent sampling. Analytically, it is difficult to express the variance of the Metropolis-Hastings algorithm sample. However, we do want to know whether the Metropolis-Hastings Markov chain has converged to f_X, and if the samples seem to be independent. Determining whether or not a Markov chain has converged can be done by analysing trace plots in the univariate case. However, this is too difficult for multi-dimensional PDFs. Fortunately, there exist methods, such as the Gelman-Rubin statistic, that can test for convergence of Markov chains without the necessity of plotting. Furthermore, testing whether a Metropolis-Hastings sample is independent can be done by using autocorrelation plots. These methods will be demonstrated in Example 4.5.

Example 4.5: Bimodal Univariate PDF
Suppose we want to draw samples form the following unnormalized bimodal PDF:

$$\tilde{f_X} : \mathbb{R} \to \mathbb{R}_{>0} \text{ defined by } \tilde{f}_X(x) := 0.3e^{-0.2x^2} + 0.7e^{-0.2(x-10)^2}. \tag{107}$$

Bimodal refers to the fact that the PDF possesses two modes, i.e. two local maxima. See Figure 20 for the graph of $\tilde{f}_X$. Figure 20 also shows the normalized PDF $f_X \propto \tilde{f}_X$.

We cannot directly sample from the unnormalized PDF $\tilde{f}_X$. Therefore, we are going to use the Metropolis-Hastings algorithm to create a Markov chain whose collection of states resembles a sample from $\tilde{f}_X$.

```
%% Bimodal PDF with Normal Proposal PDF
rng('default')
n = 100000;
x_Values = -10:0.1:20;
Tilde_f_X = 0.3*exp(-0.2*x_Values.^2) + 0.7*exp(-0.2*(x_Values-10).^2);
% Calculating Normalization Constant using Monte Carlo integration
Uniform_Realization = random('Uniform',-5,15, [1 n]);
Tilde_f_X_Uniform = 0.3*exp(-0.2*Uniform_Realization.^2) + 0.7*exp( ...
    -0.2*(Uniform_Realization-10).^2);
NormalizationConstant = mean(Tilde_f_X_Uniform);

%%% Plotting Tilde_f_X and f_X
plot(x_Values,Tilde_f_X,'Color','magenta','LineWidth',2);
hold on;
plot(x_Values,NormalizationConstant*Tilde_f_X, ...
    'Color','green','LineWidth',2);

title('Graphs of $f_X$ and $\tilde{f}_X$', ...
    'FontSize',34,'Interpreter','latex')
xlabel('x','FontSize',21,'Interpreter','latex')
legend({'$\tilde{f}_X$','$f_X$'}, ...
    'Location','northeast','FontSize',30,'Interpreter','latex');
hold off;
```

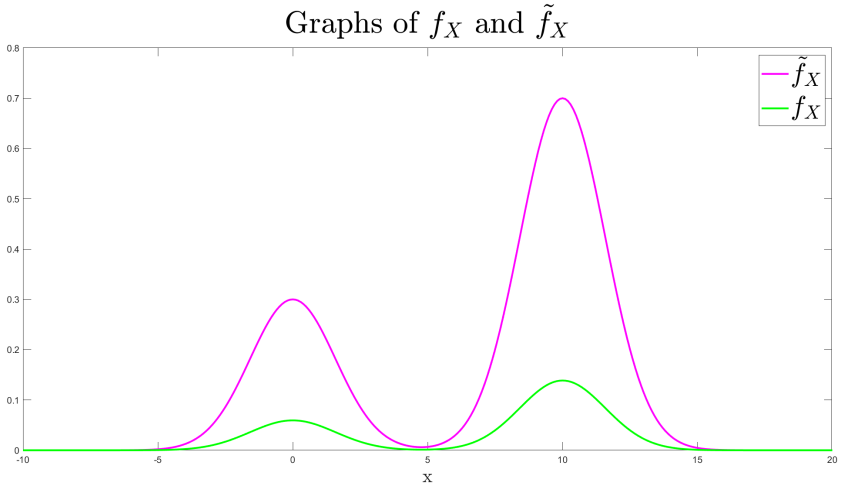

Figure 20

Before implementing the Metropolis-Hastings algorithm, let us explain in a visual manner how the proposal mechanism of this algorithm works. Suppose the Metropolis-Hastings Markov chain is in some state $x_i = 14$, at an arbitrary point in time $i \in \{0, ..., n-1\}$. Let us

use one of the most common proposal PDFs: $p(y \mid x_i) := \mathcal{N}(x_i, \sigma^2)$, where $\sigma^2 = 1$ is this case. Notice that the Metropolis-Hastings algorithm equals the Metropolis algorithm in this case, because $\frac{p(x_i|y)}{p(y|x_i)} = 1$ by symmetry of p. This proposal PDF p is depicted by the blue graph in Figure 21. Next, the algorithm generates a proposal y from the proposal PDF p, i.e. y is a realization of $Y \sim p(y \mid x_i)$.

Figure 21 shows two possible scenarios for y. First, the left-hand side plot displays the scenario where $y < x_i$. Since y and x_i lie to the right of the global maximum of $\tilde{f}_X$, we observe that $\tilde{f}_X(y) > \tilde{f}_X(x_i)$. Hence, the acceptance rate r becomes $\frac{\tilde{f}_X(y)}{\tilde{f}_X(x_i)} > 1$. Therefore, the proposal will be accepted with probability one because regions of higher densities are preferred. Second, the right-hand side plot displays the scenario where $y > x_i$. In this case, $\tilde{f}_X(y) < \tilde{f}_X(x_i)$. Therefore, this proposal is accepted with probability $\alpha = r = \frac{\tilde{f}_X(y)}{\tilde{f}_X(x_i)} < 1$.

```
%% Visual Explanation of Metropolis-Hastings Algorithm Mechanism
% For example, suppose x_i = 14 and p(y | x_i) = N(x_i, 1)
x_i = 14;
PD_Normal = makedist('Normal','mu',x_t,'sigma',1);
PDF_Normal = pdf(PD_Normal,x_Values);

%%% Plotting Tilde_f_X and the proposal PDF p
plot(x_Values,Tilde_f_X,'Color','magenta','LineWidth',2);
hold on;
plot(x_Values,PDF_Normal,'LineWidth',2,'Color','blue')
Grey = [0.7 0.7 0.7];
plot([x_i x_i],[0 max(PDF_Normal)],'--','LineWidth',2,'Color',Grey);
text(x_i-0.25,-0.02,'$x_t$','FontSize',21,'Interpreter','latex')
scatter(x_i,max(PDF_Normal),'LineWidth',2, ...
    'MarkerEdgeColor',Grey,'MarkerFaceColor',Grey)

% Proposal Y, 2 cases: 12 and 15.5
y = 15.5;
plot([y y],[0 PDF_Normal(256)],'--','LineWidth',2,'Color','green');
text(y-0.2,-0.02,'$y$','FontSize',21,'Interpreter','latex')
scatter(y,PDF_Normal(256),'LineWidth',2, ...
    'MarkerEdgeColor','green','MarkerFaceColor','green')
% y = 13 corresponds to x_Values' index (13 - -10)*10 + 1 = 231
% y = 15 corresponds to x_Values' index (15.5 - -10)*10 + 1 = 256
title(['Proposal Mechanism with $y= $ ' num2str(y)], ...
    'FontSize',34,'Interpreter','latex')
xlabel('x','FontSize',21,'Interpreter','latex')
legend({'$\tilde{f}_X$','$p(y \mid x_i) = \mathcal{N}(x_i,1)$'}, ...
    'Location','northwest','FontSize',30,'Interpreter','latex');
hold off;
```

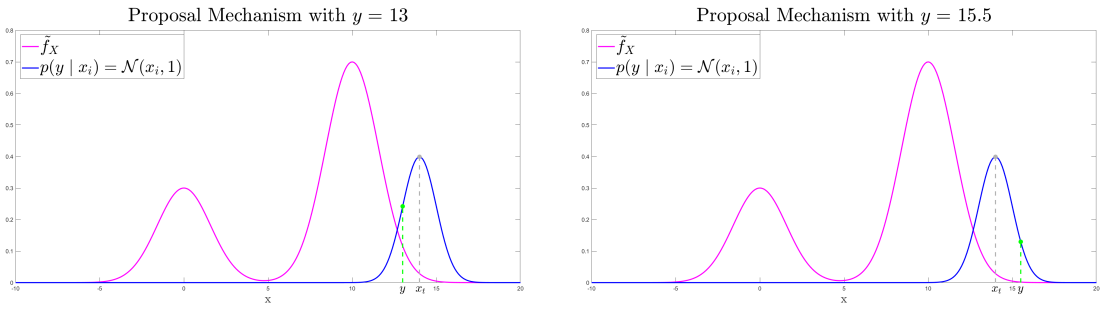

Figure 21

Now, let us implement the Metropolis(-Hastings) algorithm. For computational efficiency, the algorithm is programmed as a function in a separate function-file. A function cannot be placed inside the main `MATLAB` file of interest, but we can call this function from a function-file which is saved as the name of the function of interest. `MH_Algorithm` in this case. The input arguments of this function are the number of iterations `n`, the initial value `x_0` and the deterministic parameter `Sigma2` of our normal proposal PDF.

```
function [X] = MH_Algorithm (n,x_0,Sigma2)
     % A function that accepts inputs and returns output X
     % x_0 = Initial state of Markov chain
     % Sigma2 = Deterministic parameter of proposal PDF p
     X = zeros(1,n);  % Create vector to store the Markov chain
     X(1) = x_0;  % In MATLAB, the first index is defined by a 1

     for i = 1:(n-1)
          % For X_1,...,X_n. Note: x_0 corresponds to index 1 of X
          U = random('Uniform',0,1);
          % Proposal PDF p(y | x_i)
          p_y_xi = random('Normal',X(i),Sigma2);
          p_xi_y = random('Normal',p_y_xi,Sigma2);  % p(x_i | y)

          Tilde_f_X_xi = 0.3*exp(-0.2*X(i)^2) + 0.7*exp( ...
               -0.2*(X(i)-10)^2);  % f_X(x_i)
          Tilde_f_X_y = 0.3*exp(-0.2*p_y_xi^2) + 0.7*exp( ...
               -0.2*(p_y_xi-10)^2);  % f_X(y)

          AcceptanceRate = (Tilde_f_X_y*p_xi_y)/(Tilde_f_X_xi*p_y_xi);

          if U <= AcceptanceRate
               X(i+1) = p_y_xi;
          else
               X(i+1) = X(i);
          end
     end
end
```

Next, we can plot the output of the Metropolis-Hastings algorithm for different input arguments. Figure 22 depicts the plots of two different Metropolis-Hastings Markov chains. These plots showing the state values of the Metropolis-Hastings Markov chain over the iterations i are called trace plots. Observe that sometimes the value of X_i is constant for some successive iterations. This is because a proposal is rejected for these iterations which implies

that the Markov chain stays in the same state. Furthermore, observe that the jumps between different states is larger in the right-hand side plot than in the left-hand side plot. This is because of the usage of different variance parameters: $\sigma^2 = 10$ versus $\sigma^2 = 1$. It is probably beneficial to use a proposal PDF with a higher variance parameter, because then it is more likely that most of the state space is visited in some finite number of iterations. However, the variance must not be too large, since frequent visits of high density regions becomes less likely in that case.

```
%% Metropolis-Hastings Algorithm
% The MetropolisHastingsAlgorithm() function is defined in a separate
% function file.

%%% Plotting a Metropolis-Hastings Markov Chain
rng('default')
n = 210; x_0 = 5; Sigma2 = 1;
MC = MH_Algorithm(n,x_0,Sigma2);

plot(1:length(MC),MC,'Color','blue')
hold on;
xlim tight  % Ensure the correct x-axis
title(['M-H Markov Chain with $n = $ ' num2str(n) '$, \ x_0 = $ ' ...
     num2str(x_0) '$\ \ \& \ \ p(y \mid x_i) = \mathcal{N}(x_i,$' ...
     num2str(Sigma2) '$)$'],'FontSize',27,'Interpreter','latex')
xlabel('Iteration $i$','FontSize',21,'Interpreter','latex')
ylabel('$X_i$','FontSize',21,'Interpreter','latex')
hold off;
```

