An Introduction to Markov chain Monte Carlo methods

Tim Dockhorn*

Department of Applied Mathematics, University of Waterloo

July 1, 2018

Abstract

This report gives a brief introduction to Markov chain Monte Carlo methods using a step-by-step procedure. The Monte Carlo method for computing integrals is reviewed and demonstrated using a straightforward example. The issue of computing independent and identically distributed random numbers is discussed. Basic properties of Markov chains are reviewed in order to introduce the Markov chain Monte Carlo methods, in particular the Metropolis algorithm. Several results of numerical simulations are shown throughout the report.

*Email address: tim.dockhorn@uwaterloo.ca

Chapter 1

Introduction

The Monte Carlo methods are an extensive class of computational algorithms that utilize random sampling to approximate the solutions of deterministic problems [1]. These methods are used in mathematical optimization, numerical integration and probability theory, etc. Mathematicians such as Stan Ulam and Enrico Fermi used Monte Carlo methods for the calculations of neutron diffusion [1] already in the 1930s. The first public document on Monte Carlo methods was then published by Nick Metropolis and Ulam in 1949 [7]. Since then, there have been many publications on a vast spectrum of Monte Carlo methods.

One of the main difficulties in Monte Carlo methods is the demand for random numbers. Generally, it is impossible to produce random numbers with a computer, however, pseudo random numbers can be generated using sophisticated algorithms, e.g., a linear congruential generator.

Markov chain Monte Carlo methods are a special class of algorithms that are well-suited for drawing samples from complex probability distributions. The Metropolis algorithm [6], an instance of the Markov chain Monte Carlo methods, has been named one of the "ten algorithms with the greatest influence of the developrneut and practice of science and engineering in the 20th century" [2].

The remainder of this report is organized as follows. In Chapter 2, the basic principles of Monte Carlo methods are described and a numerical simulation for the approximation of π is shown. Chapter 3 explains algorithms of sampling methods for basic probability density distributions. In Chapter 4, the properties of Markov chains are reviewed and their importance in providing a basis for Markov chain Monte Carlo methods are demonstrated. In Chapter 5, The Metropolis algorithm is described and a numerical simulation for drawing samples from a complex probability density function is presented. Chapter 6 summarizes the results and demonstrates the continued relevance of Markov chain Monte Carlo methods.

Chapter 2

The Monte Carlo method

The Monte Carlo method can be used to provide estimates to integrals, e.g., the expectation value of a function $h(X)$ of a random variable X with probability density function $f_X(x)$

$$\mathbb{E}[h(X)] = \int_{-\infty}^{\infty} h(x) f_X(x) dx \ . \tag{2.1}$$

The estimates are based on an independent and identically distributed set of random numbers $\{x^{(i)}\}_{i=1}^{N}$ drawn from the distribution $f_X(x)$. A Monte Carlo estimator $I_N(h)$ is then given as

$$I_N(h) = \frac{1}{N} \sum_{i=1}^{N} h(x^{(i)}) \ . \tag{2.2}$$

The estimator $I(h)$ is unbiased and converges almost surely to the exact solution $\mathbb{E}[h(X)]$, i.e.,

$$P\Big(\lim_{N\to\infty} I_N(h) = \mathbb{E}[h(X)] \Big) = 1 \ , \tag{2.3}$$

by the strong law of large numbers [5]. The estimator itself is a random variable with mean

$$\mathbb{E}[I_N(h)] = \frac{1}{N} \mathbb{E}\left[\sum_{i=1}^{N} h(x^{(i)}) \right] = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}[h(x^{(i)})] = \mathbb{E}[h(X)] \ , \tag{2.4}$$

