Ivo F. Sbalzarini & Christoph Zechner

Stochastic Modeling and Simulation

Lecture Notes

TU Dresden, Faculty of Computer Science
Center for Systems Biology Dresden

Prof. Dr. Ivo F. Sbalzarini & Dr. Christoph Zechner

Winter 2021/22

This page is intentionally left blank

Contents

Foreword

These lecture notes were created for the course “Stochastic Modeling and Simulation”, taught as part of the mandatory electives module “CMS-COR-SAP” in the Masters Program “Computational Modeling and Simulation” at TU Dresden, Germany. The notes are based on handwritten notes by Prof. Sbalzarini and Dr. Zechner, which have been typeset in LaTeXby Fahad Fareed as a paid student teaching assistantship during his first term of studies in the Masters Program “Computational Modeling and Simulation” at TU Dresden, Germany, and subsequently extended and corrected by Prof. Sbalzarini and Dr. Zechner.

Chapter 1

Introduction

The goal of stochastic Modeling is to predict the (time evolution of the) probability over system states:

$$P(\vec{X}, t|\vec{X_0}, t_0). \tag{1.1}$$

This is the probability that a system which was in state $\vec{X_0}$ at time t_0 is found in state $\vec{X}$ at time t. Here, both $\vec{X}$ and $\vec{X_0}$ are random variables. In a deterministic model, it is possible to predict exactly which state a system is going to be found in. In contrast, a stochastic model does not predict states, but only probabilities of states. One hence never knows for sure which state the system is going to be in, but one can compute the probability of it to be a certain state. For example, in stochastic weather modeling, the model is not going to predict whether it is going to rain or not, but rather the probability of rain (e.g., 60% chance of rain tomorrow).
A stochastic simulation then generates specific realizations of state trajectories:

$$\vec{x_0}, \vec{x_1}, \vec{x_2}, \ldots, \vec{x_t}, \tag{1.2}$$

which are simulated from the model in Eq. 1.1 using a suitable stochastic simulation algorithm, such that for each time point t:

$$\vec{x_t} \sim P(\vec{X}, t|\vec{X_0}, t_0), \tag{1.3}$$

i.e., the states are distributed according to the desired probability function.
While this sounds pretty straightforward conceptually, there are a number of technical difficulties that frequently occur, among them:

- $P(\cdot)$ may not be known in closed form, but only through a governing (partial) differential equation that is often not analytically solvable $\rightarrow$ Master equations.

- Generating random numbers from the correct distribution $\sim P(\cdot)$ is often not directly possible $\rightarrow$ sampling algorithms.

- Time may be a continuous variable in the model, so there exists no natural discrete sequence of states $\vec{X}_1, \vec{X}_2, \ldots$, but rather a continuous $\vec{X}(t) \to$ stochastic calculus.
- The model probability $P(\cdot)$ may itself depend on system history $\to$ non-Markovian processes.

In the rest of this course, we will go through all of these points and show some commonly used solutions. We first consider the easier case when time is discrete, so $\vec{X}_1, \vec{X}_2, \ldots, \vec{X}_t$ with $t \in \mathbb{N}$. Then, we introduce classic algorithms (some of them from the very dawn of computing) before generalizing to continuous time. Before starting, however, we are going to refresh some basic concepts from probability theory, which will be useful for understanding the algorithms introduces later. We do not provide a comprehensive treatment of probability theory, but simply recall some basic principles and revisit them by means of example. We generally consider this material a prerequisite to the course, but still provide it here for the sake of completeness and of being explicit about the prerequisites. Students who do not feel savvy in the basics reviewed in this chapter are recommended to revise basic probability theory before continuing with the course.

1.1 Elementary Probabilities

We start by defining some basic vocabulary:

Definition 1.1 (Elementary Probability)**.** *We define the following terms:*

- Population*: A collection of objects.*
- Sample*: A subset of a Population (possibly random).*
- Experiment*: The process of measuring certain characteristics of a sample.*
- Outcome/Event*: The concretely measured values from an experiment.*
- Sample Space*: The space of all possible outcomes (usually denoted Ω).*
- Probability*: The likelihood of occurrence of an outcome in the sample space (usually denoted $P(A)$ for outcome A).*

1.1.1 Events and Axioms

All of probability theory is derived from three axioms. Axioms are postulates that cannot be proved within the framework of the theory, but from which all the rest of the theory can be proven if one accepts them as true. There are slightly different versions in the literature of stating the basic axioms of probability theory, but they are all equivalent to the following:

1. $P(A) \geq 0 \ \forall A$,

2. $P(\Omega) = 1$,
3. $P(A \cup B) = P(A) + P(B)$ if $P(A \cap B) = \emptyset$.

From these three axioms, the rest of probability theory has been proven. The first axiom simply states that probabilities are always non-negative. The second axiom states that the probability of the entire sample space is 1, i.e., the probability that *any* of the possible events happens is 1. the third axiom states that the probability of either A or B happening, but not both, (i.e., an exclusive OR) is the sum of the probability that A happens and the probability that B happens. All three axioms perfectly align with our intuition of probability as "chance", and we have no difficulty accepting them.
A first thing one can derive from these axioms are the probabilities of logical operations between events. For example, we find:

$$\text{AND} : P(A \cap B) = P(A)P(B) \text{ if } A \text{ and } B \text{ are independent},$$

$$\text{XOR} : P(A \cup B) = P(A) + P(B) - P(A \cap B),$$

$$\text{NOT} : P(\neg A) = 1 - P(A).$$

From these three basic logical operations, all of Boolean logic can be constructed for stochastic events.
Our intuitive notion of probability as "chance of an event happening" is quantified by counting. Intuitively, we attribute a higher probability to an event if it has happened more often in the past. We can thus state the frequentist interpretation of probability:

$$P(A) = \frac{\#\text{ways/times } A \text{ happens}}{\text{total } \#\Omega} \tag{1.4}$$

as the fraction of the sample space that is covered by event A. Therefore, $P(A)$ can conceptually simply be determined by enumerating all events in the entire sample space and counting what fraction of them belongs to A. For example, the sample space of rolling a fair dice is $\Omega = \{1, 2, 3, 4, 5, 6\}$. If we now define A the event that an *even* number of eyes shows, then we have $A = \{2, 4, 6\}$ and hence $\#A = 3$ whereas $\#\Omega = 6$ and, therefore according to the above definition of probability: $P(A) = 3/6 = 1/2$. Often, however, it is infeasible to explicitly enumerate all possible events and count them because their number may be very large. The field of combinatorics then provides some useful formulas to compute the total number of events without having to explicitly list all of them.

1.1.2 Combinatorics

Combinatorics can be used to compute outcome numbers even when explicit counting is not feasible. The basis of combinatorics is the *multiplication principle*: if experiment 1 has m possible events and experiment 2 has n possible events, then the total number of different events over both experiments is mn.

Example 1.1. *Let experiment 1 be the rolling of a dice observing the number of eyes shown. The* $m = 6$ *possible events of experiment 1 are:* $\{1, 2, 3, 4, 5, 6\}$*. Let experiment 2 be the tossing of a coin. The* $n = 2$ *possible events of experiment 2 are: {head, tail}. Then, the combination of both experiments has* $mn = 12$ *possible outcomes: {1-head, 1-tail, 2-head, 2-tail, ..., 6-head, 6-tail}.*

There are two basic cases to compute numbers of combinations:

- **Permutations**: Number of distinct *arrangements* of r out of n distinct objects, where order matters

$$P_{n,r} = \frac{n!}{(n-r)!} \tag{1.5}$$

- **Combinations**: Number of distinct groups (ordering within the group does not matter) of r objects that can be chosen from a total of n objects

$$C_{n,r} = \frac{n!}{r!(n-r)!} = \binom{n}{r}. \tag{1.6}$$

1.1.3 Probability Spaces (Ω, P)

Combining a finite sample space Ω with a function P, called *probability measure*, that assigns to each event in Ω its probability of happening, yields the notion of a *probability space* (Ω, P). Therefore, a probability space is the space of all possible events including their corresponding probabilities.

Example 1.2. *The probability space for the experiment of rolling a fair dice and observing the number of eyes shown is:* $(\{1, 2, 3, 4, 5, 6\}, \{\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\})$.

A notion that is frequently used in conjunction with probability spaces is that of a *σ-Algebra*. While the concept is not essential for this course, we state it briefly for completeness.

Definition 1.2 (σ-Algebra)**.** *A* σ-Algebra *is a collection of subsets of* Ω *that is closed under union, intersection, and complement.*

Example 1.3. *Let* Ω *= (a,b,c,d). Then the following is a σ-Algebra:* $\Sigma = \{\emptyset, \{a, b\}, \{c, d\}, \{a, b, c, d\}\}$*. Any intersection, union, or complement of any of these sets again yields a member of* Σ*. Please go ahead and convince yourself of this. Of course, this particular* Σ *is just one of several σ-Algebras that can be defined on this sample space.*

σ-Algebras allow us to restrict a probability space to (Σ,P) while still be able to construct all elementary events and have the same information represented via the Boolean formulas above.

1.2 Conditional Probabilities

A conditional probability expresses the probability of an event given/knowing another event has happened. For example, we might want to know the probability of rain tomorrow. After waking up in the morning, we see that the sky is overcast, but there is (still?) no rain. So now we want to know the probability that there will be rain given the sky is overcast. This conditional probability is likely different (higher) than the probability we estimated the day before when we did not yet know what the sky looks like the next morning. Conditional probabilities hence include the effect of certain events on the chance of an uncertain event.

1.2.1 Definition and properties

Definition 1.3 (Conditional Probability)**.** *The* conditional probability *of an event A given an event B is:*

$$P(A|B) = \frac{P(A \cap B)}{P(B)}. \tag{1.7}$$

The symbol | means "given", so $A|B$ is the event "A given B".

Example 1.4. *As an example, consider the standard deck of 52 playing cards. Now we might ask:*

1. *What is the probability that the other player has an Ace of Hearts given s/he told us that the card is red? We have:*

   $$P(\textit{Ace of Hearts} \mid \textit{card is red}) = \frac{1/52}{1/2} = \frac{1}{26}.$$

   So, knowing the card is red doubles the probability of it being an Ace of Hearts, because Hearts are red.

2. *What is the probability that the card is a King given it is red? We have:*

   $$P(\textit{King} \mid \textit{card is red}) = \frac{2/52}{1/2} = \frac{1}{13}.$$

   Note that in this example, knowing that the card is red does not change the probability of it being a King, since we also have:

   $$P(\textit{King}) = \frac{4}{52} = \frac{1}{13},$$

   which is the same since there are 4 Kings in the deck in total, two of them red and two black.

In the second part of the above example, the probability and the conditional probability are the same. We then say that the two events are *independent.* That is, the probability of a card being a King is independent of the probability of it being red, because there are just as many red Kings as black Kings.

Definition 1.4 (Independence)**.** *Two events A and B are called* independent *if and only if:*

$$P(A|B) = P(A).$$

This definition implies that for independent events A and B, we have

$$P(A \cap B) = P(A)P(B),$$

which directly follows from combining Definitions 1.3 and 1.4.
Using this result, we can now also write the logical AND for general, non-independent events as:

$$P(A \cap B) = P(A|B)P(B) = P(A, B), \tag{1.8}$$

which is called the *joint probability* of A and B, i.e., the probability that both A and B happen.

1.2.2 Bayes' Theorem

An important theorem for probabilities is Bayes' Rule. It relates conditional probabilities of two events in both orderings, i.e., it relates the probability of "A given B" to the probability of "B given A". This is useful, because one is often interested in the one, but can only measure or observe the other. Bayes' Theorem holds in general and not only for independent events. It is stated as:

Theorem 1.1 (Bayes)**.** *For any two random events A and B, there holds:*

$$\boxed{P(A|B) = \frac{P(B|A)P(A)}{P(B)}.} \tag{1.9}$$

In this theorem, the probability $P(A|B)$ is called "Posterior", $P(B|A)$ is called "Likelihood", and $P(A)$ is called "Prior".
Bayes' Theorem follows directly from the following identity:

$$P(A, B) = P(A|B)P(B) = P(B|A)P(A)$$

when solving for $P(A|B)$. Independence is not assumed.

1.2.3 Law of Total Probabilities

Another useful identity for conditional probabilities is the law of total probability. Let $B_1, B_2, \ldots, B_n$ be a set of mutually disjoint events, i.e., $P(B_i \cap B_j) = \emptyset \ \forall \ i \neq j$ and $B_1 \cup B_2 \cup \ldots \cup B_n = \Omega$. Then, for any event A, we have:

$$P(A) = P(A|B_1)P(B_1) + P(A|B_2)P(B_2) + \ldots + P(A|B_n)P(B_n). \tag{1.10}$$

This follows directly from the logical OR between events, Eq. 1.8, and the fact that the B_i are mutually disjoint.

1.2.4 Probability Expansion

For any set of events $A_1, A_2, \ldots A_n$ we can write the probability that all of them happen as:

$$P(A_1 \cap A_2 \cap \ldots \cap A_n) = P(A_1)P(A_2|A_1)P(A_3|A_1 \cap A_2) \ldots \\ P(A_n|A_1 \cap A_2 \cap ... \cap A_{n-1}). \quad (1.11)$$

This formula is known as conditional probability expansion and will be very important when we consider random processes later on. For now, consider the following classic example:

Example 1.5. *The "Birthday Problem": Assume there are n, $2 \leq n \leq 365$ people in a room, each equally likely to be born on any day of the year (ignoring leap years). Now define the events:*

- *A_i: "the birthday of person $i+1$ is different from those of all persons $1, \ldots, i$. Using simple counting (i.e., how many days of the year are still available to choose from for a different birthday), we find:*

$$P(A_1) = \frac{365-1}{365}$$
$$P(A_2|A_1) = \frac{365-2}{365}$$
$$\vdots$$
$$P(A_i|A_1 \cap A_2 \cap \ldots \cap A_{i-1}) = \frac{365-i}{365}.$$

  Note that in this case it is easy to write down that conditional probabilities knowing that the birthdays of all people before are different. Directly writing down $P(A_i)$ would not be so easy.

- *Now consider the event B: "at least two people in the room share a birthday". Clearly, it is:*

$$P(B) = 1 - P(A_1 \cap A_2 \cap \ldots \cap A_{n-1}).$$

  Using the probability expansion from Eq. 1.11, this is:

$$P(B) = 1 - P(A_1)P(A_2|A_1)P(A_3|A_1 \cap A_2) \ldots P(A_{n-1}|A_1 \cap A_2 \cap \ldots \cap A_{n-2})$$
$$P(B) = 1 - \prod_{i=1}^{n-1} \frac{365-i}{365},$$

  providing us with an easily computable formula. The results of this are somewhat counter-intuitive, as the probability of shared birthdays is higher than we intuitively tend to expect given the year has 365 days to choose from. Indeed, already for $n = 23$ people, we have $P(B) = 0.5$ and for $n = 50$ people $P(B) = 0.97$, hence a 97% chance that at least two share a birthday.

1.3 Random Variables

The notion of events is very general and, as we have seen in the above example, events can be described in words in arbitrary ways. In order to enable easier mathematical treatment and numerical computation, however, we often restrict ourselves to events that can be expressed by numbers. We then define:

Definition 1.5 (Random Variable)**.** *A* random variable *(RV) is a number assigned to the outcome of an experiment.*

Example 1.6. *When rolling a fair dice, the following random variables can, e.g., be defined:*

- *the number of eyes shown*
- *the sum of eyes shown over the past 10 rolls*
- *the number of times a 5 showed*
- ...

In the following, we denote RVs by uppercase latin characters and the values they assume by the corresponding lowercase character. Depending on the set of numbers from which the values of a random variable come, we distinguish three types of random variables: indicator/binary RV, discrete RV, and continuous RV.

1.3.1 Indicator/binary random variables

An indicator RV I only takes values from the set $\mathbb{B} = \{0, 1\}$. It therefore is a binary variable. It is defined as:

$$I : \Omega \to \mathbb{B} \qquad A \mapsto I(A) = \begin{cases} 1 & \text{if } A \\ 0 & \text{else.} \end{cases} \tag{1.12}$$

Indicator RVs directly generalize the concept of events to random variables with an easy mapping:

$$\begin{aligned} P(I(A) = 1) &= P(A) \\ I(A \cap B) &= I(A)I(B) \\ I(A \cup B) &= I(A) + I(B) - I(A)I(B) \end{aligned}$$

1.3.2 Discrete random variables

A discrete RV X takes values from a finite or countably infinite set of outcomes $\mathbb{S} = \{x_1, x_2, \ldots\}$. It is defined as:

$$X : \Omega \to \mathbb{S} \qquad A \mapsto X(A) = x_A \in \mathbb{S}. \tag{1.13}$$

For discrete random variables, we can write $P(X = x_i)$ the probability that it assumes one of the possible values x_i.

1.3.3 Continuous random variables

A continuous RV X takes values from a continuum, i.e., any subset $\mathbb{J} \subset \mathbb{R}$ of the set of real numbers $\mathbb{R}$. It is defined as:

$$X : \Omega \to \mathbb{J} \tag{1.14}$$

In this case, it is not possible any more to directly map to events, because there are infinitely many events/points in $\mathbb{J}$, each with infinitesimal probability.

1.4 Probability Distributions

Probability distributions are mathematical functions that assign probability to events, i.e., they govern how the total probability mass of 1 is distributed across the events in the sample space, or what the probabilities are of the possible values of a RV. Again, we distinguish the discrete and the continuous case.

1.4.1 Discrete Distributions

For discrete RVs, each possible outcome $x_i \in \mathbb{S}$ can directly be assigned a probability, as:

$$P(X = x_i) = P_X(x_i). \tag{1.15}$$

The function $P_X(x)$ is called the *probability mass function* (PMF) or the *probability distribution function* of the RV X. For discrete RV, we can also define the *Cumulative Distribution Function* (CDF) $F_X(x)$ of the RV X, as:

$$F_X(x) = P(X \leq x) = \sum_{x_i \leq x} P(X = x_i). \tag{1.16}$$

Note that by definition $0 \leq F_X(x) \leq 1$ and $F_X(x)$ is monotonically increasing, as it simply is a cumulative sum over probabilities, which are all non-negative by definition.

Example 1.7. *Consider the experiment of rolling a fair dice once, and define the RV X: number of eyes shown. Then, the probability mass function is:*

$$P_X(x) = \begin{cases} \frac{1}{6} & \textit{if } x \in \{1,2,3,4,5,6\} \\ 0 & \textit{else} \end{cases}$$

and the cumulative distribution function of X is:

$$F_X(x) = \frac{x}{6} \qquad x \in \{1,2,3,4,5,6\}.$$

1.4.2 Continuous Distributions

For continuous RVs, probability mass functions do not exist. However, since there are infinitely many infinitesimally small probabilities, we can define the *probability density function* (PDF), as:

$$f(x) = \frac{\mathrm{d}F(x)}{\mathrm{d}x}, \tag{1.17}$$

using the analogy between summation and integration in continuous spaces. Note that this is **not** a probability. Rather, for any $a, b \in \mathbb{J}$, we have:

$$P(a \leq X \leq b) = \int_a^b f(x)\,\mathrm{d}x. \tag{1.18}$$

Therefore, the probability of the RV taking values in a certain interval is given by the area under the curve of the PDF over this interval. Obviously, from the definition of the PDF:

$$P(X \in \mathbb{J}) = \int_{-\infty}^{\infty} f(x)\,\mathrm{d}x = 1, \tag{1.19}$$

so the total probability is correct. We can also compute the CDF from the PDF by inverting Eq. 1.17:

$$F_X(x) = \int_{-\infty}^{x} f(\tilde{x})\,\mathrm{d}\tilde{x}. \tag{1.20}$$

Finally, since the CDF is monotonic, its slope is always non-negative, thus:

$$f(x) \geq 0 \quad \forall x. \tag{1.21}$$

1.4.3 Joint and Marginal Distributions

Considering more than one RV, one can define joint, marginal, and conditional distributions analogously to joint and conditional probabilities. Obviously, all RVs must be of the same type (continuous/discrete). In the following, we give the definitions for two RVs.

- **Joint**: The joint CDF of two RVs X and Y is:

  $$F_{X,Y}(x, y) = P(X \leq x, Y \leq y). \tag{1.22}$$

  For discrete RVs, the joint PMF is:

  $$P_{X,Y}(x, y) = P(X = x, Y = y) = P(X = x \cap Y = y) \quad \text{for discrete } X, Y \tag{1.23}$$

  and for continuous RVs, the joint PDF is:

  $$f_{X,Y}(x, y) = \frac{\mathrm{d}^2}{\mathrm{d}x\mathrm{d}y} F_{X,Y}(x, y) \quad \text{for continuous } X, Y. \tag{1.24}$$

- **Marginal**: The marginal distribution over one of the two RVs is obtained from their joint distribution by summing or integrating over the other RV. For the CDF, this again looks the same for discrete and continuous RVs:

$$F_X(x) = P(X \leq x) = P(X \leq x, Y \leq \infty) = F_{X,Y}(x, \infty). \tag{1.25}$$

  And for the other distributions:

$$\text{Discrete PMF: } P_X(x) = \sum_y P_{X,Y}(x, y)$$

$$\text{Continuous PDF: } f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x, y)\,\mathrm{d}y.$$

- **Conditional**: The conditional probability distribution is the distribution of one of the two RVs given the other. Again, the conditional CDF is the same for continuous and discrete RVs:

$$F_{X|Y}(x|y) = P(X \leq x | Y \leq y). \tag{1.26}$$

  And for the other distributions:

$$\text{Discrete PMF: } P_{X|Y}(x|y) = \frac{P_{X,Y}(x, y)}{P_Y(y)}$$

$$\text{Continuous PDF: } f_{X|Y}(x|y) = \frac{f_{X,Y}(x, y)}{f_Y(y)}.$$

  Note that the conditional distribution is the joint distribution divided by the marginal distribution of the given RV, i.e., of the condition.

1.4.4 Moments of Probability Distributions

It is powerful to consider the moments of different order of a probability distribution. On the one hand, this is because these moments correspond to descriptive summary statistics of the RV governed by the distribution. On the other hand, it is often easier to write models for the moments of a distribution than for the distribution itself. It may even be the case that a distribution is not known or cannot be modeled explicitly, but one can still compute or model (some of) its moments. A fundamental theorem in mathematics guarantees that knowing all moments of a function is equivalent to knowing the function itself. This is because moments relate to Taylor expansions of the function. Often in stochastic modeling, considering just the first few moments of a probability distribution provides a good-enough approximation for a simulation.
The moment of order μ of a discrete RV X is defined as:

$$M_\mu[X] = \sum_x x^\mu P_X(x). \tag{1.27}$$

And for a continuous RV X:

$$M_\mu[X] = \int_{-\infty}^{\infty} x^\mu f_X(x)\,\mathrm{d}x. \tag{1.28}$$

Considering the first three moments, we find:

- $M_0[X] = 1$ because the total probability over the entire sample space is always 1.
- $M_1[X] = \mathbb{E}[X]$, which is the *expectation value* of RV X.

  Example 1.8. *Rolling a fair dice. X is the number of eyes shown. And from Eq. 1.27, we find:*

  $$\mathbb{E}[X] = \sum_{i=1}^{6} \frac{i}{6} = 3.5.$$

  This is the expected value when rolling the dice many times. It is related to the statistical mean of the observed values of X.

- $M_2[X] = \mathrm{Var}(X) + \mathbb{E}[X]^2 = \mathbb{E}[X^2]$, where Var is the *variance* of the RV X.

  Example 1.9. *For the same dice example, we find:*

  $$\mathbb{E}[X^2] = \sum_{i=1}^{6} \frac{i^2}{6} = 15.167$$

  and therefore:

  $$Var(X) = 15.167 - (3.5)^2 = 2.9167.$$

Higher moments relate to higher cumulants of the distribution, like skewness, Kurtosis etc. One can also define central moments that are shifted by the expectation value, as this is sometimes more convenient. We refer to the literature for these additional concepts.
The most important moments of a RV are the expectation value and the variance. They behave as follows when computed over sums or affine transforms of RVs:

$$\mathbb{E}[aX + b] = a\mathbb{E}[X] + b \tag{1.29}$$
$$\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y] \tag{1.30}$$
$$\mathrm{Var}(aX + b) = a^2\mathrm{Var}(X) \tag{1.31}$$
$$\mathrm{Var}(X + Y) = \mathrm{Var}(X) + \mathrm{Var}(Y) + 2\mathrm{Cov}(X, Y), \tag{1.32}$$

where $\mathrm{Cov}(X, Y)$ is the covariance of the two random variables (related to their correlation).

1.5 Common Examples of Distributions

We provide some examples of commonly used and classic probability distributions with their key properties and results. Keep in mind, though, that this is only a very small selection of the many more distributions that are known and characterized, like Gamma, Beta, Chi-squared (χ^2), Geometric, Weibull, Gumbel, etc.

1.5.1 Common discrete distributions

1. **Binomial (Bernoulli)**

$$P(X=k)=\binom{n}{k}p^k(1-p)^{n-k} \qquad k\in\{0,1,2,3,...,n\}. \tag{1.33}$$

For $X\sim \mathrm{Bin}(n,p)$:

$$\mathbb{E}[X]=np \tag{1.34}$$
$$\mathrm{Var}(X)=np(1-p). \tag{1.35}$$

The binomial distribution with parameters (n,p), evaluated at k, gives the probability of observing exactly k successes from n independent events, each with probability p to succeed; $P(X=k)=P(\text{exactly } k \text{ successes})$.

Example 1.10. *Imagine an exam with 20 yes/no questions. What is the probability of getting all answers right by random guessing?*

$$X\sim Bin(20,\frac{1}{2})\Rightarrow P(X=20)=\binom{20}{20}\cdot 0.5^{20}\cdot 0.5^{0}=1\cdot\frac{1}{2^{20}}\cdot 1=9.537\cdot 10^{-7}.$$

You would have to retake the exam 1,048,576 times until you could expect a pass. Clearly not a viable strategy.

2. **Poisson**

$$P(X=k)=\frac{\mathrm{e}^{-\lambda}\lambda^k}{k!} \qquad k\in\{0,1,2,3,...\}=\mathbb{N}. \tag{1.36}$$

For $X\sim \mathrm{Poiss}(\lambda)$:

$$\mathbb{E}[X]=\lambda \tag{1.37}$$
$$\mathrm{Var}(X)=\lambda. \tag{1.38}$$

In the Poisson distribution, the variance and the mean are identical. It has only one parameter. The Poisson distribution gives the probability that a random event is counted k times in a certain time period, if the event's rate of happening (i.e., the expected number of happenings in the observation time period) is λ. It is therefore also called the "counting distribution".

Example 1.11. *Ceramic tiles crack with rate $\lambda = 2.4$ during firing. What is the probability a tile has no cracks?*

$$X \sim \mathit{Poiss}(2.4) \Rightarrow P(X=0) = \frac{\mathrm{e}^{-2.4} \cdot 1}{1} = 0.0907.$$

So only about 9% of tiles would survive and a better production process should be found that has a lower crack rate.