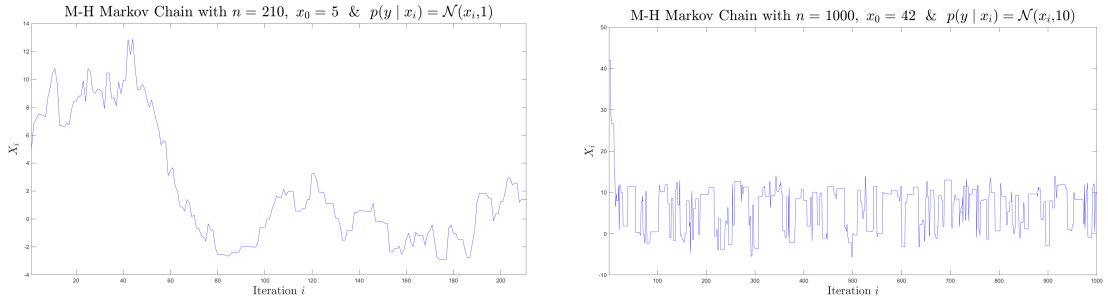

Figure 22

The Metropolis-Hastings Markov chain can be plotted as a histogram enabling us to verify whether or not it follows the same distribution as the target PDF $\tilde{f}_X$. Figure 23 shows the histograms, together with the graph of $\tilde{f}_X$, of the Metropolis-Hastings Markov chains with the same input arguments as in Figure 22. Observe that even though a much worse initial state value is chosen in the right-hand side plot, the histogram in the left-hand side plot is a much worse approximation of $\tilde{f}_X$ than the histogram in the right-hand side plot. This is solely because a proposal PDF with a small variance does not enable the algorithm to visit the entire state space with appropriate frequencies.

```
%%% Plotting the histogram and theoretical PDF
histogram(MC,'Normalization','pdf','FaceColor','cyan')
hold on;
plot(x_Values,NormalizationConstant*Tilde_f_X, ...
```

```
    'LineWidth',2,'Color','magenta')

title(['M-H: $\tilde{f}_X$ with $n = $ ' num2str(n) '$, \ x_0 = $ ' ...
num2str(x_0) '$\ \ \& \ \ p(y \mid x_i) = \mathcal{N}(x_i,$' ...
num2str(Sigma2) '$)$'],'FontSize',27,'Interpreter','latex')
xlabel('$x$','FontSize',21,'Interpreter','latex')
legend({'MCMC $\tilde{f}_X$','True PDF'}, ...
    'Location','northeast','FontSize',30,'Interpreter','latex');
hold off;
```

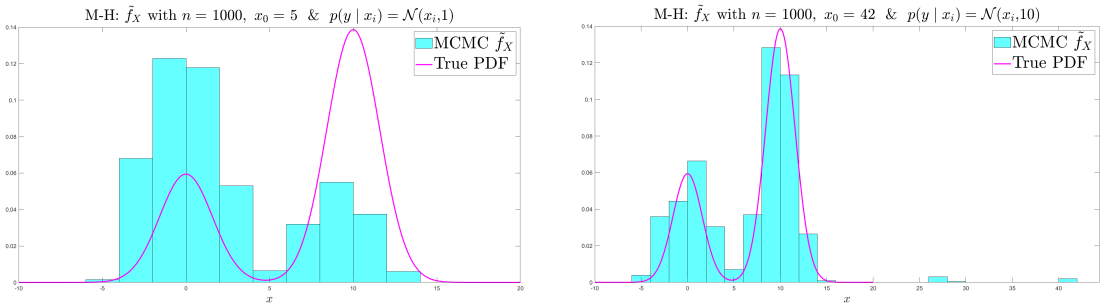

Figure 23

Figure 24 shows that increasing the number of iterations n improves the sample approximation. Observe that the left-hand side plot is still not a good approximation of $\tilde{f}_X$. But the right-hand side plot looks like a good approximation with only a relatively small number of iterations for computers nowadays. In this case, a 'good' approximation refers to a relatively small difference in surface between the histogram and the graph of $\tilde{f}_X$.

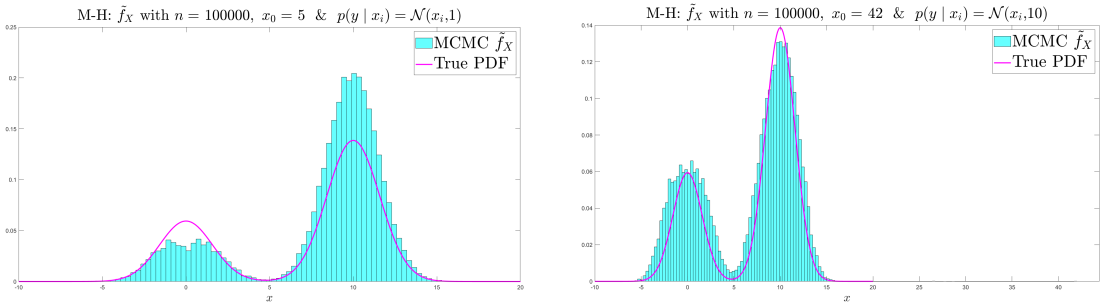

Figure 24

So far, we have encountered examples of the algorithm's sensitivity to different choices of deterministic parameters of the proposal PDF p. But how does an initial state value x_0 influence the Metropolis-Hastings Markov chain. Figure 25 shows the trace plots of Metropolis-Hastings Markov chains with different initial values $x_0 \in \{5, 42, 100, 121\}$ and $\sigma^2 = 10$. Observe that the number of iterations needed for the Markov chain to be in the interval of main interest, $[-5, 15]$, increases when the distance of x_0 from this interval increases. In particular, a starting value of $x_0 = 121$ with $\sigma^2 = 10$ does not visit $[-5, 15]$ in the first $n = 1000$ iterations. Every proposal gets rejected in this case. Observe that assumption

MH.1. is violated in this case, because $\tilde{f}_X(121) = 0$. However, increasing the variance to $\sigma^2 = 21$ does result in convergence to $[-5, 15]$ after around 100 iterations.

In this univariate case, convergence of the Metropolis-Markov chain can be assessed by looking at the relevant trace plot. But for multi-dimensional proposal PDFs p this is not easy to verify visually. Fortunately, there exist statistics that can assist in determining whether or not a Markov chain has converged.

```
%% Trace Plot of Multiple Metropolis-Hastings Markov Chains
TracePlot_Length = 1000;
rng('default')

plot(1:TracePlot_Length+1,MH_Algorithm(TracePlot_Length,5,10), ...
     'Color','green','LineWidth',1.5)
hold on;
plot(1:TracePlot_Length+1,MH_Algorithm(TracePlot_Length,42,10), ...
     'Color','blue','LineWidth',1.5)
plot(1:TracePlot_Length+1,MH_Algorithm(TracePlot_Length,100,10), ...
     'Color','cyan','LineWidth',1.5)
plot(1:TracePlot_Length+1,MH_Algorithm(TracePlot_Length,121,10), ...
     'Color','red','LineWidth',1.5)
Orange = [240 100 10]/256;
plot(1:TracePlot_Length+1,MH_Algorithm(TracePlot_Length,121,21), ...
     'Color',Orange,'LineWidth',1.5)
xlim tight  % Ensure the correct x-axis
title(['Trace Plot of Markov Chains with $n = $ ' ...
num2str(TracePlot_Length)],'FontSize',27,'Interpreter','latex')
xlabel('Iteration $i$','FontSize',21,'Interpreter','latex')
ylabel('$X_i$','FontSize',21,'Interpreter','latex')
legend({'$x_0 = 5, \ \sigma^2 = 10$','$x_0 = 42, \ \sigma^2 = 10$', ...
     '$x_0 = 100, \ \sigma^2 = 10$','$x_0 = 121, \ \sigma^2 = 10$', ...
     '$x_0 = 121, \ \sigma^2 = 21$'}, ...
     'Location','northeast','FontSize',30,'Interpreter','latex');
hold off;
```

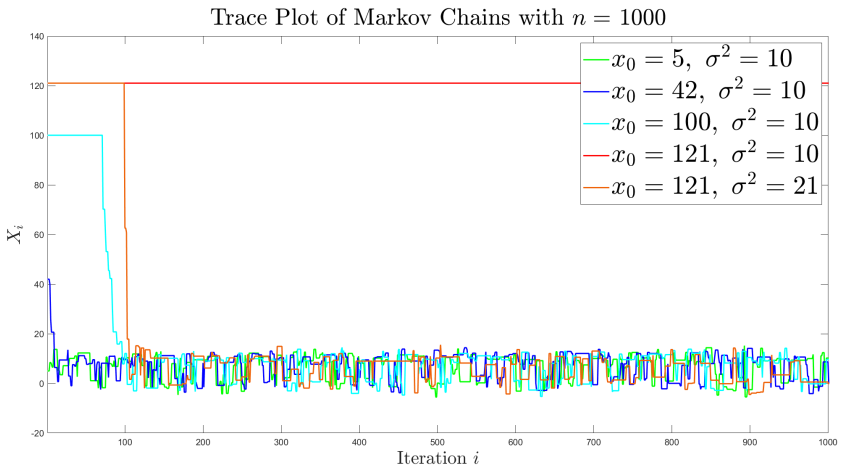

Figure 25

The Gelman-Rubin diagnostic statistic, developed by Gelman and Rubin (1992), is one of the most popular methods to determine convergence of a Monte Carlo Markov chain. Gelman (2004) suggests that the Monte Carlo Markov chain has converged after n iterations when the Gelman-Rubin statistic $|R| \leq 1.1$. This threshold has been adopted widely by practitioners (Vats and Knudson, 2021).