and variance

$$\begin{aligned}
\mathrm{Var}[I] &= \mathbb{E}\left[\left(\frac{1}{N}\sum_{i=1}^{N} h(x^{(i)}) - \mathbb{E}[h(X)]\right)^2\right] \\
&= \frac{1}{N^2}\mathbb{E}\left[\sum_{i=1}^{N}\left(h(x^{(i)}) - \mathbb{E}[h(X)]\right)^2 - \sum_{i=1}^{N}\sum_{j=1, i\neq j}^{N}\big(h(x^{(i)}) - \mathbb{E}[h(X)]\big)\big(h(x^{(j)}) - \mathbb{E}[h(X)]\big)\right] \\
&= \frac{1}{N^2}\mathbb{E}\left[\sum_{i=1}^{N}\left(h(x^{(i)}) - \mathbb{E}[h(X)]\right)^2\right] \quad \text{(since draws } \{x^{(k)}\}_{k=1}^{N} \text{ are independent)} \\
&= \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{Var}[h(X)] = \frac{1}{N}\mathrm{Var}[h(X)] \; .
\end{aligned} \tag{2.5}$$

The major advantage of the Monte Carlo method compared to other numerical integration algorithms is that its convergence is independent of the dimension of the random variable X [5].

Example 1. We can use the Monte Carlo method to estimate π numerically. The area of a circle with radius one is π, thus we want to approximate the integral

$$A = \int_0^1 \int_0^{2\pi} r dr d\theta = \pi \; . \tag{2.6}$$

Note that the integral can be rewritten in Cartesian coordinates using the indicator function

$$\mathbb{1}_A(x, y) \coloneqq \begin{cases} 1 & \text{if } x^2 + y^2 \leq 1 \\ 0 & \text{otherwise} \end{cases} \; , \tag{2.7}$$

as

$$\pi = \int_{-1}^{1}\int_{-1}^{1} \mathbb{1}_A(x, y) dx dy = 4\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathbb{1}_A(x, y) f_{X,Y}(x, y) dx dy \; , \tag{2.8}$$

where $f_{X,Y}(x, y) = f_X(x) f_Y(y) = f(x) f(y)$ with

$$f(x) = \begin{cases} \frac{1}{2} & \text{if } x \in [-1, 1] \\ 0 & \text{otherwise} \end{cases} \; . \tag{2.9}$$

The integral A can then be approximated by

$$\pi \approx I_N(\mathbb{1}_A(x, y)) = 4\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_A(x^{(i)}, y^{(i)}) \tag{2.10}$$

where the tuples $\{x^{(i)}, y^{(i)}\}_{i=1}^{N}$ are drawn from the uniform distribution $\mathcal{U}([-1,1] \times [-1,1])$. The process of drawing independent and identically distributed random numbers is addressed in Chapter 3.
Numerical simulations have been carried out for a set of increasing numbers of draws $\hat{N} = \{10^2, 10^3, 10^4, 10^5, 10^6\}$. For each number of draws ($N \in \hat{N}$), ten simulations have been run; the results are documented in Table A.1. In Figure 2.1, it can be seen that the variance of the expected value is decreasing with increasing N. The expected value $I(h)$ clearly converges to π.