1.5.2 Common continuous distributions

1. **Uniform**

$$f_X(x) = \frac{1}{b-a} \qquad a \leq x \in \mathbb{R} \leq b \tag{1.39}$$

$$F_X(x) = \frac{x-a}{b-a} \qquad a \leq x \in \mathbb{R} \leq b. \tag{1.40}$$

For $X \sim \mathcal{U}(a,b)$:

$$\mathbb{E}[X] = \frac{1}{2}(a+b) \tag{1.41}$$

$$\mathrm{Var}[X] = \frac{1}{12}(b-a)^2. \tag{1.42}$$

The uniform distribution formalizes the case where all values within an interval $[a,b]$ over the real numbers are equally likely.

Example 1.12. *Random number generation $\rightarrow$ See next chapter.*

2. **Exponential**

$$f_X(x) = \lambda \mathrm{e}^{-\lambda x} \qquad x \geq 0 \tag{1.43}$$

$$F_X(x) = 1 - \mathrm{e}^{-\lambda x} \qquad x \geq 0. \tag{1.44}$$

For $X \sim \mathrm{Exp}(\lambda)$:

$$\mathbb{E}[X] = \lambda^{-1} \tag{1.45}$$

$$\mathrm{Var}[X] = \lambda^{-2}. \tag{1.46}$$

The exponential distribution governs waiting times between memoryless random events occurring at rate λ.

Example 1.13. *An engine has a probability to fail once in 100,000 km. How likely is it to last 200,000 km?*

$$X \sim \mathit{Exp}(1/100{,}000) \Rightarrow P(X > 200{,}000) = 1 - P(X \leq 200{,}000) =$$

$$1 - F_X(200{,}000) = 1 - (1 - \mathrm{e}^{-10^{-5} \cdot 2 \cdot 10^5}) = \mathrm{e}^{-2} = 0.1353.$$

So about 13% of all engines will reach the double of their mean lifetime.

3. **Normal/Gaussian**

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \qquad x \in \mathbb{R} \tag{1.47}$$

$$F_X(x) = \Phi\left(\frac{x-\mu}{\sigma}\right) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right]. \tag{1.48}$$

The CDF of the Gaussian distribution, Φ, has no analytical form, but can be computed numerically due to its relation with the *error function* erf$(\cdot)$, which is a so-called "special function" for which good numerical approximations exist. For $X \sim \mathcal{N}(\mu, \sigma^2)$:

$$\mathbb{E}[X] = \mu \tag{1.49}$$

$$\mathrm{Var}[X] = \sigma^2. \tag{1.50}$$

The normal distribution governs measurements with a true value of μ and measurement uncertainty/error of σ (called *standard deviation*).

Example 1.14. *IQs of people are distributed around $\mu = 100$ with a standard deviation of $\sigma = 15$. What is the probability of having an IQ > 110?*

$$X \sim \mathcal{N}(100, 15^2) \Rightarrow P(X > 110) = 1 - P(X \leq 110) =$$

$$1 - \Phi(110 - 100/15) = 1 - \Phi(2/3) = 1 - 0.7454 = 0.2546.$$

So about a quarter of all people have an IQ higher than 110. The value of Φ was taken from a table.

1.5.3 Scale-free distributions

A special class of distributions are the so-called "scale-free" distributions. Intuitively, we often ask ourselves: "Why does one so often observe rare events?". This is because our intuition likes normal distributions where occurrences are relatively tightly distributed around a mean (e.g., body sizes) and exponential distributions (e.g., time to wait for the bus). According to this intuition, earthquakes of magnitude 8 or 9 should basically never occur given that the mean earthquake magnitude is a 2.3. However, they do occur much more frequently than we expect. This is, of course, because earthquake magnitudes are not normally or exponentially distributed.

Distributions we find "intuitive" have a characteristic scale (μ for the normal and λ for the exponential distribution), which somewhat sets the order of magnitude of the values that occur. However, many real-world processes are scale-free with values spanning several orders of magnitude without any preferred "typical value". Examples include city sizes (from a few 100 to tens of millions without any preferential scale), wealth (from a few cents to billions of dollars), strength of earthquakes, stock market values, etc. In a scale-free distribution, very large and very small values are not rare.

Example 1.15. *A classic example of a scale-free distribution is the* power law:

$$P(x) = x^{-\alpha}, \quad \alpha > 0. \tag{1.51}$$

Another example is the Gutenberg-Wagner law:

$$\log N = a - bM, \tag{1.52}$$

for the frequency $N(M)$ of earthquakes of magnitude M. Also the log-normal distribution is an example of a scale-free distribution.

Scale-free distributions are typically the result of multiplicative processes where effects multiply instead of just adding up (e.g., stock market). Their moments are often infinite, such that a variance cannot really be computed (or it would be infinity), and sometimes even the mean cannot be computed. Nevertheless, scale-free distributions are very important, e.g., when studying criticality in physics, scaling in biology, or processes on networks in engineering.

Chapter 2

Random Variate Generation

All (and more than) you ever wanted to know about the topics covered in this chapter can be found in the book: "Non-Uniform Random Variate Generation" by Luc Devroye (Springer, 859 pages).

The most basic task in any stochastic simulation is the ability to simulate realizations of a random variable from a given/desired probability distribution. This is called *random variate generation* or *simulation of random variables.* On electronic computers, which are deterministic machines, the simulated random variates are not truly random, though. They just appear random (e.g., by statistical test) and follow the correct distribution, which is why we say that the random variable is "simulated" and it is not the real random variable itself. A notable exception is special hardware, such as crypto cards, that generates true random numbers. This often involves a radioactive source, where the time between decay events is truly random and exponentially distributed.

2.1 Transformation of Random Variables

Many simulation methods for random variables are based on transformation of random variables. We therefore start by providing the basic background on random variable transformation. Let a random variable X be given and define a second random variable from it, as:

$$Y = g(X)$$

for a given transformation function g with $\mathrm{supp}(g) \supseteq \mathrm{dom}(X)$, i.e., the support of the function g has to include the entire domain of the RV X. For a valid transform, the inverse

$$g^{-1}(A) := \{x : g(x) \in A\} \tag{2.1}$$

for any set A exists, but is not necessarily unique. The function g maps from the space in which X takes values to the space in which Y takes values.

Example 2.1. *Consider the following transformation functions:*

- $g(x) = x^3$
$$\Rightarrow \quad g^{-1}(\{1,8\}) = \{1,2\}. \tag{2.2}$$
In this case, the inverse is unique, as g is a bijection.

- $g(x) = \sin x$
$$\Rightarrow \quad g^{-1}(0) = k\pi; \quad k \in \mathbb{Z}. \tag{2.3}$$
Here, the inverse in not unique and maps to a countably infinite set.

The map $A \mapsto P(g(X) \in A) = P(X \in g^{-1}(A))$ satisfies the axioms of probability and therefore allows us to define probability distributions over $Y = g(X)$. How these distributions are defined, and what they look like, depends on whether (X, Y) are continuous or discrete random variables.

2.1.1 Transformed discrete distributions

For X and Y discrete with probability mass functions $P_X(x), P_Y(y)$, we have for $Y = g(X)$:
$$P_Y(y) = \sum_{x \in g^{-1}(y)} P_X(x). \tag{2.4}$$

Example 2.2. *Let X be uniformly distributed over the integers from 1 to n. Then, the probability mass function of X is:*
$$P_X(x) = \begin{cases} \frac{1}{n} & x \in \{1, 2, \dots, n\} \\ 0 & \textit{else.} \end{cases}$$
Now consider the transformation $Y = X + a$, which adds a given constant a to each realization of X. From Eq. 2.4, we find:
$$P_Y(y) = \begin{cases} \frac{1}{n} & y \in \{a+1, \dots, a+n\} \\ 0 & \textit{else.} \end{cases}$$

2.1.2 Transformed continuous distributions

Let X and Y be continuous random variables with the CDF of X being $F_X(x)$. Further assume that g is uniquely invertible, i.e., it is a bijection. While this assumption is not necessary, the case where g is not a bijection (e.g., the sin example above) is more difficult to treat and omitted here. Since g is a bijection, it is a monotonic function. There are two cases:
For monotonically increasing g, we have for the CDF of $Y = g(X)$:
$$F_Y(y) = P(Y \le y) = P(g(X) \le y) = P(X \le g^{-1}(y)) = F_X(g^{-1}(y)) \tag{2.5}$$
and for the PDF:
$$f_Y(y) = F_Y'(y) = f_X(g^{-1}(y)) \frac{\mathrm{d}}{\mathrm{d}y} g^{-1}(y), \tag{2.6}$$

Where we have used the chain rule of differentiation in the last step.
For monotonically decreasing g, we analogously find:

$$F_Y(y) = P(g(X) \leq y) = P(X > g^{-1}(y)) = 1 - F_X(g^{-1}(y)) \tag{2.7}$$

$$f_Y(y) = -f_X(g^{-1}(y)) \frac{\mathrm{d}}{\mathrm{d}y} g^{-1}(y) \tag{2.8}$$

This is easily understood by drawing the graphs of g for the two cases and observing that the half-space $Y \leq y$ gets mapped to $X \leq g^{-1}(y)$ in one case, and to $X > g^{-1}(y)$ in the other.

Example 2.3. *Let $X \sim \mathcal{U}(0,1)$. The PDF of this continuous random variable is:*

$$f_X(x) = \begin{cases} 1 & \textit{if } x \in [0,1] \\ 0 & \textit{else.} \end{cases}$$

Now consider the transformed random variable $Y = g(X) = 1 - X$. This function in uniquely invertible as:

$$y = g(x) = 1 - x \quad \Rightarrow \quad g^{-1}(y) = 1 - y \quad \textit{for } x, y \in [0,1].$$

The function g is monotonically decreasing. From Eq. 2.8, we thus find:

$$f_Y(y) = -f_X(1-y)(-1) = \begin{cases} 1 & \textit{if } y \in [0,1] \\ 0 & \textit{else,} \end{cases}$$

which is the same as the PDF of X. We hence find the important result that if $X \sim \mathcal{U}(0,1)$, then also $Y = 1 - X \sim \mathcal{U}(0,1)$, i.e., the probability of a uniformly distributed event to not *happen, is also uniformly distributed.*

2.1.3 The Inversion Transform

The inversion transform is an important identity when generating random variates from a given continuous distribution. Let X be a continuous random variable. It's CDF $F_X(x)$ is monotonically increasing by definition. Now set the special transform $Y = g(X) = F_X(X)$. Then:

$$\begin{aligned} F_Y(y) &= P(Y \leq y) = P(F_X(X) \leq y) \\ &= P(X \leq F_X^{-1}(y)) = F_X(F_X^{-1}(y)) = y. \end{aligned}$$

The distribution with $F_Y(y) = y$ is the uniform distribution over the interval $[0,1]$. Therefore, random variables from a given cumulative distribution F_X can be simulated from uniform ones by $X = F_X^{-1}(\mathcal{U}(0,1)) \sim F_X(x)$. This endows uniform random numbers with special importance for stochastic simulations.

2.2 Uniform Random Variate Generation

Due to the inversion transform, random variables of any given distribution can be simulated if we can simulate uniformly distributed random variables over the interval $[0, 1]$. We therefore first consider the problem of simulating $\mathcal{U}(0, 1)$ on a deterministic computer. There are two approaches to this: algorithmic or data-driven. Data-driven approaches include, e.g., measuring the time between two keystrokes of the user, taking every y-th byte of network traffic flowing through the network interface of the computer, or reading every x-th byte of data from a storage medium. Of course, neither the data on a storage medium, nor network traffic, nor user keystrokes are truly random, but they may appear random in the absence of a predictive model. Data-driven approaches are often used by cryptography applications because the sequence of numbers is truly unpredictable (even if it is not truly random). For simulations, however, the algorithmic approach is more practical, because it allows us to generate many random numbers quickly, not limited by the speed of a data stream, and it ensures reproducible simulation results. We hence focus on the algorithmic approach here.
Algorithms for random variable simulation are called *Random Number Generators* (RNG). Two types of RNGs exist:

- **pseudo-RNGs** use deterministic algorithms to produce sequences of numbers that appear random by statistical tests, but are predictable.
- **quasi-RNGs** use deterministic algorithms to produce low-discrepancy sequences, sampling more "evenly" while still appearing random but being predictable.

In both cases, we require:

$$\lim_{n\to\infty} |\hat{F}_n(x) - x| = 0 \quad \forall x \in [0, 1], \tag{2.9}$$

where $\hat{F}_n(x)$ is the empirical CDF over n samples from the RNG and $F(x) = x$ is the uniform CDF we want to simulate. This requirement means that if we generate infinitely many random numbers, then their CDF is identical to the CDF we want to simulate.

2.2.1 Uniform Pseudo-Random Number Generation

Uniform pseudo-random number generators over the interval $[0, 1]$ are available in all programming languages as basic intrinsic functions, and they are the core building block of any stochastic simulation. There exists a host of well-known pseudo-RNG algorithms. The classic example, which illustrates the working principle of pseudo-RNGs, is the **Linear Congruential Generator** (a.k.a. multiplicative congruential generator, Lehmer generator), which computes the sequence of numbers:

$$z_i = az_{i-1} \mod m, \quad i = 1, 2, 3, \dots \tag{2.10}$$

which means that it computes the division rest of az_{i-1}/m. By definition of the division rest, $|z_i| < m$. Thus:

$$u_i = \frac{z_i}{m} \in [0, 1). \tag{2.11}$$

The start value (called "seed") z_0 and the two integers $m > 0$ and $0 < a < m$ must be chosen by the user. There are also versions in the literature that include an additional constant shift, providing one more degree of freedom, but the general principle remains the same. It can be shown that for the linear congruential generator,

$$|\hat{F}_n(u) - u| \leq \varepsilon(m) \quad \forall u \tag{2.12}$$

with $\varepsilon(m) \downarrow 0$ as $m \to \infty$. Therefore, the linear congruential generator is a valid pseudo-RNG in the sense of Eq. 2.9. The question, remains, however, how to choose a and m. Usually (a, m) are chosen very large and mutually prime. As the upper error bound $\varepsilon(m)$ becomes smaller for larger m, it is good to choose m as large as possible. The following is a good choice on a 32-bit numeric type (C++11 standard):

$$a = 48\,271, \qquad m = 2^{31} - 1.$$

The linear congruential generator is very simple, easy to implement, and illustrates well how pseudo-RNGs work. However, much better pseudo-RNGs are known nowadays, and the linear congruential generator is not used any more in practice. Its main problem is the relatively short *cycle length*. Since the set of machine numbers (i.e., finite-precision floating-point numbers) is finite, the values u_i necessarily start to repeat after some point T. Therefore, u_T it the first number produced that is identical to an already seen number, say u_0. Since the algorithm of a pseudo-RNG is deterministic, the entire sequence $u_{T+1}, u_{T+2}, \ldots = u_1, u_2, \ldots$ repeats itself. Until it hits u_T again, and then it repeats itself again identically. Therefore, the number T is called the *cycle length* of the RNG. Every deterministic pseudo-RNG has a finite cycle length. Obviously, any stochastic simulation that requires more pseudo-random numbers than T is not going to advance any more beyond T, as it is simply going to recompute results it has already computed. In practice, stochastic simulations routinely require millions of random numbers. The linear congruential generator has a relatively short cycle length. The actual value of T depends on m and a. If m is a power of 2, then the cycle length cannot be more than $m/4$, with the low bits having an even shorter period than the high bits. Better algorithms with longer cycles are known and mostly used nowadays, e.g.:

- Mersenne Twister (1998) – Most commonly used
- Park-Miller (1988) – C/C++ std
- XOR-Shift (2003) – Good for GPU
- Xoroshiro128+ (2018) – Fastest on 64-bit CPU

They are all simple recursion formulas operating on integers or bit strings, which makes them very fast and easy to implement.
A particular problem occurs when using pseudo-RNGs in parallel computing. There, obviously, one wants that each processor or core simulates a different random number sequence. If they all compute the identical thing, the parallelism is wasted. The simplest way to have different random number sequences is to use a different seed on each processor or core. However, using different seeds may not change the cycle, nor its length, but could simply starts the parallel sequences at different locations in the cycle. So when using P processors, the effectively usable cycle length on each processor is T/P on average. Beyond this, processors recompute results that another processor has already computed before. Special *Parallel Random Number Generators* (PRNGs) therefore exist, which one should use in this case. They provide statistically independent streams of random numbers that have full cycle length on each processor and guarantee reproducible simulation results when re-running the simulation on different numbers of processors. We do not go into detail on PRNGs here, but refer to the literature and the corresponding software libraries available.

2.2.2 Uniform Quasi-Random Number Generation

It is a general property of uniformly distributed pseudo-random numbers that they form "clusters" and "holes" in high-dimensional spaces. If one for example generates uniform pseudo-random numbers (x, y) and plots them as points in the $x-y$ plane, they do not look homogeneously distributed. Instead, they form visible clusters, despite the fact that they are statistically uniformly distributed. Of course, for $n \to \infty$, the entire plane will be covered and the distribution is indistinguishable from the exact uniform distribution, as required. But for finite numbers of samples, the clustering is clearly visible.
Quasi-RNGs solve this problem by generating more evenly scattered samples with the same statistical properties. These sequences of numbers are also called *low-discrepancy sequences* because they aim to reduce a quality measure called "discrepancy":

Definition 2.1 (Discrepancy)**.** *The* discrepancy D_n *of a finite sequence of* n *numbers* x_i*,* $i = 1, 2, \dots, n$ *is:*

$$D_n = \max_{0 \le u \le 1} \left| \sum_{i=1}^{n} I_{[0,u]}(x_i) - u \right|, \tag{2.13}$$

where the indicator function over the interval $[0, u]$*,* $I_{[0,u]}(x) = \begin{cases} 1 & \textit{if } x \in [0, u] \\ 0 & \textit{else.} \end{cases}$

The sum over the indicator is the total number of samples in the interval $[0, u]$. The expected number of samples under the uniform distribution is u. The discrepancy therefore measures the largest deviation of the empirical CDF from the true CDF for any finite number n of samples. For small sample counts, the CDF of a low-discrepancy sequence converges faster to the true CDF than that

of a pseudo-random number sequence. However, low-discrepancy sequences may suffer from aliasing artifacts for large sample counts.
A simple classic quasi-RNG is the **Additive Recurrence Sequence**:

$$u_{i+1} = (u_i + \alpha) \mod 1 \tag{2.14}$$

with an irrational number α. This sequence has $D_n \in O_\varepsilon(n^{-1+\varepsilon})$, which means that the discrepancy scales inversely proportional to n. Good choices of α are: $\frac{1}{2}(\sqrt{5}-1)$ or $\sqrt{2}-1$.
The Additive Recurrence Sequence is easy to generate and understand, but is not the best low-discrepancy sequence known. Better ones include:

- Van-der-Corput sequence
- Halton sequence
- Faure sequence
- Niederreiter sequence
- Sobol sequence
- ...

We refer to the literature for more information about them.

2.3 Inversion Transform for Non-uniform Distributions

Using the result from Section 2.3, sequences of uniformly distributed pseudo- or quasi-random numbers can be transformed to another distribution with known and invertible CDF F_X, as:

$$X \sim F_X = F_X^{-1}(Y) \text{ with } Y \sim \mathcal{U}(0,1). \tag{2.15}$$

In the best case, the inverse CDF F_X^{-1} can be computed analytically and can directly be used as a transform. If this is not possible, numerical methods of inversion can be used, such as line search or Regula Falsi.

Example 2.4. *We want to generate exponentially distributed pseudo-random numbers $X \sim Exp(\lambda)$. From the CDF of the exponential distribution, we analytically find the inverse:*

$$F_X = 1 - e^{-\lambda x} = y \tag{2.16}$$

$$e^{-\lambda x} = 1 - y \tag{2.17}$$

$$-\lambda x = \log(1-y) \tag{2.18}$$

$$x = -\frac{1}{\lambda}\log(1-y) = F_X^{-1}. \tag{2.19}$$

As $y \sim \mathcal{U}(0,1)$, also $r = 1 - y \sim \mathcal{U}(0,1)$, according to Example 2.3.

$$\Rightarrow X = -\frac{1}{\lambda}\log(R) \text{ for } R \sim \mathcal{U}(0,1). \tag{2.20}$$

This simple formula then generates exponentially distributed pseudo-random numbers with parameter λ.

While the above example illustrates the concept of inversion transforms, practical cases are not always this simple. A prominent example is the generation of normally/Gaussian distributed pseudo-random numbers. A simple idea would be to use the Central Limit Theorem, stating that the average of a large number of random variables is normally distributed, regardless of how the individual random variables are distributed. So one could compute averages over large, independent collections of uniform pseudo-random numbers, and get a standard-normal variable:

$$X = \frac{1}{\sqrt{n/12}}\left(\sum_{i=1}^{n} R_i - \frac{n}{2}\right) \tag{2.21}$$

from n uniformly distributed $R_i \sim \mathcal{U}(0,1)$, $i = 1, 2, \dots n$. Here, $n/2$ is the mean of the n uniformly distributed values, and $n/12$ their variance (see Sec. 1.5.2). However simple, this method is not very good. Due to the slow convergence of the Central Limit Theorem, the number n needs to be very large in order to get a good approximation to the Gaussian distribution. Therefore, this method requires many (typically thousands) uniform random numbers to generate a single Gaussian random number, with obvious problems for the cycle length of the generator.

A better way of generating Gaussian random numbers is the *Box-Muller transform*. It is a classic inversion transform method. However, the CDF of the Gaussian distribution is not analytically invertible (in fact, the CDF is not an analytical function, see Section 1.5.2). But a little trick helps: considering two independent Gaussian random numbers (x_1, x_2) with mean 0 and variance 1 (i.e., standard-normal distribution) jointly as the coordinates of a point in the 2D plane, and converting to polar coordinates (r, θ):

$$r^2 = x_1^2 + x_2^2 \sim \chi^2(2) = \mathrm{Exp}(1/2)$$
$$\theta \sim \mathcal{U}(0, 2\pi).$$

The sum of squares of k independent standard Gaussian random variables is Chi-square distributed with parameter k, $\chi^2(k)$, which is $\chi^2(2)$ for 2 standard Gaussian random variables. The Chi-square distribution with parameter 2 is identical to the exponential distribution with parameter 1/2. And, since the 2D standard Gaussian is rotationally symmetric, the angle is uniformly distributed. From above, we know how to invert the exponential distribution, so we can analytically invert the coordinate transform and find:

$$x_1 = \cos(2\pi r_2)\sqrt{-2\log(r_1)}$$
$$x_2 = \sin(2\pi r_2)\sqrt{-2\log(r_1)}. \tag{2.22}$$

This is the *Box-Muller transform.* It generates two independent standard-normal random numbers (x_1, x_2) from two independent uniformly distributed random numbers $(r_1, r_2) \sim \mathcal{U}(0, 1)$, which makes it optimally efficient. Because it is based on an analytical inversion, it is also accurate to machine precision.
Note that the Box-Muller transform is not the only way of generating normally distributed random variates. Another famous example is the *Marsaglia polar method*, which avoids the evaluation of trigonometric functions. This may be preferred in practical implementations, since trigonometric functions are expensive to compute on CPUs, where they are typically approximated by finite series expansions. We derived the Box-Muller transform here because it is easiest to understand and illustrates well the inversion principle. The other standard normal transforms can be understood from it.
In descriptive statistics, there is also a class of transforms called *power transforms*, which are designed to make empirical data "look more normally distributed". Examples include the Box-Cox transform or the Yeo-Johnson transform. There are, however, not directly suited to simulate Gaussian random variables, as they are only approximate, i.e., they produce "Gaussian-like" distributions as suited for statistical tests of normality.
For the following distributions, pseudo-random number generators based on the inversion transform are also known: Poisson, Chi-square, Beta, Gamma, and Student t (see literature).

2.4 Accept-Reject Methods for Non-uniform Distributions

Accept-Reject methods are an algorithmic alternative to inversion-transform methods for simulating continuous random variables from non-uniform distributions. As they do not require the (analytical or numerical) computation of an inverse function, accept-reject methods are very general and easy to implement. They may, however, be rather inefficient, requiring multiple uniformly distributed random numbers to generate one random number from the target distribution. Accept-Reject methods are based on the following theorem:

Theorem 2.1. *Simulating $X \sim f_X(x)$ is equivalent to simulating:*

$$(X, U) \sim \mathcal{U}\{(x, u) : 0 < u < f_X(x)\}. \tag{2.23}$$

This is because

$$f_X(x) = \int_0^{f_X(x)} \mathrm{d}u,$$

but we omit the formal proof here. Theorem 2.1 provides a recipe for simulating random variables X with given PDF $f_X(x)$ using pairs of independent uniform random numbers (x, u), as illustrated in Fig. 2.1. The theorem tells us that generating random numbers from the PDF f_X can be done by uniformly

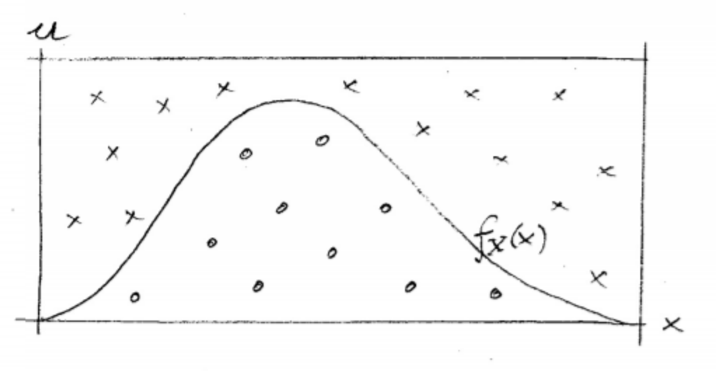

Figure 2.1: Illustration of the Accept-Reject method for simulating a random variable X with PDF f_X. Crosses are rejected, circles are accepted.

sampling (x, u) in the bounding box of f_X and only accept points for which $0 < u < f_X(x)$ (circles in Fig. 2.1). The x-component of the accepted point is then used as a pseudo-random number. Points above the graph of f_X (crosses in Fig. 2.1) are rejected and not used.
This is very easy to implement and always works. However, it may be inefficient if the area under the curve of f_X only fills a small fraction of the bounding box, i.e., $Vol(\{x, u\}) \gg 1$, which for example is the case if f_X is very peaked or has long tails. In particular, this method becomes difficult for PDFs with infinite support, such as the Gaussian, where one needs to truncate somewhere, incurring an approximation error in addition to inefficiency.
Fortunately, we are not limited to sampling points in the bounding box of f_X, but we can use any other probability density function $g(x)$ for which $f_X(x) \leq \mu g(x)$ for some $\mu > 0$, i.e., the graph of $\mu g(x)$ is always above the graph of f_X for some constant μ. Then, simulating $X \sim f_X(x)$ is equivalent to simulating pairs (y, u) such that:

$$y \sim g(x), \qquad u \sim \mathcal{U}(0, 1) \tag{2.24}$$

and accepting the sample $x = y$ if $u \leq f_X(x)/\mu g(x)$. Obviously, this requires one to be able to efficiently generate random numbers from the *proposal* distribution $g(x)$, so typically one wants to use a $g(x)$ for which an explicit inversion formula exists, like an exponential or Gaussian distribution.