The Gelman-Rubin statistic defines convergence of MCMC when they have 'forgotten' their initial states and present similar behaviour. It presumes approximate convergence when the variance between different Markov chains is smaller than the variance of each individual Markov chain. Let m denote the number of considered different Markov chains. In addition, let $i \in \{1, ...n\}$ denote the time index of each Markov chain, and let $j \in \{1, ..., m\}$ denote the m different Markov chains. The sample variance between Markov chains is defined as (Martínez, 2022):

$$B := \frac{n}{m-1} \sum_{j=1}^{m} (\bar{X}_j - \bar{X}) = \frac{n}{m-1} \sum_{j=1}^{m} \left[\frac{1}{n} \sum_{i=1}^{n} X_{i,j} - \frac{1}{m} \sum_{j=1}^{m} \left(\frac{1}{n} \sum_{i=1}^{n} X_{i,j} \right) \right], \tag{108}$$

where $\bar{X}_j := \frac{1}{n} \sum_{i=1}^{n} X_{i,j}$ is the average of a particular Markov chain, and $\bar{X} := \frac{1}{m} \sum_{j=1}^{m} \bar{X}_j$ is the average of all Markov chain averages. Moreover, sample variance within Markov chains is defined as:

$$W := \frac{1}{m} \sum_{j=1}^{m} s_j^2 = \frac{1}{m} \sum_{j=1}^{m} \left[\frac{1}{n-1} \sum_{i=1}^{n} (X_{i,j} - \bar{X}_j)^2 \right], \tag{109}$$

where s_j^2 denotes the sample variance of a single Markov chain. Hence, the Gelman-Rubin statistic is defined as:

$$R := \frac{n-1}{n} + \frac{1}{n} \cdot \frac{B}{W} \xrightarrow{n \to \infty} 1. \tag{110}$$

```
%% Gelman-Rubin Convergence Statistic
% i = 1,...n denotes the time index of a single M-H Markov chain
% j = 1,...,m labelling of different chains
rng('default')
n = 100000;  % Number of iterations
x_0_mValues = [5 42 100 121];  % Initial state values
m = length(x_0_mValues);  % Number of different Markov chains considerd
Sigma2 = 10;

% Store the M-H Markov chains, where each column corresponds to one of
% the m initial values & the rows correspond to the time index
VBS_Chains = zeros(1,m);  % Variance between sequences
VWS_Chains = zeros(1,m);  % Variance within sequences

for j = 1:4
     Exp_Sequence = mean( MH_Algorithm(n,x_0_mValues(j),Sigma2) );
     VBS_Chains(:,j) = Exp_Sequence;  % 1/n * sum_{i=1}^n X_{i,j}

     Var_Sequence = var( MH_Algorithm(n,x_0_mValues(j),Sigma2) );
     % 1/(n-1) * sum_{i=1}^n (X_{i,j} - mean(X_i))^2
     VWS_Chains(:,j) = Var_Sequence;
end

% Variance between sequences
B = n/(m-1) * sum(VBS_Chains - mean(VBS_Chains));
```

```
W = mean(VWS_Chains);  % Variance within sequences
R = 1 - 1/n + B/(n*W);  % Gelman-Rubin Convergence Statistic
```

The code above calculates the Gelman-Rubin statistic for the Metropolis-Hastings Markov chains in Figure 25. This yields `R = 0.9999` which implies that the Markov chains have converged after `n = 100,000` iterations.

Optionally, a burn-in period can be implemented. This refers to the practice of deleting some iterations at the beginning of a Metropolis-Hastings Markov chain. Sometimes, this is also called a transient phase. The primary reason for applying this is a preference to exclude starting values that are not in the neighbourhood of the centre of the stationary distribution. For example, in Figure 25, we might want to exclude the first 120 samples from the Markov chain with $x_0 = 100$. The implementation of a burn-in period is mostly harmless, which is probably why the practice persists. However, it is not necessary and it does not produce any signifiant unbiasedness. The created bias is of order $\mathcal{O}(n^{-1})$, and hence is often neglected for sufficiently large n (Brooks et al., 2011).

Moreover, we also want the samples from $\tilde{f}_X$ to be independent. The proposal mechanism in the Metropolis-Hastings algorithm introduces a dependency. We can test for possible correlation by means of an autocorrelation function (ACF). ACF denotes a correlation with a lagged value of itself, hence the term 'auto', as a function of delay.

Definition 4.8. *The autocovariance function (ACVF) of a stochastic process* $\{X(t) : t \in \mathbb{N}\}$ *at lag* k *is defined as:*

$$\gamma_{k,i} \equiv \gamma_X(k,i) := Cov(X_i, X_{i+k}) = \mathbb{E}\left[(X_i - \mathbb{E}[X_i]) \cdot (X_{i+k} - \mathbb{E}[X_{i+k}])\right]. \tag{111}$$

The autocorrelation function (ACF) $\rho_k : \mathbb{N} \to \mathbb{R}$ *of a stochastic process* $\{X(t) : t \in \mathbb{N}\}$ *at lag* k *is defined as:*

$$\rho_{k,i} \equiv \rho_X(k,i) := Corr(X_i, X_{i+k}) = \frac{\gamma_{k,i}}{Var(X_i)}. \tag{112}$$

The ACVF and variance of X_i can be calculated using their sample estimators. Figure 26 shows the ACF plots for the same Metropolis-Hastings Markov chains as in Figure 22 and 23. Observe that the left-hand side plot, with a smaller variance than the right-hand side plot, depicts a large autocorrelation which indicates a correlated sample. This is not surprising since successive states are close to each other when the variance is small. The right-hand side plot informs us that corresponding Markov chain shows independent behaviour when there is a minimal time difference of twenty units.

Alternative methods for testing independence of the Metropolis-Hastings sample are thinning and usage of effective sample size.

```
%% Autocorrelation Plot
rng('default')
n = 100000; x_0 = 5; Sigma2 = 10;
MC = MH_Algorithm(n,x_0,Sigma2);

MaxLag = 100;  % Maximum number of lags
% Create vector to store autocorrelations
AutoCorrelation = zeros(1,MaxLag);

for k = 0:MaxLag
     Cov_Xk = mean( (MC(1:end-k) - mean(MC)).*(MC(k+1:end) - mean(MC)) );
     AutoCorrelation(k+1) = Cov_Xk/var(MC);
end
```

```
14
15 %%% Plotting the autocorrelation function
16 stem(0:MaxLag,AutoCorrelation, ...
17      'Marker','none','LineWidth',2,'Color','green')
18 hold on;
19 title(['ACF for $n = $ ' num2str(n) '$, \ x_0 = $ ' ...
20 num2str(x_0) '$\ \ \& \ \ \sigma^2 =$ ' num2str(Sigma2)], ...
21      'FontSize',27,'Interpreter','latex')
22 xlabel('Lag $k$','FontSize',21,'Interpreter','latex')
23 ylabel('$\rho_k$','FontSize',21,'Interpreter','latex')
24 hold off;
```

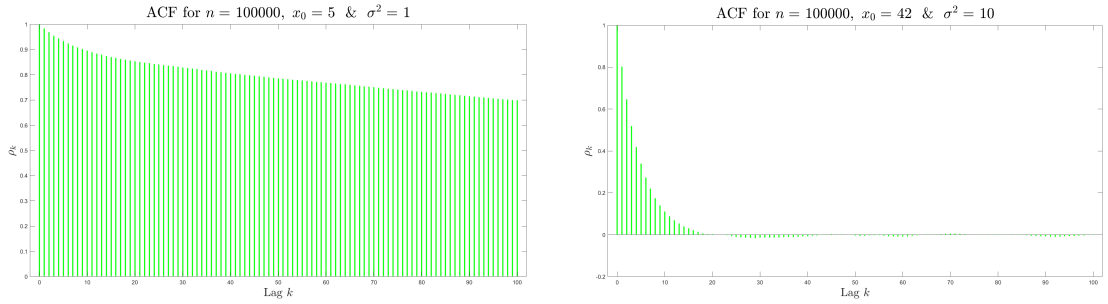

Figure 26

▶

In turns out that MCMC algorithms are relatively slow for sampling from low-dimensional PDFs. Fortunately, the Metropolis-Hastings algorithm can be generalized in relatively straightforward manner. Let $f_{\boldsymbol{X}} : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ be the d-dimensional PDF of interest which is known up to its normalization constant. Observe that $f_{\boldsymbol{X}}$ takes a realization $(x_1, ..., x_d)^T \in \mathbb{R}^d$ of a random vector $X = (X_1, ..., X_d)^T$ as input argument, and it takes on some scalar value $f_{\boldsymbol{X}}\left((x_1, ..., x_d)\right) \in \mathbb{R}$.

Metropolis-Hastings Algorithm: Pseudo-Code for **Multi-Dimensional PDFs**

1: Input: The number of iterations n and an initial value $\boldsymbol{x}_0 \in \mathbb{R}^d$
2: **for** $i = 0, ..., n-1$ % `In MATLAB, the first index takes value 1`
3: Generate $U \sim \text{Unif}(0, 1)$
4: Generate proposal $\boldsymbol{Y} \sim p(\boldsymbol{y} \mid \boldsymbol{x}_i)$
5: Calculate acceptance ratio $r \equiv r(\boldsymbol{x}_i, \boldsymbol{y}) := \frac{f_{\boldsymbol{X}}(\boldsymbol{y}) \cdot p(\boldsymbol{x}_i | \boldsymbol{y})}{f_{\boldsymbol{X}}(\boldsymbol{x}_i) \cdot p(\boldsymbol{y} | \boldsymbol{x}_i)} \in \mathbb{R}$
6: **if** $U \leq r$ **then**
7: $\boldsymbol{X}_{i+1} = \boldsymbol{y}$ % `Accept`
8: **else**
9: $\boldsymbol{X}_{i+1} = \boldsymbol{x}_i$ % `Reject`
10: **end**
11: **end**

Gibbs sampling can also be used to sample from a multi-dimensional target PDF $f_{\boldsymbol{X}}$, but we only consider the Metropolis-Hastings algorithms since it is the most general MCMC

algorithm. Depending on the problem, a special case of the Metropolis-Hastings algorithm can be used.

In econometrics, MCMC algorithms are not only applied to integration problems (Chib and Greenberg, 1969). For example, MCMC is used to efficiently sample from the posterior distribution of the parameters of the seemingly unrelated Bayesian regression model, enabling exact inference in small samples. In particular, Metropolis-Hastings algorithm can be used to estimate parameters in Bayesian regression models with autoregressive errors. Furthermore, MCMC algorithms can be useful to simulate the posterior distributions of model parameters when dealing with censored data problems by introducing latent variables. Moreover, Geweke, Koop, and van Dijk (2011) present other applications of MCMC algorithms in jump-diffusion and DSGE models.

4.6 Application

An application of Monte Carlo integration with a MCMC algorithm in econometrics is calculating the probability that a new response variable is a success by using the predictive posterior distribution of a Bayesian logistic regression. An example of this application is given below. However, before doing so, we first introduce Bayesian statistics and generalized linear model.

Example 4.6: Bayesian Logistic Regression
There exist two 'schools' of statistics: the traditional frequentist school and the Bayesian school. The traditional school of frequentist's statistics assumes that unknown parameters are constants which can be estimated from the data. Given a random sample $Z_1, ..., Z_n$ that follows some distribution p which depends on the parameter $\theta \in \Theta$, the unknown parameter θ is often estimated by maximum likelihood estimation. The likelihood function is denoted by $\mathcal{L}(\theta) := p(z_1, ..., z_n \mid \theta)$. Frequentist's statistics often uses asymptotic theory to infer, i.e. learn about, some parameters.