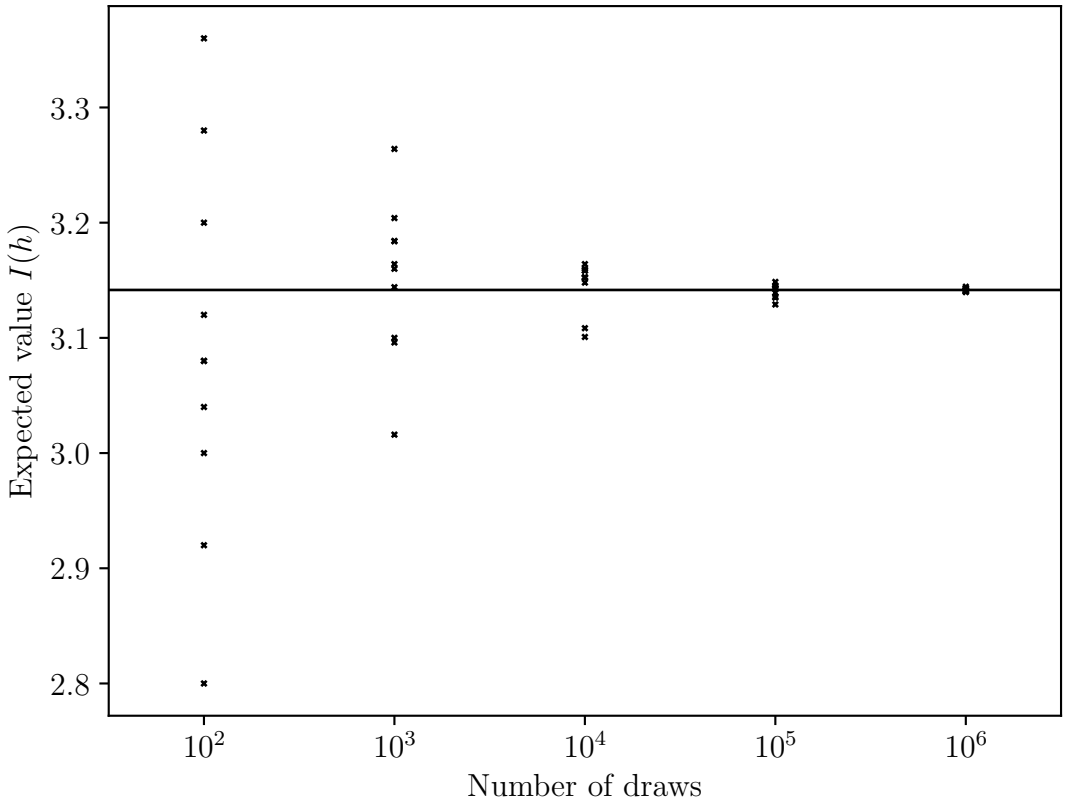


Figure 2.1: Monte Carlo method for approximating π

Chapter 3

Sampling methods

In the Example 1, we have used a set of independent and identically distributed random numbers drawn from the uniform distribution $\mathcal{U}([-1,1] \times [-1,1])$. The obvious question to ask is how the samples have been drawn.

The core of any algorithm that simulates random variables for a complex probability density function is to first draw random variables from the uniform distribution $\mathcal{U}[0,1]$ and modify the samples subsequently [5]. For a suitable choice of parameters, independent random numbers can be drawn from $\mathcal{U}[0,1]$ using Algorithm 1.

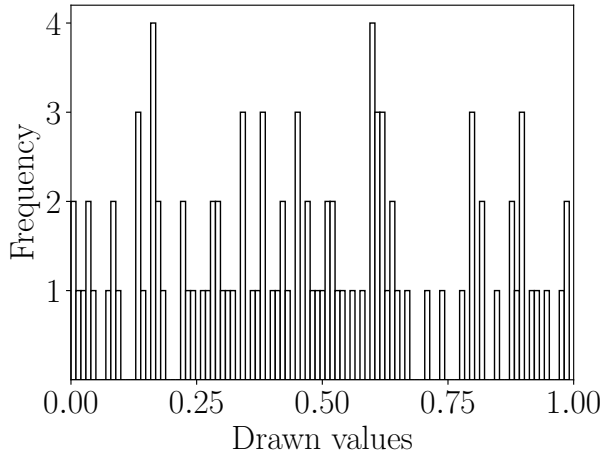
Algorithm 1 Drawing samples from $\mathcal{U}[0,1]$ using a linear congruential generator [5]

1: Choose $x^{(0)} \in \mathbb{R}$ and $a, c, m, N \in \mathbb{N}$
2: $i = 0$
3: **for** $i < N$ **do**
4: $\quad x^{(i+1)} = (ax^{(i)} + c) \mod m$
5: $\{u^{(i)}\}_{i=1}^{N} = \{x^{(i)}/m\}_{i=1}^{N}$