2.4.1 Composition-Rejection Method

Another possibility of improving the efficiency (i.e., increasing the fraction of accepted samples) of the Accept-Reject method is *composition-rejection sampling*. The idea here is to bin the target distribution f_X into dyadic intervals, as illustrated in Fig. 2.2. The first bin contains all blocks of (x, u) for which $0 \leq f_X(x) \leq a$ for some constant a. The second bin contains all blocks with $a < f_X(x) \leq 2a$, the third $2a < f_X(x) \leq 4a$, and so on.

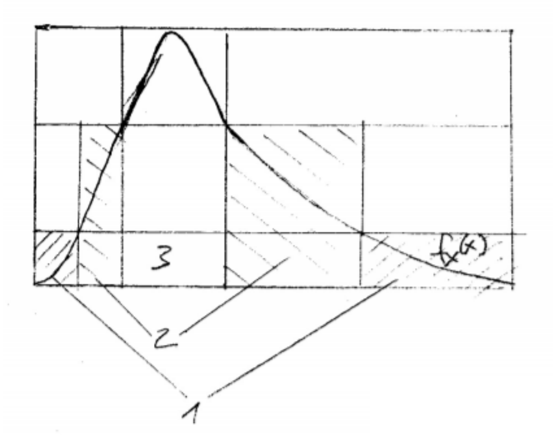

Figure 2.2: Illustration of composition-rejection sampling.

This is called dyadic binning because the bin limits grow as powers of two. If one then performs Accept-Reject sampling independently in these bins (i.e., first choose a bin proportional to its total probability, the do accept-reject inside that bin), the efficiency is better than when performing Accept-Reject sampling directly on f_X. This is obvious because the "empty" blocks that do not overlap with the area under the curve of f_X are never considered.

Chapter 3

Discrete-Time Markov Chains

Markov Chains are one of the most central topics in stochastic modeling and simulation. Both discrete-time and continuous-time variants exist. Since the discrete-time case is easier to treat, that is what we are going to start with. Markov Chains are a special case of the more general concept of stochastic processes.

3.1 Discrete-Time Stochastic Processes

In a discrete-time model, time is a discrete variable and can therefore be represented by integer numbers. The intuition is not to talk about time as a continuum, but about *time steps* or *time points*, such as for example years (2018, 2019, 2020, ...) or hours of the day (0, 1, 2, ..., 23).

Definition 3.1 (Discrete-Time stochastic process)**.** *A* stochastic process in discrete time $n \in \mathbb{N} = \{0, 1, 2, \ldots\}$ *is a sequence of random variables* $X_0, X_1, X_2, \ldots$ *denoted by* $\{X_n : n \geq 0\}$*. The value* x_n *of* X_n *is called the* state *of the process at time n; The value* x_0 *of* X_0 *is the initial state.*

So, in the most general case, a stochastic process is any sequence of random variables. In the discrete-time case, time is given by an integer index n. We further distinguish between:

1. *Discrete-state processes* where the random variables X_n take values in a discrete space (e.g., $X_n \in \mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$).

2. *Continuous-state processes* where the random variables X_n take values in a continuous space (e.g., $X_n \subset \mathbb{R}^d, d \geq 1$).

Definition 3.2 (State space)**.** *The* state space $\mathbb{S}$ *of a stochastic process* $\{X_n : n \geq 0\}$ *is the space of all possible values of the random variables* X_n*.*

Therefore, a discrete-state stochastic process has a discrete state space, whereas a continuous-state stochastic process has a continuous state space.
Stochastic processes are widely used to model the time evolution of a random phenomenon, such as:

- Population size in year n.
- Amount of water in n^{th} rain of the year.
- Time the n^{th} patient spends in hospital.
- Outcome of n^{th} rolling of a dice.

These are all discrete-time processes, some with discrete state space (e.g., dice) and some with continuous state space (e.g., amount of rain).
The challenge in stochastic modeling is to find a stochastic process model $\{X_n : n \geq 0\}$ that is complex enough to capture the phenomenon of interest, yet simple enough to be efficiently computable and mathematically tractable.

Example 3.1. *Consider the example of repeatedly tossing a fair coin and let X_n be the outcome (head or tail) of n^{th} toss. Then:*

$$X_n \sim Bin\left(\frac{1}{2}\right) \Rightarrow P(X_n = head) = P(X_n = tail) = \frac{1}{2} \quad \forall n.$$

In this example, all X_n in $\{X_n : n \geq 0\}$ are independent of each other and identically distributed (i.i.d.).

i.i.d. random processes are the easiest to deal with. They:

1. are defined by a single distribution,
2. obey the central limit theorem,
3. allow simply multiplying the probabilities of elementary events.

3.2 Markov Chains

For many phenomena, *i.i.d.* processes are not powerful enough to be a good model. For example, one would expect the population size of a country in year $n+1$ to be large if it was already large in year n. Often, one therefore wants a model where the distribution of X_{n+1} depends on the value x_n of X_n in order to allow for correlations between random variables in the stochastic process. However, allowing for all possible correlations between all n random variables would be computationally infeasible and in many cases overkill, as dependence on the current value of the stochastic process is often sufficient. This then defines:

Definition 3.3 (Markov chain)**.** *A discrete-time stochastic process* $\{X_n : n \geq 0\}$ *is called a* Markov Chain *if and only if for all times* $n \geq 0$ *and all states* $x_i, x_j \in \mathbb{S}$*:*

$$\begin{aligned} P(X_{n+1} = x_j | X_n = x_i, X_{n-1} = x_{i-1}, \dots, X_0 = x_0) \\ = P(X_{n+1} = x_j | X_n = x_i) = P_{ij}. \end{aligned} \tag{3.1}$$

This means that the probability distribution of the next state only depends on the current state, but not on the history of how the process arrived at the current state. Markov Chains are the next-more-complex stochastic process after *i.i.d.* processes. They implement a one-step dependence between subsequent distributions. Despite their simplicity, Markov Chains are very powerful and expressive, while still remaining mathematically tractable. This explains the widespread use and central importance of Markov Chains in stochastic modeling and simulation.
Equation 3.1 is called the *Markov Property.* For discrete-state processes (i.e., $\mathbb{S}$ is discrete), the number $0 \leq P_{ij} \leq 1$ is the probability to move to state x_j whenever the current state is x_i. It is called the *one-step transition probability* of the Markov Chain. For discrete $\mathbb{S}$ with finite $|\mathbb{S}|$, the matrix of all one-step transition probabilities $\mathbf{P} = (P_{ij})$, $\forall (i, j) : x_i, x_j \in \mathbb{S}$, is the *transition matrix* of the Markov Chain. It is a square, positive-semi-definite matrix.
We have $\sum_{j \in \mathbb{S}} P_{ij} = 1$, since upon leaving state x_i, the chain must move to one of the states x_j (possibly the same state, $x_j = x_i$, which is the diagonal element of the matrix). Therefore, each row i of $\mathbf{P}$ is a probability distribution over states reachable from state x_i.
Due to the Markov property, the future process $X_{n+1}, X_{n+2}, \dots$ is independent of the past process $X_0, X_1, \dots, X_{n-1}$, given the present state X_n.

3.3 Markov Chains as Recursions

There is a close relationship between Markov Chains and recursive functions. Indeed:

Theorem 3.1. *Let* $f(x, u)$ *be a real-valued function of two variables. Let* $\{U_n : n \geq 0\}$ *be an i.i.d. discrete-time random process. Then:*

$$x_{n+1} = f(x_n, U_n), n \geq 0 \tag{3.2}$$

is a Markov Chain if x_0 *is independent of* $\{U_n : n \geq 0\}$.

The transition probabilities of this Markov Chain are given by $P_{ij} = P(f(x_i, u_n) = x_j)$.
It is clear that X_{n+1} only depends on x_n and U_n. Since the random variables U_n are independent of the past (they are *i.i.d.* by definition), the Markov property holds. Note that the transition probability is still allowed to change over time, which can be made explicit by writing $f(x_n, U_n, n)$.
Remarkably, however, the converse is also true:

Theorem 3.2. *Every Markov Chain $\{X_n = x_n : n \geq 0\}$ can be represented as a recursion*

$$x_{n+1} = f(x_n, U_n), n \geq 0 \tag{3.3}$$

for some suitable f and i.i.d. $\{U_n : n \geq 0\}$.

In particular, the $\{U_n : n \geq 0\}$ can always be chosen *i.i.d.* from $\mathcal{U}(0,1)$ with f adjusted accordingly. Therefore, every Markov Chain can be simulated if one can simulate uniformly distributed random variables (see Section 2.2). The proof is more involved, but is based on the inversion transform.
If the transition probabilities are independent of time, i.e. $P_{ij} = \text{const}$ for each given pair (i,j) (time-invariant Markov Chain), we can compute the *n-step transition matrix* as:

$$\mathbf{P}^{(n)} = \mathbf{P}^n = \underbrace{\mathbf{P} \times \mathbf{P} \times \ldots \times \mathbf{P}}_{n \text{ times}}. \tag{3.4}$$

The fact that the n-step transition matrix is simply the n^{th} power of the one-step transition matrix follows from the *Chapman-Kolmogorov equation*:
For any $n \geq 0, m \geq 0, x_i, x_j, x_k \in \mathbb{S}$, we have:

$$P_{ij}^{n+m} = \sum_{k \in \mathbb{S}} P_{ik}^n P_{kj}^m, \tag{3.5}$$

where:

$$\begin{aligned} P_{ik}^n &= P(X_n = x_k | X_0 = x_i) \\ P_{kj}^m &= P(X_{m+n} = x_j | X_n = x_k). \end{aligned}$$

Because of the Markov property (the future is independent of the past) and Eq. 1.10, it is easy to see that

$$\begin{aligned} P_{ik}^n P_{kj}^m &= P(X_{m+n} = x_j | X_n = x_k) P(X_n = x_k | X_0 = x_i) \\ &= P(X_n = x_k, X_{m+n} = x_j | X_0 = x_i). \end{aligned}$$

Summing over all k, i.e., all possible states x_k the path from x_i to x_j could pass through, i.e., marginalizing this joint probability over k, yields the formula in Eq. 3.5. Since this is the formula for matrix multiplication, Eq. 3.4 is shown.

3.4 Properties of Markov Chains

Markov Chains have some useful and elegant mathematical properties. It is mainly thanks to these properties that Markov Chains are so popular and widely used. We start by giving a few definitions of properties a Markov Chain may have and then discuss their importance.

Definition 3.4 (Closed set of states). *A set $\mathbb{C} = \{x_i\}$ of states is* closed *if and only if no state outside $\mathbb{C}$ can be reached from any $x_0 \in \mathbb{C}$.*

If a closed set contains only one state, that state is called *absorbing*. After the Markov Chain reaches an absorbing state, it will never leave it again (hence the name).

Example 3.2. *Consider the Markov Chain defined by the recursion $f(x, u) = xu$, where the U_n are uniformly distributed random numbers in the interval $[0, 1]$ and the continuous state space $\mathbb{S} = [0, 1]$. That is, the next value (between 0 and 1) of the chain is given by the current value multiplied with a uniform random number between (and including) 0 and 1. The value 0 is an absorbing state. Once the chain reaches 0, it is never going to show any value other than 0 any more. Even more, every lower sub-interval $C = [0, \nu] \subseteq S$ for all $\nu \in [0, 1]$ is a closed set of states, because the state of the chain is never increasing.*

Definition 3.5 (Irreducibility)**.** *A Markov Chain is* irreducible *if and only if there exists no closed set other than $\mathbb{S}$ itself.*

Therefore, in an irreducible chain, every state can be reached from every other state, eventually. There is no closed set from which the chain could not escape any more. Every state is reachable, one just has to wait long enough (potentially infinitely long).

Example 3.3. *Clearly, the Markov Chain from Example 3.2 is not irreducible, because is has infinitely many closed sets. However, if we consider i.i.d. random variables $U_n \sim \mathcal{U}[\epsilon, 1/x]$ for some arbitrarily small $\epsilon > 0$, then the chain has no absorbing state and no closed set any more, and it becomes irreducible. The state space is then $\mathbb{S} = (0, 1]$.*

Definition 3.6 (Periodicity)**.** *A Markov Chain is* periodic *if and only if for at least one state x_j, $P_{jj}^{(n)} = 0$ for all $n \neq kt$ with $k, t \in \mathbb{N}$, but $P_{jj}^{(kt)} = 1$. $t > 1$ is called the* period*. If no such $t > 1$ exists for none of the states, the Markov Chain is* aperiodic.

A periodic Markov Chain revisits one or several states in regular time intervals t. Therefore, if we find the chain in a periodic state x_j at time t, we know that it is going to be in the same state again at times $2t, 3t, 4t, \ldots$. A Markov chain that is not periodic is called *aperiodic*.

Definition 3.7 (Ergodicity)**.** *A Markov Chain is* ergodic *if and only if it is irreducible and aperiodic.*

An ergodic chain revisits any state with finite mean recurrence time. It is not possible to predict when exactly the chain is going to revisit a given state, like in the periodic case, but we know that it will in finite time. While an irreducible Markov Chain is eventually going to revisit any state, the recurrence time may be infinite, so irreducibility alone is not sufficient for ergodicity.
One of the interesting properties of Markov Chains for practical applications is that they can have *stationary distributions*. This means that when running

the Markov Chain for $n \to \infty$, states are visited according to an invariant probability distribution, i.e., the probability

$$P_k = \lim_{n \to \infty} P_{jk}^{(n)} \quad \forall k, \tag{3.6}$$

is independent of x_j. Since $P_k \geq 0$ and $\sum_k P_k = 1$, this defines a probability distribution over states, where state x_k is visited with probability P_k, independent of what the previous and current states are.

Example 3.4. *Any i.i.d. random process is also a Markov Chain (but not vice versa!), with recursion function $f(x, u) = u$. Because it is i.i.d., it will visit each state with given probability. Therefore, i.i.d. processes are Markov Chains with trivial stationary distribution.*

While it may seem that in such cases an *i.i.d.* model would be simpler to use, it may not always be computable. The stationary Markov Chain then provides a simple algorithm for (approximately) simulating the process. Markov Chains with known stationary distribution can be used to simulate random numbers from that distribution. One just has to recurse the chain long enough and then record its states, which will be distributed according to the desired stationary distribution. A problem in practice is to know what "long enough" means. Equation 3.6 talks about time going to infinity. In practice, stationary distributions may be (approximately) reached after a finite time. This time is called the *mixing time* of the Markov Chain. After the initial mixing time, the chain samples values from its stationary distribution forever onward. For a given chain, mixing times can be hard to known and are usually determined empirically, i.e., by simulating the chain and using statistical tests to decide when the values are statistically indistinguishable from samples of the stationary distribution. Part of the problem is that while the stationary distribution is independent of the initial state by definition, the mixing time may depend on the initial state. Only for a few special cases, mixing times are known analytically.
Another important question is when a Markov Chain *has* a stationary distribution at all. Fortunately, this can be clearly answered:

Theorem 3.3. *A Markov Chain possesses a unique stationary distribution if and only if it is ergodic.*

In other words, a Markov Chain must never be trapped in any closed set of states (irreducible) and it must not have predictable periods. While it is intuitive that these are necessary conditions for the existence of a stationary distribution, they are also sufficient, and the distribution is unique (proof not given here).

Example 3.5. *The* Random Walk *is a classic example of a Markov Chain with great importance as a model in physics, finance, biology, and other fields. In physics, for example, random walks model the process of diffusion caused by Brownian motion of microscopic particles, such as molecules. In biology, random walks model population dynamics or cell motion, and in economics they are used to model stock market fluctuations or gambling.*

Mathematically, a random walk is defined by an i.i.d. sequence of "increments" $\{D_n = d_n : n \geq 0\}$ *and the stochastic process:*

$$x_n := \sum_{i=1}^{n} d_i \,, \quad x_0 = 0. \tag{3.7}$$

While it might superficially seem that X_n *depended on the entire history of the process, it is in fact a Markov Chain, since* $x_n = x_{n-1} + D_n$ *is independent of the previous states. The recursion formula of this Markov Chain is* $f(x, d) = x + d$, *which, according to Theorem 3.1, proves the Markov property.*
If the X_n *are scalar and the increments are either* $+1$ *or* -1, *we obtain a one-dimensional discrete-state random walk with*

$$P(D = +1) = p, \quad P(D = -1) = 1 - p. \tag{3.8}$$

Such a random walk is called simple. *It models the random motion of an object on a regular 1D lattice. For the special case that* $p = \frac{1}{2}$, *the random walk becomes* symmetric, *i.e., it has equal probability to go left or right in each time step.*
In a simple random walk, the only states that can be reached from x_i *are* x_{i+1} *and* x_{i-1}. *Therefore, we have the one-step transition probabilities* $P_{i,i+1} = p$ *and* $P_{i,i-1} = 1 - p$. *Consequently,* $P_{ii} = 0$. *This would form one row of the transition matrix. However, we can only write down the matrix once we restrict the chain to a bounded domain and hence a finite state space.*

Chapter 4

Monte Carlo Methods

There exist a variety of computational problems, which are very expensive or even impossible to solve using conventional numerical and analytical techniques. Monte Carlo (MC) methods are an important class of computational algorithms, which can be used to efficiently approximate solutions of such problems by exploiting (quasi-/pseudo-)random numbers generated on a computer. MC methods find application in many disciplines of engineering and science, including operations research, finance, and machine learning. They are most commonly used to (a) numerically calculate integrals (b) sample from complex distributions (c) solve optimization problems.

The name "Monte Carlo Method" goes back to the 1940s when John von Neumann and Stanislaw Ulam developed the computational aspects of this class of algorithms in the US nuclear weapons projects at Los Alamos National Laboratory. Since the project was top-secret, they required a code name for their new algorithm. They chose the code name "Monte Carlo Method", as suggested by Nicholas Metropolis in reference to the famous gambling casino in the city of Monte Carlo (Principality of Monaco) due to its use of random numbers.

To motivate the core idea of MC methods, we consider a classical example.

Example 4.1. *How to compute π using random numbers? This historic example goes back to Buffon's "needle problem", first posed in the 18th century by Georges-Louis Leclerc, Comte de Buffon. It shows a very simple and intuitive way to approximate π using MC simulation. We begin by first drawing a unit square on a blackboard or a piece of paper. Moreover, we draw a quarter of a unit circle into the square such that the top-left and bottom-right corners of the square are connected by the circle boundary. We then generate N random points uniformly distributed across the unit square, i.e., $\vec{x}^{(i)} = (u_1^{(i)}, u_2^{(i)})$ with $u_1^{(i)} \sim \mathcal{U}(0,1)$ and $u_2^{(i)} \sim \mathcal{U}(0,1)$ for $i = 1, \ldots, N$. You can achieve this, for instance, by throwing pieces of chalk at the board, assuming that your shots are uniformly distributed across the square. Subsequently, you count the points that* *made it into the circle and calculate*

$$4\frac{\#\ \textit{of samples inside the circle}}{N} \approx \frac{\textit{area of circle}}{\textit{area of square}} = \frac{r^2\pi}{r^2} = \pi, \tag{4.1}$$

which will give you an approximation of π. The more samples N you use, the more accurate your estimate will be. We shall see later why this algorithm works, but an intuitive explanation can be provided right away. In particular, the fraction of points that fall inside the circle approximates how much of the area of the unit square is occupied by the quarter circle. We know that the area of a unite square is $A_s = r^2 = 1$ and that the area of the quarter of a unit circle is $A_c = r^2\pi/4 = \pi/4$. The ratio between the two is therefore $A_c/A_s = \pi/4$. In Eq. 4.1 we first calculate an estimate of this ratio and then multiply it by four, which will therefore result in an approximation of π. Using MC simulations, the ratio A_c/A_s is not determined exactly, but approximated using standard uniform random numbers, which we can easily generate on a computer.

Remarks:

- Solutions obtained by MC simulation are random. Repeated MC simulations will not give you the exact same answer (unless the same sequence of quasi- or pseudo-random numbers is used).
- It is crucial to know how accurate a MC approximation is. How much can you trust your results?
- MC methods converge to the exact answer for $N \to \infty$. This is a fundamental result known as the law of large numbers.

4.1 The Law of Large Numbers

The law of large numbers gives a rigorous argument why MC methods converge to the exact solution if N becomes sufficiently large. To show this, let us consider a sequence of *i.i.d.* RVs $X_1, \ldots, X_N$ with finite expectation value $\mathbb{E}\{X_i\} = \mathbb{E}\{X_j\} = \mu$. Moreover, we define the average of these variables as

$$\bar{X}_N = \frac{1}{N}\sum_{i=1}^{N} X_i. \tag{4.2}$$

The law of large numbers states that

$$\bar{X}_N \xrightarrow{N\to\infty} \mu, \tag{4.3}$$

i.e., the average of the RVs "converges" to the expectation μ of the individual variables. This implies that if we have a large number of independent realizations of the same probabilistic experiment, we can estimate the expected outcome of this experiment by calculating the empirical average over these realizations. This is the main working principle behind MC methods.

Depending on how we specify convergence, we can distinguish two forms of the law of large numbers. The first one is called the *weak* law of large numbers and it states that

$$\lim_{N\to\infty} P(|\bar{X}_N - \mu| > \varepsilon) = 0, \tag{4.4}$$

with $\varepsilon > 0$ an arbitrary positive constant. The weak law implies that the average $\bar{X}_N$ converges *in probability / in distribution* to the true mean μ, which means that for any ε we can find a sufficiently high N such that the probability that $\bar{X}_N$ differs from μ by more than ε will be arbitrarily small.
The second version of the law of large numbers is called the *strong* law of large numbers, which states that

$$P\Big(\lim_{N\to\infty} \bar{X}_N = \mu\Big) = 1, \tag{4.5}$$

meaning that $\bar{X}_N$ converges to μ almost surely, that is, with probability one *in value*. The difference between the two laws is that in the weak law, we leave open the possibility that $\bar{X}_N$ deviates from μ by more than ε infinitely often (although unlikely) along the path of $N \to \infty$. So there is no sufficiently large finite N such that we can *guarantee* that $\bar{X}_N$ stays within the ε margin. The strong law, by contrast, states that for a sufficiently large N, $\bar{X}_N$ will remain within the margin with probability one. Therefore, the strong law implies the weak law, but not vice-versa. One is the probability of a limit, whereas the other is a limit over probabilities. There are certain cases, where the weak law holds, but the strong law does not. Moreover, there are cases where neither of them apply: if the samples are Cauchy-distributed, for instance, the mean μ does not exist and therefore, we can not use sample averages to determine the expected outcome.

4.1.1 Proof of the Weak Law

While the proof of the strong law is rather technical, the weak law can be derived in a relatively straightforward manner using the concept of *characteristic functions.*

Definition 4.1 (Characteristic function)**.** *The characteristic function $\phi_X(t)$ of a continuous RV X is defined as*

$$\phi_X(t) := \mathbb{E}[\mathrm{e}^{iXt}], \tag{4.6}$$

with i the imaginary unit and t the real argument of this function. Making use of the definition of the expectation,

$$\phi_X(t) = \int_{-\infty}^{\infty} \mathrm{e}^{ixt} p(x)\,\mathrm{d}x, \tag{4.7}$$

we recognize $\phi_X(t)$ as the Fourier transform of the PDF $p(x)$ of X.

Characteristic functions provide an alternative characterization of RVs that can be beneficial in certain cases. This is due to the mathematical properties of the characteristic functions. The following two properties are required for our proof:

- $\phi_{X+Y}(t) = \phi_X(t)\phi_Y(t)$ if X and Y are independent,
- $\phi_{\alpha X}(t) = \phi_X(\alpha t)$ with α any a real constant.

We begin the proof by setting up the characteristic function of the sample average $\bar{X}_N = 1/N \sum_{i=1}^N X_i$:

$$\phi_{\bar{X}_N}(t) = \mathbb{E}\left[\mathrm{e}^{i\frac{t}{N}\sum_{i=i}^N X_i}\right]. \tag{4.8}$$

By rearranging the exponent we obtain

$$\phi_{\bar{X}_N}(t) = \mathbb{E}[\mathrm{e}^{it(X_1/N+X_2/N+\cdots+X_N/N)}], \tag{4.9}$$

which is the characteristic function of a sum of N independent and rescaled RVs. Therefore, by exploiting the two properties of characteristic functions above, we obtain

$$\phi_{\bar{X}_N}(t) = \phi_{X_1}(t/N)\phi_{X_2}(t/N)\cdots\phi_{X_N}(t/N). \tag{4.10}$$

Now, since all X_i are identically distributed, i.e., $\phi_{X_i}(t) = \phi_{X_j}(t) = \phi_X(t)$, we further obtain

$$\phi_{\bar{X}_N}(t) = \phi_X(t/N)^N. \tag{4.11}$$

The specific form of $\phi_X(t)$ depends on the PDF of the RV X, but we can get a general form of it by Taylor expanding it around $t = 0$ up to order one. In particular, we obtain

$$\phi_X(t) = \phi_X(0) + t\frac{\partial}{\partial t}\phi_X(t)\Big|_{t=0} + o(t), \tag{4.12}$$

whereas $o(t)$ summarizes all higher-order terms (little-o notation). Applying the definition of the characteristic function, we obtain

$$\phi_X(0) = \mathbb{E}[\mathrm{e}^{i0X}] = \mathbb{E}[1] = 1 \tag{4.13}$$

and

$$\begin{aligned}\frac{\partial}{\partial t}\phi_X(t)\Big|_{t=0} &= \frac{\partial}{\partial t}\mathbb{E}[\mathrm{e}^{itX}]\Big|_{t=0} = \mathbb{E}\left[\frac{\partial}{\partial t}\mathrm{e}^{itX}\right]\Big|_{t=0} \\ &= \mathbb{E}\left[iX\mathrm{e}^{i0X}\right] = i\mathbb{E}\left[X\right] = i\mu.\end{aligned} \tag{4.14}$$

In summary, we thus obtain

$$\phi_X(t) = 1 + it\mu + o(t), \tag{4.15}$$

and inserting this into Eq. (4.11) yields

$$\phi_{\bar{X}_N}(t) = \left[1 + i\frac{t}{N}\mu + o(t/N)\right]^N . \tag{4.16}$$

The term $o(t/N)$ tends to zero faster than the other two terms for large N since it contains higher orders of t/N. We can therefore neglect it asymptotically. Letting N go to infinity for the remaining expression, we obtain

$$\lim_{N\to\infty} \phi_{\bar{X}_N}(t) = \lim_{N\to\infty} \left[1 + i\frac{t}{N}\mu\right]^N , \tag{4.17}$$

which is the definition of the exponential function $\mathrm{e}^{it\mu}$. This, in turn, is the characteristic function of a constant (deterministic) variable μ, which means that the sample average converges in density to μ. Intuitively, this says that the PDF of the sample average will be squeezed together as $N \to \infty$ such that all its probability mass concentrates at the value μ, resulting in a Dirac delta distribution $\delta(\bar{x} - \mu)$.