Second, Bayesian statistics assumes that unknown parameters are random variables themselves. A parameter θ has prior distribution with PDF $p(\theta)$, which is imposed by a Bayesian statistician. A prior distribution can be based upon prior beliefs. For example, in a regression model for modelling an exam's pass rate based upon the number of hours of preparation, there might be a prior belief that the regression coefficient corresponding to the study hours is not centred around zero. Alternatively, a convenient prior PDF is chosen when it's difficult to form a prior belief. The aim of Bayesian statistics is to infer the posterior distribution $p(\theta \mid z_1, ..., z_n)$. This is done by updating the prior distribution $p(\theta)$ to the posterior distribution $p(\theta \mid z_1, ..., z_n)$ using Bayes's rule:

$$p(\theta \mid z_1, ..., z_n) = \frac{p(\theta, z_1, ..., z_n)}{p(z_1, ..., z_n)} = \frac{p(z_1, ..., z_n \mid \theta) \cdot p(\theta)}{p(z_1, ..., z_n)} \propto p(z_1, ..., z_n \mid \theta) \cdot p(\theta), \tag{113}$$

where $p(z_1, ..., z_n) = \int_\Theta p(z_1, ..., z_n, \theta)\, d\theta$ is a normalization constant called the marginal likelihood, and Θ denotes the parameter space of θ. Equation (113) is sometimes stated to enhance intuition in the following manner:

$$\text{Posterior} = \frac{\text{Likelihood} \cdot \text{Prior}}{\text{Evidence}}. \tag{114}$$

Let us consider an example to clarify the difference between the two schools of statistics. Suppose that a potentially biased die is rolled n times. Let Z_i denote the random variable

such that $Z_i = 1$ if we roll a '6' in the i^{th} trial, and $Z_i = 0$ otherwise. We want infer the probability of throwing a '6' which we denote by $\theta \in \Theta$. A frequentist formulates θ as the proportion $\frac{1}{n}\sum_{i=1}^{n} Z_i$ when n becomes infinitely large. In contrast, a Bayesian approach would work as follows. Consider some conditional data:

$$Z_1, ..., Z_n \mid \theta \sim \text{Bern}(\theta). \tag{115}$$

The parameter θ is unknown, and the realizations of n die rolls $z_1, ..., z_n$ are observed. Next, a prior distribution is imposed. For instance:

$$\theta \sim \text{Beta}(a, b), \quad a, b \in \mathbb{R}_{>0}. \tag{116}$$

Analytically, it can be shown that updating our prior distribution to the posterior distribution yields (Hoff, 2009):

$$\theta \mid z_1, ..., z_n \sim \text{Beta}\left(a + \sum_{i=1} z_i, b + n - \sum_{i=1} z_i\right) \tag{117}$$

This is one of the most well-known Bayesian models called the Bernoulli-Beta model. Hence, according to our posterior belief, θ is a realization of the beta distribution in equation (117). Moreover, observe that frequentist's statistics yields point estimation while Bayesian statistics results in a posterior distribution.

In many complicated econometrics and statistics problems, there are no obvious non-Bayesian methods for estimation or inference. An example of such a situation could be when the aim is to compute the probability that a new categorical response is a success when considering a Bayesian linear regression model. Observe that the Bayesian approach can be applied to regression coefficients $(\beta_0, \beta_1, ..., \beta_{d-1})^T = \boldsymbol{\beta} \in \mathbb{R}^d$.

Suppose we have n observations of some categorical response variable Y_i, where $Y_i = 1$ if a success is observed and $Y_i = 0$ otherwise. We aim to model these categorical observations using some regressor variables $(X_1, ..., X_{d-1})^T$ that we observed. For example, the response variable Y_i could indicate whether it rains on a particular day, and the regressors of the previous day consist of the humidity level $\in [0, 100]$ and a categorical variable specifying whether it rains on that day. The observations of the regressor variables are collected in a design matrix:

$$\mathbf{X} := \begin{bmatrix} 1 & x_{i1} & \dots & x_{i(d-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1n} & \dots & x_{n(d-1)} \end{bmatrix}, \tag{118}$$

where the column vector of ones corresponds to the intercept coefficient β_0 of the model.

A classical normal regression model cannot be used in this case, because the necessary assumption that the response variable Y_i is normally distributed, i.e. $Y_i \not\sim \mathcal{N}(\cdot, \cdot)$, is violated. Instead we should use a generalized linear model (GLM). GLMs allow us to model categorical response variables because the binomial distribution, and hence also the Bernoulli distribution, belongs to the exponential family. Furthermore, GLMs enable us to model non-linear relationships between the response and regressors by means of a link function g (Dobson and Barnett, 2018). Table 2 shows the main differences between classical linear regressions and GLMs in frequentist's statistics.

Classical Normal Linear Regression	**GLM**
Independent $Y_i \sim \mathcal{N}(\mathbb{E}[Y_i], \sigma^2)$	Independent $Y_i \sim$ Exponential Family
$\mathbb{E}[Y_i] = \boldsymbol{x}_t^T \boldsymbol{\beta}$ linear in β_i	$g(\mathbb{E}[Y_i]) = \boldsymbol{x}_t^T \boldsymbol{\beta}$ linear in β_i
Set of parameters $\boldsymbol{\beta} \in \mathbb{R}^d$ and explanatory variables $X \in \mathbb{R}^{n \times d}$	

Table 2: Comparison of Classical Linear Regression and GLM in Frequentist's Statistics.

The function g is called a link function and it is assumed to be monotonic, continuous and bijective. Notice that the latter two assumptions ensure that the inverse of g exist. Furthermore, in GLM, the Exponential Family of distributions is commonly used, but it is not the only choice. The Exponential Family includes many well-known distributions such as the binomial, normal and gamma distribution.

In frequentist's statistics, the parameters of both a classical linear regression and a GLM can be solved by maximum likelihood estimation which is a point estimation. However, in this example, our goal is to model a Bayesian logistic regression where point estimates are irrelevant. Let π_i denote the probability that Y_i is a success, where π_i is our parameter of interest. Table 3 shows the main components of a Bayesian GLM.

Bayesian GLM
Likelihood (Data Generating Process): independent $Y_i \mid \pi_i \sim p(Y_i \mid \pi_i)$, where $p \in$ exponential family
Link function g such that $g(\mathbb{E}[Y_i \mid \pi_i]) = \boldsymbol{x}_i^T \boldsymbol{\beta}$, ensuring a linear relationship in β_i.
Prior Distribution: parameters $\boldsymbol{\beta} \sim p(\boldsymbol{\beta})$, imposing prior belief about predictors

Table 3: Comparison of Classical Linear Regression and GLM in Bayesian Statistics.

Since we have a categorical response $Y_i \in \{0, 1\}$, a Bernoulli probability model is a natural choice for the data:

$$Y_i \mid \pi_i \sim \text{Bern}(\pi_i) \quad \text{with} \quad \mathbb{E}[Y_i \mid \pi_i] = \pi_i. \tag{119}$$

Next, we want to define a link function such that the image of $g(\mathbb{E}[Y_i \mid \pi_i]) = g(\pi_i)$ is $\mathbb{R}$ because $\boldsymbol{x}_i^T \boldsymbol{\beta} \in \mathbb{R}$. A common choice for g in this case is a logit link function

$$g(\pi_i) = \log\left(\frac{\pi_i}{1 - \pi_i}\right) =: \log(\text{odds}_i) \in \mathbb{R}. \tag{120}$$

Notice that odds_i denotes the probability that the event $Y_i = 1$ happens relative to the probability that it does not happen. In addition, since odds_i take values in $[0, \infty)$, $\log(\text{odds}_i)$ takes values in $\mathbb{R}$. Using the logit link function in combination with categorical responses Y_i results in a well-known GLM called the logistic regression. Furthermore, the probabilities π_i are unknown and are of interest for inference.

Moreover, a prior distribution of $\boldsymbol{\beta}$ must be specified. Usually, this prior is based upon the observed data of the regressors X. However, since we do not use any real data in this example, we need to artificially create a dataset by drawing random numbers. For simplicity, we assume $d = 2$, but it can be extended to any $d \geq 2$. The dataset of regressors is generated in the following way:

$$X_{i1} := X_i \sim \mathcal{N}(5, 7). \tag{121}$$

Additionally, we impose the following prior distributions on the regression parameters:

$$\beta_0, \beta_1 \sim \mathcal{N}(0, 5^2). \tag{122}$$

Observe that the mean of the prior distribution is zero. This implies that the possibility that the parameter β_j, $j \in \{0, 1\}$, does not influence the response variable is included.

Having introduced the basics of Bayesian statistics, we are now ready to consider the Bayesian logistic regression model which takes the following form:

$$\begin{aligned} \text{Data}: \quad & Y_i \mid \beta_0, \beta_1 \overset{ind.}{\sim} \text{Bern}(\pi_i) \quad \text{with} \quad \log\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_0 + \beta_1 \cdot X_{i1}, \\ \text{Priors}: \quad & \beta_0, \beta_1 \overset{ind.}{\sim} \mathcal{N}(0, 5^2), \end{aligned} \tag{123}$$

where $i \in \{1, ..., n\}$ refers to an index of the dataset. Assume that the true parameter values are $\beta_0^{\text{TRUE}} := 4.2$ and $\beta_1^{\text{TRUE}} := 2.1$. However, this model is often transformed from the $\log(\text{odds}_i)$ scale to a more meaningful probability scale:

$$\pi_i = \frac{e^{\boldsymbol{X}_i^T \boldsymbol{\beta}}}{1 + e^{\boldsymbol{X}_i^T \boldsymbol{\beta}}} = \frac{1}{1 + e^{-\boldsymbol{X}_i^T \boldsymbol{\beta}}} = \frac{1}{1 + e^{-\beta_0 - \beta_1 \cdot X_i}} \in [0, 1], \tag{124}$$

where $\boldsymbol{X}_i^T = (1, X_i)^T$ for $d = 2$.

Since we are dealing with a Bayesian model, we want to derive the posterior distribution. In other words, we want to update our prior belief $p(\boldsymbol{\beta})$ about the parameters $\boldsymbol{\beta}$ to our posterior belief $p(\boldsymbol{\beta} \mid \boldsymbol{y})$ after having observed the data $\boldsymbol{y}$ of the categorical response variable Y. The procedure of updating is according to equation (113). Furthermore, the data of the regressors $\mathbf{X}$ is also known, hence we condition on it. Therefore, by using the facts that the prior distribution of $\boldsymbol{\beta}$ does not depend on $\mathbf{X}$ and that the components of $\boldsymbol{Y}$ and $\boldsymbol{\beta}$ are i.i.d., we obtain that the posterior distribution is proportional to $\tilde{p}(\boldsymbol{\beta} \mid \boldsymbol{y}, \mathbf{X})$:

$$\begin{aligned} p(\beta_0, \beta_1 \mid \boldsymbol{y}, \mathbf{X}) &= p(\boldsymbol{\beta} \mid \boldsymbol{y}, \mathbf{X}) \\ &\propto p(\boldsymbol{y} \mid \boldsymbol{\beta}, \mathbf{X}) \cdot p(\boldsymbol{\beta} \mid \mathbf{X}) \\ &= p(\boldsymbol{y} \mid \boldsymbol{\beta}, \mathbf{X}) \cdot p(\boldsymbol{\beta}) \\ &\overset{ind.}{=} \left[\prod_{i=1}^{n} (\pi_i)^{y_i} \cdot (1-\pi_i)^{1-y_i}\right] \cdot \left[\frac{1}{\sqrt{2\pi \cdot 5}} \cdot e^{-\frac{(\beta_0 - 0)^2}{2 \cdot 5^2}} \cdot \frac{1}{\sqrt{2\pi \cdot 5}} \cdot e^{-\frac{(\beta_1 - 0)^2}{2 \cdot 5^2}}\right] \\ &\propto \left[\prod_{i=1}^{n} (\pi_i)^{y_i} \cdot (1-\pi_i)^{1-y_i}\right] \cdot e^{\frac{-\beta_0^2 - \beta_1^2}{50}} \\ &= \left[\prod_{i=1}^{n} \left(\frac{1}{1 + e^{-\beta_0 - \beta_1 \cdot x_i}}\right)^{y_i} \cdot \left(\frac{1}{1 + e^{\beta_0 + \beta_1 \cdot x_i}}\right)^{1-y_i}\right] \cdot e^{\frac{-\beta_0^2 - \beta_1^2}{50}} \\ &=: \tilde{p}(\boldsymbol{\beta} \mid \boldsymbol{y}, \mathbf{X}). \end{aligned} \tag{125}$$