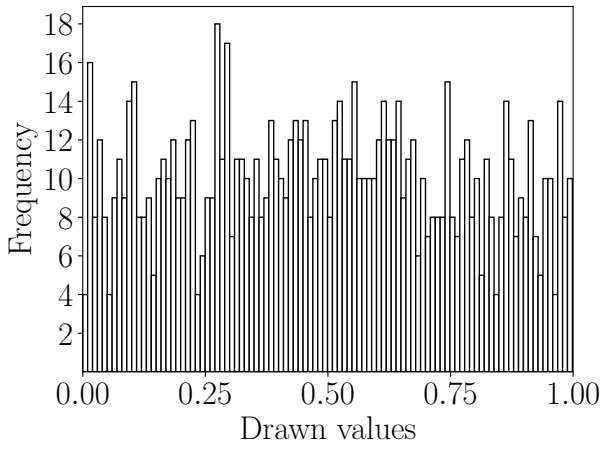
Numerical simulations for drawing samples from $\mathcal{U}[0,1]$ have been carried out using the set of parameters: $m = 2^{32}$, $a = 1664525$, $c = 1013904223$, and $x^{(0)} = 0$ [8]. The draws $\{u^{(i)}\}_{i=1}^{N}$, which are clustered in 100 bins, are plotted in a histogram in Figure 3.1. For 100 draws it does not seem like the draws are identically distributed, however, for an increasing number of draws (e.g. 10^5) the samples clearly approach a uniform distribution.

If the cumulative distribution function is known, the inverse transformation method (Algorithm 2) can be used to draw samples from a desired probability distribution. For higher-dimensional normal distributions Algorithm 3 can be used.

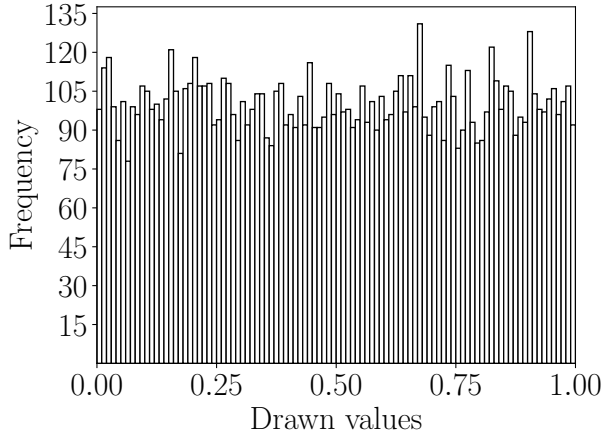
These so-called transformation methods are well-suited for "easy" probability distributions. More sophisticated sampling methods are needed when the probability distributions are complex (or even non-normalized).



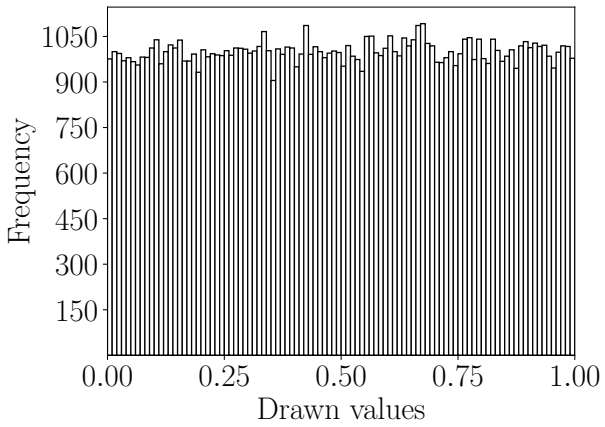
(a) 10^2 draws



(b) 10^3 draws



(c) 10^4 draws



(d) 10^5 draws

Figure 3.1: Samples from the distribution $\mathcal{U}[0,1]$ using a linear congruential generator

Algorithm 2 Inverse transformation method [5]

1: Draw samples $\{u^{(i)}\}_{i=1}^{N}$ from $\mathcal{U}[0,1]$.
2: $i = 0$
3: **for** $i < N$ **do**
4: $\quad x^{(i)} = F_X^{-1}(u^{(i)})$