Remark: Strictly speaking, our proof has only shown convergence in density (in characteristic functions), but not in probability as stated by the weak law. However, it can be shown further that since μ is a constant, convergence in density implies convergence in probability, which completes the proof. This, however, is beyond the scope of this lecture.

4.2 Monte Carlo Integration

One main application of MC methods is the numerical computation of complicated and/or high-dimensional integrals, for which analytically solutions do not exist and numerical quadrature may be computationally inefficient. The key idea is to reformulate an integral in terms of an expectation, which can then be approximated using a conventional sample average.

Let us first consider a one-dimensional integral of the form

$$F = \int_a^b f(x)\,\mathrm{d}x \tag{4.18}$$

with a and b the integration limits. In order to reformulate this integral in terms of an expectation, we first multiply the integrand by $1 = (b - a)/(b - a)$, which leaves the value of the integral unaffected, i.e.,

$$\begin{aligned} \int_a^b f(x)\,\mathrm{d}x &= \int_a^b f(x)\frac{b-a}{b-a}\,\mathrm{d}x & (4.19)\\ &= \int_a^b f(x)(b-a)p(x)\,\mathrm{d}x, & (4.20) \end{aligned}$$

where we recognize $p(x) = \frac{1}{b-a}$, $x \in (a,b)$, as the PDF of a uniform continuous RV $X \sim \mathcal{U}(a,b)$. We can therefore express the integral as an expectation

$$\int_a^b f(x)\,\mathrm{d}x = (b-a)\mathbb{E}[f(X)], \tag{4.21}$$

taken with respect to a uniform RV $X \sim \mathcal{U}(a,b)$. We can therefore approximate this integral by the sample average

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{N}\sum_{i=1}^{N} f(x_i) =: \theta_N, \tag{4.22}$$

with *i.i.d.* samples $x_i \sim \mathcal{U}(a,b)$ for $i = 1,\ldots,N$. In the following, we refer to the sample average θ_N as the *Monte Carlo estimator.* Since a uniform RV has a finite mean, we know by the law of large numbers that θ_N will converge to the correct solution of the integral as $N \to \infty$.

As indicated earlier in this chapter, it is crucial to know how much uncertainty we have to expect in our MC estimator for a certain finite sample size N. To this end, we can calculate the variance of the MC estimator, i.e.,

$$\begin{aligned}
\mathrm{Var}[\theta_N] &= \mathrm{Var}\left[\frac{(b-a)}{N}\sum_{i=1}^{N} f(x_i)\right] && (4.23)\\
&= \frac{(b-a)^2}{N^2}\sum_{i=1}^{N}\mathrm{Var}[f(X_i)] && (4.24)\\
&= \frac{(b-a)^2}{N^2}N\mathrm{Var}[f(X)] && (4.25)\\
&= O(1/N). && (4.26)
\end{aligned}$$

In the first step, we have used the linearity of the variance operator, and in the second step we have used the fact that the RVs are *i.i.d.* This shows that the MC variance scales with $1/N$ if independent samples are used. If the variance of the transformed RV $f(X)$ is hard to compute analytically, an empirical estimate can be determined from the N samples, i.e.,

$$\mathrm{Var}[f(X)] \approx \frac{1}{N-1}\sum_{i=1}^{N}(f(x_i) - \langle f(X)\rangle)^2, \tag{4.27}$$

with

$$\langle f(X)\rangle = \frac{1}{N}\sum_{i=1}^{N} f(x_i). \tag{4.28}$$

Remember that the partition function in the empirical sample variance is $1/(N-1)$ in order for the estimator to be unbiased (a single sample has no variance) — see statistics course.

4.2.1 Multidimensional Extension

MC integration can be extended to multidimensional integrals of the form

$$F = \int_\Omega f(x)\,\mathrm{d}^m x, \quad x \in \mathbb{R}^m, \tag{4.29}$$

with $f : \mathbb{R}^m \to \mathbb{R}$ and Ω the domain of this integral. Just as in the one-dimensional case, we can approximate this integral by an MC estimate

$$F \approx V \frac{1}{N} \sum_{i=1}^{N} f(x_i) =: \theta_N, \tag{4.30}$$

with x_i as N *i.i.d.* RVs distributed uniformly in Ω. The scaling factor V is the volume of the domain Ω, i.e.,

$$V = \int_\Omega \mathrm{d}^m x. \tag{4.31}$$

Note that in the one-dimensional case, this volume reduces to the length of the domain $(b-a)$, as derived above. The corresponding MC estimator variance, by analogy, is given by

$$\mathrm{Var}[\theta_N] = \frac{V^2}{N} \mathrm{Var}[f(X)]. \tag{4.32}$$

4.3 Importance sampling

A major drawback of standard MC integration is that it relies on uniform RVs. This means that each part of Ω is sampled equally likely. However, depending on the integrand $f(x)$, there may be certain regions of Ω where $f(x)$ contributes more to the value of the integral than in other regions. One would therefore expect more accurate results if samples were drawn more frequently in these "important" regions than in regions that only contribute marginally to the integral. This leads to the idea of *importance sampling.* For the sake of illustration, we introduce the concept in the context of MC integration. However, keep in mind that it can also be applied to other MC problems.

As before, we consider a general m-dimensional integral of the form

$$F = \int_\Omega f(x)\,\mathrm{d}^m x, \tag{4.33}$$

with $f : \mathbb{R}^m \to \mathbb{R}$ and Ω as the domain of this integral. In order to derive the importance sampler, we first multiply the integrand by $1 = q(x)/q(x)$, where $q(x)$ is an m-dimensional *proposal* PDF, whose support $\mathcal{S}$ contains the domain

Ω, i.e., $\mathcal{S} \supseteq \Omega$. We obtain:

$$\int_\Omega f(x)\,\mathrm{d}^m x = \int_\Omega f(x)\frac{q(x)}{q(x)}\,\mathrm{d}^m x \tag{4.34}$$

$$= \int_{\mathcal{S}} \frac{f(x)}{q(x)}\mathbf{1}_{X\in\Omega} q(x)\,\mathrm{d}^m x \tag{4.35}$$

$$= \mathbb{E}\left[\frac{f(X)}{q(X)}\mathbf{1}_{X\in\Omega}\right], \tag{4.36}$$

with X a continuous RV with PDF $q(x)$, and $\mathbf{1}_{X\in\Omega}$ the indicator function that restricts the samples to the original domain of integration. Assuming that we are given N *i.i.d.* realizations $x_1, \ldots, x_N$ of X, we can compute an MC estimate as

$$\int_\Omega f(x)\,\mathrm{d}^m x \approx \underbrace{\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{q(x_i)}\mathbf{1}_{X\in\Omega}}_{=:\theta_N}. \tag{4.37}$$

The corresponding MC estimator variance is given by:

$$\mathrm{Var}[\theta_N] = \frac{1}{N}\mathrm{Var}\left[\frac{f(x)}{q(x)}\mathbf{1}_{X\in\Omega}\right]. \tag{4.38}$$

This shows that the MC variance depends on the proposal distribution $q(x)$. In order to study which proposals achieve the lowest MC variance, and hence the most accurate approximations, we consider the following toy examples, before we proceed to a general discussion of this topic in the next chapter.

Example 4.2 (Optimal proposal distribution)**.** *We first consider a proposal distribution that is proportional to the function $f(x)$, i.e.,*

$$q(x) = Kf(x) \tag{4.39}$$

for some constant K that ensures that q integrates to one. Remember that $f(x)$ is not a PDF, but just a function to be integrated. For the MC variance, we obtain from Eq. 4.38:

$$\mathrm{Var}\left[\theta_N\right] = \frac{1}{N}\mathrm{Var}\left[\frac{f(x)}{Kf(x)}\right] = \frac{1}{N}\mathrm{Var}\left[\frac{1}{K}\right] = 0, \tag{4.40}$$

because $1/K$ is a deterministic value. The indicator function was dropped since f and q have the same support. Paradoxically, this result implies that a single sample from this proposal would suffice to fully determine the true value of the integral, for any integrand $f(x)$, since the MC estimator variance is zero. However, at a second look we realize that in order to evaluate the ratio $f(x)/q(x)$ in the MC estimator, we require knowledge of the constant K, since $f(x)/q(x) = 1/K$. This constant, however, is equal to the solution of the integral we are after, since it must hold that

$$\frac{1}{K} = \int_\Omega f(x)\,\mathrm{d}^m x \tag{4.41}$$

for $q(x)$ to integrate to one. Because the constant already contains the result, one "sample" of that constant would indeed suffice. Clearly, this choice of proposal distribution is not realizable, but it still reveals important information about how suitable proposals should look like: a good $q(x)$ should follow $f(x)$ as closely as possible, up to a constant scaling factor.

Example 4.3 (Calculating π revisited)**.** *We have shown at the beginning of this chapter that π can be approximated using random numbers. It now becomes clear that the scheme we thus used can be understood as an importance sampler. Indeed, the number π can be written as the integral*

$$\pi = 4A_\Omega = 4\iint_\Omega \mathrm{d}u_1 \mathrm{d}u_2, \tag{4.42}$$

where Ω is the quarter circle and $\vec{x} = (u_1, u_2)$ a point in the 2D plane. Note that in this case, the function f is one everywhere. This can be rewritten as

$$\begin{aligned} \pi &= 4\int_0^1 \int_0^1 \frac{1}{q(u_1,u_2)} \mathbf{1}_{(u_1,u_2)\in\Omega} q(u_1,u_2)\,\mathrm{d}u_1\mathrm{d}u_2 \\ &= 4\mathbb{E}\left[\frac{1}{q(U_1,U_2)}\mathbf{1}_{(U_1,U_2)\in\Omega}\right]. \end{aligned} \tag{4.43}$$

In the introductory example, we have chosen $q(u_1, u_2)$ to be a uniform proposal distribution on the unit square. Since this proposal is constant and $q = 1$ inside the entire unit square, we further obtain

$$\pi = 4\mathbb{E}\left[\mathbf{1}_{(U_1,U_2)\in\Omega}\right] \approx \frac{4}{N}\sum_{i=1}^{N} \mathbf{1}_{(u_1,u_2)\in\Omega} = 4\frac{\#\ \textit{samples inside the circle}}{N}, \tag{4.44}$$

which is the formula introduced at the beginning of this chapter.

Example 4.4. *Let us consider a function $f(x) = x^2$ for $x \in [0,1]$, which we want to integrate using MC integration $\int_0^1 x^2\,\mathrm{d}x$. We consider the standard MC estimator*

$$\theta_N^{(1)} = \frac{1}{N}\sum_{i=1}^{N} f(x_i) \quad \textit{with} \quad x_i \sim \mathcal{U}(0,1) \tag{4.45}$$

as well as an estimator that uses importance sampling using a proposal distribution $q(x) = 2x$, $x \in [0,1]$:

$$\theta_N^{(2)} = \frac{1}{N}\sum_{i=1}^{N} \frac{f(x_i)}{q(x_i)} \quad \textit{with} \quad x_i \sim q(x) = 2x. \tag{4.46}$$

Which of the two estimators is better? To answer this question, we calculate the

MC variances of both estimators. For estimator 1 we find:

$$\mathrm{Var}[\theta_N^{(1)}] = \frac{1}{N}\mathrm{Var}[f(x)] \tag{4.47}$$

$$\mathrm{Var}[f(x)] = \mathrm{Var}[X^2] = \int_0^1 (X^4 - \mathbb{E}[X^2]^2)p(x)\,\mathrm{d}x = \mathbb{E}[X^4] - \mathbb{E}[X^2]^2 \tag{4.48}$$

$$\mathbb{E}[X^2] = \int_0^1 x^2 p(x)\,\mathrm{d}x = \int_0^1 x^2\,\mathrm{d}x = \left.\frac{X^3}{3}\right|_0^1 = \frac{1}{3} \tag{4.49}$$

$$\mathbb{E}[X^4] = \int_0^1 x^4 p(x)\,\mathrm{d}x = \int_0^1 x^4\,\mathrm{d}x = \left.\frac{X^5}{5}\right|_0^1 = \frac{1}{5}. \tag{4.50}$$

Therefore, the MC variance of estimator 1 is:

$$\mathrm{Var}[\theta_N^{(1)}] = \frac{1}{N}\left(\frac{1}{5} - \frac{1}{9}\right) = \frac{1}{N}\frac{4}{45} = \frac{0.0889}{N}. \tag{4.51}$$

For estimator 2 we obtain:

$$\mathrm{Var}[\theta_N^{(2)}] = \frac{1}{N}\mathrm{Var}\left[\frac{f(x)}{q(x)}\right] = \frac{1}{N}\mathrm{Var}\left[\frac{X^2}{2X}\right] \tag{4.52}$$

$$= \frac{1}{4N}\mathrm{Var}[X]. \tag{4.53}$$

Again, the indicator function can be dropped since f and q have the same support. We know that $\mathrm{Var}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$ *and since*

$$\mathbb{E}[X] = \int_0^1 2x^2\,\mathrm{d}x = \frac{2}{3} \tag{4.54}$$

$$\mathbb{E}[X^2] = \int_0^1 2x^3\,\mathrm{d}x = \frac{1}{2}, \tag{4.55}$$

we get for the MC variance of estimator 2

$$\mathrm{Var}[\theta_N^{(2)}] = \frac{1}{4N}\left(\frac{1}{2} - \frac{4}{9}\right) = \frac{1}{N}\frac{1}{72} = \frac{0.0139}{N}. \tag{4.56}$$

Since

$$0.0139\frac{1}{N} < 0.0889\frac{1}{N} \tag{4.57}$$

for any $N > 0$, $\theta_N^{(2)}$ is a better estimator. This means that for the same sample size N, estimator 2 achieves a higher (about 6-fold) accuracy, in expectation over many realizations of the MC procedure.

Chapter 5

Variance Reduction

As discussed in the previous sections, a key property of any MC estimator is the variance associated with it. Ultimately, the MC variance determines how many samples N one has to generate in order to achieve a certain statistical precision. We have also shown that different MC estimators of the same quantity can have very different MC variances in the context of importance sampling. In this chapter we will discuss techniques that allow to improve the accuracy of MC estimators, commonly known as *variance reduction* techniques. While there exists a broad range of variance reduction methods, we will focus on two of them in this lecture: *antithetic variates* and *Rao-Blackwellization.*

5.1 Antithetic Variates

So far, a key assumption of the discussed MC estimators was that the individual samples used for constructing the estimate are *i.i.d.* We recall that in this case, the MC variance is given by

$$Var\{\theta_N\} = Var\left\{\frac{1}{N}\sum_{i=1}^{N} f(X_i)\right\} = \frac{1}{N^2}\sum_{i=1}^{N} Var\{f(X_i)\}, \tag{5.1}$$

with X_i as *i.i.d.* RVs and $f(x)$ as some function. We now consider the case where the RVs X_i are not independent, in which case the covariance between any X_i and X_j will be non-zero. In this case, it is straightforward to show that the variance of an MC estimator is given by

$$Var\left\{\frac{1}{N}\sum_{i=1}^{N} X_i\right\} = \frac{1}{N^2}\left[\sum_{i=1}^{N} Var\{f(X_i)\} + \sum_{i\neq j} Cov\{f(X_i), f(X_j)\}\right]. \tag{5.2}$$

We see that if the covariance terms are positive, the MC variance will be larger than in the i.i.d. case. However, if the covariances become negative, we can achieve variance reduction. This is the key idea underlying a popular variance reduction method called *antithetic variates*. While this approach is fairly general, we will illustrate it here in the context of a simple example.

Our goal is to use Monte Carlo estimation to calculate the expectation $\mathbb{E}\{f(X)\}$ with $f(X)$ as some function and X as a uniform RV. To do so, we generate N uniformly distributed random numbers

$$X_i \sim \mathcal{U}(0,1) \quad \forall i = 1, \ldots, N. \tag{5.3}$$

Our canonical MC estimator would take the form

$$\frac{1}{N} \sum_{i=1}^{N} f(X_i). \tag{5.4}$$

The goal of antithetic variates is to inversely correlate the generated samples. To this end, we define the augmented sample

$$Z = \{X_1, \ldots, X_N, \bar{X}_1, \ldots, \bar{X}_N\} = \{Z_1, \ldots, Z_{2N}\}, \tag{5.5}$$

with $\bar{X}_i = 1 - X_i$. On the one hand, this means that we have doubled the number of samples of our estimator. On the other hand, we have artificially introduced negative correlations between pairs of samples since X_i and $\bar{X}_i$ will be anticorrelated. It can be shown that if the function f is monotonically increasing or decreasing, this implies that also the correlations between $f(X_i)$ and $f(\bar{X}_i)$ will be negative. In particular, the variance of the antithetic MC estimator becomes

$$\begin{aligned} Var\{\theta_N^A\} &= Var\left\{\frac{1}{2N}\sum_{i=1}^{2N} f(Z_i)\right\} && (5.6) \\ &= \frac{1}{4N^2}\left[\sum_{i=1}^{2N}\underbrace{Var\{f(Z_i)\}}_{\sigma_f^2} + 2\sum_{i=1}^{N}\underbrace{Cov\{f(X_i), f(\bar{X}_i)\}}_{\gamma_f}\right] && (5.7) \\ &= \frac{1}{2N}\left(\sigma_f^2 + \gamma_f\right). && (5.8) \end{aligned}$$

Now, since γ_f is strictly negative for monotonic functions f we obtain variance reduction when compared to an MC estimator that uses $2N$ independent samples. Another important advantage of antithetic variates is that we can double the sample size N "for free": we can use $2N$ samples, even though we had to draw only N random numbers.

5.2 Rao-Blackwellization

In this section we will discuss the concept of Rao-Blackwellization to reduce the variance of MC estimators. This method is suited for Monte Carlo problems that depend on multiple random variables. While they apply to arbitrary

multidimensional problems, we limit ourselves to the two-dimensional case for simplicity.
Assume we want to use MC estimation to calculate the expectation $\mathbb{E}\{f(X,Y)\}$ with X and Y as dependent RVs and f as some function that maps a tuple (x,y) to a real or integer number. In the following, we assume that $X, Y \in \mathbb{R}$ and $f : \mathbb{R}^2 \to \mathbb{R}$, but keep in mind that the same concept applies also to integer-valued random variables and functions f. Analogously to the one-dimensional case, we can construct an MC estimator as

$$\theta_N = \frac{1}{N}\sum_{i=1}^{N} f(x_i, y_i), \tag{5.9}$$

with (x_i, y_i) as *i.i.d.*random samples from a joint probability distribution $p(x_i, y_i)$. The corresponding MC variances is given by

$$Var\{\theta_N\} = \frac{1}{N^2} Var\{f(X,Y)\} = \frac{1}{N}\sigma_f^2, \tag{5.10}$$

with

$$\sigma_f^2 = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (f(x,y) - \mu_f)^2 p(x,y)\mathrm{d}x\mathrm{d}y \tag{5.11}$$

and

$$\mu_f = \mathbb{E}\{f(X,Y)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)p(x,y)\mathrm{d}x\mathrm{d}y. \tag{5.12}$$

Let's now consider another estimator, given by

$$\hat{\theta}_N = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}\{f(X, y_i) \mid y_i\}, \tag{5.13}$$

with y_i as *i.i.d.*samples from the marginal distribution $\sim p(y) = \int_{-\infty}^{\infty} p(x,y)\mathrm{d}x$ and $\mathbb{E}\{f(X, y_i) \mid y_i\}$ as a *conditional expectation* defined by

$$\mathbb{E}\{f(X, y_i) \mid y_i\} = \int_{-\infty}^{\infty} f(x, y_i)p(x \mid y_i)\mathrm{d}x. \tag{5.14}$$

This estimator is called Rao-Blackwellized (RB) estimator. One important difference between the standard MC and RB estimators is that the latter one uses samples only from one of the two variables (in this case Y). The second variable has been "integrated out" analytically, which means that we effectively deal with lower-dimensional sampling space. Let us now look into the properties of the RB estimator. We first realize that on expectation, the RB estimator will deliver the correct result just as the original MC estimator since

$$\mathbb{E}\{\hat{\theta}_N\} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}\{\mathbb{E}\{f(X,Y) \mid Y\}\} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}\{f(X,Y)\} = \mathbb{E}\{f(X,Y)\}. \tag{5.15}$$

Most importantly, the variance of the RB estimator,

$$Var\{\hat{\theta}_N\} = \frac{1}{N} Var\{\mathbb{E}\{f(X,Y) \mid Y)\}\} = \frac{1}{N}\hat{\sigma}_f^2, \tag{5.16}$$

is lower than that of the standard MC estimator as summarized in the following theorem.

Theorem 5.1. *For a given sample size N, the Rao-Blackwellized estimator $\hat{\theta}_N$ is guaranteed to achieve lower or equal variance than the standard Monte Carlo estimator θ_N, i.e.,*

$$Var\{\hat{\theta}_N\} \leq Var\{\theta_N\} \tag{5.17}$$

for any N.

Proof: *A more formal proof of this theorem can be obtained by employing the Rao-Blackwell theorem. However, a simple derivation of this result is possible using the properties of conditional expectations. We begin by rewriting the variance σ_f^2 that appears in the MC variance of the standard estimator θ_N as*

$$\sigma_f^2 = \mathbb{E}\{f(X,Y)^2\} - \mu_f^2 \tag{5.18}$$
$$= \mathbb{E}\{\mathbb{E}\{f(X,Y)^2 \mid Y\}\} - \mu_f^2. \tag{5.19}$$

We then realize that the inner expectation $\mathbb{E}\{f(X,Y)^2 \mid Y\}$ is a conditional expectation taken over the squared function f. We can therefore make use of the relation

$$Var\{f(X,Y) \mid Y\} = \mathbb{E}\{f(X,Y)^2 \mid Y\}\} - \mathbb{E}\{f(X,Y) \mid Y\}^2, \tag{5.20}$$

to replace $\mathbb{E}\{f(X,Y)^2 \mid Y\}$ in (5.19) by $Var\{f(X,Y) \mid Y\} + \mathbb{E}\{f(X,Y) \mid Y\}^2$ such that we obtain

$$\sigma_f^2 = \mathbb{E}\{Var\{f(X,Y) \mid Y\}\} + \mathbb{E}\{\mathbb{E}\{f(X,Y) \mid Y\}^2\} - \mu_f^2. \tag{5.21}$$
$$\tag{5.22}$$

Using the same idea, we can now replace $\mathbb{E}\{\mathbb{E}\{f(X,Y) \mid Y\}^2\}$ by $Var\{\mathbb{E}\{f(X,Y) \mid Y\}^2\} + \mathbb{E}\{\mathbb{E}\{f(X,Y) \mid Y\}\}^2$ which yields

$$\sigma_f^2 = \mathbb{E}\{Var\{f(X,Y) \mid Y\}\} + Var\{\mathbb{E}\{f(X,Y) \mid Y\}^2\} + \underbrace{\mathbb{E}\{\mathbb{E}\{f(X,Y) \mid Y\}\}^2}_{\mu_f^2} - \mu_f^2 \tag{5.23}$$

$$= \underbrace{\mathbb{E}\{Var\{f(X,Y) \mid Y\}\}}_{\geq 0} + \underbrace{Var\{\mathbb{E}\{f(X,Y) \mid Y\}^2\}}_{\hat{\sigma}_f^2}. \tag{5.24}$$

Therefore, since the first term on the r.h.s is non-negative, we conclude that $\hat{\sigma}_f^2 \leq \sigma_f^2$ and correspondingly $Var\{\hat{\theta}_N\} \leq Var\{\theta_N\}$. □

Chapter 6

Markov Chain Monte-Carlo

In the previous chapters on Monte Carlo methods we have so far considered cases where the random numbers used for constructing the Monte Carlo estimators are easy to generate. In particular, we have focused on one- or two-dimensional problems that used "simple" RVs such as uniform random numbers. In many practical scenarios, however, this may not be the case. In Chapter 2 we have discussed several methods to generate more complex RVs but these methods largely apply to low-dimensional problems. So how can we generate random samples from higher-dimensional and possibly complex distributions? Markov chain Monte Carlo (MCMC) methods provide a powerful framework to address this problem. In this chapter we will discuss the core idea of MCMC and introduce two of the most popular MCMC sampling algorithms, commonly known as the Gibbs- and Metropolis-Hastings samplers.

Let us assume we want to sample from an arbitrary multidimensional target distribution $\Pi(x)$ with $x \in \mathcal{S}$. For the sake of illustration, we consider the case where x is a discrete-valued k-dimensional RV such that $\mathcal{S} = \mathbb{N}^k$. However, the following concepts apply in a similar fashion to other scenarios (e.g., when x is real-valued). The key idea of MCMC is to construct a Markov chain X_n, which has exactly Π as its stationary distribution. If we would simulate such a Markov chain until it reaches stationarity, we can use the states that it occupies as random samples from the correct target distribution Π.

We recall from Chapter 3 that the time-evolution of a discrete-time Markov chain X_n is governed by a transition kernel

$$P(X_{n+1} = x_j | X_n = x_i) = P_{ij}, \tag{6.1}$$

which characterizes the probability of moving from state i to state j. We furthermore know that if Π is a stationary distribution of X_n, then it must hold that

$$\sum_i P_{ij} \Pi(x_i) = \Pi(x_j), \tag{6.2}$$

whereas the sum goes over all states in $\mathcal{S}$. The kernel P is then said to be *invariant* with respect to the distribution Π. Intuitively, this means that if we start at the stationary distribution, then applying the invariant kernel P to it will leave it unaffected. The goal of MCMC is to find an invariant transition kernel P which satisfies (6.2) for a *given* target distribution Π. If we then simulate the Markov chain and wait until it reaches stationarity, then the resulting samples are distributed according to Π.
A condition that is related to (6.2) is called *detailed balance*, which states

$$P_{ij}\Pi(x_i) = P_{ji}\Pi(x_j). \tag{6.3}$$

Importantly, also detailed balance guarantees that Π is a stationary distribution of the considered Markov chain, which can be seen immediately by summing over i (or j) on both sides of (6.3), which leads directly to (6.2). The advantage of detailed balance is that it is a *local* condition that is easier to check. Correspondingly, many MCMC methods construct Markov chains that obey detailed balance such as the Metropolis-Hastings algorithm. Note that while (6.2) and (6.3) guarantee that Π is a stationary distribution, they do not necessarily ensure that Π is the only stationary distribution. Correspondingly, Π may not always be reached from any initial condition. This can be achieved by additionally requiring the Markov chain to be ergodic, which implies that it has a unique stationary distribution, which must be Π when also (6.2) and/or (6.3) are satisfied.

6.1 Gibbs Sampling

The first MCMC sampler that we discuss is known as the *Gibbs sampler.* Before we consider the general case, we will motivate the idea using a two-dimensional problem. Let us assume we want to sample from a two-dimensional distribution $\Pi(x) = \Pi(y, z)$. A key assumption of the Gibbs sampler is that while it is hard to sample directly from the joint distribution Π, it is easy to sample from the conditional distributions $\Pi(y \mid z)$ and $\Pi(z \mid y)$. While at first sight, this looks like a pretty strong assumption, this is indeed the case for many statistical problems encountered in practice. The two-dimensional Gibbs sampler iteratively samples from the conditional distributions starting from an arbitrary initial condition, i.e.,

$$y_{n+1} \sim \Pi(y \mid z_n) \tag{6.4}$$
$$z_{n+1} \sim \Pi(z \mid y_{n+1}) \tag{6.5}$$

for $n = 1, 2, \ldots n_{max}$.