In Bayesian statistics, the predictive posterior distribution is the distribution of a new response variable conditional on new observed regressors. In this example, we aim to calculate the probability that a new response variable Y_{new} is a success, i.e. $Y_{\text{new}} = 1$. Let y_{new} denote the realization of a new response and let $\boldsymbol{x}_{\text{new}}$ denote the newly observed regressors. The probability of interest is denoted by $p(y_{\text{new}} = 1 \mid \boldsymbol{y}, \boldsymbol{x}_{\text{new}}, \mathbf{X})$, because it depends on the previously observed data, $\boldsymbol{y}$ and $\mathbf{X}$, and on the newly observed regressors $\boldsymbol{x}_{\text{new}}$. Furthermore, it is a realization of the predictive posterior distribution $p(y_{\text{new}} \mid \boldsymbol{y}, \boldsymbol{x}_{\text{new}}, \mathbf{X})$ which is a

marginalization of the distribution of y_{new} that integrates over the uncertainties β_0 and β_1. Thus, we want to calculate the following integral:

$$\begin{aligned} p(y_{\text{new}} = 1 \mid \boldsymbol{x}_{\text{new}}, \boldsymbol{y}, \mathbf{X}) &= \int_{\mathbb{R}} \int_{\mathbb{R}} \underbrace{p(y_{\text{new}} = 1 \mid \boldsymbol{x}_{\text{new}}, \beta_0, \beta_1)}_{\text{Likelihood of } y_{\text{new}}=1} \cdot \underbrace{p(\beta_0, \beta_1 \mid \boldsymbol{y}, \mathbf{X})}_{\text{Posterior of } \boldsymbol{\beta}} \, d\beta_0 \, d\beta_1 \\ &=: \int_{\mathbb{R}^2} g(\boldsymbol{\beta}) \cdot f_{\boldsymbol{\beta}}(\boldsymbol{\beta}) \, d\boldsymbol{\beta} \ \in [0,1] \subsetneq \mathbb{R}. \end{aligned} \tag{126}$$

Observe that the integral is over $\mathbb{R}^2$ because the parameter space for both parameters is $\Theta = \mathbb{R}$. Furthermore, the second line in equation (126) denotes the link between the predictive posterior distribution and the integral in equation (73). Notice that g is a function of $\boldsymbol{\beta}$ because $p(y_{\text{new}} \mid \boldsymbol{x}_{\text{new}}, \beta_0, \beta_1) \propto \frac{1}{p(\boldsymbol{\beta})} \cdot p(\boldsymbol{\beta} \mid y_{\text{new}}, \boldsymbol{x}_{\text{new}})$ which depends on $\boldsymbol{\beta}$. Additionally, the posterior distribution of $\boldsymbol{\beta}$ is used as the target PDF.

The integral in equation (126) corresponding to the predictive posterior distribution is analytically intractable, mainly because of the composition of non-linear terms and the existence of a normalization constant in the integrand. However, this integral can be approximated numerically. Monte Carlo integration in combination with a MCMC algorithm is an appropriate technique to approximate this integral, because the target PDF, which is the posterior distribution $p(\beta_0, \beta_1 \mid \boldsymbol{y}, \mathbf{X})$ in this case, is only known up to a normalization constant (See subsection 4.5).

We use the following plan of attack to approximate the integral in equation (126):

Step-by-Step Plan to Compute Predictive Posterior Probability

1. Generate synthetic data of regressors $X_i \sim \mathcal{N}(5, 7)$ to construct the design matrix
$$\mathbf{X} := \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix} \in \mathbb{R}^{n \times 2}. \tag{127}$$
2. Generate response variables $\boldsymbol{y} \in \mathbb{R}^n$ by drawing $Y_i \sim \text{Bern}(\pi_i)$, where the success probability $\pi_i = \left(1 + e^{-\boldsymbol{x}_i^T \boldsymbol{\beta}_i^{\text{TRUE}}}\right)^{-1} := \left(1 + e^{-\beta_0^{\text{TRUE}} - \beta_1^{\text{TRUE}} \cdot x_1}\right)^{-1}$ is calculated using $\mathbf{X}$ and the true parameter values $\beta_0^{\text{TRUE}} := 4.2$ and $\beta_1^{\text{TRUE}} := 2.1$.
3. Define a function for the unnormalized posterior distribution $\tilde{p}(\boldsymbol{\beta} \mid \boldsymbol{y}, \mathbf{X})$.
4. Draw n samples, $\{(\beta_{0,i}, \beta_{1,i})^T\}_{i=1}^n =: \{\boldsymbol{\beta}_i\}_{i=1}^n$, from $\tilde{p}(\boldsymbol{\beta} \mid \boldsymbol{y}, \mathbf{X})$ by using the Metropolis-Hastings algorithm. Use normal proposal PDFs $q(\cdot \mid \cdot)$.
5. Auditing the generated samples from the unnormalized posterior $\tilde{p}(\boldsymbol{\beta} \mid \boldsymbol{y}, \mathbf{X})$.
6. Use Ergodic Theorem 4.7 to approximate the integral in equation (126):
$$\begin{aligned} \frac{1}{n} \sum_{i=1}^{n-1} p(y_{\text{new}} = 1 \mid \boldsymbol{x}_{\text{new}}, \boldsymbol{\beta}_i) = \frac{1}{n} \sum_{i=1}^{n-1} \pi_{\text{new},i} &\xrightarrow{a.s.} \mathbb{E}_{p(\boldsymbol{\beta} \mid \boldsymbol{y}, \mathbf{X})}[\pi_{\text{new},i}] \\ &= p(y_{\text{new}} = 1 \mid \boldsymbol{x}_{\text{new}}, \boldsymbol{y}, \mathbf{X}), \end{aligned} \tag{128}$$
where $\pi_{\text{new},i} := \left(1 + e^{-\boldsymbol{x}_{\text{new}}^T \boldsymbol{\beta}_i}\right)^{-1} = \left(1 + e^{-\beta_{0,i} - \beta_{1,i} \cdot x_{\text{new}}}\right)^{-1}$ and $\boldsymbol{x}_{\text{new}} := (1, x_{\text{new}})^T$.
7. Visualize the empirical predictive probabilities of $p(y_{\text{new}} = 1 \mid \boldsymbol{x}_{\text{new}}, \boldsymbol{y}, \mathbf{X})$ for different x_{new}.

Observe that the probability that a new categorical response variable is a success,

$p(y_{\text{new}} = 1 \mid \boldsymbol{x}_{\text{new}}, \boldsymbol{\beta}_i)$, is the probability in equation (124) with the new regressor observations and some parameter values.

The fourth step in the plan above needs further clarification because the notation is adjusted in the Metropolis-Hastings algorithm to avoid confusion. First, since the parameters in the Bayesian linear regression (BLR) constitute the sample of interest, the relevant Markov chain is denoted by $\{\boldsymbol{\beta}_i\}_{i=1}^n$. Second, the proposal random variable changes from $\boldsymbol{Y}$ to $\boldsymbol{Z}$ because $\boldsymbol{Y}$ refers to the response variable. Lastly, the notation of the proposal PDF changes from $p(\cdot \mid \cdot)$ to $q(\cdot \mid \cdot)$. We choose a normal proposal PDF. Furthermore, observe that the number of iterations n equals the number of data points considered.

Metropolis-Hastings Algorithm: Pseudo-Code for **BLR** with $d = 2$

Input: The number of iterations n and an initial value $\boldsymbol{\beta}_0 := (\beta_{0,0}, \beta_{1,0})^T \in \mathbb{R}^2$
for $i = 0, ..., n-1$ % `In MATLAB, the first index takes value 1`
    Generate $U \sim \text{Unif}(0, 1)$
    Generate proposal $\boldsymbol{Z} \sim q(\boldsymbol{z} \mid \boldsymbol{\beta}_i) := \left(\mathcal{N}(\beta_{0,i}, \sigma^2),\ \mathcal{N}(\beta_{1,i}, \sigma^2)\right)^T$
    Calculate acceptance ratio $r \equiv r(\boldsymbol{\beta}_i, \boldsymbol{z}) := \frac{\tilde{p}(\boldsymbol{z} \mid \boldsymbol{y}, \mathbf{X})}{\tilde{p}(\boldsymbol{\beta}_i \mid \boldsymbol{y}, \mathbf{X})} \in \mathbb{R}$
    if $U \leq r$ **then**
        $\boldsymbol{\beta}_{i+1} = \boldsymbol{z}$ % `Accept`
    else
        $\boldsymbol{\beta}_{i+1} = \boldsymbol{\beta}_i$ % `Reject`
    end
end

Since the parameters β_0 and β_1 are assumed to be independent in equation (123), we generate both components of the proposal $\boldsymbol{Z} := (Z_0, Z_1)^T$ separately which yields a realization $\boldsymbol{z} := (z_0, z_1)^T$. In contrast, if β_0 and β_1 were not assumed to be independent, then the proposal $\boldsymbol{Z}$ is drawn from a multivariate normal distribution with some covariance matrix that has non-zero off-diagonal elements. Furthermore, since the normal distribution is symmetric, the acceptance ratio simplifies to:

$$r = \frac{\tilde{p}(\boldsymbol{z} \mid \boldsymbol{y}, \mathbf{X}) \cdot q(\boldsymbol{\beta}_i \mid \boldsymbol{z})}{\tilde{p}(\boldsymbol{\beta}_i \mid \boldsymbol{y}, \mathbf{X}) \cdot q(\boldsymbol{z} \mid \boldsymbol{\beta}_i)} = \frac{\tilde{p}(\boldsymbol{z} \mid \boldsymbol{y}, \mathbf{X})}{\tilde{p}(\boldsymbol{\beta}_i \mid \boldsymbol{y}, \mathbf{X})} \cdot 1. \tag{129}$$

We are now in a position to execute the seven steps for calculating and visualizing the predictive posterior probabilities of a success. The first two steps are relatively straightforward.

```
%% Step 1: Generate Regressor Data
rng('default')
n = 10000;  % Length of dataset considered
x = random('Normal',5,7,[n 1]);
X = [ones(n,1) x];  % Design matrix
```

```
%% Step 2: Generate Response Observations
Beta_True = [4.2; 2.1];  % Column vector

Pi = zeros(n,1);  % Store succes probabilities
for i = 1:n
     Pi(i) = ( 1 + exp( -X(i,:)*Beta_True ) )^(-1);  % \pi_i
```

```
     % Note: operator '*' is used instead of '.*' to compute the
     %   multiplication of a row vector times a column vector.
end

y = zeros(n,1);  % Store categorical response observations
for i = 1:n
     y(i) = random('Binomial',1,Pi(i));
     % Note: a binomial distribution with one trial equals a Bernoulli
     %   distribution.
end
```