Algorithm 3 Drawing samples from $\mathcal{N}(\mu, \Sigma)$ [5]

1: Decompose the covariance matrix $\Sigma = LL^T$
2: Draw samples $Z = \{z^{(i)}\}_{i=1}^{N}$ from $\mathcal{N}(0,1)$
3: Compute $\{x^{(i)}\}_{i=1}^{N} = X = \mu + LZ$

Let us again consider the integral $\mathbb{E}[h(X)]$ and introduce an arbitrary distribution function $q(x)$ such that

$$\operatorname{supp}\big(q(x)\big) = \operatorname{supp}\big(f_X(x)\big) \ . \tag{3.1}$$

From equation (3.1) it follows that the so-called importance weight $w(x) = f_X(x)/q(x)$ can be used to rewrite the expected value of $h(X)$, i.e.,

$$\mathbb{E}[h(X)] = \int_{-\infty}^{\infty} h(x)w(x)q(x)dx \ . \tag{3.2}$$

An unbiased Monte Carlo estimator that converges to $\mathbb{E}[h(X)]$ is then given as [1]

$$\hat{I}(h) = \frac{1}{N}\sum_{i=1}^{N} h(x^{(i)})w(x^{(i)}) \ , \tag{3.3}$$

where the N independent and identically distributed samples $\{x^{(i)}\}_{i=1}^{N}$ are drawn from $q(x)$. We are especially interested in those distribution functions $q(x)$ that minimize the variance of $\hat{I}(h)$ [1], i.e,

$$\min_{q(x)} \operatorname{Var}\big[\hat{I}(h)\big] \ . \tag{3.4}$$

It has been found in [4] that an optimal distribution function $q(x)$ is proportional to $|h(x)|f_X(x)$, where $|\cdot|$ denotes the absolute value. Drawing samples from $|h(x)|f_X(x)$ is generally complicated, however, the result tells us that it would be efficient to sample $q(x)$ on "important regions" of $\operatorname{supp}(f_X(x))$ where $|h(x)|f_X(x)$ is relatively large [1]. Moreover, it is shown in [1] that drawing samples from a well-suited $q(x)$ can result in a lower variance compared to when drawing from $f_X(x)$ itself, i.e, $\operatorname{Var}\big[\hat{I}(h)\big] < \operatorname{Var}[I(h)]$. Methods based on this approach are called importance sampling.

A strategy to increase the efficiency of importance sampling algorithms is to equip $q(x)$ with a parameter θ that is adopted during the simulation. However, even those more sophisticated methods are not always efficient enough; for this reason, we will introduce sampling algorithms that are based on Markov chains of continuous state spaces in Chapter 5.

Chapter 4

The Markov chain

Before we introduce Markov chain Monte Carlo methods, we review the basic properties of Markov chains on finite state spaces. A Markov chain is a sequence of possible events in which the probability of jumping in one event solely depends on its present state, i.e.,

$$P(x^{(k)} = x_k | x^{(k-1)} = x_{k-1}, x^{(k-2)} = x_{k-2}, ..., x^{(1)} = x_1, x^{(0)} = x_0) = P(x^{(k)} = x_k | x^{(k-1)} = x_{k-1}) \ . \tag{4.1}$$

We only consider Markov chains for which this dependence is independent of k, i.e., time-homogeneous Markov chains. An example of a Markov chain with state space $\Psi = \{x_1, x_2, x_3\}$ is given in Figure 4.1. The weights associated with the directed edges represent the probability of jumping from state x_i to state x_j.