Theorem 6.1. *$X_n = (Y_n, Z_n)$ is a Markov Chain with stationary distribution $\Pi(y, z)$.*

Proof: *We have to show that*

$$\sum_y \sum_z P(\hat{y}, \hat{z} \mid y, z)\Pi(y, z) = \Pi(\hat{y}, \hat{z}). \tag{6.6}$$

The transition kernel P of the Gibbs sampler is given by

$$P(\hat{y}, \hat{z} \mid y, z) = \Pi(\hat{z} \mid \hat{y})\Pi(\hat{y} \mid z), \tag{6.7}$$

whereas we assume that in each iteration n, we first resample y and subsequently z. Note that also the reverse order would be possible. Inserting the transition kernel into 6.6 yields

$$\sum_y \sum_z \Pi(\hat{z} \mid \hat{y})\Pi(\hat{y} \mid z)\Pi(y, z) = \Pi(\hat{y}, \hat{z}). \tag{6.8}$$

By manipulating this expression, we can show that

$$\begin{aligned}
\sum_y \sum_z \Pi(\hat{z} \mid \hat{y})\Pi(\hat{y} \mid z)\Pi(y, z) &= \sum_z \Pi(\hat{z} \mid \hat{y})\Pi(\hat{y} \mid z) \sum_y \Pi(y, z) && (6.9)\\
&= \sum_z \Pi(\hat{z} \mid \hat{y})\Pi(\hat{y} \mid z)\Pi(z) && (6.10)\\
&= \sum_z \Pi(\hat{z} \mid \hat{y})\Pi(\hat{y}, z) && (6.11)\\
&= \Pi(\hat{z} \mid \hat{y}) \sum_z \Pi(\hat{y}, z) && (6.12)\\
&= \Pi(\hat{z} \mid \hat{y})\Pi(\hat{y}) && (6.13)\\
&= \Pi(\hat{z}, \hat{y}). \quad \square && (6.14)
\end{aligned}$$

This shows that the Gibbs sampler has the correct target distribution $\Pi(y, z)$.

6.1.1 Multivariate case

The Gibbs sampler can be extended to distributions with more than two variables in a straightforward manner. For instance, if we want to sample from a K-dimensional probability distribution $\Pi(x_1, \dots, x_K)$, the Gibbs sampler would iteratively sample from the so-called *full conditional distributions* where one of the K variables is resampled conditionally on all other variables, i.e.,

1. $x_{1,n+1} \sim \Pi(x_1 \mid x_{2,n}, x_{3,n}, \dots, x_{K,n})$
2. $x_{2,n+1} \sim \Pi(x_2 \mid x_{1,n+1}, x_{3,n}, \dots, x_{K,n})$
   $\vdots$
3. $x_{K,n+1} \sim \Pi(x_K \mid x_{1,n+1}, \dots, x_{K-1,n+1})$,

with $x_{k,n}$ as the kth variable at iteration n. We remark that the order in which the variables are resampled can be chosen. In the algorithm above, we consider a fixed-sequence scan, which means that we resample the variables in a round-robin fashion $(1 \to 2 \to \ldots \to K \to 1 \to 2 \ldots)$. An alternative strategy is called random-sequence scan, where the update sequence is chosen randomly (e.g., $5 \to 2 \to 9 \ldots$). Moreover, one can group several variables together and update them jointly within a single step conditionally on all other variables (e.g., $\Pi(x_1, x_2 \mid x_3) \to \Pi(x_3 \mid x_1, x_2) \to \ldots$), which can improve the convergence of the sampler. While all variants of the Gibbs sampler have the right stationary distribution Π, they may differ in certain properties. For instance, certain random-sequence scan algorithms exhibit the detailed balance property, while this is generally not true for fixed-scan algorithms.

Example 6.1. *Imagine we want to study the statistical relationship between price λ and service cost γ of cars. We are told that the price of cars can be well described by a Gamma distribution $p(\lambda) = \Gamma(\alpha, \beta)$, with α and β as the shape and inverse scale parameter of this distribution. Furthermore, we find out that for a given price λ, the service cost γ follows an exponential distribution with mean $\frac{1}{c\lambda}$ and c as a known constant such that $p(\gamma \mid \lambda) = Exp(c\lambda)$. Our goal is to analyze the joint distribution*

$$p(\lambda, \gamma) = p(\gamma \mid \lambda) p(\lambda), \tag{6.15}$$

which captures the statistical relation between price and service cost. To do so, we try to generate random numbers from this distribution using Gibbs sampling, which we could then use to estimate several statistical properties of the model such as the correlation between price and service cost. To perform Gibbs sampling, we need the conditional distributions $p(\gamma \mid \lambda)$ and $p(\lambda \mid \gamma)$. The former one, we already know from our statistical assumptions. To calculate the latter one, we can use Bayes' rule:

$$p(\lambda \mid \gamma) = \frac{p(\gamma \mid \lambda) p(\lambda)}{p(\gamma)} \propto p(\gamma \mid \lambda) p(\lambda). \tag{6.16}$$

If we plug in the definitions of the exponential and Gamma distributions, we obtain

$$p(\lambda \mid \gamma) \propto c\lambda e^{-c\lambda\gamma} \frac{\beta^\alpha}{\gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda} \tag{6.17}$$

$$\propto \lambda^\alpha e^{-\lambda(c\gamma+\beta)}. \tag{6.18}$$

The last expression is equivalent to a Gamma distribution $\Gamma(\alpha + 1, c\gamma + \beta)$ up to a scaling constant that is independent of λ. We can therefore conclude that $p(\lambda \mid \gamma) = \Gamma(\alpha+1, c\gamma+\beta)$. Therefore, we can sample from the joint distribution $p(\lambda, \gamma)$ by iteratively sampling from

1. $\gamma_{n+1} \sim Exp(c\lambda_n)$
2. $\lambda_{n+1} \sim \Gamma(\alpha + 1, c\gamma_{n+1} + \beta)$.

6.2 Metropolis-Hastings Sampling

Consider a Markov Chain with one-step transition matrix $\mathbf{P}$. A key question in Markov Chains is to ask when a chain has a stationary distribution $\mathbf{P}^n = \pi$ for $n \to \infty$, and what that distribution is. This was discussed in Chapter 3. Markov-Chain Monte Carlo (MCMC) methods as introduced in Chapter **??** consider the inverse problem: given a known *target distribution* π, how can one generate a Markov Chain (or its transition matrix) that has exactly this distribution as its stationary distribution.

The standard Markov-Chain Monte Carlo method was developed by Ulam and von Neumann in the 1940s and published in 1949. It was Nicolas Metropolis, however, who is said to have suggested the name "Monte Carlo" for this class of algorithms. The Metropolis algorithm, a classical sampler to computationally solve MCMC problems, originated from work toward the hydrogen bomb at Los Alamos National Laboratory (New Mexico, USA) and can be seen as the first Markov-Chain Monte Carlo (MCMC) algorithm in history. It was published by Metropolis in 1953. The original Metropolis algorithm was only used in physics, but it was generalized to applications in statistics by Hastings in 1970. Since then, the resulting form of the algorithm is commonly referred to as the *Metropolis-Hastings Algorithm.* It became famous because it performs much better in high-dimensional spaces than the original Monte Carlo methods, such as accept-reject sampling or umbrella sampling (i.e., it is less prone to fall victim to the curse of dimensionality). Today, the Metropolis-Hastings sampler is the basis of modern Bayesian computation.

The Metropolis-Hastings algorithm is related to Gibbs sampling, but instead of updating components of $\vec{X}_t = (X_1, \dots, X_n)_t$ one at a time, it uses a local proposal and a rejection mechanism to compute $\vec{X}_n$ from $\vec{X}_{n-1}$. Also in this case, the sequence $\vec{X}_0, \vec{X}_1, \dots$ is a Markov Chain. The main advantage over Gibbs sampling is that the Metropolis-Hastings sampler does not require that it is possible/easy to sample from the full conditional probability distributions, and it generally has smaller mixing times than the Markov Chain produces by Gibbs sampling, in particular in cases when components of $\vec{X}$ are highly correlated.

6.2.1 Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm is an MCMC sampler that generates a Markov Chain with a given stationary (target) distribution π. Like Gibbs sampling, the algorithm generates samples from a proposal distribution $q(\vec{y}|\vec{x})$ given the current state of the chain $\vec{x}$. Unlike Gibbs sampling, however, π only needs to be known up to a constant. This is particularly important for Bayesian computation, where the posterior is often available only in unnormalized form. The Metropolis-Hastings algorithm generates a Markov Chain with stationary distribution π from any $f = C\pi$ with C any constant.

The proposal distribution is required to be properly normalized, i.e.,

$$\int q(\vec{y}|\vec{x})\mathrm{d}\vec{y} = 1,$$

so why not directly use $q(\vec{y}|\vec{x})$ as a transition kernel for the chain? The reason is that this would not satisfy detailed balance (see **??**), which is a sufficient condition for π being an equilibrium distribution. Detailed balance requires that the probability of being in state $\vec{x}$ and going to $\vec{y}$ from there is the same as the probability of doing the reverse transition if the current state is $\vec{y}$. This is true if and only if

$$\pi(\vec{x})q(\vec{y}|\vec{x}) = \pi(\vec{y})q(\vec{x}|\vec{y}), \tag{6.19}$$

which is not the case for arbitrary $q(\vec{y}|\vec{x})$. If the two sides are not the same in general, then one must be bigger than the other. Consider for example the case where

$$\pi(\vec{x})q(\vec{y}|\vec{x}) > \pi(\vec{y})q(\vec{x}|\vec{y}),$$

i.e., moves $\vec{x} \to \vec{y}$ are more frequent than the reverse. The other case is analogous. The idea is then to adjust the transition probability $q(\vec{y}|\vec{x})$ with an additional probability of move $0 < \rho \leq 1$ in order to reduce it to what is should be according to detailed balance:

$$\pi(\vec{x})q(\vec{y}|\vec{x})\rho = \pi(\vec{y})q(\vec{x}|\vec{y}). \tag{6.20}$$

Thus, the condition of detailed balance can be used to determine ρ by solving Eq. 6.20 for ρ as:

$$\rho = \frac{\pi(\vec{y})}{\pi(\vec{x})}\frac{q(\vec{x}|\vec{y})}{q(\vec{y}|\vec{x})} = \frac{C\pi(\vec{y})}{C\pi(\vec{x})}\frac{q(\vec{x}|\vec{y})}{q(\vec{y}|\vec{x})} = \frac{f(\vec{y})}{f(\vec{x})}\frac{q(\vec{x}|\vec{y})}{q(\vec{y}|\vec{x})}. \tag{6.21}$$

In the second step, we realized that replacing the target distribution π with an unnormalized version $f = C\pi$ for any constant C leaves ρ unchanged because the unknown normalization factor C cancels.
The resulting Metropolis-Hastings algorithm produces a Markov Chain $\{\vec{x_k}\}$, as given in Algorithm 1.

Algorithm 1 Metropolis-Hastings

```
1: procedure METROPOLISHASTINGS($\vec{x}_0$)                ▷ start point $\vec{x}_0$
2:     for $k = 0, 1, 2, \ldots$ do
3:         Sample $\vec{y}_k \sim q(\vec{y}|\vec{x}_k)$
4:         Compute $\rho = \min\left\{1, \frac{f(\vec{y}_k)q(\vec{x}_k|\vec{y}_k)}{f(\vec{x}_k)q(\vec{y}_k|\vec{x}_k)}\right\}$
5:         With probability $\rho$, set $\vec{x}_{k+1} = \vec{y}_k$; else set $\vec{x}_{k+1} = \vec{x}_k$.
6:     end for
7: end procedure
```

The factor ρ is called the *Metropolis-Hastings acceptance probability.* The minimum operator in line 4 of Algorithm 1 accounts for the case when moves $\vec{y} \to \vec{x}$

are too frequent and hence the correction factor, when placed on $\vec{x}$ would be larger than 1. In this case, the move is always accepted (and the reverse move reduced to satisfy detailed balance).
The Metropolis-Hastings algorithm has some similarity with the accept-reject sampling method (see Chapter 2.4). Both depend only on ratios of probabilities. Therefore, Algorithm 1 can also be used for random variate generation from π. The important difference is that in Metropolis-Hastings sampling, it may be that $\vec{x}_{k+1} = \vec{x}_k$, which has probability zero in a continuous Accept-Reject method. This means that Metropolis-Hastings samples are correlated (as two subsequent samples are identical with probability $1 - \rho$ and therefore perfectly correlated in this case) and not *i.i.d.*, as in accept-reject sampling.

6.2.2 Convergence properties

Any Markov Chain generated by the Metropolis-Hastings algorithm has the following properties:

1. Irreducibility $\iff q(\vec{y}|\vec{x}) > 0 \quad \forall \vec{x}$,
2. Aperiodicity $\iff \rho < 1$ (i.e., non-zero probability of $\vec{x}_{k+1} = \vec{x}_k$),
3. Ergodicity.

In addition, it satisfies detailed balance by design. The first two are easy to see, while proving ergodicity is technically involved. Irreducibility requires that no absorbing states exist, which is clearly the case if the outgoing transition probability from any state $\vec{x}$ is always non-zero. Aperiodicity requires that there is no sequence of states that exactly repeats with a certain period. This is clearly the case if every now and then, randomly, the chain stays in its present state, disrupting any regular cycle.

6.2.3 One-step transition kernel

The one-step transition probability of the Markov Chain generated by Algorithm 1 is:

$$P_{\vec{x},\vec{y}} = \rho(\vec{y},\vec{x})q(\vec{y}|\vec{x}) + (1-a)\delta(\vec{x}), \tag{6.22}$$

where δ is the Dirac delta distribution and

$$a = \int \rho(\vec{y},\vec{x})q(\vec{y}|\vec{x})\mathrm{d}\vec{y}. \tag{6.23}$$

Theorem 6.2. *If the Markov Chain generated by Metropolis-Hastings sampling is irreducible, then for any integrable real-valued function h*

$$\lim_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} h(\vec{x}_t) = \mathbb{E}[h(\vec{X})] \tag{6.24}$$

for every starting point $\vec{x}_0$.

This means that expectations can be replaced by empirical means, which provides a powerful property in practical applications where expectations of unnormalized distributions are to be estimated. The reason this works is because the chain is ergodic. Indeed, for ergodic stochastic processes, ensemble averages and time averages are interchangeable. However, it is important to only use the samples of the Metropolis-Hastings chain after the algorithm has converged. The first couple of samples generated must be ignored because they do not yet come from the correct target distribution π. This immediately raises the question how to detect if the chain has converged, i.e., after how many iterations one can start using the samples.

6.2.4 Special Proposal Choices

The main degree of freedom one has to tune the Metropolis-Hastings algorithm to specific applications is the choice of the proposal distribution $q(\vec{y}|\vec{x})$. While a myriad of possible choices exist, we here review two particularly important special cases.

6.2.4.1 Symmetric proposals

A symmetric proposal is a proposal that only depends on the distance between the two states and not on the actual source state, thus:

$$q(\vec{y}|\vec{x}) = g(\vec{y} - \vec{x})$$

with symmetric (i.e., even) distribution $g(-\vec{z}) = g(\vec{z})$. This greatly simplifies the algorithm, as the proposal does not depend on the source state any more, but only on the "step length", which is typically the case when simulating conservative systems (i.e., potential fields). In this case, the proposed new state $\vec{y}_k$ can be written as:

$$\vec{y}_k = \vec{x}_k + \vec{\varepsilon}$$

with the random variable $\vec{\varepsilon} \sim g$. Then, the Metropolis-Hastings acceptance probability simplifies to:

$$\rho = \min\left\{1, \frac{f(\vec{y}_k)}{f(\vec{x}_k)} \frac{g(\vec{y}_k - \vec{x}_k)}{g(\vec{x}_k - \vec{y}_k)}\right\} = \min\left\{1, \frac{f(\vec{y}_k)}{f(\vec{x}_k)}\right\} \tag{6.25}$$

due to the symmetry of g. With this choice of proposal, the algorithm is also called *Random Walk Metropolis-Hastings algorithm.* It accepts every move to a more probable state with probability 1 and moves to less probable states with probability $f(\vec{y})/f(\vec{x}) = \pi(\vec{y})/\pi(\vec{x}) < 1$. A popular (because easy to simulate from) choice for g is the multivariate Gaussian distribution, in which case the Markov chain generated is a Brownian walk (see Chapter **??**).

6.2.4.2 Independent proposals

A further simplification is given by assuming an independent proposal for which the probability only depends on the target state, but neither on the source state nor on the distance between source and target. Therefore, an independent proposal can be written as:

$$q(\vec{y}|\vec{x}) = g(\vec{y})$$

for all $\vec{x}$ and simply gives the probability of going to state $\vec{y}$ regardless from where. The Metropolis-Hastings acceptance probability then becomes:

$$\rho = \min\left\{1, \frac{f(\vec{y}_k)}{f(\vec{x}_k)}\frac{g(\vec{x}_k)}{g(\vec{y}_k)}\right\}. \tag{6.26}$$

With this choice of proposal, the algorithm is called *Independent Metropolis-Hastings algorithm.* It generates $\vec{y}_k \sim g(\vec{y})$ *i.i.d.*. Still, the $\vec{x}_k$ are not *i.i.d.*, since not every move is accepted. The independent Metropolis-Hastings algorithm converges if $g(\vec{y}) > 0$ for all $\vec{y}$ in the support of f. In particular, it converges if there exists a constant $M > 0$ such that $f(\vec{x}) \leq Mg(\vec{x})$ for all $\vec{x}$. Every such pair of functions (f, g) can also be used in Accept-Reject sampling, but independent Metropolis-Hastings has a higher expected accept probability (namely $\frac{1}{M}$ at stationary) and is therefore typically more efficient.

6.2.5 Stopping Criteria

The question when to stop a Metropolis-Hastings sampler is mostly application dependent. The short answer is: when enough samples have been collected. However, it is important to keep in mind that the samples generated by Metropolis-Hastings are not independent. Since the same sample may appear repeatedly, the information gathered about the target distribution π is not directly given by the number of samples generated. It is instructive to compute the autocorrelation $c(\tau)$ of a given chain and thin it by using only every τ-th sample, where τ is chosen such that $c(\tau) \approx 0$. This is particularly important when estimating variances (or higher-order moments) of π, where only independent samples guarantee convergence to the correct result.

6.3 Thinning and Convergence Diagnostics

It takes time until the Markov Chain generated by the Metropolis-Hastings algorithm has converged (in distribution!), i.e., until $\{x_k\} \sim \pi(x)$. The time until convergence is called *burn-in* period. The samples generated during the burn-in period are not to be used as they do not come from the desired distribution (the Markov Chain has not yet reached stationarity). Convergence Diagnostics are designed to determine when the burn-in period is over or to decide on *thinning* of the chain. "Thinning" refers to the process of only using every k-th sample from the chain (e.g., every 10-th) and is used to reduce correlation between samples (remember that there is a non-zero probability that the Metropolis-Hastings algorithm outputs the same sample repeatedly). While thinning is irrelevant when using the chain to estimate means, variances can only be estimated from uncorrelated samples.

The design of convergence diagnostics for Markov Chains is an open research topic and most available methods are heuristics. This means that they usually work in practice, but are not based on theory and provide no guaranteed performance. What may immediately come to mind is to run many chains and use the Law of Large Numbers (see Section 4.1) to check for normality of the means computed across chains. While this would work, it is very inefficient as it requires a very large (potentially millions) number of chains to be generated. Much more efficient convergence tests are available. We present the two most frequently used diagnostics for Markov Chains: one based on a statistical test, and one based on empirical correlation. Both are general to arbitrary stationary Markov Chains, and not limited to the Metropolis-Hastings algorithm.

6.3.1 Gelman-Rubin Test

The Gelman-Rubin test is a statistical test that provides a general convergence diagnostic for stationary Markov Chains. It is based on comparing intra-chain variance with inter-chain variance and requires generating M independent realizations of the chain from different starting points ${x_0}^m$, $m = 1, \dots, M$. Consider scalar chains and denote these M chains by

$$\{x_k\}^m, \quad m = 1, \dots, M \tag{6.27}$$

and only use the N samples $k = l, \dots, N + l$ of each chain, i.e., discard the first $l - 1$ samples.
Then, calculate:

- the mean of each chain $\hat{\mu}_m = \frac{1}{N}\sum_{i=1}^{N} x_i^m$,
- the empirical variance of each chain $\hat{\sigma}_m^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i^m - \hat{\mu}_m)^2$,
- the mean across all chains $\hat{\mu} = \frac{1}{M}\sum_{m=1}^{M}\hat{\mu}_m$,
- the variation of the means across chains $B = \frac{N}{M-1}\sum_{m=1}^{M}(\hat{\mu}_m - \mu)^2$, and
- the average chain variance $W = \frac{1}{M}\sum_{m=1}^{M}\hat{\sigma}_m^2$.

Then, compute the Gelman-Rubin test statistic, defined as:

$$V = \left(1 - \frac{1}{N}\right)W + \frac{M+1}{MN}B. \tag{6.28}$$

The chain has converged if $\hat{R} := \sqrt{\frac{V}{W}} \approx 1$ (in practice $|\hat{R} - 1| < 0.001$). Choose the smallest possible l (i.e., length of the burn-in period) such that this is the case.
The reason this test works is that both W and V are unbiased estimators of the variance of π (not proven here). Therefore, for converged chains, they should be the same. For increasing l, $\hat{R}$ usually approaches unity from above (for initially over-dispersed chains).

6.3.2 Autocorrelation Test

The Gelman-Rubin test works well in practice and is based on solid reasoning, but it requires M independent replica of the chain. If only a single realization of the chain is available, it cannot be used. The (weaker, in terms of statistical power) autocorrelation test can then be used instead. It is based on computing the autocorrelation function of the Markov chain, defined as:

$$c(\tau) = \frac{\sum_{i=1}^{N-\tau}(x_i - \hat{\mu})(x_{i+\tau} - \hat{\mu})}{\sum_{i=1}^{N}(x_i - \hat{\mu})^2}, \tag{6.29}$$

where $\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$ is the mean of the chain samples. The autocorrelation function $c(\tau)$ is a function of the *time lag* τ and is, in practice, computed iteratively for increasing $\tau = 1, 2, \ldots$. For each τ, $c(\tau)$ tells us what the average correlation is between samples that are τ time points apart. A sufficient condition for convergence is that the samples are uncorrelated. Therefore, choose the smallest τ for which $c(\tau) \approx 0$ as the length of the burn-in period. Beyond this time point, samples are effectively independent.

While uncorrelatedness is a sufficient condition for convergence, it is not necessary. Recall that the Metropolis-Hastings algorithm may produce correlated samples even at stationarity. In order to obtain uncorrelated samples, the chain then needs to be thinned. The autocorrelation test is also useful in deciding the thinning factor. Indeed, using only use every τ-th sample from the chain provides roughly uncorrelated samples and is a good choice for thinning.

Chapter 7

Stochastic Optimization

Optimization problems are amongst the most widespread in science and engineering. Many applications, from machine learning over computer vision to parameter fitting, can be formulated as optimization problems. An optimization problem consists of finding the optimal $\vec{\vartheta}^*$ such that

$$\vec{\vartheta}^* = \arg\min_{\vec{\vartheta}} h(\vec{\vartheta})$$
$$\text{for } h : \mathbb{R}^n \to \mathbb{R}, \quad \vec{\vartheta} \mapsto h(\vec{\vartheta}) \tag{7.1}$$

for some given function h. This function is often called the *cost function*, *loss function*, *fitness function*, or *criterion function*. Following the usual convention, we define an optimization problem as a minimization problem, where maximization is equivalently contained when replacing h with $-h$.
Problems of this type, where h maps to $\mathbb{R}$ are called *scalar real-valued optimization problems* or *real-valued single-objective optimization*. Of course, one can also consider complex-valued or integer-valued problems, as well as vector-valued (i.e., multi-objective) optimization. Many of the concepts considered here generalize to those cases, but we do not describe such generalizations here.
For large n (i.e., high-dimensional domains) or non-convex $h(\cdot)$, there are no efficient deterministic algorithms to solve the above optimization problem. In fact, a non-convex function in n dimensions can have exponentially (in n) many (local) minima. Since the goal according to Eq. 7.1 is to find the best local minimum, i.e., the *global minimum*, deterministic approaches have a worst-case runtime that is exponential in n. A typical way out is the use of randomized stochastic algorithms. While randomized algorithms can be efficient, they provide no performance guarantees, i.e., they may not converge, may not find any minimum, or may get stuck in a sub-optimal local minimum. The only tests that can be used to compare and select randomized optimization algorithms are heuristic benchmarks, typically obtained by running large ensembles of Markov chains. There are two widely used standard suites of test problems: the IEEE CEC2005-2020 standard and the ACM GECCO BBOB. They define test problems with known exact solutions, as well as detailed evaluation protocols, on

which algorithms are to be compared and tested. About 200 different stochastic optimization algorithms have been benchmarked on these tests with the test results publicly available.

Many stochastic optimization methods have the big advantage that they do not require h to be known in closed form. Instead, it is often sufficient that h can be *evaluated* point-wise. Therefore, h does not have to be a mathematical function, but can also be a numerical simulation, taking a laboratory measurement, or user input. Algorithms of this sort are called *black-box optimization algorithms*, and optimization problems with an unknown, but evaluatable h are called *black-box optimization problems.*

Designing good stochastic optimization algorithms is a vast field of research, which is in itself split into sub-fields such as *evolutionary computing*, *randomized search*, and *biased sampling*. Many exciting concepts, from evolutionary biology over information theory to Sobolev calculus, are being exploited on this problem. Here, we exemplarily discuss examples of classes of algorithms for Monte-Carlo optimization from each of these sub-fields: Stochastic descent and random pursuit from the class of randomized search heuristics, simulated annealing from the class of biased sampling methods, and evolution strategies from the class of evolutionary algorithms.

7.1 Stochastic Exploration

Stochastic exploration is a classic approach for the case when the *search space*, i.e., the domain of $\vec{\vartheta} \in \Theta \subset \mathbb{R}^n$ is bounded. In this case, a straightforward approach is to sample:

$$\vec{\mu}_1, \dots, \vec{\mu}_m \sim \mathcal{U}(\Theta) \tag{7.2}$$

uniformly over Θ and then use:

$$\vec{\vartheta}^* \approx \arg\min(h(\vec{\mu}_1), \dots, h(\vec{\mu}_m)). \tag{7.3}$$

This clearly converges to the correct global minimum for $m \to \infty$ and has a linear computational complexity in $O(m)$. For general h, the number of samples required to reach a given probability of finding the global minimum is $m \propto |\Theta| = C^n$, where the constant $C > 0$ is the linear dimension of the search space Θ.

Stochastic exploration therefore converges exponentially slowly and is particularly impractical in cases where $h(\cdot)$ is costly to evaluate, e.g., where it is given by running a simulation or performing a measurement. This is because stochastic exploration “blindly” samples the search space without exploiting any structure or properties of h that may be known.