In the third step, we define a `MATLAB` function that computes $\tilde{p}(\boldsymbol{\beta} \mid \boldsymbol{y}, \mathbf{X})$. This function is saved in a function file named `Posterior.m`.

```
%% Step 3: Posterior Function
function [Output] = Posterior (Beta_0,Beta_1,y,X)
     % A function that accepts inputs and returns Output
     % This is saved in a function file called 'Posterior.m'
     Prior_Distribution = exp( (-Beta_0^2 - Beta_1^2)/50 );  % p(Beta)

     n = length(y);
     Marginal_Likelihood = zeros(1,n);  % p(y | Beta, X)

     Pi = zeros(n,1);  % Store succes probabilities
     for i = 1:n
          Pi(i) = ( 1 + exp( -X(i,:)*[Beta_0; Beta_1] ) )^(-1);  % \pi_i
     end

     % Calculating the marginal likelihoods for all i
     for i = 1:n
          Marginal_Likelihood(i) = Pi(i)^(y(i)) * (1 - Pi(i))^(1-y(i));
     end

     Output = prod(Marginal_Likelihood) * Prior_Distribution;
end
```

In the fourth step, we define another function that executes the relevant Metropolis(-Hastings) algorithm. This function is saved in a function file named `MH_Algorithm_BLR.m`. In this case, the usage of a function is beneficial because several Metropolis(-Hastings) Markov chains are needed in step five. Furthermore, observe that the `Posterior` function defined in the third step is used twice in this new function.

```
%% Step 4: Metropolis-Hastings Algorithm for BLR
function [Beta] = MH_Algorithm_BLR (Beta_Initial,Sigma2,y,X)
     % A function that accepts inputs and returns output Beta
     % This is saved in a function file called 'Posterior.m'
     % Beta_Initial = Initial state of Markov chain (row vector)
     % Sigma2 = Deterministic parameter of proposal PDF q
     % y,X = Observed data needed to calculate the posterior

     n = length(y);
     Beta = zeros(n,2);  % Create vector to store the Markov chain
     Beta(1,:) = Beta_Initial;  % In MATLAB, the first index is defined by a 1

     %AcceptanceRate = zeros(n,1);
```

```
     %AR_Numerator = zeros(n,1);
     %AR_Denominator = zeros(n,1);


     for i = 1:(n-1)  % For Beta_1,...,Beta_n.
          U = random('Uniform',0,1);

          % Generate proposal Z = (z_0, z_1)
          z_0 = random('Normal',Beta(i,1),Sigma2);
          z_1 = random('Normal',Beta(i,2),Sigma2);
          % q(Beta_i | z) / q(z | Beta_i) = 1 by symmetry of normal PDF

          AR_Numerator = Posterior(z_0,z_1,y,X);
          AR_Denominator = Posterior(Beta(i,1),Beta(i,2),y,X);
          AcceptanceRate = AR_Numerator/AR_Denominator;

          if U <= AcceptanceRate
               Beta(i+1,:) = [z_0 z_1];
          else
               Beta(i+1,:) = Beta(i,:);
          end
     end
end
```

In the fifth step, we examine four Metropolis(-Hastings) Markov chains with initial values `Beta_Initials`. Furthermore, after a trial-and-error procedure, we use the deterministic parameter `Sigma2 = 3` for the proposal PDF q. First, trace plots are created for both β_0 and β_1. See Figure 27. The left-hand side plot shows that all four Markov chains seem to converge to $\beta_0^{\text{TRUE}} := 4.2$ after only approximately 500 iterations. Similarly, the right-hand side plot shows that all four Markov chains seem to converge to $\beta_1^{\text{TRUE}} := 2.1$ after only approximately 500 iterations. This is an excellent result that all four generated samples of the posterior distribution $\tilde{p}(\boldsymbol{\beta} \mid \boldsymbol{y}, \mathbf{X})$ seem to have a burn-in period of merely 500 iterations. Moreover, observe that we want the Metropolis(-Hastings) Markov chains to fluctuate around their true parameter values because we imposed some prior uncertainty about the parameters.

```
%% Step 5: Auditing Generated Samples
Beta_Initials = [[0 0]; [1 1]; [2 2]; [4 2]];  % Matrix with 4 rows
Sigma2 = 3;

tic
Sample_1 = MH_Algorithm_BLR(Beta_Initials(1,:),Sigma2,y,X);
Sample_2 = MH_Algorithm_BLR(Beta_Initials(2,:),Sigma2,y,X);
Sample_3 = MH_Algorithm_BLR(Beta_Initials(3,:),Sigma2,y,X);
Sample_4 = MH_Algorithm_BLR(Beta_Initials(4,:),Sigma2,y,X);
toc  % Elapsed time is 322.9203 seconds.

Index = 1;  % Selects either the \beta_0s or \beta_1s from sample
% Index = 1 corresponds to \beta_0, and Index = 2 to \beta_1

plot(1:n,Sample_1(:,Index),'Color','green','LineWidth',2)
hold on;
plot(1:n,Sample_2(:,Index),'Color','blue','LineWidth',2)
plot(1:n,Sample_3(:,Index),'Color','cyan','LineWidth',2)
plot(1:n,Sample_4(:,Index),'Color','red','LineWidth',2)
```

```
20
21 title(['Trace Plot of $\beta_0$-Markov Chains with $n = $' num2str(n)], ...
22      'FontSize',27,'Interpreter','latex')
23 xlabel('$i$','FontSize',21,'Interpreter','latex')
24 ylabel('$\beta_{0,i}$','FontSize',21,'Interpreter','latex')
25 legend({'\mbox{\boldmath$\beta_0$} $= (0,0)^T, \ \sigma^2 = 3$', ...
26      '\mbox{\boldmath$\beta_0$} $= (1,1)^T, \ \sigma^2 = 3$', ...
27      '\mbox{\boldmath$\beta_0$} $= (2,2)^T, \ \sigma^2 = 3$', ...
28      '\mbox{\boldmath$\beta_0$} $= (4,2)^T, \ \sigma^2 = 3$'}, ...
29      'Location','northeast','FontSize',24,'Interpreter','latex');
30 % Note: Adjust the beta subscript when changing 'Index'
31 hold off;
```

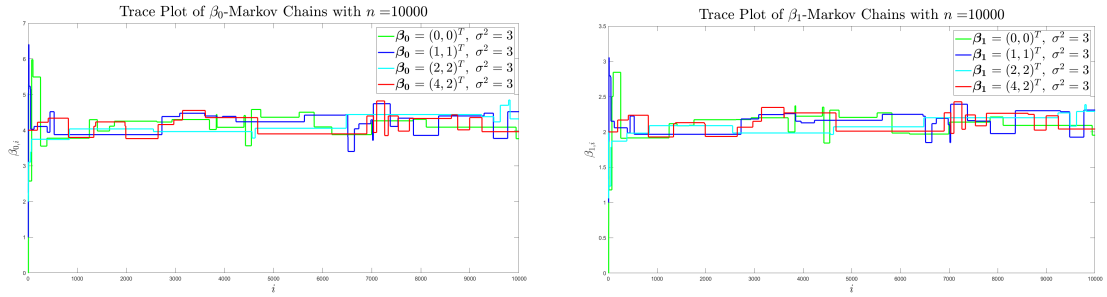

Figure 27

Next, we calculate the Gelman-Rubin statistics R that tests whether a Markov chain has converged after a certain number of iterations. Using this statistics, we aim to examine whether a burn-in period of 500 iterations is sufficient.

```
1  %%% Gelman-Rubin Convergence Statistic
2  % i = 1,...n denotes the time index of a single M-H Markov chain
3  % j = 1,...,m labelling of different chains
4  n_GR = 500;  % n corresponding to the Gelman-Rubin statistic
5  m = 4;  % Number of different Markov chains considerd
6
7  % Store the M-H Markov chains, where each column corresponds to one of
8  % the m initial values & the rows correspond to the time index
9  VBS_Chains = zeros(2,m);  % Variance between sequences
10 VWS_Chains = zeros(2,m);  % Variance within sequences
11
12 Merged_Samples = [Sample_1(1:n_GR, ...
13      :) Sample_2(1:n_GR,:) Sample_3(1:n_GR,:) Sample_4(1:n_GR,:)];
14 B = [1 2]; R = [1 2];  % To store the relevant statistics
15
16 for l = 1:2  % For beta_0 and beta_1
17      for j = 1:m
18           Sequence = Merged_Samples(:,j+(l-1));
19           VBS_Chains(l,j) = mean(Sequence);
20           VWS_Chains(l,j) = var(Sequence);
21      end
22
23      % Variance between sequences
```

```
     B(l) = n_GR/(m-1) * sum( VBS_Chains(l,:) - mean(VBS_Chains(l,:)) );
     W = mean(VWS_Chains(l,:));  % Variance within sequences
     R(l) = 1 - 1/n_GR + B(l)/(n_GR*W);  % Gelman-Rubin Convergence Statistic
end
```

The Gelman-Rubin statistic with `n_GR = n` yields `R = 0.9999` for both parameters. Hence, since $|R| \leq 1.1$, the generated Metropolis(-Hastings) Markov chains have converged after $10,000$ iterations. Similarly, the Gelman-Rubin statistic with `n_GR = 500` yields `R = 0.9980` for both parameters. Thus, a burn-in period of 500 iterations is sufficient for all four generated Metropolis(-Hastings) Markov chains.

Moreover, we also want to test whether the generated samples show independent behaviour. Let us use the generated Metropolis(-Hastings) Markov chain `Sample_3` with initial conditions $\beta_{0,0} = \beta_{1,0} = 2$ for the further computations. Next, we calculate the autocorrelation plot (ACF) for this chosen sample. Figure 28 displays the relevant ACF plot. Observe that the generated samples do not show independent behaviour because even the ACF with a lag of $k = 100$ produces a value of approximately 0.75. Nevertheless, we are still going to use this sample for further calculations.

```
%%% Autocorrelation Plot
% Let us use 'Sample_3' to approximate the integral of interest
Beta_Sample = Sample_3(500:end,:);  % Delte Burn-in period of 500 samples

MaxLag = 100;  % Maximum number of lags
AutoCorrelation = zeros(2,MaxLag);  % Store autocorrelations

for l = 1:2  % For beta_0 and beta_1
     for k = 0:MaxLag
          Cov_Xk = mean( (Beta_Sample(1:end-k,l) - mean( ...
               Beta_Sample(:,l))) .* (Beta_Sample(k+1:end,l) - ...
               mean(Beta_Sample(:,l))) );
          AutoCorrelation(l,k+1) = Cov_Xk/var(Beta_Sample(:,l));
     end
end

%%% Plotting the autocorrelation function
stem(0:MaxLag,AutoCorrelation(1,:), ...
     'Marker','none','LineWidth',2,'Color','green')
hold on;
stem(0:MaxLag,AutoCorrelation(2,:), ...
     'Marker','none','LineWidth',2,'Color','green')
title(['ACF for $n = $ ' num2str(n) '$, \ $ \mbox{\boldmath$\beta_0$}' ...
     ' $= (2,2)^T \ \& \ \ \sigma^2 =$ ' num2str(Sigma2)], ...
     'FontSize',27,'Interpreter','latex')
xlabel('Lag $k$','FontSize',21,'Interpreter','latex')
ylabel('$\rho_k$','FontSize',21,'Interpreter','latex')
hold off;
```