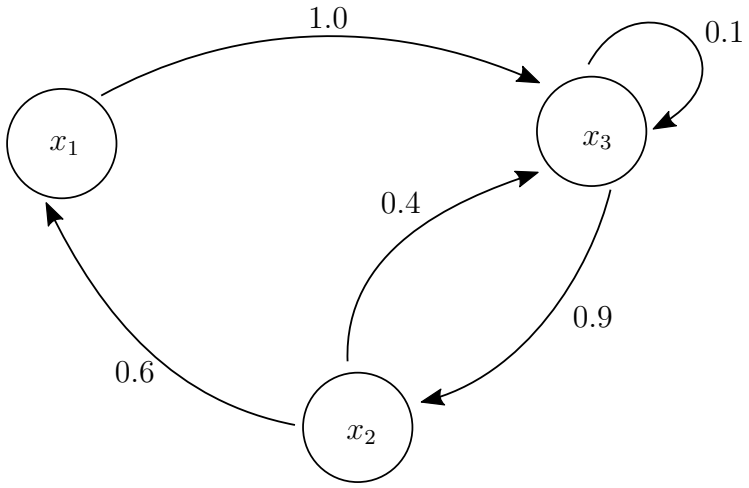


Figure 4.1: State transition graph for a three-state Markov chain [1]

The transition matrix T is then defined as

$$T_{ij} = P(x^{(k)} = x_j | x^{(k-1)} = x_i) \ . \tag{4.2}$$

It can be shown that starting from any initial state (e.g. $P(x^{(0)} = x_i) = [0.4, 0.5, 0.1]^T$), the Markov chain will converge against a stationary probability mass function $p_M(x)$ with the property that

$$p_M(x) = Tp_M(x) \ , \tag{4.3}$$

if the the state transition graph is both connected (irreducibility) and the chain does not get trapped in a cycle (aperiodicity) [1].

Let us now expand the idea of Markov chains to continuous state spaces. Assume that every state represents a point of a real interval, i.e, $\hat{x} \in I \subset \mathbb{R}$. In the continuous state spaces, the transition matrix develops into an integral kernel K [1]

$$\int p(y)K(x|y)dy = p(x) \ , \tag{4.4}$$

and p is a stationary probability density function. Those Markov chains serve as a basis for the Markov chain Monte Carlo methods described in the next chapter.

Chapter 5

The Markov chain Monte Carlo method

Markov chain Monte Carlo methods are sampling algorithms based on continuous Markov chains that have the desired probability distribution (from which samples should be drawn from) as a stationary probability density function (see equation (4.4)). One of those algorithms – the Metropolis algorithm [7] – has been named one of the "ten algorithms with the greatest influence on the development and practice of science and engineering in the 20th century" [2].

Algorithm 4 The Metropolis algorithm

1: Choose a $x^{(0)} \in \text{supp}(f_X(x))$
2: Draw samples $\{u^{(i)}\}_{i=1}^{N}$ from $\mathcal{U}[0,1]$
3: $i = 0$
4: **for** $i < N$ **do**
5: Draw a sample x^* from $q(x^*|x^{(i)})$
6: **if** $u^{(i)} < \min\left\{1, \frac{f_X(x^*)}{f_X(x^{(i)})}\right\}$ **then**
7: $x^{(i+1)} = x^*$
8: **else**
9: $x^{(i+1)} = x^{(i)}$

The Algorithm 4 works as follows: Giving a current Markov state $x^{(i)}$, a candidate value x^* is computed using the symmetrical proposal distribution $q(x^*|x^{(i)})$. The state of the Markov chain advances $x^{(i+1)} = x^*$ with the acceptance rate $\min\left\{1, \frac{f_X(x^*)}{f_X(x^{(i)})}\right\}$, otherwise it remains at the current state, i.e., $x^{(i+1)} = x^{(i)}$. Due to the normalization in the sixth step of the algorithm, normalized samples will be produced even if the input probability density function $f_X(x)$ is non-normalized. This property is demonstrated in the following numerical example.