7.2 Stochastic Descent

A first attempt to reduce the number of samples required by exploiting additional information about h is to use the gradient of h. This is inspired by deterministic gradient descent as a popular algorithm to find local minima:

$$\vec{\vartheta}_{j+1} = \vec{\vartheta}_j - \alpha_j \nabla h(\vec{\vartheta}_j), \quad j = 0, 1, 2, \dots \tag{7.4}$$

with step size $\alpha_j > 0$. If the step size is well chosen (or dynamically adjusted), this converges to *a* (not necessarily the nearest) local minimum of h around the given starting point $\vec{\vartheta}_0$. In practice, however, the gradient ∇h may not be known analytically (in black-box problems, even h itself may not be known analytically), and numerically approximating gradients may be expensive for large n.
Stochastic descent therefore replaces the iteration over the gradient with:

$$\vec{\vartheta}_{j+1} = \vec{\vartheta}_j - \frac{\alpha_j}{2\beta_j} \Delta h(\vec{\vartheta}_j, \beta_j \vec{u}_j)\vec{u}_j, \quad j = 0, 1, 2, \dots \tag{7.5}$$

with $\vec{u}_j$ *i.i.d.* uniform random variates on the unit sphere (i.e., $|\vec{u}_j| = 1$) and

$$\Delta h(\vec{x}, \vec{y}) = h(\vec{x} + \vec{y}) - h(\vec{x} - \vec{y}) \approx 2|\vec{y}| \nabla h(\vec{x}) \cdot \vec{y}. \tag{7.6}$$

The latter is because the finite difference

$$\frac{h(\vec{x} + \vec{y}) - h(\vec{x} - \vec{y})}{2|\vec{y}|} \approx \nabla h(\vec{x}) \cdot \vec{y}$$

is an approximation to the directional derivative of h in direction y. This iteration does not proceed along the steepest slope and therefore has some potential to overcome local minima. Stochastic descent has two algorithm parameters:

- α_j: step size,
- β_j: sampling radius.

One can show that stochastic descent converges to a local optimum if $\alpha_j \downarrow 0$ for $j \to \infty$ and $\lim_{j \to \infty} \frac{\alpha_j}{\beta_j} = \text{const} \neq 0$. There are no guarantees of global convergence. The problem in practice usually is the correct choice and adaptation of α_j and β_j. The biggest advantage over stochastic exploration is a greatly increased convergence speed, and that the method also works in unbounded search spaces.
A side note on uniform random numbers on the unit sphere: Simply sampling uniformly in the spherical angles leads to a bias toward the poles. One needs to correct the samples with the arccos of the polar angle in order for them to have uniform area density on the unit sphere.

Example 7.1. *For example, in 3D, uniform random points $\vec{u} \sim \mathcal{U}(\mathcal{S}^2)$ on the unit sphere S^2 can be sampled by:*

$$\text{sampling } \varphi \sim \mathcal{U}(0, 2\pi) \tag{7.7}$$

$$\text{sampling } w \sim \mathcal{U}(0, 1) \tag{7.8}$$

$$\text{computing } \theta = \arccos(2w - 1) \tag{7.9}$$

and then using $(r = 1, \varphi, \theta)$ as the spherical coordinates of the sample point on the unit sphere, where the polar angle θ runs from 0 (north pole) to π (south pole) and the azimuthal angle φ from 0 to 2π.

In higher dimensions $n > 3$, uniform random points on the unit hypersphere S^{n-1} can be sampled by successive planar rotations, which can be efficiently computed using Givens rotations. This is, e.g., described in: *C. L. Müller, B. Baumgartner, and I. F. Sbalzarini. Particle swarm CMA evolution strategy for the optimization of multi-funnel landscapes. In Proc. IEEE Congress on Evolutionary Computation (CEC), pages 2685-2692, Trondheim, Norway, May 2009.*

7.3 Random Pursuit

The next example of a classic algorithm is Random Pursuit. It is related to stochastic descent, but does not require choosing or adapting any step-size parameters (like α_j and β_j in stochastic descent). It tries to keep the good convergence properties of stochastic descent, while relaxing its most salient drawback. The procedure is given in Algorithm 2.

Algorithm 2 Random Pursuit

```
1: procedure RandomPursuit($\vec{\vartheta}_0$)                         ▷ start point $\vec{\vartheta}_0$
2:     for $k = 0, 1, \dots, N-1$ do
3:         $\vec{u}_k \sim \mathcal{U}(S^{n-1})$                              ▷ uniform on the unit sphere
4:         $\vec{\vartheta}_{k+1} = \vec{\vartheta}_k +$ LineSearch($\vec{\vartheta}_k, \vec{u}_k$)$\vec{u}_k$
5:     end for
6:     return $\vec{\vartheta}_N$
7: end procedure
```

The subroutine LineSearch($\vec{x}, \vec{y}$) finds the distance to the minimum of h along a line in direction $\vec{y}$ starting from $\vec{x}$. Since this is a 1D problem, it can efficiently be solved using stochastic exploration, usually followed by local descent. Note that bisection search only works for convex h.

Random Pursuit does not require setting any step sizes, thanks to the line search subroutine. It is guaranteed to converge to a (local) minimum of h but only works for bounded search spaces Θ. There are no guarantees of global convergence.

7.4 Simulated Annealing

A classic stochastic optimization method is Simulated Annealing, as developed in 1953 by Metropolis. Strictly speaking, this is not a pure Monte-Carlo method, but a Markov-Chain Monte-Carlo method, because the samples are not *i.i.d.* The basic idea is to perform stochastic exploration with a proposal that focuses on promising regions. This should speed up stochastic exploration and reduce its computational cost (i.e., number of samples required) to something tractable. Like stochastic exploration, simulated annealing only works for bounded search spaces Θ.

The simulated annealing algorithm is inspired from physics, particularly from statistical mechanics. There, the probability of finding a collection of particles (e.g., atoms) in a certain state is proportional to the *Boltzmann factor*:

$$P(\text{state}) \propto \mathrm{e}^{-E/T}, \tag{7.10}$$

where E is the energy of the state and T is the temperature in the system. Upon cooling ($T \downarrow$), the system settles into low-energy states, i.e., it finds minima in $E(\cdot)$. In Algorithm 3, this analogy is exploited to perform stochastic optimization over general functions h.

Algorithm 3 Simulated Annealing

```
1: procedure SimulatedAnnealing($\vec{\vartheta}_0$)                ▷ start point $\vec{\vartheta}_0$
2:     for $k = 0, 1, \ldots, N-1$ do
3:         $\vec{u}_k \sim \mathcal{U}(B(\vec{\vartheta}_k))$                   ▷ uniform in $B$
4:         $\Delta h = h(\vec{u}_k) - h(\vec{\vartheta}_k)$
5:         accept $\vec{\vartheta}_{k+1} = \vec{u}_k$ with probability $\rho = \min\{\mathrm{e}^{-\Delta h/T}, 1\}$,
6:             else set $\vec{\vartheta}_{k+1} = \vec{\vartheta}_k$
7:         $T_{k+1} = f(T_k)$                                ▷ adjust temperature
8:     end for
9:     return $\vec{\vartheta}_N$
10: end procedure
```

The Simulated Annealing algorithm actually amounts to a Metropolis-Hastings sampler, as discussed in Section 6.2, where the acceptance probability is given by the Boltzmann factor of the change in h the move would cause. This means that the Markov chain thus generated has a higher probability of moving in directions of decreasing h, but it can also "climb uphill" with some lower probability in order to escape local minima.

The proposal focuses on promising regions by only sampling over a *neighborhood B around* the current state. This neighborhood is typically chosen to be a box or a sphere, because sampling uniformly over these shapes is easy.

The key mechanism to push for lower-energy solutions is the gradual adaptation of the "temperature" T. Due to the physical analogy, the parameter T is called "temperature", although it is not actually a physical temperature, but an exploration range parameter for the sampler. In each iteration, T is reduced according to a fixed "cooling schedule" $f(\cdot)$, which is a function that computes the new temperature from the current one (e.g., as $T_{k+1} = 0.95\, T_k$).

Simulated Annealing converges to a (local) minimum of h for bounded search spaces Θ and for $T \downarrow 0$ not too fast. If the cooling happens too fast, then the sampler finds no acceptable points any more at all and gets stuck, i.e., does not converge to any (not even a local) minimum. Clearly, choosing the right cooling schedule $f(\cdot)$ is of paramount importance. If the cooling is too fast, the algorithm does not converge to a minimum. If the cooling is too slow, convergence is slow. Together with the need to choose the right neighborhood size B, these are the main difficulties in using Simulated Annealing in practice. Clearly, the choice of f and B has to depend on what the function h looks like, which may not always be known in a practical application.

7.5 Evolutionary Algorithms

Another classic family of nature-inspired stochastic optimization algorithms are Evolutionary Algorithms. This time, the motivation does not come from statistical mechanics, but from Darwinian evolution of genomes. The basic idea is to apply random "mutations" to the state $\vec{\vartheta}$ and then to evaluate in a "selection" whether or not to keep the new state. Like in simulated annealing, the nomenclature of the inspiring domain (here: evolutionary biology) is used. The key differences to simulated annealing are that:

- mutation is not done by uniformly sampling over a bounded neighborhood, but usually from a multivariate Gaussian with mean 0 and covariance matrix $\mathbf{\Sigma}$;
- selection can be seen as a cooling schedule f that depends on Δh rather than being pre-determined.

This addresses the biggest problem of simulated annealing: choosing the cooling schedule.
Evolutionary Algorithms were introduced by Ingo Rechenberg in 1971 and Hans-Paul Schwefel in 1974. Today, they comprise a large class of bio-inspired optimization algorithms, including genetic algorithms, ant-colony optimization, particle-swarm optimization, and evolution strategies. The nomenclature is somewhat fuzzy, but methods that operate over discrete domains Θ are generally referred to as *genetic algorithms*, whereas methods for continuous domains, such as the one in Eq. 7.1 where the domain of the cost function is a continuous set, are generally called *evolution strategies*.
In its simplest form, an evolution strategy proceeds as outlined in Algorithm 4.

Algorithm 4 (1+1)-ES Evolution Strategy

```
1: procedure 1+1-ES($\vec{\vartheta}_0$)                                    ▷ start point $\vec{\vartheta}_0$
2:     for $k = 0, 1, \dots, N-1$ do
3:         $\vec{u}_k = \vec{\vartheta}_k + \mathcal{N}(0, \mathbf{\Sigma}_k)$                     ▷ mutation
4:         $\vec{\vartheta}_{k+1} = \begin{cases} \vec{u}_k & \text{if } h(\vec{u}_k) \le h(\vec{\vartheta}_k) \\ \vec{\vartheta}_k & \text{else} \end{cases}$   ▷ selection
5:     end for
6:     return $\vec{\vartheta}_N$
7: end procedure
```

This type of evolution strategy is called a *(1+1)-ES*, where "ES" is short for "Evolution Strategy". The (1+1)-ES converges linearly on unimodal $h(\cdot)$, i.e.,

its convergence rate is asymptotically as good as that of deterministic gradient descent, but without requiring the gradient of h or any approximation of it.
ES are not limited to generating only one sample per iteration. This gives rise to an entire family of methods, classified as:

- (1+1)-ES: one "parent sample" $\vec{\vartheta}_k$ and one "offspring sample" $\vec{\vartheta}_{k+1}$ in each iteration ("generation"); see above.
- (1,λ)-ES: sample λ new points $\vec{u}_{k,1}, \dots, \vec{u}_{k,\lambda}$ *i.i.d.* from the same Gaussian mutation distribution in each iteration and set $\vec{\vartheta}_{k+1} = \arg\min_{\vec{u}_{k,i}}(h(\vec{u}_{k,i}))_{i=1}^{\lambda}$, i.e., keep the best offspring to become the parent of the next generation.
- (1+λ)-ES: same as above, but include the parent in the selection, $\vec{\vartheta}_{k+1} = \arg\min\{h(\vec{u}_{k,1}), \dots, h(\vec{u}_{k,\lambda}), h(\vec{\vartheta}_k)\}$, i.e., stay at the old point if none of the new samples are better.
- (μ,λ)-ES and (μ+λ)-ES: retain the best μ samples for the next iteration, which then has μ "parents". Use a linear combination (i.e., "genetic recombination", e.g., their mean or pairwise averages) of parents as the center for the mutation distribution of the new generation. In the comma version, do not include the parents in the selection; in the plus version, do include them, as above.

Evolution strategies are further classified with into those where the covariance matrix $\mathbf{\Sigma}$ of the Gaussian mutation distribution, i.e. the *mutation rates*, is constant, and those where it is dynamically adapted according to previously seen samples.

7.5.1 ES with fixed mutation rates

In an ES with fixed mutation rates, $\mathbf{\Sigma}$ is a constant and does not depend on k, thus: $\mathbf{\Sigma}_k = \mathbf{\Sigma} = \text{const}$ for all k. Typically, it is even chosen as $\mathbf{\Sigma} = \sigma^2\mathbf{1}$ for some constant scalar mutation rate σ. This means that the same mutation rate is applied to all elements of the vector $\vec{\vartheta}$ and mutations only act along coordinate axes.

7.5.2 ES with adaptive mutation rates

In all of the above variants of evolution strategies, the mutation rates can also be dynamically adapted depending on previous progress. The question of course is *how*, i.e., how to determine $\mathbf{\Sigma}_{k+1}$ from $\mathbf{\Sigma}_k$ and the current fitness values of the sample population.
There are three classic approaches to this, which we present below. Many more exist and can be found in the literature. This is an active field of research. Typically, these adaptive evolution strategies are empirically found to work well on test problems and in practical applications, but their convergence has so far not been proven and convergence rates are unknown.

7.5.2.1 Rechenberg's 1/5 rule

We keep the covariance matrix of the mutation distribution proportional to the identity matrix, i.e., isotropic: $\mathbf{\Sigma}_k = \sigma_k^2 \mathbf{1}$, but we adapt the scalar mutation rate σ_k over iterations/generations k as follows:
If less than 1/5 (i.e., 20%) of the new samples are better than the (best) old sample, then decrease $\sigma_{k+1} = d\sigma_k$ with some decrease factor $d < 1$. Else, increase $\sigma_{k+1} = e\sigma_k$ with some expansion factor $e > 1$. The fraction of new samples that are better than the (best) old sample is called the *success rate*.
For λ-sample strategies, the success rate can be computed per iteration if $\lambda \geq 5$. Otherwise, the success rate is computed as a running average over the past > 5 iterations, possibly with exponential forgetting.
The idea behind this empirical rule is that if less than 1/5 of the samples are successful, the algorithm is hopping around a minimum and the step size should be decreased in order to force convergence into that minimum. If more than 1/5 are successful, the algorithm is on a run toward a minimum (i.e., down a slope) and larger steps can be tried to increase convergence speed. Of course, the art becomes choosing the factors d and e.

7.5.2.2 Self-adaptation

Self-adaptation does away with the need of choosing decrease and expansion factors for the mutation rate. Instead, it lets the Darwinian selection process itself take care of adjusting the mutation rates. For this, each sample (i.e., "individual") has its own mutation rate $\sigma_{k,i}$, $i = 1, \dots \lambda$ in iteration k, and we again use isotropic mutations $\mathbf{\Sigma}_{k,i} = \sigma_{k,i}^2 \mathbf{1}$, but now with a different covariance matrix for each offspring sample

$$\vec{u}_{k,i} \sim \mathcal{N}(\vec{\vartheta}_k, \sigma_{k,i}^2 \mathbf{1}), \qquad i = 1, \dots, \lambda. \tag{7.11}$$

The individual mutation rates for each offspring are themselves sampled from a Gaussian distribution, as:

$$\sigma_{k,i} = \mathcal{N}(\sigma_k, s^2), \tag{7.12}$$

where σ_k is the mutation rate of the parent (or the h-weighted mean of the mutation rates of the parents for a μ-strategy), and s is a step size. This self-adaptation mechanism will automatically take care that samples with "good" mutation rates have a higher probability of becoming parents of the next generation and hence the mutation rate is inherited by the offspring as one of the "genetic traits". Choosing the step size s is unproblematic. Performance is robust over a wide range of choices. However, the main drawback of self-adaptation is its reduced efficiency. The same point potentially needs to be tried multiple times for different mutation rates, therefore increasing the number of samples required to converge.

7.5.2.3 Covariance matrix adaptation (Hansen & Ostermeier, 1993)

A more efficient (if not the most efficient known) way of adapting the mutation rates is by considering the path the sampler has taken over the past iterations

and using that to adapt a fully anisotropic covariance matrix $\mathbf{\Sigma}_k$ that can also have off-diagonal elements. This allows the mutation rates of different elements of the vector $\vec{\vartheta}$ to be different, in order to account for different scaling of the parameters in different coordinate directions of the domain. The off-diagonal elements are used to exploit correlations between search dimensions.
The classic algorithm to achieve this is CMA-ES, the evolution strategy with Covariance-Matrix Adaptation (CMA). The algorithm adapts the covariance matrix of the mutation distribution by rank-μ updates of $\mathbf{\Sigma}$ based on correlations between the previous best samples, which can be interpreted nicely in terms of information geometry. The algorithm uses Cholesky decompositions and eigenvalue calculations, and we are not going to give it in full detail here. We refer to online resources (e.g., wikipedia) for details.
CMA-ES roughly proceeds by:

1. sampling λ offspring from $\mathcal{N}(\vec{\vartheta}_k, \mathbf{\Sigma}_k)$
2. choosing the best $\mu < \lambda$: $\vec{u}_1, \dots, \vec{u}_\mu$
3. recombining them as $\vec{\vartheta}_{k+1} = \text{mean}(\vec{u}_1, \dots, \vec{u}_\mu)$
4. rank-μ update of $\mathbf{\Sigma}_k$ using $\vec{u}_1, \dots, \vec{u}_\mu \Longrightarrow \mathbf{\Sigma}_{k+1}$

A few remarks about CMA-ES:

- The rank-μ update requires an eigendecomposition of $\mathbf{\Sigma}$ and therefore has a computational cost of $O(n^3)$.
- No formal convergence proof exists for CMA-ES.
- CMA-ES can be interpreted as natural gradient descent in the space of sample distributions.
- CMA-ES usually is among the top performers in benchmarks (IEEE CEC 2005, ACM BBOB 2015).
- The Markov Chain generated by CMA-ES has nice stationarity and invariance properties. Note that CMA-ES still is a Markov-Chain Monte-Carlo method, since the rank-μ update of the covariance matrix only depends on the current samples.
- On convex deterministic problems, CMA-ES is about 10-fold slower than the deterministic BFGS optimizer, but CMA-ES is black-box and also works on non-convex and stochastic problems.

Chapter 8

Random Walks

In Chapter 3 we briefly discussed a special class of discrete-time Markov chains called *random walks*. Intuitively, a random walk (RW) is a stochastic process defined on a discrete lattice, which in each step, moves to any of its neighbouring states with a certain probability. RWs play a central role in many different fields including finance, physics or biology. In this chapter we will first discuss some properties of discrete RWs and show how those relate to continuous-time RWs. We consider a discrete-time, integer-valued Markov chain $X_n \in \mathbb{Z}$. At each time step, the state of the chain either increase or decrease by one. The corresponding transition kernel of this Markov chain is given by

$$P(X_{n+1} = x + 1 | X_n = x) = p \tag{8.1}$$

$$P(X_{n+1} = x - 1 | X_n = x) = q = 1 - p, \tag{8.2}$$

with $p \in [0, 1]$.

Definition 8.1. *X_n is called a one-dimensional Random Walk.*

8.1 Characterization and Properties

8.1.1 Kolmogorov-forward Equation

As any other Markov chain, a random walk can be characterized in terms of a Kolmogorov-forward equation

$$P(X_n = x) = pP(X_{n-1} = x - 1) + qP(X_{n-1} = x + 1). \tag{8.3}$$

This recursive equation can be intuitively interpreted. If the RW is in state x at time n, this implies that it was either in state $x + 1$ at time $n - 1$ and decreased by one, or it was at state $x - 1$ and increased by one. This provides a recursive way to calculate the probability of finding the RW at any state x at time n. The probability $P(X_n = x)$ will therefore consist of two additive contributions corresponding to the two outcomes. The first contribution is the probability that the walker was in $x-1$ at $n-1$ (i.e., $P(X_{n-1}=x-1)$) times the probability of moving upwards (i.e., p). The second contribution is the probability of finding the walker in $x+1$ at $n-1$ (i.e., $P(X_{n-1}=x+1)$) times the probability of moving downwards (i.e., q). Note that eq. (8.3) is generally infinite-dimensional, since we need to consider an equation for every possible x.

8.1.2 State Equation

Instead of characterizing the probability distribution of the RW, we can also formulate a stochastic equation that describes the stochastic time evolution of a single path of the RW. In particular, such equation reads

$$X_n = X_{n-1} + \Delta X_n = X_0 + \sum_{i=1}^{n} \Delta X_i, \tag{8.4}$$

with ΔX_i as the random increments of the RW and X_0 as some known starting point of the RW. These increments are *i.i.d.*binary random variables, i.e.,

$$\Delta X_i \sim \begin{cases} +1 & \text{with probability } p \\ -1 & \text{with probability } q. \end{cases} \tag{8.5}$$

8.1.3 Mean and Variance

We next study how the mean and variance of a RW evolve with time. To this end, we first calculate the mean and variance of the random increments ΔX_i. For the mean we obtain

$$\begin{aligned} \mu = \mathbb{E}\{\Delta X_n\} &= -1q + p && (8.6)\\ &= p - q && (8.7)\\ &= p - (1-p) && (8.8)\\ &= 2p - 1. && (8.9) \end{aligned}$$

For the variance we obtain correspondingly,

$$\begin{aligned} \sigma^2 = \mathbb{E}\{\Delta X_n^2\} - \mu^2 &= p + q - (2p-1)^2 && (8.10)\\ &= 1 - 4p^2 + 4p - 1 && (8.11)\\ &= 4p(1-p) && (8.12) \end{aligned}$$

Using these results, we can now calculate the mean and variance of X_n for any n. For the mean, we obtain

$$\mathbb{E}\{X_n\} = X_0 + \mathbb{E}\Big[\sum_{i=1}^{n} \Delta X_i\Big] = X_0 + \sum_{i=1}^{n} \mathbb{E}\{\Delta X_i\} = X_0 + n\mu. \tag{8.13}$$

The variance becomes

$$Var\{X_n\} = Var\Big[X_0 + \sum_{i=1}^{n} \Delta X_i\Big] = \sum_{i=1}^{n} Var\{\Delta X_i\} = n\sigma^2, \tag{8.14}$$

whereas the second last equality follows from the fact that (a) X_0 has zero variance (i.e., it is deterministic) and (b) the variance of the sum of independent RVs is just the sum of the variances of each individual RV.

Definition 8.2. *The mean increment μ is called the drift of a random walk.*

Definition 8.3. *A random walk with $\mu = 0$ (p=q=1/2) is called symmetric.*

8.1.4 Restricted Random Walks

In many practical scenarios, it is useful to define RWs on a bounded state space. For instance, this is the case if one models the random motion of a molecule in a volume of fixed size or gambler with finite capital. Mathematically, this can be accounted for by introducing appropriate boundary conditions in the transition kernel of the underlying Markov chain. For instance, if a process cannot go negative, we should set $P(X_{n+1} = -1 \mid X_n = 0) = 0$. More generally, we can adjust the transition probabilities to prevent a RW to leave a certain region. Such RWs are called *restricted* RWs. An important distinction of restricted RWs can be made based on how they behave when they reach the boundary of their state space. For instance, a gambler that reaches zero capital will no longer be able to participate in the game, which would mean that the RW stays at zero for ever. In this case, we would have $P(X_{n+1} = 1 \mid X_n = 0) = 0$ and $P(X_{n+1} = 0 \mid X_n = 0) = 1$. We refer to such a process as a restricted RW with absorbing boundary. In other cases, the RW may "bounce back" as soon as it hits its boundary, which would for instance be the case if we have $P(X_{n+1} = 1 \mid X_n = 0) = 1$ and $P(X_{n+1} = 0 \mid X_n = 0) = 0$. Such RWs are called restricted RWs with reflecting boundary.

8.1.5 Relation to the Wiener process (continuous limit)

Let us consider an unrestricted 1D random walk. If we observe this process on very long time scales, it will look almost like a continuous-time random walk. In particular, since $Var\{X_n\}$ scales linearly with n, we will see increasingly large deviations from the starting point as time increases, such that the relative change between X_n and X_{n+1} will appear almost continuous. Indeed, it can be shown mathematically, that a discrete-time RW converges to a continuous-time RW as $n \to \infty$. An intuitive justification of this result can be given by the central limit theorem (CLT). Let us now consider a discrete-time RW X_n within some fixed time window. We assume that each time step of the RW corresponds to an elapsed time r such that the total time that elapses after n iterations is $t = nr$. This allows us to reparameterize the RW in terms of t such that

$$Y(t) = X_{t/r} = X_0 + \sum_{i=1}^{t/r} \Delta X_i. \tag{8.15}$$

We see that the random process $Y(t)$ is given by the initial condition X_0 and a sum of i.i.d. RVs. The number of summands inside this sum will increase linearly with t. We know from the CLT that the sum of many i.i.d. RVs will converge to a normally distributed RV. More precisely, we have that

$$Y(t) = X_0 + \sum_{i=1}^{t/r} \Delta X_i \xrightarrow{t>>r} N\left(X_0 + \frac{t}{r}\mu, \frac{t}{r}\sigma^2\right), \tag{8.16}$$

whereas $t >> r$ is equivalent to letting n be very large. If we now interpret t as a continuous variable, we can approximate the RW as

$$Y(t) \approx \frac{\mu}{r}t + \frac{\sigma}{\sqrt{r}}W(t), \tag{8.17}$$

with $W(t)$ as a standard *Wiener process*. This is because the standard Wiener process is normally distributed for all t with mean $\mathbb{E}\{W(t)\} = 0$ and variance $Var\{W(t)\} = t$. We will have a more detailed discussion about Wiener processes in Chapter **??**.

8.1.6 Random Walks in higher dimensions

Random walks represent a very general class of stochastic processes and can be extended to many different scenarios. For instance, it is straightforward to extend RWs to a two-dimensional lattice, in which case a random walker can move from its current state to its four neighbouring states. Correspondingly, the transition kernel consists of four transition probabilities. Similarly, we would obtain six transition probabilities for the three-dimensional case and so forth. However, RWs can exhibit very different properties depending on their dimension. For instance, 1-and 2D RWs have the property that no matter where the walker starts, it will at some point in the future return to its starting point (almost surely, i.e., with probability one). Surprisingly, this is not the case for RWs of dimension three or higher.