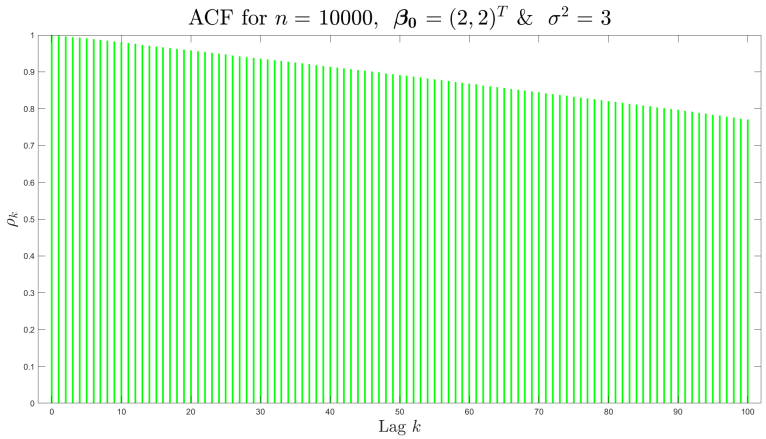

Figure 28

In the sixth step, an estimator for the predictive posterior probability of success $p(y_{\text{new}} = 1 \mid \boldsymbol{x}_{\text{new}}, \boldsymbol{y}, \mathbf{X})$, given by Ergodic Theorem 4.7, is computed. For this computation we use the chosen sample `Beta_Sample = Sample_3` with a deleted burn-in period of 500 iterations. However, one particular outcome lacks significant meaning since a pseudo-randomly generated value of `x_new` is used in this computation. Therefore, we plot the estimated probability of interest for a range of `x_new` values in step seven.

```
%% Step 6: Ergodic Theorem
n_AfterBurnIn = length(Beta_Sample(:,1));

x_new = [1 random('Normal',5,7)];  % x_{new}
Pi_new = zeros(n_AfterBurnIn-1,1);
for i = 1:(n_AfterBurnIn-1)  % \pi_{new,i}
     Pi_new(i) = ( 1 + exp(-x_new*transpose(Beta_Sample(i,:))) )^(-1);
end

Predictive_Probability_Success = (1/n_AfterBurnIn)*sum(Pi_new);
```

```
%% Step 7: Visualizing Predictive Probability
x_new_Values = -20:0.1:20;
PPS_Image = zeros(length(x_new_Values),1);
for k = 1:length(x_new_Values)
     PPS_Image(k) = PPS(x_new_Values(k),Beta_Sample);
end

plot(x_new_Values,PPS_Image,'Color','magenta','LineWidth',2)
hold on;
plot(x_new_Values,( 1 + exp(-x_new_Values) ).^(-1), ...
     'Color','cyan','LineWidth',2)  % Plotting the Sigmoid function

title(['$p(y_{new} = 1 \mid$ \mbox{\boldmath$x$}$_{new},$ ' ...
     '\mbox{\boldmath$y$}, \mbox{\boldmath$X$}$)$'], ...
     'FontSize',27,'Interpreter','latex')
xlabel('$x_{new}$','FontSize',21,'Interpreter','latex')
legend({'Empirical Plot','$t \mapsto (1+e^{-t})^{-1}$'}, ...
     'Location','northeast','FontSize',30,'Interpreter','latex');
```

```
hold off;
```

While plotting the predictive probability of success in step seven, a function `PPS` is used to reiterate step six for all `x_new_Values`. Figure 29 shows the approximated predictive posterior probability that a new response variable is a success, i.e. $p(y_{\text{new}} = 1 \mid \boldsymbol{x}_{\text{new}}, \boldsymbol{y}, \mathbf{X})$, over a range of possible newly observed regressor values `x_new`. Observe that empirical probability plot display similar behaviour to the sigmoid function $t \mapsto (1 + e^{-t})^{-1}$. However, this is not surprising, since $\pi_{\text{new},i}$ is also a sigmoid function. Moreover, if the newly observed regressor values is positive, then the response variable is almost surely a success. Similarly, if the newly observed regressor values is smaller than -5, then the response variable is almost surely not a success.

```
%% Function for Visualization of Probabilities
function[Predictive_Probability_Success] = PPS(x_new_i,Beta_Sample)
     % This is saved in a function file called 'PPS.m'
     x_new = [1 x_new_i];  % x_{new}

     N = length(Beta_Sample);
     Pi_new = zeros(N-1,1);
     for j = 1:(N-1)  % \pi_{new,i}
          Pi_new(j) = ( 1 + exp(-x_new*transpose(Beta_Sample(j,:))) )^(-1);
     end

     Predictive_Probability_Success = (1/N)*sum(Pi_new);
end
```

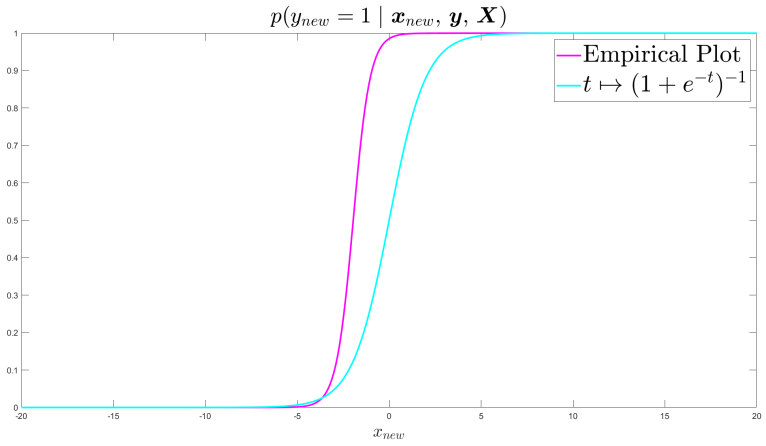

Figure 29

5 Conclusion

In this paper, a survey of the basics of Monte Carlo integration is presented. The primary objective of this survey is to explain the relevant theory in sufficient depth, and accompany each part of the theory with at least one application.

In section 2, we have shown that simply rewriting an integral in terms of an expectation operator results in a powerful technique to approximate an integral. If we can easily sample from a target PDF, which we obtain by rewriting the one-dimensional integral according to the Monte Carlo integration principle, then it is not difficult to approximate the one-dimensional integral of interest. An analogous result holds when a multi-dimensional integral can be rewritten in terms of a product of univariate common PDFs.

However, the efficiency of the Monte Carlo integration estimator can be improved, not through the conventional method of increasing the sample size, but by a technique called importance sample that has been studied in section 3. Caution must be taken when choosing a proposal PDF, because a bad choice for the proposal PDF can diminish the accuracy of the estimator.

In section 4, we consider the most realistic scenario where we cannot directly sample from the target PDF. Subsection 4.2 explains that the rejection sampling algorithm can be used to generate samples from a low-dimensional unnormalized PDF. In one-dimension, we have demonstrated that it is not difficult to choose a good proposal after having plotted the, possibly unnormalized, target PDF. When a bad proposal PDF is chosen, the acceptance rate decreases, making the algorithm less efficient. Choosing a good proposal PDF becomes significantly more challenging for a multi-dimensional PDF. Hence, rejection sampling is not recommended when the goal is to sample from a multi-dimensional target PDF. In this case, a widely used method is a MCMC algorithm, called the Metropolis-Hastings algorithm, that we present in subsection 4.5. An overview is given of some other well-known MCMC algorithms that are special cases of the Metropolis-Hastings algorithm. MCMC algorithms are preferred to rejection sampling when dealing with multi-dimensional target PDFs, because analysing plots is not necessary and situation-based adaptation is possible. Furthermore, by utilizing the Ergodic Theorem 4.7 from subsection 4.4, the Monte Carlo integration principle can be used to approximate multi-dimensional integrals.

Finally, subsection 4.6 provides an application where a multi-dimensional integral corresponding to a Bayesian linear regression is approximated by the Metropolis-Hastings algorithm and the Ergodic Theorem. Hence, even in a more complex scenario, the Metropolis-Hastings algorithm is still relatively simple and compact. However, caution must be taken when choosing the initial conditions for the algorithm.

We conclude that Monte Carlo integration is a useful technique for approximating either an one-dimensional integral, or multi-dimensional integrals. Therefore, Monte Carlo integration constitutes a beneficial collection of algorithms for econometrics, data science, operations research and actuarial science.

However, not every Monte Carlo integration technique is considered in this survey. In particular, there exists a diverse range of MCMC algorithms. Well-known MCMC algorithms that are not covered in this study are Hamiltonian MCMC, slice sampling, reversible-jump MCMC. Moreover, there are also many techniques that build upon or improve the importance sampling technique for specific scenarios. Such techniques include sequential importance sampling, exponential tilting and cross-entropy. In addition, quasi-Monte Carlo integration can also be considered. Instead of using pseudo-random generated numbers, quasi-Monte Carlo integration uses quasi-random sequences. Alternatively, probabilistic numerical techniques, such as Bayesian quadrature, can also be used to approximate integrals.

References

Agapiou, S., O. Papaspiliopoulos, D. Sanz-Alonso, and A.M. Stuart (2017). Importance Sampling: Intrinsic Dimension and Computational Cost. *Statistical Science 32*(3), 405–431.

Akyildiz, O.D. (2022). Stochastic Simulation. Lecture notes, Department of Mathematics, Imperial College London.

Andrieu, C., N. de Freitas, A. Doucet, and M.I. Jordan (2003). An Introduction to MCMC for Machine Learning. *Machine Learning 50*(1-2), 5–43.

Bierens, H.J. (Ed.) (2005). *Introduction to the Mathematical and Statistical Foundations of Econometrics*. Cambridge, New York: Cambridge University Press.

Blitzstein, J.K. and J. Hwang (Eds.) (2019). *Introduction to Probability*. Boca Raton: CRC Press, Taylor and Francis Group.

Brandimarte, P. (Ed.) (2014). *Handbook in Monte Carlo simulation: applications in financial engineering, risk management, and economics*. Hoboken, New Jersey: John Wiley and Sons.

Brooks, S., A. Gelman, G. Jones, and X.L. Meng (Eds.) (2011). *Handbook of Markov chain Monte Carlo*. Boca Raton, London: CRC Press.

Cameron, A.C. and P.K. Trived (Eds.) (2005). *Microeconometrics: Methods and Applications*. Cambridge: Cambridge University Press.

Chib, S. and E. Greenberg (1969). Markov Chain Monte Carlo Simulation Methods in Econometrics. *Econometric Theory 12*(3), 409–431.

Cochran, W.G. (Ed.) (1977). *Sampling Techniques*. New York, London: Wiley.

Dobson, A.J. and A.G. Barnett (Eds.) (2018). *An Introduction to Generalized Linear Models*. Boca Raton: CRC Press, Taylor and Francis Group.

Dodge, Y. (Ed.) (2008). *The Concise Encyclopedia of Statistics*. New York: Springer.

Dongarra, J. and F. Sullivan (2000). Guest Editors Introduction to the Top 10 Algorithms. *Computing in Science and Engineering 1*(1), 22–23.

Durrett, R.T. (Ed.) (2019). *Probability: Theory and Examples*. Cambridge, New York: Cambridge University Press.

Gelman, A. (Ed.) (2004). *Bayesian Data Analysis*. Boca Raton: Chapman and Hall/CRC.

Gelman, A. and D.B. Rubin (1992). Inference from Iterative Simulation using Multiple Sequences. *Statistical Science 7*(4), 457–472.

Geman, S. and D. Geman (1984). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 6*(6), 721–741.

Geweke, J. (1989). Bayesian Inference in Econometric Models Using Monte Carlo Integration. *Econometrica 57*(6), 1317–1339.