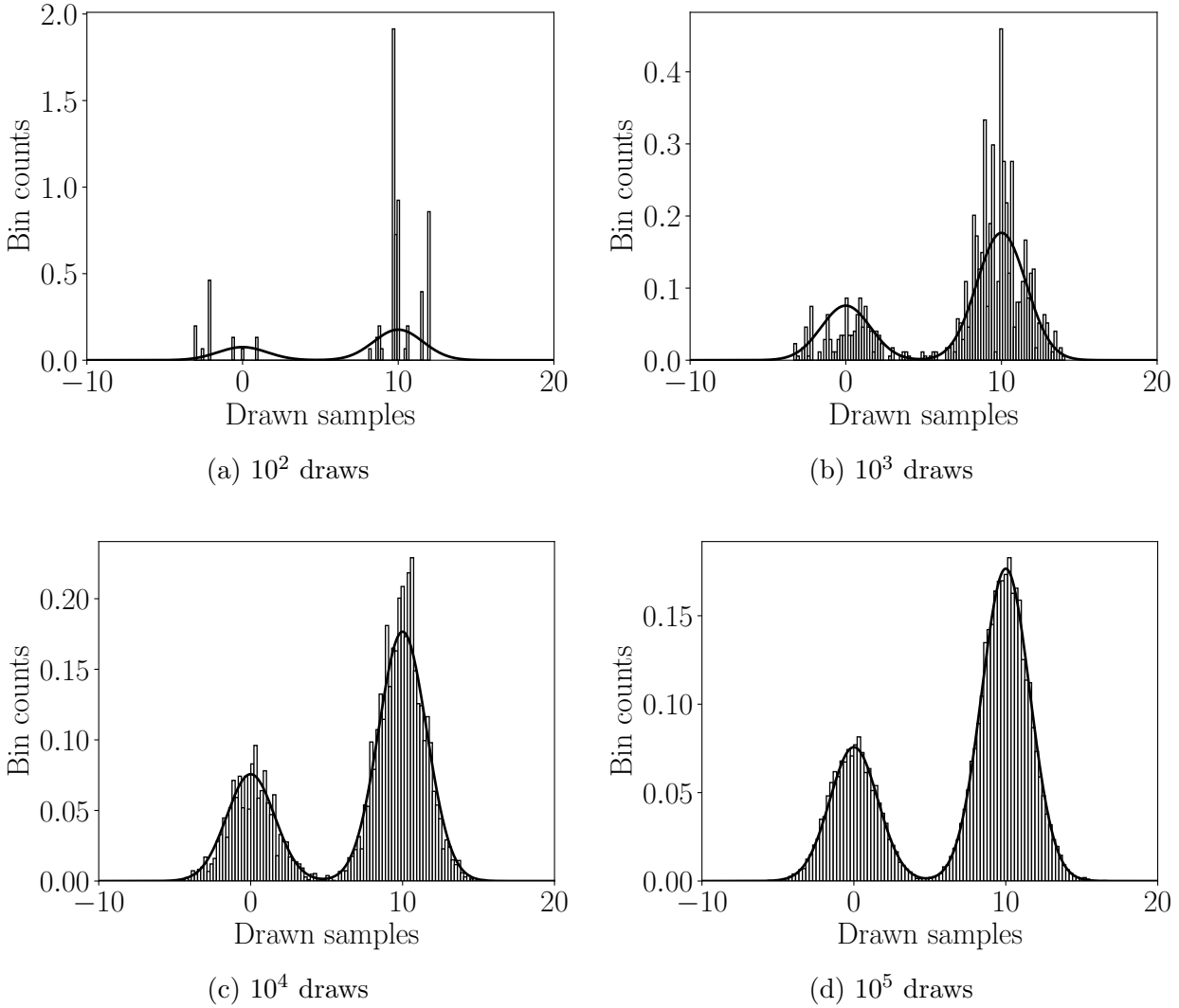
The implementation of the Metropolis algorithm is simple, however, the selection of a proposal distribution requires careful design in order to achieve optimal convergence rates. Due

to its complexity, this issue cannot be further adressed in this paper. The reader is reffered to [3] for an extensive discussion.