Chapter 9

Stochastic Calculus

In the previous chapters, we have focused predominantly on discrete-time stochastic processes such as Markov chains or random walks. In this chapter we focus on continuous-time stochastic processes, which play an important role in modeling the stochastic dynamics of physical an natural phenomena.
As a motivation, let us for the moment consider a deterministic dynamical system described by an ordinary differential equation

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = f(x(t), t) \quad x(0) = x_0, \tag{9.1}$$

with $x(t)$ as the state of the system at time t and f as some vector field. In this system, the time evolution of $x(t)$ is uniquely determined by the initial value x_0, such that $x(t)$ moves along a fixed trajectory as time increases. Many real-world systems, however, evolve stochastically: if we observe the same system several times using the same initial condition, then the trajectory $x(t)$ may vary from experiment to experiment. For instance, if we analyze a chemical reaction in a test tube, then the exact number of reaction products at some fixed time t will vary over repeated experiments due to thermal fluctuations.
To account for stochasticity in the time-evolution of dynamical systems we can introduce a stochastic driving term in the differential equation, i.e.,

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = f(x(t), t) + u(t), \tag{9.2}$$

whereas $u(t)$ is a "white noise" signal. This means that $u(t)$ is statistically independent of $u(s)$ for any $s \neq t$. Now, the time evolution of $x(t)$ will be stochastic, which means that if we repeatedly observe this system, the trajectory $x(t)$ will be different each time. Using the substitution $u(t)\mathrm{d}t = \mathrm{d}W(t)$, we can rewrite (9.2) as

$$\mathrm{d}x(t) = f(x(t), t)\mathrm{d}t + \mathrm{d}W(t). \tag{9.3}$$

with $\mathrm{d}W(t)$ as the differential version of a standard Wiener process as briefly introduced in Chapter 8. We remark here that while equation (9.3) is mathematically sound, the original version (9.2) comes with certain trouble. In particular, the white noise process $u(t)$ would correspond to the time derivative of $W(t)$. It is known, however, that a Wiener process is not continuously differentiable making (9.2) problematic. Therefore, continuous-time stochastic processes are generally given in the form of (9.3).

9.1 Stochastic differential equations

Definition 9.1. *A stochastic differential equation (SDE) is defined as*

$$\mathrm{d}X(t) = \mu(X(t), t)\mathrm{d}t + \sigma(X(t), t)\mathrm{d}W(t), \tag{9.4}$$

with $\mathrm{d}W(t)$ *as the infinitesimal increment of a standard Wiener process. The terms* μ *and* σ *are commonly referred to as drift and diffusion terms, respectively.*

Definition 9.2. *The standard Wiener process is characterized by the following properties:*

1. *The Wiener process has independent increments:* $W(t+\tau) - W(t)$ *is independent of* $W(s)$ $\forall s \leq t$*.*
2. *The Wiener process is stationary: the increments* $W(t+\tau) - W(t)$ *do not depend on* t*.*
3. *The Wiener process has Gaussian increments:* $W_{t+\tau} - W_t \sim \mathcal{N}(0, \tau)$*.*

From the last definition, it follows that $\mathbb{E}\{W(t+\tau) - W(t)\} = 0$ *and* $Var\{W(t+\tau) - W(t)\} = \tau$*.*

9.1.1 Ito integrals

SDE's can also be written in integral form

$$X(t) = X_0 + \int_0^t \mu(X(s), s)\mathrm{d}s + \int_0^t \sigma(X(s), s)\mathrm{d}W(s), \tag{9.5}$$

whereas the first integral corresponds the a classical Riemann integral. The second integral is called a *stochastic integral*, where in this case, the function $\sigma(X(t), t)$ is integrated with respect to a standard Wiener process $W(t)$. Mathematically, this integral can be defined as

$$\int_0^t H(s)\mathrm{d}W(s) = \lim_{n\to\infty} \sum_{i=0}^{n} H(t_i)(W(t_{i+1}) - W(t_i)) \quad t_i = t\frac{i}{n}. \tag{9.6}$$

Eq. (9.6) is generally known as the Ito integral. This integral converges (in probability) if:

- $H(t)$ depends only on $\{W(t-h) \mid h \leq 0\}$. $H(t)$ is then said to be non-anticipating.

- It holds that $\mathbb{E}\{\int_0^t H(s)^2 \mathrm{d}s\} < \infty$.

The mean and variance of the Ito integral are given by

$$\mathbb{E}\left[\int_0^t H(s)\mathrm{d}W(s)\right] = 0 \iff \mathbb{E}\{H(t)\mathrm{d}W(t)\} = 0 \tag{9.7}$$

$$Var\left[\int_0^t H(s)\mathrm{d}W(s)\right] = \int_0^t \mathbb{E}\{H(s)^2\}\mathrm{d}s. \tag{9.8}$$

9.1.2 Transformation of Wiener processes

A fundamental result in stochastic calculus is *Ito's lemma*, which can be understood as a generalization of the chain rule for stochastic processes. Assume that we are given an SDE of the form

$$\mathrm{d}X(t) = \mu(X(t),t)\mathrm{d}t + \sigma(X(t),t)\mathrm{d}W(t), \tag{9.9}$$

with $W(t)$ as a standard Wiener process. Furthermore, consider a non-linear transformation $f : \mathbb{R} \to \mathbb{R}$.

Theorem 9.1. *The transformed process $Y(t) = f(X(t))$ satisfies the SDE*

$$\begin{aligned}\mathrm{d}Y(t) = &\left[\frac{\partial}{\partial x} f(X(t))\mu(X(t),t) + \frac{1}{2}\frac{\partial}{\partial x^2} f(X(t))\sigma^2(X(t),t)\right]\mathrm{d}t \\ &+ \frac{\partial}{\partial x} f(X(t))\sigma(X(t),t)\mathrm{d}W(t).\end{aligned} \tag{9.10}$$

This is known as Ito's lemma.

Remark: Note that (9.10) is valid only if f does not explicitly depend on t. While Ito's lemma can be extended also to time-dependent f, we restrict ourselves to the case where f depends only on $X(t)$ in this lecture.

9.1.3 Mean and Variance of SDE's

While solutions of SDEs are inherently stochastic, their temporal dynamics can be characterized in terms of statistical moments. For instance, we can calculate the mean and variance of $X(t)$ in order to study the "average" solution of an SDE and how much it varies between different realizations. In order to calculate the expectation, we can apply the expectation operator on both sides the SDE, i.e.,

$$\mathbb{E}\{\mathrm{d}X(t)\} = \mathbb{E}\{\mu(X(t),t)\}\mathrm{d}t + \mathbb{E}\{\sigma(X(t),t)\mathrm{d}W(t)\}. \tag{9.11}$$

Due to the properties of the standard Wiener process we know that the second term will be zero such that we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\{X(t)\} = \mathbb{E}\{\mu(X(t),t)\}. \tag{9.12}$$

Calculating the variance, is slightly more complicated since we need a dynamic equation that describes how $Var\{X(t)\}$ evolves with time. In order to get such an equation, we can make use of Ito's lemma to first derive an SDE for $Y(t) = f(X(t)) = X(t)^2$. Taking the expectation of the SDE then gives an equation for the (non-central) second order moment $\mathbb{E}\{X(t)^2\}$, which we can then use to derive an equation for the variance according to

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}Var\{X(t)\} &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\mathbb{E}\{X(t)^2\} - \mathbb{E}\{X(t)\}^2\right) \\ &= \frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\{X(t)^2\} - 2\mathbb{E}\{X(t)\}\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\{X(t)\}.\end{aligned} \tag{9.13}$$

We will illustrate this approach using the following simple example.

Example 9.1 (Mean and Variance of the Ornstein-Uhlenbeck (OU) process)**.** *The Ornstein-Uhlenbeck process is defined as*

$$\mathrm{d}X(t) = \theta(\mu - X(t))\mathrm{d}t + \sigma\mathrm{d}W(t), \tag{9.14}$$

with $\theta > 0$, $\sigma > 0$ and μ as real parameters.
To calculate an equation for the mean of $X(t)$, we take the expectation of (9.14)

$$\mathrm{d}\mathbb{E}\{X(t)\} = \theta(\mu - \mathbb{E}\{X(t)\})\mathrm{d}t + \underbrace{\mathbb{E}\{\sigma\mathrm{d}W(t)\}}_{0}, \tag{9.15}$$

which after rearranging yields

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\{X(t)\} = \theta(\mu - \mathbb{E}\{X(t)\}). \tag{9.16}$$

To calculate the average of the OU process at stationarity (i.e., when $t \to \infty$), we can set the left hand side of this equation to zero and solve for $\mathbb{E}\{X(t)\}$. We obtain

$$\lim_{t\to\infty}\mathbb{E}\{X(t)\} = \mu. \tag{9.17}$$

To calculate the variance, we first use Ito's lemma to derive an SDE for $Y(t) = f(X(t)) = X(t)^2$. We first calculate the first and second order derivatives f, i.e.,

$$\frac{\partial}{\partial x}f(x) = 2x \tag{9.18}$$

$$\frac{\partial}{\partial x^2}f(x) = 2. \tag{9.19}$$

Using these derivates within Ito's lemma gives us

$$\begin{aligned}\mathrm{d}Y(t) = \mathrm{d}[X(t)^2] &= \left(2X(t)\theta(\mu - X(t)) + \sigma^2\right)\mathrm{d}t + 2\sigma X(t)\mathrm{d}W(t) \\ &= \left(2\theta(\mu X(t) - Y(t)) + \sigma^2\right)\mathrm{d}t + 2\sigma X(t)\mathrm{d}W(t).\end{aligned} \tag{9.20}$$

Taking the expectation on both sides yields

$$\mathrm{d}\mathbb{E}\{Y(t)\} = \left(2\theta(\mu\mathbb{E}\{X(t)\} - \mathbb{E}\{Y(t)\}) + \sigma^2\right)\mathrm{d}t + 2\sigma\mathbb{E}\{X(t)\mathrm{d}W(t)\}. \tag{9.21}$$

The second order non-central moment $\mathbb{E}\{Y(t)\} = \mathbb{E}\{X(t)^2\}$ *therefore satisfies the differential equation*

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\{X(t)^2\} = 2\theta(\mu\mathbb{E}\{X(t)\} - \mathbb{E}\{X(t)^2\}) + \sigma^2. \tag{9.22}$$

For the variance, we obtain correspondingly,

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}Var\{X(t)\} &= \frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\{X(t)^2\} - 2\mathbb{E}\{X(t)\}\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\{X(t)\}\\ &= 2\theta(\mu\mathbb{E}\{X(t)\} - \mathbb{E}\{X(t)^2\}) + \sigma^2 - 2\mathbb{E}\{X(t)\}\theta(\mu - \mathbb{E}\{X(t)\})\\ &= -2\theta\left(\mathbb{E}\{X(t)^2\} - \mathbb{E}\{X(t)\}^2\right) + \sigma^2\\ &= -2\theta Var\{X(t)\} + \sigma^2.\end{aligned} \tag{9.23}$$

The long-term variance therefore becomes,

$$\lim_{t\to\infty} Var\{X(t)\} = \frac{\sigma^2}{2\theta}. \tag{9.24}$$

We finally remark that the same approach can in principle be used to calculate mean and variance of any SDE driven by a Wiener process. However, one should keep in mind that if the SDE is non-linear, one may encounter a so-called moment-closure problem. That means, that the equation for the mean of $X(t)$ may depend on the second order moment, which in term depends on the third-order moment and so forth. In this case, certain approximate techniques can be considered. Those techniques, however, are beyond the scope of this lecture.

Chapter 10

Numerical Methods for Stochastic Differential Equations

Stochastic differential equations (SDE), as introduced in the previous chapter, form the basis of modeling continuous-time stochastic processes. While in some cases, the solution (or its moments) of a SDE can be computed analytically (see, e.g., the Ornstein-Uhlenbeck process in the previous chapter), most SDEs need to be simulated or solved numerically. In this chapter, we present the classic numerical methods for solving or simulating SDEs.

10.1 Refresher on SDEs

Before we start, we briefly refresh the main concepts of SDEs as they pertain to numerical methods, and introduce the notation. A scalar SDE governs the dynamics of a continuous random variable $X(t) \in \mathbb{R}$ by:

$$\frac{\mathrm{d}X}{\mathrm{d}t} = v_0(X(t),t) + v_1(X(t),t)\frac{\mathrm{d}W_1}{\mathrm{d}t} + \ldots + v_n(X(t),t)\frac{\mathrm{d}W_n}{\mathrm{d}t} \tag{10.1}$$

with given functions $v_0, v_1, \ldots, v_n$ and Wiener processes $W_1, \ldots, W_n$. The first term on the right-hand side governs the deterministic part of the dynamics through the function v_0. The remaining terms govern the stochastic influences on the dynamics, of which there could be more than one, each with its own Itô transformation $v_1, \ldots, v_n$. The Wiener processes $W_i(t)$ are continuous functions of time that are almost certainly not differentiable anywhere. Therefore, the $\frac{\mathrm{d}W_i}{\mathrm{d}t}$ are pure white noise and the equation cannot be interpreted mathematically. However, if we multiply the entire equation by $\mathrm{d}t$, we get:

$$\mathrm{d}X(t) = v_0(X(t),t)\mathrm{d}t + v_1(X(t),t)\mathrm{d}W_1(t) + \ldots + v_n(X(t),t)\mathrm{d}W_n(t), \tag{10.2}$$

where everything is still time-continuous and the solution therefore is a continuous function in time. This is the usual way of writing SDEs. In the following, for simplicity, we only consider SDEs where the stochastic terms can all be collected into one and write:

$$\mathrm{d}X(t) = \mu(X(t), t)\mathrm{d}t + \sigma(X(t), t)\mathrm{d}W(t). \tag{10.3}$$

The deterministic part μ is called *drift* and the stochastic part σ is called *diffusion*, because Wiener increments $\mathrm{d}W$ are normally distributed (see previous chapter).

Example 10.1. *Consider as an example the classic Langevin equation:*

$$\mathrm{d}X(t) = -aX(t)\mathrm{d}t + \sqrt{b}\,\mathrm{d}W(t),$$

where $a > 0$ and $b > 0$ are constants. This equation describes the dynamics of the velocity $X(t)$ of a particle (point mass) under deterministic friction (friction coefficient a) and stochastic Brownian motion (diffusion constant b). It is a central equation in statistical physics, chemistry, finance, and many other fields.

10.2 Solving an SDE

It is not immediately clear what "solving an SDE" means. Since $X(t)$ is a continuous-time stochastic process, every realization of it is different, so we cannot find *the* solution $x(t)$. Consider $t \in [0, T]$. Each realization of the Wiener process produces a different $x_T = X(t = T)$ when always starting from the same initial $X(t = 0) = x_0$. As a "solution", we may want to know:

1. The probability density function (PDF) of $x_T = X(t = T)$,
2. $\mathbb{E}[X(t = T)]$ or any $\mathbb{E}[g(X(t = T))]$.

(1) is referred to as the *strong solution* of the SDE, and (2) as the *weak solution* of the SDE.

10.2.1 Solution methods

10.2.1.1 Weak solution: Feynman-Kac formulae

Weak solutions can be computed analytically or numerically by reducing the stochastic problem to a deterministic one. Using Feynman-Kac formulae, every $\mathbb{E}[g(X(t))]$ of every SDE has an associated deterministic parabolic PDE with solution:

$$u(t, x) = \mathbb{E}[g(X(t)) \,|\, X(0) = x].$$

This deterministic parabolic PDE can then be solved analytically or using methods from numerical analysis (e.g., finite differences). Many quantities of the original SDE are thus accessible, e.g.:

- g a monomial $\rightarrow$ corresponding moment of $X(t)$,
- $g = \delta$ (i.e, the Dirac delta) $\rightarrow$ transition PDF of $X(t)$,
- $g = \mathcal{H}$ (i.e., the Heaviside step function) $\rightarrow$ transition CDF of $X(t)$,
- $g = \exp(\cdot)$ $\rightarrow$ the Laplace transform of the solution.

10.2.1.2 Strong solution: Analytical solution

If the strong solution is required, the SDE can sometimes be solved analytically. A famous example is the Black-Scholes equation that models stock option prices:

$$\mathrm{d}X = X\mu \mathrm{d}t + X\sigma \mathrm{d}W(t)$$

with constants μ and σ.

10.2.1.3 Strong and weak solution: Numerical integration

The above two methods are exact. If a numerical approximation to the solutions is sufficient, stochastic numerical integration of the SDE can be used to approximate both weak and strong solutions. This is analogous to MCMC simulation in the discrete-time case, except that now there are infinitely many infinitesimal stochastic transitions, so extra care is required.

10.3 Stochastic Numerical Integration: Euler-Maruyama

The solution of the SDE in Eq. 10.3 can be written in integral form as:

$$X(t) = x_0 + \int_0^t \mu(X(\tilde{t}), \tilde{t})\,\mathrm{d}\tilde{t} + \int_0^t \sigma(X(\tilde{t}), \tilde{t})\,\mathrm{d}W(\tilde{t}), \tag{10.4}$$

where the first integral is a deterministic Riemann integral, and the second one is a stochastic Itô (or Stratonovich) integral.
In order to numerically approximate the solution, we discretize the time interval $[0, T]$ in which we are interested in the solution into N finite-sized time steps of duration $\delta t = \frac{T}{N}$ such that $t_n = n\delta t$ and $X_n = X(t = t_n)$, $W_n = W(t = t_n)$. Due to the 4^{th} property of the Wiener process from the previous chapter, which states that the differences between any two time points of a Wiener process are normally distributed, we can also discretize:

$$W_{n+1} = W_n + \Delta W_n \tag{10.5}$$

with ΔW_n *i.i.d.* $\sim \mathcal{N}(0, \delta t)$ and $W_0 = 0$. The starting value for the Wiener process, $W_0 = 0$ is chosen arbitrarily, since the absolute value of W is inconsequential for the SDE; only the increments dW matter, so we can start from an arbitrary point.

The integrals in Eq. 10.4 can be interpreted as the continuous limits of sums. The deterministic term can hence be discretized by a standard quadrature (numerical integration). The stochastic term is discretized using the above discretization of the Wiener process, hence, for any time T,

$$\int_0^T \sigma(X(\tilde{t}),\tilde{t})\,\mathrm{d}W(\tilde{t}) \approx \sum_{n=0}^{N} \sigma(X(t_n),t_n)\Delta W_n.$$

Using the rectangular rule (i.e., approximating the integral by the sum of the areas of rectangular bars) for the deterministic integral, and the above sum over one time step for the stochastic integral, we find:

$$\int_{t_n}^{t_{n+1}} \mu(X(\tilde{t}),\tilde{t})\,\mathrm{d}\tilde{t} \approx \mu(X_n,t_n)\delta t\,,$$
$$\int_{t_n}^{t_{n+1}} \sigma(X(\tilde{t}),\tilde{t})\,\mathrm{d}W(\tilde{t}) \approx \sigma(X_n,t_n)\Delta W_n\,.$$

This yields the classic Euler-Maruyama method:

$$\boxed{X_{n+1} = X_n + \mu(X_n,t_n)\delta t + \sigma(X_n,t_n)\Delta W_n} \tag{10.6}$$

with

$$\begin{aligned}\Delta W_n &= W_{n+1} - W_n \sim \mathcal{N}(0,\delta t)\ \textit{i.i.d.},\\ W_0 &= 0,\\ X_0 &= x_0.\end{aligned}$$

Iterating Eq. 10.6 forward in time $n = 0, 1, \ldots, N$ yields a numerical approximation of *one* trajectory/realization of the stochastic process $X(t)$ governed by the SDE from Eq. 10.3.

10.4 Convergence

The error of the numerical solution is defined with respect to the exact stochastic process $X(t)$ for decreasing δt. To make this comparison possible, we introduce $X_{\delta t}(t)$, the continuous-time stochastic process obtained by connecting the points (t_n, X_n) by straight lines, i.e.:

$$X_{\delta t}(t) = X_n + \frac{t-t_n}{t_{n+1}-t_n}(X_{n+1}-X_n) \quad \text{for } t \in [t_n, t_{n+1}). \tag{10.7}$$

Comparing this continuous-time process to the exact process, we can define convergence. Note that in the deterministic case this is not possible as there we can simply evaluate the analytical solution at the simulation time steps and compare the values. In a stochastic simulation, however, this is not possible as each realization of the process has different values. The only thing we can compare are moments, which can only be computed over continuous processes. So with the above trick we can define:

Definition 10.1 (strong and weak convergence)**.** *A numerical method is* strongly convergent *if and only if*

$$\lim_{\delta t \to 0} \mathbb{E}\Big[|X(T) - X_{\delta t}(T)|\Big] = 0 \tag{10.8}$$

and weakly convergent *if*

$$\lim_{\delta t \to 0} \Big|\mathbb{E}[g(X(T))] - \mathbb{E}[g(X_{\delta t}(T))]\Big| = 0 \tag{10.9}$$

for every polynomial g and every time point T.

Strong convergence is also sometimes called *convergence in value*, and it implies convergence to the strong solution of the SDE. Weak convergence is also called *convergence in distribution* and it implies convergence to the weak solution of the SDE.
For the Euler-Maruyama algorithm, the following result is known:

Theorem 10.1. *The Euler-Maruyama method is both strongly and weakly convergent if $\mu(\cdot,\cdot)$ and $\sigma(\cdot,\cdot)$ are four times continuously differentiable and have bounded first derivatives. This condition is sufficient, but not necessary.*

Strong convergence implies weak convergence, but not the other way around. Strong convergence intuitively means that for a given simulation with $\delta t \to 0$, the trajectory exactly matches *one of the* trajectories of the analytical process. This is because the above has to hold for all T. So any simulation converges to something that is *a* valid trajectory of the analytical process. Weak convergence does not require this. If only required the moments to match. For example, if the true trajectory goes to value 0.5, a strong simulation would also have to go to 0.5. A weak simulation, however, can go to 1.0 half of the time and to 0.0 half of the time. The average is still 0.5, albeit all of the simulated trajectories are arbitrarily far away from any true trajectory.
The next question then is, how *fast* the algorithm converges. For this, we define the order of convergence in both the weak and strong sense, as:

Definition 10.2 (strong convergence order)**.** *A numerical method has* strong convergence order $\gamma \geq 0$ *if and only if*

$$\mathbb{E}\Big[|X(T) - X_{\delta t}(T)|\Big] \leq C(T)\delta t^{\gamma} \tag{10.10}$$

for every time T, where the constant $C(T) > 0$ depends on T and on the SDE considered.

Definition 10.3 (weak convergence order)**.** *A numerical method has* weak convergence order $\gamma \geq 0$ *if and only if*

$$\left|\mathbb{E}\Big[g(X(T))\Big] - \mathbb{E}\Big[g(X_{\delta t}(T))\Big]\right| \leq C(T,g)\delta t^{\gamma} \tag{10.11}$$

for every time T, where the constant $C(T,g) > 0$ depends on T, g, and the SDE considered.

For the Euler-Maruyama algorithm, the following result is known:

Theorem 10.2. *The Euler-Maruyama method has weak convergence order* 1 *and strong convergence order* $\frac{1}{2}$.

The proof of this theorem involves stochastic Taylor expansions (into Stratonovich integrals) and is omitted here.
In principle, the Euler-Maruyama method allows us to compute both the strong and weak solution of any SDE to any precision, if we just choose the time step δt small enough. The strong convergence order of 1/2 is rather slow, though. It means that in order to get a solution that is 10 times more accurate, we need 100 times smaller time steps. In practice, however, δt cannot be chosen too small because of finite-precision arithmetic rounding errors and computer time. Therefore, there is an ongoing search for stochastic integration methods with higher orders of convergence.

10.5 Milstein Method

The standard way to increase the convergence order of a numerical method is to take into account additional higher-order terms in the Taylor expansion of the solution. Keeping terms up to and including order 2 in the stochastic Taylor expansion of $X(t + \delta t)$, or recursively applying Itô's Lemma to the coefficient functions of the autonomous SDE

$$\mathrm{d}X(t) = \mu(X(t))\mathrm{d}t + \sigma(X(t))\mathrm{d}W(t)$$

yields the Milstein method:

$$\boxed{X_{n+1} = X_n + \mu(X_n)\delta t + \sigma(X_n)\Delta W_n + \frac{1}{2}\sigma'(X_n)\sigma(X_n)((\Delta W_n)^2 - \delta t)} \tag{10.12}$$

with:

$$\begin{aligned} \Delta W_n &= W_{n+1} - W_n \sim \mathcal{N}(0, \delta t) \ \textit{i.i.d.}, \\ W_0 &= 0, \\ X_0 &= x_0, \\ \sigma'(x) &= \frac{\mathrm{d}\sigma(x)}{\mathrm{d}x}. \end{aligned}$$

This formulation of the Milstein method only works for autonomous SDEs, i.e., SDEs where the coefficient functions $\mu(\cdot)$ and $\sigma(\cdot)$ depend on $X(t)$, but not explicitly on t. For non-autonomous SDEs, where the coefficients are also explicit functions of time, the formulation of the Milstein method involves Stratonovich integrals that need to be numerically approximated by Lévy area calculation if they cannot be analytically solved. Here, we only consider the autonomous case, where these complications do not appear.
Regarding the convergence order of the Milstein method, we have:

Theorem 10.3. *The Milstein method has both strong and weak orders of convergence of 1.*

The Milstein method therefore is more accurate than the Euler-Maruyama method in the strong sense, but has the same weak order of convergence. Using a 100-fold smaller time step reduces the numerical error 100 times when using Milstein. This allows larger time steps compared to Euler-Maruyama and relaxes the numerical rounding issues.
Of course, Euler-Maruyama and Milstein are not the only known stochastic numerical integration methods. Other methods also exist (e.g. Castell-Gaines, stochastic Lie integrators, etc.), some with lower pre-factors C in the error bounds of Eqs. 10.10 and 10.11. However, no method is known with strong convergence order > 1, unless we can analytically solve the corresponding Stratonovich integrals.
The numerical stability properties (with δt) of stochastic numerical integration methods are unclear. Only few results are known on almost-sure stability, e.g., for linear scalar SDEs. No A-stable stochastic numerical integrator is known.

10.6 Weak Simulation

While no method with strong order of convergence larger than 1 is known, the weak order of convergence can be increased at will if strong convergence is not required. Therefore, if one is only interested in the weak solution of an SDE, simpler methods exist to approximate $\mathbb{E}[g(X(T))]$ without needing to simulate the entire path of the process.
It turns out that one can, e.g., simply choose binomial increments $\Delta\widetilde{W} = \pm\sqrt{\delta t}$ with $P(\Delta\widetilde{W} = \pm\sqrt{\delta t}) = \frac{1}{2}$. This does not require simulating Gaussian random variates. Instead, at each time point one simply chooses $\underset{\smile}{i.i.d.}$ increments of $+\sqrt{\delta t}$ or $-\sqrt{\delta t}$, each with probability $\frac{1}{2}$. Then, one uses $\Delta\widetilde{W}$ in the stochastic integration method instead of ΔW. Depending on the numerical integration (quadrature) scheme used to approximate Eq. 10.4, any weak order of convergence can be achieved. E.g.:

- Rectangular rule $\longrightarrow$ Euler-Maruyama $\longrightarrow$ weak 1-st order
- Trapezoidal rule $\longrightarrow$ weak 2-nd order

  $\vdots$

If the quadrature approximates the integrand by a piecewise polynomial of degree p, the weak order of convergence of a simulation using binomial increments $\Delta\widetilde{W}$ is $p+1$. There is no strong convergence in any of these cases (i.e., strong order of convergence is 0).