Geweke, J., G. Koop, and H.K. van Dijk (Eds.) (2011). *The Oxford handbook of Bayesian econometrics*. Oxford, New York: Oxford University Press.

Guilhoto, L.F. (2017). Applying Markov Chains to Monte Carlo Integration.

Hastings, W. K. (1970). Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika 57*(1), 97–109.

Henning, P., M.A. Osborne, and H.P. Kersting (Eds.) (2022). *Probablistic Numerics: Computation as Machine Learning*. Cambridge University Press.

Hoff, P.D. (Ed.) (2009). *A First Course in Bayesian Statistical Methods*. New York, London: Springer.

Jameson, G.J.O. (Ed.) (2003). *The Prime Number Theorem*. Cambridge, New York: Cambridge University Press.

Kloek, T. and H.K. van Dijk (1978). Bayesian Estimates of Equation System Parameters: An Application of Integration by Monte Carlo. *Econometrica 46*(1), 1–19.

Levin, D.A. and Y. Peres (Eds.) (2017). *Markov Chains and Mixing Times*. Rhode Island: American Mathematical Society.

Liu, J.S. (Ed.) (2001). *Monte Carlo Strategies in Scientific Computing*. New York: Springer.

Luenberger, D.G. (Ed.) (2013). *Investment science*. New York: Oxford University Press.

Martínez, A.G. (2022). Markov chains and Markov chain Monte Carlo methods. Research paper, Department of Mathematics and Computer Science, University of Barcelona.

Metropolis, N. (1987). The Beginning of the Monte Carlo Method. *Los Alamos Science Special Issue 15*, 125–130.

Metropolis, N., A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics 21*(6), 1087–1092.

Metropolis, N.C. and S.M. Ulam (1949). The Monte Carlo Method. *Journal of the American Statistical Association 44*(247), 335–341.

von Neumann, J. (1951). Various techniques used in connection with random digits. Monte Carlo methods. *National Bureau of Standards Applied Math Series 12*, 36–38.

Riemann, G.F.B. (1859). Ueber die Anzahl der Primzahlen unter einer gegebenen Grösse. *Monatsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin*.

Robbins, H. (1948). On the Asymptotic Distribution of the Sum of a Random Number of Random Variables. *Proceedings of the National Academy of Sciences of the United States of America 34*(4), 162–163.

Robert, C.P. and G. Casella (Eds.) (2004). *Monte Carlo Statistical Methods*. New York: Springer.

Robert, C.P. and G. Casella (Eds.) (2010). *Introducing Monte Carlo Methods with R*. New York, London: Springer.

Ross, S.M. (Ed.) (2019). *Introduction to probability models*. London, San Diego: Academic Press, an imprint of Elsevier.

Rubinstein, R.Y. (Ed.) (2016). *Simulation and the Monte Carlo Method*. New York: John Wiley and Sons.

Shonkwiler, R. W. and F. Mendivil (Eds.) (2024). *Explorations in Monte Carlo methods*. Cham, Switzerland: Springer Nature Switzerland.

Sutherland, W. A. (Ed.) (2009). *Introduction to Metric and Topological Spaces*. Oxford: Oxford University Press.

Train, K.E. (Ed.) (2009). *Discrete Choice Methods with Simulation*. Cambridge: Cambridge University Press.

Vats, D. and C. Knudson (2021). Revisiting the Gelman-Rubin Diagnostic. *Statistical Science 36*(4), 518–529.

6 Appendix

6.1 Symbols and Notation

Notation	*Meaning*
$\mathbb{N}$	The natural numbers $\{1, 2, 3, ...\}$.
$\mathbb{R}$	The real numbers.
$\mathbb{R}_{>0}$	The positive real numbers.
$\mathbb{R}_{\geq x}$	The real numbers greater than or equal to $x \in \mathbb{R}$.
$\#S$	The cardinality of a set S.
$A \times B$	The Cartesian product of two sets A and B, i.e. the set $\{(a, b) \mid a \in A,\ b \in B\}$.
$\mathbb{R}^d$	d-dimensional real space.
$A \subseteq B$	A is an improper subset of B.
$A \subsetneq B$	A is a proper subset of B.
$\mathbb{I}\{\cdot \in S\}$	The indicator function of a set S.
$\text{int}\,\{S\}$	The interior of a subset S of the standard topological space $\mathbb{R}$.
$\phi(A)$	The image of a real-valued function $\phi : X \to Y$ is $\phi(A) := \{\phi(a) : a \in A \subseteq X\}$.
$\phi^{-1}(B)$	The pre-image of a real-valued function $\phi : X \to Y$ is $\phi(A) := \{x \in X : \phi(x) \in B \subseteq Y\}$.
$\log(\cdot)$	The natural logarithmic function: $\mathbb{R}_{>0} \to \mathbb{R}$.
$\phi(x) \propto \tilde{\phi}(x)$	A real-valued function $\phi : \mathbb{R} \to \mathbb{R}$ is proportional to $\tilde{\phi}(x)$, if $\exists$ a constant $c \in \mathbb{R}$ such that $\phi(x) = c \cdot \tilde{\phi}(x)$, $\forall x \in \mathbb{R}$.
$\phi_2 \circ \phi_1(x)$	The composition of two real-valued function $\phi_1, \phi_2 : \mathbb{R} \to \mathbb{R}$ is $\phi_2 \circ \phi_1(x) := \phi_2(\phi_1(x))$, $\forall x \in \mathbb{R}$.
$\boldsymbol{x}, \boldsymbol{X}$	Real-valued vectors in $\mathbb{R}^n$, where $n \in \mathbb{N}$.
$\mathbf{A}$	Real-valued matrix in $\mathbb{R}^{n \times m}$, where $m, n \in \mathbb{N}$.
A^T	The transpose of some real-valued matrix $A \in \mathbb{R}^{n \times m}$, where $m, n \in \mathbb{N}$.
X	Real-valued random variable $X : \Omega \subseteq \mathbb{R} \to \mathbb{R}$ defined on probability measure space $(\Omega, \mathcal{B}(\Omega), \mathbb{P})$, with realization x.
$\boldsymbol{X}$	Real-valued n-dimensional random vector $\boldsymbol{X} : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}^n$ defined on probability measure space $(\Omega, \mathcal{B}(\Omega), \mathbb{P})$, with realization $\boldsymbol{x}$.
f_X	The probability density function (PDF) of a random variable X.
$f_{\boldsymbol{X}}$	The joint PDF of a random vector $\boldsymbol{X}$.
F_X	The cumulative distribution function (CDF) of a random variable X.
$F_{\boldsymbol{X}}$	The joint CDF of a random vector $\boldsymbol{X}$.
$\mathbb{E}_{f_X}[g(X)]$	The expectation of the random variable $g(X)$ with PDF f_X, where g is a real-valued function.
$\mathbb{E}_{f_{\boldsymbol{X}}}[g(\boldsymbol{X})]$	The expectation of the random vector $g(\boldsymbol{X})$ with PDF $f_{\boldsymbol{X}}$, where g is a real-valued function.

$\text{Var}_{f_X}[g(X)]$	The variance of the random variable $g(X)$ with PDF f_X, where g is a real-valued function.
$\text{Var}_{f_{\boldsymbol{X}}}[g(\boldsymbol{X})]$	The variance of the random vector $g(\boldsymbol{X})$ with PDF $f_{\boldsymbol{X}}$, where g is a real-valued function.
$Supp(f)$	The support of a real-valued function $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$ is $\{\boldsymbol{x} \in X \ : \ f(\boldsymbol{x}) \neq 0\}$.
$X \sim f_X$	The random variable X is distributed according to PDF f_X.
$f_X \overset{d}{=} g_X$	Two PDFs of a random variable X are equivalent.
$X_n \xrightarrow{d} X$	A sequence of real-valued random variables $X_1, ..., X_n$ converges almost surely if $\mathbb{P}\left(\{\omega \in \Omega : \lim_{n\to\infty} X_n(\omega) = X(\omega)\}\right) = 1$. This is convergence is denotes by $X_n \xrightarrow{d} X$
$F_n(x) \xrightarrow{d} F(x)$	A sequence of real-valued random variables $X_1, ..., X_n$ with CDFs F_i, for $i \in \{1, ..., n\}$, converges in distribution if $\lim_{n\to\infty} F_n(x) = F(x)$. This is convergence is denotes by $F_n(x) \xrightarrow{d} F(x)$.
$\text{Bern}(p)$	The Bernoulli distribution with success probability $p \in [0, 1]$. The corresponding PMF $f : \{0, 1\} \to [0, 1]$ is defined by $f(x) = p \cdot \mathbb{I}\{x = 1\} + (1 - p) \cdot \mathbb{I}\{x = 0\}$.
$\text{Unif}(a, b)$	The continuous uniform distribution over the interval $[a, b] \subsetneq \mathbb{R}$, $a < b$. The corresponding PDF $f : [a, b] \to \{0, \frac{1}{b-a}\}$ is defined by $f(x) = \frac{1}{b-a}\mathbb{I}\{x \in [a, b]\}$.
$\mathcal{N}(\mu, \sigma^2)$	The normal/Gaussian distribution with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 \in \mathbb{R}_{>0}$. The corresponding PDF $f : \mathbb{R} \to \mathbb{R}_{>0}$ is defined by $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-\frac{(x-\mu)^2}{2\sigma^2}\}$.
$\mathcal{T}f_X(\boldsymbol{\theta}, a)$	The truncated distribution with parameters $\boldsymbol{\theta} \in \mathbb{R}^k$ by $a \in \mathbb{R}$. The corresponding truncated PDF $\mathcal{T}f_X : \mathbb{R}_{\geq a} \to \mathbb{R}$ is $\mathcal{T}f_X(x) = \left[\int_a^\infty f_X(x)\, dx\right]^{-1} \cdot \mathbb{I}\{x \geq a\} \cdot f_X(x)$.
$\text{Cauchy}(x_0, \gamma)$	The Cauchy distribution with location parameter $x_0 \in \mathbb{R}$ and scale parameter $\gamma \in \mathbb{R}$.
$\text{Expo}(\lambda)$	The exponential distribution with rate parameter $\lambda \in \mathbb{R}_{>0}$. The corresponding PDF $f : \mathbb{R} \to \mathbb{R}_{>0}$ is defined by $f(x) = -\lambda e^{-\lambda x} \cdot \mathbb{I}\{x \geq 0\}$.
$\text{Beta}(\alpha, \beta)$	The beta distribution with shape parameters $\alpha, \beta \in \mathbb{R}_{>0}$.
$\text{Gamma}(\alpha, \beta)$	The gamma distribution with shape parameter $\alpha \in \mathbb{R}_{>0}$ and rate/scale parameter $\beta \in \mathbb{R}_{>0}$. The corresponding PDF $f : \mathbb{R} \to \mathbb{R}_{>0}$ is defined by $f(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x} \cdot \mathbb{I}\{x > 0\}$.
$\text{Weib}(\alpha, \beta)$	The Weibull distribution with shape parameter $\alpha \in \mathbb{R}_{>0}$ and scale parameter $\beta \in \mathbb{R}_{>0}$. The corresponding PDF $f : \mathbb{R} \to \mathbb{R}_{>0}$ is defined by $f(x) = \frac{\alpha}{\beta^\alpha} x^{\alpha-1} e^{-(x/\beta)^\alpha} \cdot \mathbb{I}\{x \geq 0\}$.