Figure 5.1 shows the result of the Metropolis algorithm using a (normal) proposal distribution $q(x^*|x^{(i)}) = \mathcal{N}(x^{(i)}, 100)$ to draw samples from a non-normalized target distribution $f_X(x) = 0.3\exp\big(-0.2x^2\big) + 0.7\exp\big(-0.2(x-10)^2\big)$ for 10^2, 10^3, 10^4, and 10^5 iterations. As expected, the histogram approximates the normalized target distribution

$$\hat{f}_X(x) = \frac{f_X(x)}{\int_{-\infty}^{\infty} f_X(x)dx}\,, \tag{5.1}$$

well for increasing iteration numbers.



(a) 10^2 draws

(b) 10^3 draws

(c) 10^4 draws

(d) 10^5 draws

Figure 5.1: Markov chain Monte Carlo approximation of the normalized target distribution

Chapter 6

Conclusion

This report gives a brief introduction to Markov chain Monte Carlo methods. The Monte Carlo method for computing integrals is reviewed, and the expected value as well as the variance of a general Monte Carlo estimator is computed. To gain a better understanding of the method, π is numerically approximated by a Monte Carlo estimator. In this example, the question of how to sample independent and identically distributed random numbers arises.

A linear congruential generator that draws samples from the uniform distribution $\mathcal{U}([0,1])$ is presented. This algorithms serves as a basis for drawing samples for more sophisticated probability distributions of which the cumulative distribution functions are known. Furthermore, the concept of importance sampling is introduced and used to illustrate the necessity of Markov chain Monte Carlo methods.

The concept of Markov chains is briefly reviewed and desirable properties like irreducibility and aperiodicity are explained. Markov chain Monte Carlo methods are introduced by the Metropolis algorithm. Finally, a numerical example is provided and convergence is shown.

Markov chain Monte Carlo methods are well-suited for applications in physical sciences for two reasons. On the one hand, problems are generally of a high-dimensional type as there is a large number of uncertain quantities. On the other hand, these random variables are determined by complex probability density functions, e.g., by solutions of Kramer's equation.

With the increasing interest in machine learning, Markov chain Monte Carlo methods have become more popular. Some areas that already incorporate sampling methods are computer vision, web statistics, speech and audio processing, etc.

Appendix A

Results of Monte Carlo simulation for approximating π

Table A.1: Monte Carlo method for approximating π

	Expected value $I(h)$ with N draws				
Runs	$N = 10^2$	$N = 10^3$	$N = 10^4$	$N = 10^5$	$N = 10^6$
1	3.04	3.096	3.16	3.14348	3.14143
2	3.08	3.184	3.1008	3.14856	3.14289
3	3.28	3.164	3.1584	3.13588	3.13992
4	2.92	3.1	3.1524	3.1432	3.14136
5	3.08	3.204	3.1584	3.13492	3.1407
6	3.12	3.264	3.152	3.129	3.14438
7	3.0	3.184	3.148	3.1448	3.14312
8	3.36	3.016	3.164	3.13912	3.14235
9	3.2	3.16	3.1084	3.14288	3.14062
10	2.8	3.144	3.1528	3.14264	3.13989
Arithmetic mean	3.088	3.1516	3.14552	3.14045	3.14167

Bibliography

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

[2] J. Dongarra and F. Sullivan. Guest editors introduction: The top 10 algorithms. *Computing in Science & Engineering*, 2(1):22–23, 2000.

[3] H. Haario, E. Saksman, and J. Tamminen. Adaptive proposal distribution for random walk Metropolis algorithm. *Computational Statistics*, 14(3):375–396, 1999.

[4] H. Kahn and A. W. Marshall. Methods of reducing sample size Monte Carlo computations. *Journal of the Operations Research Society of America*, 1(5):263–278, 1953.

[5] P.-S. Koutsourelakis. Lecture notes on Uncertainty Modeling in Engineering, 2017.

[6] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[7] N. Metropolis and S. Ulam. The Monte Carlo Method. *Journal of the American statistical association*, 44(247):335–341, 1949.

[8] W. H. Press. *Numerical Recipes 3rd edition: The Art of Scientific Computing*. Cambridge university press, 2007.