Chapter 11

Stochastic Reaction Networks

Reaction networks, as introduced in the previous chapter, are an important class of stochastic models that can be used to describe populations of individuals that randomly interact with one another over time. Applications range from chemical kinetics over computer networks to logistics and traffic. In this chapter, we will introduce the formalism of reaction networks and discuss how they can be described mathematically and simulated using numerical methods.

11.1 Formal Representations

11.1.1 Representation of a reaction

A reaction *reactants* $\rightarrow$ *products* is formally represented by:

$$\sum_{i=1}^{N} \nu_i^- S_i \xrightarrow{k} \sum_{i=1}^{N} \nu_i^+ S_i \,, \tag{11.1}$$

where:

S_i: species i,

N: total number of different species in the reaction,

ν_i^-: reactant stoichiometry,

ν_i^+: product stoichiometry,

k: reaction rate.

The *total stoichiometry* is $\nu_i = \nu_i^+ - \nu_i^-$, and it gives the net change in copy numbers when the reaction happens. Reactions are classified by the total number of reactant molecules $\sum_i \nu_i^-$, which is called the *order* of the reaction. Reactions having only a single reactant are of order one, or *unimolecular*. Reactions with two reactants are of order two, or *bimolecular*; and so on. Reactions of order ≤ 2 are called *elementary reactions*.

Example 11.1. *For the reaction $A + B \to C$, the above quantities are:*

- $S_i = \{A, B, C\}$
- $N = 3$
- $\vec{\nu}^- = [1, 1, 0]^\mathsf{T}$
- $\vec{\nu}^+ = [0, 0, 1]^\mathsf{T}$
- $\vec{\nu} = [-1, -1, 1]^\mathsf{T}$

11.1.2 Representation of a reaction network

A reaction network comprising M reactions between N distinct species is then represented by the indexing Eq. 11.1 with the reaction index and writing:

$$\sum_{i=1}^{N} \nu_{i,\mu}^- S_i \xrightarrow{k_\mu} \sum_{i=1}^{N} \nu_{i,\mu}^+ S_i \,, \qquad \mu = 1, \ldots, M \tag{11.2}$$

where:

μ: index of reaction R_μ,

M: total number of different reactions.

Now the stoichiometry is a matrix with one column per reaction: $\nu = \nu^+ - \nu^-$. All the stoichiometry matrices are of size $N \times M$. All elements of ν^+ and ν^- are non-negative whereas those of ν can be negative, zero or positive.

Example 11.2. *Consider the following cyclic chain reaction network with N species and $M = N$ reactions:*

$$\begin{array}{rcll} \mathrm{S}_i & \to & \mathrm{S}_{i+1}, & \quad i = 1, \ldots, N-1 \\ \mathrm{S}_N & \to & \mathrm{S}_1. & \end{array} \tag{11.3}$$

For $N = 3$ the reaction network is

$$\begin{array}{rccc} \text{Reaction 1}: & \mathrm{S}_1 & \to & \mathrm{S}_2 \\ \text{Reaction 2}: & \mathrm{S}_2 & \to & \mathrm{S}_3 \\ \text{Reaction 3}: & \mathrm{S}_3 & \to & \mathrm{S}_1. \end{array} \tag{11.4}$$

The stoichiometry matrices for this reaction network are:

$$\vec{\nu}^- = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{11.5}$$

$$\vec{\nu}^{+} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \tag{11.6}$$

and

$$\vec{\nu} = \vec{\nu}^{+} - \vec{\nu}^{-} = \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix}. \tag{11.7}$$

Example 11.3. *Consider the following colloidal aggregation reaction network with N species and $M = \left\lfloor \frac{N^2}{4} \right\rfloor$ reactions:*

$$\mathrm{S}_i + \mathrm{S}_j \quad \rightarrow \quad \mathrm{S}_{i+j}, \qquad i = 1, \ldots, \left\lfloor \tfrac{N}{2} \right\rfloor; \quad j = i, \ldots, N - i. \tag{11.8}$$

Species S_i can be considered a multimer consisting of i monomers.
For $N = 4$ the reaction network is

$$\begin{array}{lccl} \text{Reaction 1}: & \mathrm{S}_1 + \mathrm{S}_1 & \rightarrow & \mathrm{S}_2 \\ \text{Reaction 2}: & \mathrm{S}_1 + \mathrm{S}_2 & \rightarrow & \mathrm{S}_3 \\ \text{Reaction 3}: & \mathrm{S}_1 + \mathrm{S}_3 & \rightarrow & \mathrm{S}_4 \\ \text{Reaction 4}: & \mathrm{S}_2 + \mathrm{S}_2 & \rightarrow & \mathrm{S}_4. \end{array} \tag{11.9}$$

The stoichiometry matrices for this reaction network are:

$$\vec{\nu}^{-} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{11.10}$$

$$\vec{\nu}^{+} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \tag{11.11}$$

and

$$\vec{\nu} = \vec{\nu}^{+} - \vec{\nu}^{-} = \begin{bmatrix} -2 & -1 & -1 & 0 \\ 1 & -1 & 0 & -2 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}. \tag{11.12}$$

11.2 The Chemical Master Equation

We are now concerned with the temporal dynamics of stochastic reaction networks. In particular, we seek for a stochastic process that captures the random firings of reactions over time. In the previous chapters we have discussed discrete Markov chains (e.g., random walks) and continuous-time Markov processes (e.g., Wiener processes). Reaction networks do not fall in either of these classes since they evolve continuously in time, but have a discrete state space (i.e., particle numbers are integer-valued). In order to describe such systems, we employ another class of Markov processes termed *continuous-time Markov chains* (CTMCs).

We consider the state $\vec{X}(t) = (X_1(t), \ldots, X_N(t))$ of a reaction network collecting the copy numbers (population) of each species at time t. At a certain time instance, we can now assess the time evolution of $\vec{X}(t)$ with a small amount of time Δt using basic probability theory. In particular, we define the following two probabilities

$$P(\vec{X}(t+\Delta t) = \vec{x} + \nu_i \mid \vec{X}(t) = \vec{x}) = a_i(\vec{x})\Delta t + o(\Delta t) \tag{11.13}$$

$$P(\vec{X}(t+\Delta t) = \vec{x} \mid \vec{X}(t) = \vec{x}) = 1 - \sum_i a_i(\vec{x})\Delta t + o(\Delta t), \tag{11.14}$$

where $a_i(\vec{x})$ is commonly referred to as a rate function, hazard function or propensity function. Intuitively, the propensity function $a_i(x)$ tells us how likely a particular reaction i happens within an infinitely small amount of time and therefore sets the time scale of this reaction. Eq. (11.13) defines the probability that the state changes by ν_i within Δt. On the one hand, this can happen if exactly one reaction of type i happens. This term is proportional to the propensity function $a_i(x)$. However, it can also happen that a sequence of consecutive reactions yield a net change of ν_i, which is captured by the second term $o(\Delta t)$. This term, however, is of order higher than Δt, such that it tends to zero much faster than $a_i(\vec{x})\Delta t$ as Δt goes to zero. We will make use of that fact later in this section. Eq. (11.14) is the probability that the net change is zero, which is just one minus the probability that any of the M reactions happens plus an additional term $o(\Delta t)$ that accounts for the possibility that two or more reactions accumulated to a net change of zero. Note that both (11.13) and (11.14) are "instantaneous", i.e., they depend only on the current state $X(t) = x$ and not any past state. This reflects the Markovianity of the reaction networks defined in such a way. We remark that while the Markov assumption can be rigorously justified in certain physical scenarios, it can be violated in others. While non-Markovian extensions of reaction networks exist, they are beyond the scope of this lecture.

A key quantity that captures the stochastic dynamics of reaction networks is the state probability distribution $P(\vec{x}, t) := P(\vec{X}(t) = \vec{x})$. This distribution tells us how likely we will find the system in a particular molecular configuration $\vec{x}$ at any time t. While $P(\vec{x}, t)$ is generally not known explicitly, it can be described by a famous differential equation, commonly known as the Chemical Master Equation (CME). Importantly, this equation is straightforward to derive using our definitions from (11.13) and (11.14). In particular, we first write down an expression for the temporal change in $P(\vec{x}, t)$, i.e.,

$$\frac{\mathrm{d}}{\mathrm{d}t} P(\vec{x}, t) = \lim_{\Delta t \to 0} \frac{P(\vec{x}, t+\Delta t) - P(\vec{x}, t)}{\Delta t} \tag{11.15}$$

Now, in order to calculate the distribution $P(\vec{x}, t + \Delta t)$, we make use of (11.13) and (11.14). In particular, the probability of being in state $\vec{x}$ at time $t + \Delta t$ is the probability that we were brought to this state via any of the N reactions plus the probability that we have already been in state $\vec{x}$ at time t (plus some additional terms accounting for the possibility that multiple events happened). In particular, we obtain

$$\begin{aligned} P(\vec{x}, t + \Delta t) = &\left(\sum_i a_i(\vec{x} - \nu_i)\Delta t + o(\Delta t) \right) P(\vec{x} - \nu_i, t) \\ &+ \left(1 - \sum_i a_i(\vec{x})\Delta t + o(\Delta t) \right) P(\vec{x}, t). \end{aligned} \tag{11.16}$$

Plugging this expression into (11.15) yields

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} P(\vec{x}, t) = \lim_{\Delta t \to 0} \Bigg[&\frac{\sum_i a_i(\vec{x} - \nu_i)\Delta t P(\vec{x} - \nu_i, t)}{\Delta t} + \underbrace{\frac{o(\Delta t) P(\vec{x} - \nu_i, t)}{\Delta t}}_{\to 0} \\ &+ \underbrace{\frac{P(\vec{x}, t) - P(\vec{x}, t)}{\Delta t}}_{\to 0} - \frac{\sum_i a_i(\vec{x})\Delta t P(\vec{x}, t)}{\Delta t} + \underbrace{\frac{o(\Delta t) P(\vec{x}, t)}{\Delta t}}_{\to 0} \Bigg]. \end{aligned}$$

We therefore obtain for the following differential equation for $P(\vec{x}, t)$

$$\frac{\mathrm{d}}{\mathrm{d}t} P(\vec{x}, t) = \sum_i a_i(\vec{x} - \nu_i) P(\vec{x} - \nu_i) - \sum_i a_i(\vec{x}) P(\vec{x}, t), \tag{11.17}$$

known as the CME. Similarly to the discrete Markov chain scenario, the CME describes how the state distribution evolves over time and is thus a continuous-time analog of the Kolmogorov-forward equation that we have discussed in the previous chapters. Technically speaking, the CME is a difference-differential equation: it has a time-derivative on the left hand side and discrete shifts in the state on the right-hand side. Note that in general, the CME is infinite-dimensional, since for every possible $\vec{x}$, we would get an additional dimension in the CME. Unfortunately, analytical solutions of the CME do not exist in all but the simplest cases (e.g., linear chain of three reactions) and needs to be solved numerically. Traditional methods from numerical analysis, such as finite differences or finite elements, also fail due to the high dimensionality of the domain of the probability distribution $P(\vec{x}, t)$, which leads to an exponential increase in computational and memory cost with network size. However, Monte Carlo approaches can be applied to simulate stochastic reaction networks, as will be discussed in the next section.

11.3 Exact Simulation of Stochastic Reaction Networks

While direct integration of the CME is challenging, a class of algorithms exists that samples solutions (trajectories of the Markov chain) from the *exact* solution $P(\vec{x},t)$ of the Master equation without ever explicitly solving the Master equation. These are known as *exact stochastic simulation algorithms* (for short: exact SSA), and they play an important role in many practical applications.
From the given stoichiometry matrices and reaction rates introduced at the beginning of this chapter, an exact stochastic simulation algorithm samples a trajectory of the system, $\vec{x}(t) \sim P(\vec{x},t)$ from the exact solution of the Master equation. Exact Stochastic Simulation Algorithms (SSA) are a special case of the larger class of kinetic Monte Carlo methods that were introduced in the 1940s by Doob. Daniel Gillespie then formulated the modern SSA family and proved that they are exact in the sense that they sample trajectories from the (unknown) *exact* solution of the Master equation. Due to the importance of this proof, SSAs are also sometimes referred to as *Gillespie algorithms.* As a side remark, it is rare that simulations are exact. Usually, they are only approximations of the true solution. The fact that SSAs are exact gives them special importance, and it is one of the few cases where simulations can be used to validate experiments, and not the other way around.
In SSA, the probability $P(\vec{x},t)$ of finding the network in state $\vec{x}$ (vector of copy numbers) at time t, whose time evolution is given by the Master equation, is replaced by the joint probability density of a single reaction event $p(\tau,\mu \mid \vec{x}(t))$, defined as

$$p(\tau,\mu \mid \vec{x}(t))\mathrm{d}\tau \quad = \quad \text{Probability that the next reaction is } \mu \text{ and it fires in } [t+\tau, t+\tau+\mathrm{d}\tau) \text{ given } \vec{x} \text{ at time } t. \tag{11.18}$$

This probability density p is derived as follows: Consider that the time interval $[t, t+\tau+\mathrm{d}\tau)$ is divided into k equal intervals of length $\frac{\tau}{k}$ plus a last interval of length $\mathrm{d}\tau$, as illustrated in Fig. 11.1.

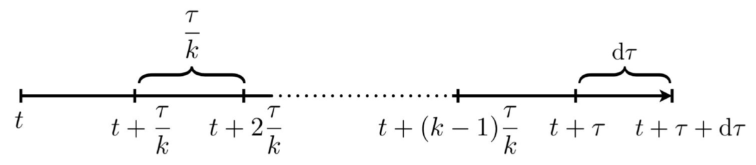

Figure 11.1: Division of the time interval $[t, t+\tau+\mathrm{d}\tau)$ into $k+1$ intervals. Here, t represents the current time. The only reaction firing is reaction μ in the $(k+1)^{\text{th}}$ infinitesimally small time interval $[t+\tau, t+\tau+\mathrm{d}\tau)$.

The definition of $p(\tau,\mu \mid \vec{x}(t))$ in Eq. 11.18 dictates that no reactions occur in all of the first k intervals, and that reaction μ fires *exactly once* in the last interval.

We recall that the Master equation has been derived from the following basic quantities:

$$\begin{aligned}
c_\mu \mathrm{d}\tau &= \text{Probability of reaction } \mu \text{ happening in the next infinitesimal time} \\
&\quad\ \text{interval } [t, t+\mathrm{d}\tau) \text{ with any randomly selected } \nu^-_{i,\mu} \text{ reactants;} \\
h_\mu(\vec{x}) &= \text{Number of distinct combinations in which the reactants of} \\
&\quad\ \text{reaction } \mu \text{ can react to form products.}
\end{aligned}$$

The product $a_\mu(\vec{x}) = h_\mu(\vec{x})c_\mu$ is called the *reaction propensity.* The probability $a_\mu \mathrm{d}\tau$ is the probability that reaction μ happens *at least once* in the time interval $\mathrm{d}\tau$. It is the product of the probability of reaction and the number of possible reactant combinations by which this can happen, as these are statistically independent events.
Now we can write:

$$\begin{aligned}
\text{Prob}\{\mu \text{ fires once in } [t+\tau, t+\tau+\mathrm{d}\tau) \text{ given } \vec{x} \text{ at } t+\tau\} &= P(\vec{x}+\vec{\nu}_\mu, t+\tau+\mathrm{d}\tau \mid \vec{x}, t+\tau) \\
&= h_\mu(\vec{x})c_\mu \mathrm{d}\tau(1-c_\mu \mathrm{d}\tau)^{h_\mu(\vec{x})-1} \\
&= c_\mu h_\mu(\vec{x})\mathrm{d}\tau + O(\mathrm{d}\tau^2) \\
&= a_\mu(\vec{x})\mathrm{d}\tau + O(\mathrm{d}\tau^2).
\end{aligned}$$

This only considers the last sub-interval of length $\mathrm{d}\tau$. The first line is simply the analytical solution of the Master equation. If reaction μ has total stoichiometry $\vec{\nu}_\mu$, then the new state of the network after reaction μ happened exactly once is $\vec{x} + \vec{\nu}_\mu$. In the second line, the first factor, $h_\mu c_\mu \mathrm{d}\tau$, is the probability that reaction μ happens from at least one of the h_μ possible reactant combinations. However, the reaction could still happen more than once. Therefore, the second factor, $(1-c_\mu \mathrm{d}\tau)^{h_\mu(\vec{x})-1}$ is the probability that none of the other $h_\mu - 1$ reactant combinations leads to a reaction. In the third line, we only multiplied out the first factor. All others are of $O(\mathrm{d}\tau^2)$ or higher. Overall, the expression thus is the probability that reaction μ happens once, and exactly once, in the last sub-interval of length $\mathrm{d}\tau$.
Further, we have for the probability that no reaction happens in *one* of the first k sub-intervals:

$$\begin{aligned}
\text{Prob}\{\text{No reaction in } [t, t+\tau/k) \text{ given } \vec{x} \text{ at } t\} &= P(\vec{x}, t+\tau/k \mid \vec{x}, t) \\
&= 1 - \sum_{\mu=1}^{M} a_\mu(\vec{x})\frac{\tau}{k} \\
&= 1 - a(\vec{x})\frac{\tau}{k},
\end{aligned}$$

where the total propensity $a(\vec{x}) = \sum_{\mu=1}^{M} a_\mu(\vec{x})$. In the second line, we sum over all reactions. Each reaction has probability $a_\mu \tau/k$ of happening at least once, so the sum over all μ is the probability that *any* reaction happens at least once. One minus this then is the probability that no reaction happens ever.

Both of the above expressions assume that the individual reaction events are statistically independent. This is an important assumption of the Master equation.

From these two expressions, we can now write an expression for Eq. 11.18 by considering *all* $k+1$ sub-intervals, again assuming that the individual sub-intervals are mutually statistically independent:

$$p(\tau, \mu \mid \vec{x}(t))\mathrm{d}\tau \quad = \quad \left[1 - a(\vec{x})\frac{\tau}{k}\right]^k \; \left[a_\mu(\vec{x})\mathrm{d}\tau + O\left(\mathrm{d}\tau^2\right)\right].$$

The term in the first square bracket is the probability that no reaction happens in any one of the k first sub-intervals. This to the power of k thus is the probability that no reaction happens in all of the k first sub-intervals. The term in the second square bracket then is the probability that reaction μ happens exactly once in the last sub-interval.

Dividing both sides of the equation by $\mathrm{d}\tau$ and taking the limit $\lim_{\mathrm{d}\tau\to 0}$, we obtain

$$p(\tau, \mu \mid \vec{x}(t)) \quad = \quad \left[1 - a(\vec{x})\frac{\tau}{k}\right]^k \; a_\mu(\vec{x}).$$

Taking the limit $\lim_{k\to\infty}$, we further get

$$p(\tau, \mu \mid \vec{x}(t)) \quad = \quad \mathrm{e}^{-a(\vec{x})\tau} a_\mu(\vec{x}), \tag{11.19}$$

because $\lim_{k\to\infty}\left(1+\frac{x}{k}\right)^k = \mathrm{e}^x$. We have thus taken the continuum limit for infinitesimally small $\mathrm{d}\tau$ and infinitely many infinitesimally small previous sub-intervals. Therefore, this is a probability density function. Sampling reactions μ and reaction waiting times τ from this density is equivalent to sampling trajectories from the exact solution of the Master equation, because they both describe the same continuous-time stochastic process. Directly doing so, however, is difficult, because τ is a continuous variable, whereas μ is a discrete variable. The density $p(\tau, \mu|\vec{x})$ therefore lives in a hybrid continuous-discrete space.

It is therefore easier to sample from the two marginals. Summing Eq. 11.19 over all reactions (i.e., summing over μ) we get the marginal probability density function of τ as

$$\begin{aligned} p(\tau \mid \vec{x}(t)) \quad &= \quad \sum_{\mu=1}^{M} a_\mu(\vec{x})\mathrm{e}^{-a(\vec{x})\tau} \\ &= \quad a(\vec{x})\mathrm{e}^{-a(\vec{x})\tau}. \end{aligned} \tag{11.20}$$

Similarly, integrating Eq. 11.19 over τ we get the marginal probability distribution function of μ as

$$\begin{aligned} p(\mu \mid \vec{x}(t)) \quad &= \quad \int_0^\infty a_\mu(\vec{x})\mathrm{e}^{-a(\vec{x})\tau}\mathrm{d}\tau \\ &= \quad \frac{a_\mu(\vec{x})}{a(\vec{x})}. \end{aligned} \tag{11.21}$$

From Eqs. 11.19, 11.20, and 11.21 we observe that

$$p(\tau, \mu \mid \vec{x}(t)) = p(\tau \mid \vec{x}(t))\, p(\mu \mid \vec{x}(t)), \tag{11.22}$$

thus inferring that μ and τ are statistically independent random variables. Sampling from the marginals in any order is therefore equivalent to sampling from the joint density. Sampling from Eq. 11.20 is easily done using the inversion method (see Section 2.3), as τ is exponentially distributed with parameter a. Eq. 11.21 describes a discrete probability distribution from which we can also sample using the inversion method. Note that Eqs. 11.20 and 11.21 also relate to a basic fact in statistical mechanics: if an event has rate a_μ of happening, then the time one needs to wait until it happens again is $\sim \mathrm{Exp}(a_\mu)$ (see Section 1.5.2).
By sampling one reaction event at a time and propagating the simulation in time according to Eq. 11.20, we obtain exact, time resolved trajectories of the population $\vec{x}$ as governed by the Master equation. The SSA, however, is a Monte Carlo scheme and hence several independent runs need to performed in order to obtain a good estimate of the probability function $P(\vec{x}, t)$, or any of its moments.
All exact formulations of SSA aim to simulate the network by sampling the random variables τ (time to the next reaction) and μ (index of the next reaction) according to Eqs. 11.20 and 11.21 and propagating the state $\vec{x}$ of the system one reaction event at a time. The fundamental steps in every exact SSA formulation are thus:

1. Sample τ and μ from Eqs. 11.20 and 11.21,
2. Update state $\vec{x} = \vec{x} + \vec{\nu}_\mu$ and time $t = t + \tau$,
3. Recompute the reaction propensities a_μ from the changed state.

We here only look at the two classical exact SSA formulations due to Gillespie. We note, however, that many more exact SSA formulations exist in the literature, including the Next Reaction Method (NRM, introducing dependency graphs for faster propensity updates), the Optimized Direct Method (ODM), the Sorting Direct Method (SDM), the SSA with Composition-Rejection sampling (SSA-CR, using composition-rejection sampling as outlined in Section 2.4.1 to sample μ), and partial-propensity methods (factorizing the propensity and independently operating on the factors), which we do not discuss here. All are different formulations of the same algorithm, exact SSA, and sample the exact same trajectories. However, the computational cost of different SSA formulations may differ for certain types or classes of networks.
A defining feature of exact SSAs is that they explicitly simulate *each and every* reaction event. Once can in fact show that algorithms that skip, miss, or lump reaction events cannot sample from the exact solution of the Master equation any more. However, they may still provide good and convergent weak approximations, at least for low-order moments of $p(\vec{x}, t)$. Examples of such *approximate SSAs* are τ-leaping, R-leaping, and Langevin algorithms. We do

not discuss them here. They are very much related to numerical discretizations of stochastic differential equations, as discussed in Chapter 10.

11.3.1 The first-reaction method (FRM)

The first-reaction method is one of the earliest exact SSA formulations, derived by Gillespie in 1976. In this formulation, the time τ_μ when reaction μ fires next is sampled according to the probability density function

$$p(\tau_\mu \mid \vec{x}(t)) = a_\mu \, \mathrm{e}^{-a_\mu \tau_\mu} \tag{11.23}$$

using the inversion method *i.i.d.* for each μ. Subsequently, the next reaction μ is chosen to be the one with the minimum τ_μ, and the time τ to the next reaction is set to the minimum τ_μ. The algorithm is given in Algorithm 5.

Algorithm 5 First Reaction Method

```
1: procedure FRM-SSA($\vec{x}_0$)                                    ▷ Initial state $\vec{x}_0$
2:     Set $t \leftarrow 0$; initialize $\vec{x}$, $a_\mu \, \forall\mu$, and $a$
3:     for $k = 0, 1, 2, \dots$ do
4:         Sample $\tau_\mu$ according to Eq. 11.23 for each reaction $\mu$: For each reaction generate an i.i.d. uniform random number $r_\mu \sim \mathcal{U}(0,1)$ and compute $\tau_\mu \leftarrow -a_\mu^{-1} \log(r_\mu)$. $\tau \leftarrow \min\{\tau_1, \dots, \tau_M\}$
5:         $\mu \leftarrow$ the index of minimum$\{\tau_1, \dots, \tau_M\}$
6:         Update: $\vec{x} \leftarrow \vec{x} + \vec{\nu}_\mu$, where $\vec{\nu}_\mu$ is the total stoichiometry of reaction $\mu$; recompute all $a_\mu$ and $a$
7:         $t \leftarrow t + \tau$
8:     end for
9: end procedure
```

The computational cost of FRM is $O(M)$ where M is the number of reactions in the network. This is due to steps 4 and 6 in Algorithm 5, both of which have a runtime of $O(M)$: step 4 involves generating M random numbers and step 6 involves recomputing all M reaction propensities.

11.3.2 The direct method (DM)

The direct method (Gillespie, 1977) first samples the next reaction index μ according to Eq. 11.21 using linear search over the reaction propensities. The time τ to the next reaction is then sampled according to Eq. 11.20. The algorithm is given in Algorithm 6.
The computational cost of DM is also $O(M)$. This is due to steps 4 and 6 in Algorithm 6, both of which have a worst-case runtime of $O(M)$. In terms of absolute runtimes, however, DM is more efficient that FRM since it does not involve the expensive step of generating M exponential random numbers for each reaction event. It only requires two random numbers in each iteration.

Algorithm 6 Direct Method

```
1: procedure DM-SSA($\vec{x}_0$)                                  ▷ Initial state $\vec{x}_0$
2:     Set $t \leftarrow 0$; initialize $\vec{x}$, $a_\mu \, \forall \mu$, and $a$
3:     for $k = 0, 1, 2, \dots$ do
4:         Sample $\mu$ using linear search according to Eq. 11.21: generate a uniform random number $r_1 \sim \mathcal{U}(0, 1)$ and determine $\mu$ as the smallest integer satisfying $r_1 a < \sum_{\mu'=1}^{\mu} a_{\mu'}$
5:         Sample $\tau$ according to Eq. 11.20: generate a uniform random number $r_2 \sim \mathcal{U}(0, 1)$ and compute $\tau$ as $\tau \leftarrow -a^{-1} \log(r_2)$
6:         Update: $\vec{x} \leftarrow \vec{x} + \vec{\nu}_\mu$, where $\vec{\nu}_\mu$ is the stoichiometry of reaction $\mu$; recompute all $a_\mu$ and $a$
7:         $t \leftarrow t + \tau$
8:     end for
9: end procedure
```